DESIGN AND APPLICATIONS OF NEW HARMONY SEARCH ALGORITHMS

Ph.D. THESIS

by

ASSIF ASSAD

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
JULY, 2017

DESIGN AND APPLICATIONS OF NEW HARMONY SEARCH ALGORITHMS

A THESIS

Submitted in partial fulfilment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

by

ASSIF ASSAD

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
JULY, 2017

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **"DESIGN AND APPLICATIONS OF NEW HARMONY SEARCH ALGORITHMS"** in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy and submitted in the Department of Mathematics of the Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during a period from July, 2014 to July, 2017 under the supervision of Dr. Kusum Deep, Professor, Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

(ASSIF ASSAD)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Kusum Deep)
Supervisor

The Ph.D. Viva-Voice Examination of **Mr. Assif Assad**, Research scholar, has been held on November 20, 2017.

Chairman, SRC **Signature of External Examiner**

This is to certify that the student has made all the corrections in the thesis.

Signature of Supervisor **Head of the Department**
Dated:

Abstract

Optimization is the process of finding the best alternate solution among a given set of solution under some given constraints. The process of finding maximum or the minimum possible value, which a function can attain in its domain, is known as optimization.

One of the most striking trend that emerged in the optimization field is the simulation of natural processes as efficient global search methods. The natural processes or phenomena are firstly analyzed mathematically and then coded as computer programs for solving complex nonlinear real world problems. The resulting methods are called “Nature Inspired Algorithms” that can often outperform classic methods. The advantages of these methods are their ability to solve various standard or application based problems successfully without any prior knowledge of the problem space. Moreover, these algorithms are more likely to obtain the global optima of a given problem. They do not require any continuity and differentiability of the objective functions. Also, they work on a randomly generated population of solutions instead of one solution. They are easy to programme and can be easily implemented on a computer. Some of the examples of Nature Inspired Optimization Techniques are Genetic Algorithm, Particle Swarm Optimization, Artificial Bee Colony Optimization and Ant Colony Optimization.

Harmony Search (HS) is a musicians behavior inspired metaheuristic algorithm developed in 2001, though it is a relatively new meta heuristic algorithm, its effectiveness and advantages have been demonstrated in various applications like traffic routing, multi objective optimization, design of municipal water distribution networks, load dispatch problem in electrical engineering, rostering problems, clustering, structural design, classification and feature selection to name a few.

The aim of this PhD Thesis is to improve the efficiency and reliability of Harmony Search algorithm in the context of solving real life problems. The organization of this thesis is as follows.

Chapter 1 is introductory in nature it states the definitions and underlines the objectives and motivation behind this Thesis. It also reviews the available literature. The chapter closes with a brief summary of the work presented in this Thesis.

Chapter 2 introduces a novel algorithm based on hybridization of Harmony search and Simulated Annealing called HS-SA to inherit their advantages in a complementary way and overcome their limitations. Taking the inspiration from Simulated Annealing the proposed HS-SA algorithm accepts even the inferior harmonies, compared to the harmonies already stored in Harmony Memory, with probability determined by parameter called Temperature. The Temperature parameter is initially kept high to favour exploration of search space and is linearly decreased to gradually shift focus to exploitation of promising search areas. The performance of HS-SA is analyzed and compared with Harmony Search algorithm and Simulated Annealing on IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed algorithm on multimodal benchmark functions. The performance of HS-SA is particularly outstanding on composition problems. Composition functions are combined, rotated, shifted, and biased version of other unimodal and multi-modal test functions and mimic the difficulties of real search spaces by providing a massive number of local optima and different shapes for different regions of the search space.

Chapter 3 introduces Two Phase Harmony Search (TPHS) algorithm that attempts to strikes a balance between exploration and exploitation by concentrating on diversification in the first phase using catastrophic mutation and then switches to intensification using local search in the second phase. Catastrophic mutation allows the evolutionary algorithm to increase diversity in the population on the cost of decreasing convergence speed so as to escape the local optimal. The key differences between standard HS and TPHS are:

1. In TPHS both PAR and BW are linearly decreased whereas both PAR and BW remain constant in standard HS (where PAR is the Pitch adjustment rate and BW is the Bandwidth).
2. The Harmony Memory is re initialized except the elite (catastrophic mutation) if the best harmony is not updated in L generations, where L is predefined number of generations.
3. Towards the end of the execution the algorithm shifts focus to exploitation by performing local search around the best solutions.

The performance of TPHS is analyzed and compared with 15 state-of-the-art metaheuristic algorithms namely Standard HS, Improved Harmony Search Algorithm, Global Best Harmony Search Algorithm, Self-adaptive Global Best Harmony Search Algorithm, Evolution Strategy with Covariance Matrix Adaptation, Comprehensive Learning PSO, Adaptive Particle Swarm Optimization, Dynamic Neighbourhood Learning PSO, Heterogeneous Comprehensive Learning PSO, Social Learning PSO, Self Regulating PSO, Social Spider Optimization Algorithm, Differential Evolution, Differential Evolution With Successful Parent Selecting Framework, Dynamic Multi-swarm Particle Swarm Optimizer with Harmony Search on IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed TPHS algorithm in terms of accuracy particularly on multimodal functions.

Chapter 4 introduces Shrinking Memory Harmony Search (SMHS) algorithm, the SMHS attempts to strike a balance between exploration and exploitation by concentrating on diversification in the beginning using extended memory and broad Bandwidth and then gradually switching to intensification by shrinking harmony memory and utilizing local search operator. The performance of SMHS is compared with nineteen state-of-the-art metaheuristic algorithms (four HS variants, five PSO variants, eight DE variants, and one variant each of GA and ES). The numerical results demonstrate the superiority of the proposed SMHS algorithm on multimodal function and its performance is outstanding on composition functions.

In Chapter 5 the performance of the three proposed algorithms namely HS-SA, TPHS & SMHS is compared on IEEE CEC 2014 benchmark suite and a real life problem called Camera Calibration. The problem of camera calibration has been studied extensively in photogrammetry and computer vision community because of its important applications such as vehicle guidance, robotic navigation and 3D-reconstruction. Camera calibration problem deals with finding the geometrical relationship between the 3D scene and its 2D images taken by two cameras. It defines exactly how the scene has been projected by the camera to result in the given image(s). Camera calibration involves:

1. Determination of the orientation and position of the camera with respect to the scene which is specified by extrinsic parameters.

2. Determination of the internal geometric and optimal characteristics of the camera, which is specified by intrinsic parameters.

It is established SMHS is the better performing algorithm on all categories of benchmark functions & Camera Calibration problem.

The development of hybrid procedures for optimization focuses on enhancing the strength and compensating for the weakness of two or more complementary approaches. The goal is to intelligently combine the key elements of the competing methodologies to create a superior solution procedure. The objective of Chapter 6 is to explore the hybridization between Harmony Search (HS) and Hill Climbing (HC) algorithm by utilizing the exploration power of the former and exploitation power of the latter in the context of solving Sudoku which is a well-known hard Combinatorial Optimization problem. We call this hybrid algorithm Harmony Search Hill Climber (HSHC). In order to extend the exploration capabilities of HSHC it is further modified to create three different algorithms namely Retrievable Harmony Search Hill Climber (RHSHC), Global Best Retrievable Harmony Search Hill Climber (GB-RHSHC) and Random Best Retrievable Harmony Search Hill Climber (RB-RHSHC). RHSHC perform significantly better than standard Harmony Search algorithm and standard Hill Climber algo-

rithm. On comparing RHSHC with the Genetic Algorithm it has been concluded that former outperforms latter both in terms of effectiveness and efficiency particularly on Hard and Expert level puzzles. Comparing RHSHC and hybrid AC3-tabu search algorithm it has been concluded that RHSHC is very competent to hybrid AC3-tabu search algorithm.

The maximum clique problem (MCP) is to determine a complete sub graph (clique) of maximum cardinality in a given graph. MCP is conspicuous for having real world applications and for its potentiality of modelling other combinatorial problems and is one of the most studied NP-hard problems. Chapter 7 investigates the capabilities of Harmony Search algorithm for solving maximum clique problem. We propose and compare two different instantiations of a generic HS algorithm namely Harmony Search for MCP (HS_MCP) and Harmony Search with idiosyncratic harmonies for MCP (HSI_MCP) for this problem. HS_MCP has better exploitation and inferior exploration capabilities than HSI_MCP whereas HSI_MCP has better exploration and inferior exploitation capabilities than HSI_MCP, it has been concluded that former performs better than latter by testing them on all the instances of DIMACS benchmark graphs. HS_MCP has been compared with a recently proposed Harmony search based algorithm for MCP called Binary Harmony search (BHS) and the simulation results show that HS_MCP significantly outperforms BHS in terms of solution quality.

Skin, rich in lycopene, is an important component of waste originating from tomato (lycopersicon esculentum) paste manufacturing plants. Lycopene belongs to the carotenoid family, is a bright red pigment that has received great interest due to its various biological activities. Lycopene is a potent antioxidant and has been found effective in reducing the risk of chronic diseases by protecting cells against oxidative damage. Various studies have shown that lycopene is associated with decreasing the risk of breast and prostate cancer. According to the World Processing Tomato Council 1,200,000 tons of tomato processing waste is produced worldwide. One of the main components of tomato processing waste is skin. At present, the tomato processing waste is either discarded or used as animal fodder, but its abundance in lycopene makes it a promising prospect as a sustainable, alternative and low-cost source of this nutraceutical compound. Chapter 8 deals with Optimization of Lycopene extraction from tomato processing waste skin using Harmony Search Algorithm.

The Thesis concludes with the Chapter 9. It derives the overall conclusions of this Thesis. It outlines the limitations and scope of the proposed algorithms. Later it suggests future scope and new directions of research in this area.

Acknowledgements

In the name of God, the infinitely Good, the All Merciful.

All praise is due to God, Lord of the worlds.

On the verge of giving a final shape to my dream, it gives me immense pleasure in expressing deep sense of gratitude to my supervisor Prof. Kusum Deep for her invaluable and scrupulous guidance, enthusiastic interest throughout my research work. Her affectionate treatment, perspicaciousness and magnanimity made feasible to conclude this gigantic task. I am highly indebted for her gentle and benign behavior, incredible broad vision and constructive criticism for this thesis work. I humbly acknowledge a lifetime‘s gratitude to her.

I express my regards to Prof. V.K. Katiyar, Head of the Department, Prof. R.C. Mittal, former Head of the Department, and Prof. S.P. Yadav, DRC Chairperson for their support. Special thanks to my SRC members: Dr. Madhu Jain and Dr. Durga Toshniwal (Dept of CSE) for spending their valuable time during the discussions over the seminars. I am indebted to the Department of Mathematics, IIT Roorkee, to all its faculty and staff for their academic support and encouragement. I am also thankful to the members of Institute Computer Center for their cooperation and support.

I am thankful to my seniors Dr. Millie Pant, Dr. Jagdish Bansal, Dr. Kedar Nath, Dr. Anupam Yadav, Dr. Amarjeet Singh, Dr. Garima Singh, Dr. Amreek Singh, Dr. Vanita Garg . I am also thankful to my colleagues Kavita Gupta, Shail Kumar Dinkar, Shubham Gupta, Teekam Singh, Amit Kumar, Vishnu Singh for providing healthy and progressive research environment. I consider myself truly blessed as I have always been in a good company of friends. Their love, inspiration, supports and cooperation is beyond the scope of any acknowledgement, yet I would like to express my heartfelt gratitude to my friends here at IIT Roorkee Jamid Ul Islam, Aashiq Hussain Ganie, Shakeel Waseem, Qazi Inam, Shujaat Buch, Towseef, John Mohd Wani, Lateef Wani, Shahnawaz Baba, Shahbaz, Suhail and back home Umar Jawaid, Ummar Muhammad, Sheraz Ahmad, Imtiyaz Ahmad.

Now I would like to thank some remarkable people who had, are having and will continue to have a tremendous influence on my life. I feel a deep sense of gratitude for my grand parents, Mr. Abdul Rahman and Mrs. Khateja, my parents Mr. Assadullah Dar and Mrs. Haleema who raised me with their unconditional love, mental support that I needed in every step and every sphere of my life. My parents & grand parents have been the biggest source of inspiration in my life and have inculcated in me the values that really matter in life. My sincere and heartfelt gratitude to my siblings Mudassir, Cheshmeeda and Rafia for their love and support. Most importantly, I want to thank my wife, Rifat, and our two sons Faouz and Fayak. During my PhD period I have not been able to fulfil all my responsibilities as a husband and a father properly. No words can express my gratitude and appreciation to her, and my family, for the kind support and tremendous care.

I thank Islamic University of Science and Technology Awantipora for granting me leave to pursue PhD in IIT Roorkee, further I would like to thank QIP center IIT Roorkee and AICTE India for sponsoring my research.

With profound gratitude, love and devotion, I dedicated this thesis to my grand father, Mr. Abdul Rahman who has always dreamed of me becoming a doctorate from a prestigious institute.

My heartfelt thanks to all who have supported me for successful realization of this Thesis. My sincere apologies, if I have forgot someone but I am grateful for their support. Finally, I would also like to thank all the readers of this work, since any piece of academia is useful if it is read and understood by others so that it can become bridge for further research.

Assif Assad

Roorkee, July 2017

Contents

List of Tables

List of Figures

List of Abbreviations

cm CENTIMETER.

DE DIFFERENTIAL EVOLUTION.

ES EVOLUTIONARY STRATEGIES.

g GRAM.

GA GENETIC ALGORITHM.

HPCL HIGH PRESSURE LIQUID CHROMATOGRAPHY.

HS HARMONY SEARCH.

mg MILI GRAM.

mm MILLIMETER.

nm NANOMETER.

PSO PARTICLE SWARM OPTIMIZATION.

RSA RESPONSE SURFACE ANALYSIS.

v/w VOLUME/WEIGHT.

Chapter 1

Introduction

This introductory Chapter states the definitions and underlines the objectives and motivation behind this Thesis. It also reviews the available literature. The chapter closes with a brief summary of the work presented in this Thesis as well as future research directions.

1.1 Optimization

Optimization is the methodology of choosing "the best" alternatives(s) among a specified set of available options. This approach of determining "the largest"/"the smallest" possible value, that a given mathematical expression can attain in its specified domain of definition, is called optimization. The mathematical expression that has to be optimized could be linear, nonlinear, integer, geometric or fractional. In some situations, explicit mathematical formulation of the function is not readily defined or may not be available. Many times the mathematical function which needs to be optimized has restrictions in the form of inequality or equality constraints. Therefore, the process of optimization can be considered as a problem of finding those values of the independent variables which do not violate the inequality and equality constraints in such a way to provide an optimal value of the mathematical function being optimized. In other words, the mathematical techniques for determining the optimal value or values ("the greatest possible value" or "the least possible value") of a mathematical function are called "Optimization Techniques". Determining the solution of most realistic problems may not be possible in the absence of robust optimization techniques. In literature, numerous books are available based on mathematical concepts of optimization and some of the references are: (Bertsekas, 2014, 2015; Boyd and Vandenberghe, 2004; Chandra et al., 2009; Himmelblau, 1972; Kapur et al., 2011; Mohan and Deep, 2009; P C Jha, 2011; Rao, 2009) .

1.2 Definition of an Optimization Problem

Mathematically speaking, the most general formulation of single objective optimization problem is:

$$\text{Minimize or Maximize } \; f(x) \tag{1.1}$$

$$x = (x_1, x_2, \ldots, x_D)$$

$$\text{Subject to } x \in F, \text{usually defined by} F = \{x \in R^D\} \;\; s.t.$$

$$h_i(x) = 0; \;\; i = 1, 2, ..., m \tag{1.2}$$

$$g_j(x) \geq 0; \;\; j = m+1, m+2, ..., p \tag{1.3}$$

$$a_i \leq x_i \leq b_i; \;\; i = 1, 2, 3..., D \tag{1.4}$$

Where $f, h_1, h_2, ..., h_m, g_{m+1}, g_{m+2}, ..., g_p$ are real valued functions on R^D.

Function $f(x)$ that is to be optimized (maximized or minimized) is called the 'objective function'. Equations $h_i(x) = 0$ for $i = 1, 2, ..., m$ are known as the equality constraints and $g_j(x) \geq 0; \;\; for \;\; j = m+1, m+2, ..., p$ are called inequality constraints. Inequality constraint of the type $g_j(x) \geq 0$ can be written as $-g_j(x) \leq 0$. It is desired to determine those values of the independent variables $x_1, x_2, ..., x_D$, which optimize the objective function $f(x)$ without violating any of the restrictions imposed in equation (1.2), (1.3) and (1.4). The variables x_i's are known as 'decision variables'. a_i's are the lower bounds and b_i's are the upper bounds of the decision variables. A decision vector $x = (x_1, x_2, ...x_D) \in R^D$ which satisfies all the constraints is called a 'feasible solution'. A feasible solution which optimizes the objective function is called a feasible optimal solution.
On the basis of presence of constraints, there are two types of optimization problems named unconstrained optimization problems and constrained optimization problems. Unconstrained optimization problems involve an objective function given by equation (1.1) or lower or upper bounds on variables given by equation (1.4). Constrained optimization problems involve an objective function given in equation (1.1), the box constraints given by equation (1.4) and linear and/or non-linear, equality constraints given by equation (1.2) and/or inequality constraints given by equation (1.3). Due to presence of equality and inequality constraints, constrained optimization problems are more difficult to solve.

1.3 Local and Global Optimal Solutions

Let F denote the feasible region of the solution vector that satisfies all the constraints of an optimization problem. Then, in case of a minimization problem, if for $\bar{x} \in F$ there exists an ϵ neighborhood $N_\epsilon(\bar{x})$ around $\bar{x}$ such that $f(x) \geq f(\bar{x})$ for each $x \in F \cap N_\epsilon(\bar{x})$ then $\bar{x}$ is known as a 'local minimum solution'. However, if, $\bar{x} \in F$ and $f(x) \geq f(\bar{x})$ for all $x \in F$ then $\bar{x}$ is known as a 'global minimum solution' of the optimization problem at hand. Figure 1.1 shows local and global optimum solutions of a mathematical function. In general it may happen that there are either no optimal solutions, or

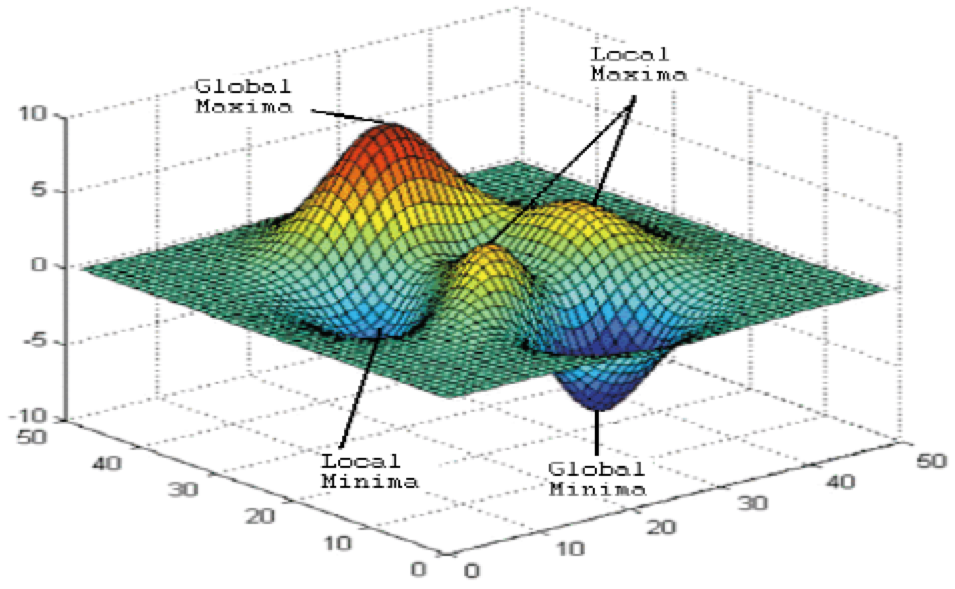


Figure 1.1: Demonstration of Local optima and Global optima.

a unique optimal solution or several optimal solutions, for a given nonlinear optimization problem. In case a problem has a single local optimal solution then it is also the global optimal solution. If, however, the optimization problem has several local optimal solutions, then, in general, one or more of them could be the global optimal solutions. In a Linear Programming Problem, it is for sure that, every local optimal solution is the global optimal solution. On the contrary, in case of a Non Linear Optimization Problem, if the objective function is convex (for minimization case) and its feasible domain is also convex, then it is guaranteed that the local optimal solution is also the global optimal solution.

In many nonlinear optimization problems, it is usually desirous to determine a global optimal solution instead of a local optimal solution. But, in general, it is often difficult to obtain the global optimal solution of a nonlinear optimization problem, rather than finding the local optimal solution. However, due to its practical significance, it becomes necessary to determine the global optimal solution.

For a mathematical function which is twice-differentiable, there exist conditions which may be used to determine a local optimal solution. In case the test fails, then due to the property of continuous differentiability of function a solution with a lesser objective function value can be determined in its neighborhood. Thus, a sequence of solutions can be constructed which converge to the local optimal solution. However, in general, such tests are not sufficient. It may be said that, a global optimization problem is not solvable in a finite number of steps. Therefore any given solution cannot be guaranteed as a solution of global minima without evaluating the objective function at least at one solution of its neighborhood. But, the neighborhoods of a solution may be unbounded; therefore, an infinite numbers of steps are required to attain the global minima.

1.4 Methods for Global Optimization

Global optimization focuses on determining the best (minimum) of the local minima. Designing global optimization techniques is not an easy task since, in general, there is no criterion for deciding whether a global optimal solution has been achieved or not. In view of the practical necessity and with the availability of fast and readily computing machines, many computational techniques are now being reported in literature for solving nonlinear optimization problems. The methods currently available in literature for solving nonlinear global optimization problems may be broadly classified as deterministic methods and probabilistic methods.

The deterministic methods try to guarantee that a neighborhood of the global optima is attained. Such methods do not use any stochastic techniques, but rely on a thorough search of the feasible domain. However, they are applicable only to a restricted class of functions. On the other hand, probabilistic methods are used to find the near optimal solution. This is achieved by assuming that the good solutions are near to each other in the search space. This assumption is valid for most real life problems (Omran, 2005). The probabilistic methods make use of probabilistic or stochastic approach to search for the global optimal solutions. Although probabilistic methods do not give an absolute guarantee, these methods are sometimes preferred over the deterministic methods because they are applicable to a wider class of functions.

1.5 Nature Inspired Computing Techniques

One of the most striking trend that emerged in the optimization field is the simulation of natural processes as efficient global search methods. The natural processes or phenomena are firstly analyzed mathematically and then coded as computer programs for solving complex nonlinear real world problems. The resulting methods are called ‘Nature Inspired Algorithms (NIA)’ that can often outperform classic methods. The advantages of these methods are their ability to solve various standard or application based problems successfully without any prior knowledge of the problem space. Moreover, these algorithms are more likely to obtain the global optima of a given problem. They do not require any continuity and differentiability of the objective functions and / or constraints. Also, they work on a randomly generated population of solutions instead of one solution. They are easy to programme and can be easily implemented on a computer.

The most primitive example of nature inspired optimization techniques is that of Genetic Algorithms (Holland, 1975). It is based on the Darwin's Theory of Evolution which is based on the property of inheritance and survival of the fittest in living organisms. The decision parameters are encoded into a encoded space (Binary / Real / Octal, etc.) and crossover, mutation and elitism is performed over a number of generations until a pre specified stopping criteria is attained. A local search based Genetic Algorithms has been proposed in (Sawyerr et al., 2014). Genetic Algorithms has been applied to many real world problems including (Ganjidoost et al., 2016; Kiran et al., 2016; Singh and Bhukya, 2011; Valente et al., 2011)

Genetic Programming (Poli et al., 2008) is one of a number of population-based evolutionary algorithms inspired by natural evolution and is widely used in machine learning. It allows a computer to automatically solve predefined tasks without requiring users to know or specify the form or structure of the solution in advance. Genetic Programming has been applied to many real world problems including (Bhardwaj et al., 2016, 2014; Liu et al., 2016; Purohit et al., 2010). Differential Evolution proposed in (Storn and Price, 1997) uses only the mutation operator on a target vector. (Ali and Zhu, 2013) proposes extended Differential Evolution for constrained optimization using penalty function.

Another important development is the introduction of Particle Swarm Optimization by Kennedy and Eberhart in (Kennedy, 2011). It mimics the behavior of a flock of birds or school of fish. All the solution or particles of the swarm fly through the search space using their personal best position in history as well as the global best position of the entire swarm. In (Ali and Kaelo, 2008) an improved PSO is proposed to obtain faster convergence. Particle Swarm Optimization has been applied to many real world problems like (Bedi et al., 2011, 2013; Nayak et al., 2015; Nwankwor et al., 2013; Roula et al., 2015; Rout et al., 2016; Shourian et al., 2008b)

Glow Worm Swarm Optimization (Krishnanand and Ghose, 2006, 2009) mimics the behavior of glow worms which emit light in order to attract the others in the group for mating. It is particularly designed to capture multiple local and global optima.

Artificial Bee colony optimization (Karaboga, 2005) is based on Self-organization and division of labor. That is, it is based on inspecting the behaviors of bees on finding nectar and sharing the information of food sources to the bees in the hive, by the employed bees, onlooker bees and scouts. Artificial Bee colony optimization has been applied to many real world problems including (Bhattacharjee et al., 2011)

Another Swarm Intelligence based algorithm is the Spider Monkey Algorithm given by Bansal et. al.

in (Bansal et al., 2014). It is based on the foraging behavior and fission-fusion social structures of spider monkeys.
Ant Colony Optimization (Dorigo et al., 2006) is proposed wherein the pheromone left behind ants and their ability to change their path as and when an obstacle is encountered on their path, is mimicked into the design of Ant colony optimization. Ant Colony Optimization has been applied to many real world problems like (Bedi et al., 2009)
The behavior of the growth of bacteria forms a basis of Bacterial Foraging Optimization Algorithm (Passino, 2002).
Grey Wolf Optimizer (Mirjalili et al., 2014) is a relatively new nature inspired optimization technique which mimics the leadership hierarchy and hunting mechanism of grey wolves. Four types of grey wolves such as alpha, beta, delta and omega are employed for simulating the leadership hierarchy by incorporating the three steps of hunting namely searching for prey, encircling the prey and attacking the prey.
Some methods draw their inspiration from the physical laws of nature. For example Gravitational Search Algorithm is based on gravitational interaction between masses (Rashedi et al., 2009). It artificially simulates the Newton's Theory, Newtonian laws of gravitation and motion. Similarly, Central Force Optimization (Formato, 2009) is based on gravitational kinematics.
Inspired by biological neural networks, Artificial Neural Networks (ANN) are massively parallel computing systems consisting of an extremely large number of simple processors with many interconnections. ANN have found diverse applications including (Shourian et al., 2008a; Verma, 2012; Verma et al., 2010; Wang, 1998; Wang et al., 2008b).

1.6 The No Free Lunch Theorem

A major and interesting result in numeric optimization literature was the presentation of the "No Free Lunch (NFL) theorem" given in (Wolpert and Macready, 1997; Wolpert et al., 1995). This theorem states "that the performance of all optimization (search) algorithms, amortized over the set of all possible functions, is equivalent." The theorem has far reaching implications, because it implies that "no algorithm can be designed so that it will be superior to a linear enumeration of the search space, or even a purely random search". Although, the theorem is defined over finite search spaces only, however, it is not proved if the result is applicable to infinite search spaces, e.g. R^D. All computer implementations of search algorithms will, in general, operate on finite search spaces; therefore the theorem is applicable to all existing algorithms. The NFL Theorem states that all search algorithms perform equally well over all functions, it does not necessarily hold for all subsets of this set. The set of all functions over a finite domain includes the set of all the permutations of this domain.

1.7 Harmony Search Algorithm

Harmony Search (HS) (Geem et al., 2001)is a musician's behavior inspired evolutionary algorithm developed in 2001, though it is a relatively new meta heuristic algorithm, its effectiveness and advantages have been demonstrated in various applications.

Weyland (Weyland, 2012) raised an issue regarding the novelty of Harmony Search algorithm by declaring it a special case of $(\mu+1)-$ES, however the pitch adjustment operator used in HS is entirely different than the mutation operator used in ES. Further HS utilizes the pitch adjustment operator (local search) probabilistically in contrast to ES's mutation operator and thus the two can't be considered same. Ample evidence has been provided in (Saka et al., 2016) to show HS is not a special case of $(\mu + 1)-$ ES even though superficially they seem to be identical.

In order to explain the Harmony Search in detail, let us first idealize the improvisation process by a skilled musician. When a musician is improvising there are three possible choices:

1. Play any piece of music exactly from his memory.

2. Play something similar to a known piece.

3. Compose new or random notes.

Geem et al. (Geem et al., 2001) formalized these three options into quantitative optimization process and the three corresponding components become usage of harmony memory (HM), pitch adjusting, and randomization. The usage of HM is similar to the choice of the best fit individuals in genetic algorithms. In order to use this memory effectively, it is typically assigned a parameter called harmony memory considering rate (HMRC $\in$ [0, 1]). If this rate is low (near 0), only few best harmonies are utilized and thus convergence of algorithm is slow. If this rate is very high (near 1), it results in exploitation of the harmonies in the HM, thus the solution space is not explored properly leading to potentially inefficient solutions. Typically HMRC $\in$ [.7, .95] is used. The second component is pitch adjustment determined by a pitch bandwidth (BW) and a pitch adjusting rate (PAR), it corresponds to generating a slightly different solution in the HS algorithm. Pitch can be adjusted linearly or nonlinearly however most often linear adjustment is used. So we have

$$H_i^{new} = H_i^{old} \quad + \quad BW \times r_i \quad where \quad r_i \in [-1, 1] \quad and \quad 1 \leq i \leq D \tag{1.5}$$

Where H_i^{old} is the i^{th} component of the existing harmony or solution and $H_i^{new}$is the i^{th} component of new harmony after the pitch adjusting action. The Equation (1.5) essentially produces a new solution around the existing solution by altering it slightly by a very small random amount. Here r_i is a random number generated in the range of [-1, 1] and D is total number of components in the harmony. The pitch adjusting rate (PAR) controls the degree of adjustment. A low pitch adjusting rate with a narrow bandwidth can slow down the convergence of HS because of limitation in exploration of only a small subspace of the whole search space. On the other extreme a very high PAR with a wide bandwidth may cause the algorithm to swing around some optimal solution. Thus the recommended value of PAR $\in$ [.1, .5]. The third component of the HS is the randomization, which is used to increase the exploration of the search space. Although pitch adjustment plays a some what similar role, but it is confined to close neighborhood of harmony and thus corresponds to local search. The use of randomization pushes the algorithm further to explore diverse search areas to find the global optima. The pseudo code of harmony search is shown as Algorithm 1. In the pseudo code H represents a potential solution or harmony, rand $\in$ [0, 1] is a uniformly distributed random number generator, rand_int(1, HMS) generates a uniformly distributed integer random number between 1 and HMS, HMS is the size of harmony memory and D is the dimension of problem.

1.7.1 Harmony Search variants based on handling of parameter

In order to enhance the performance of the standard HS algorithm several variants of HS algorithm have been proposed in literature. The parameters- HMCR, PAR and BW remain constant in standard HS. In order to strike a balance between exploration and exploitation the parameters of HS are dynamically altered giving rise to different variations of the HS algorithm. A detailed survey on variants of HS can be found in (Mohd Alia and Mandava, 2011).

1.7.1.1 Improved Harmony Search

Mahdavi et al. proposed dynamic adaptation of both pitch adjustment rate (PAR) and bandwidth (BW), the algorithm is known as Improved Harmony Search (IHS) (Mahdavi et al., 2007). In Improved Harmony Search PAR is linearly increased in each iteration using Equation (1.6) and BW is exponentially decreased in each iteration using Equation (1.7).

$$PAR_{gn} = PAR_{min} + \frac{(PAR_{max} - PAR_{min})}{NI} \times (gn - 1) \tag{1.6}$$

where PAR_{gn} is pitch adjusting rate for each generation, PAR_{min} is the minimum pitch adjusting rate, PAR_{max} is the maximum pitch adjusting rate, NI is the Maximum number of generations, gn is

Algorithm 1 HARMONY SEARCH ALGORITHM (HSA)

```
1: Let f() be the given Objective function.
2: Define harmony memory consideration rate (HMCR).
3: Define pitch adjustment rate (PAR) and bandwidth(BW).
4: Define Harmony Memory Size (HMS).
5: Initialize Harmony Memory (HM).
6: while (Stopping Criteria Not Reached) do
7:     Find current Worst and Best harmonies in HM.
8:     for $i = 1$ to $D$ do
9:         if (rand $\leq$ HMCR) then
10:            $H_i = HM_i^j$ where j=rand_int(1,HMS)
11:            if (rand $\leq$ PAR) then
12:                $H_i = H_i \pm rand \times BW$
13:            end if
14:        else
15:            Generate $H_i$ randomly within the allowed bounds.
16:        end if
17:    end for
18:    if (H is better than worst Harmony in HM) then
19:        Update HM by replacing WORST harmony by H.
20:    end if
21: end while
22: print Best Harmony as obtained solution.
```

generation number.

$$BW_{gn} = BW_{max} \times exp(c \times (gn - 1)) \quad where \quad c = \frac{Ln(\frac{BW_{min}}{BW_{max}})}{NI} \tag{1.7}$$

In Equation (1.7) BW_{gn} represents bandwidth for each generation, BW_{min} and BW_{max} are respectively the minimum and maximum bandwidth. All the other steps in IHS are similar to that of standard Harmony Search algorithm.

It has been rightly pointed out in (Taherinejad, 2009) that increasing the PAR and decreasing the BW in IHS is conspicuous because initially when the BW is high, the low value of PAR keeps it under utilized and finally when the PAR becomes high, the low value of BW seizes it to improve the harmony significantly.

1.7.1.2 Global Best Harmony Search

Inspired from Particle Swarm Optimization paradigm Omran and Mahdavi introduced another important modification to Harmony Search algorithm referred as Global-best Harmony Search (GHS) algorithm (Omran and Mahdavi, 2008). In GHS the parameter PAR is linearly increased as in IHS algorithm, however the concept of BW has completely been removed and thus pitch adjustment step (Step 12 of Algorithm 1) has been modified as Equation (1.8).
In order to generate i^{th} component of the harmony H, in pitch adjustment step, k^{th} component of the best harmony in HM is assigned to it.

$$H_i = HM_k^{Best} \quad where\ i,\ k \in \{1,2,...,D\}\ and\ D\ is\ the\ dimension\ of\ harmony. \tag{1.8}$$

Assigning the k^{th} component of the best harmony to the i^{th} component of the new harmony in Equation (1.8) is disputable because most of the times different components of problem dimension are independent of each other and second there may be vast differences in search ranges from one dimension to another. Further directing the search towards the best harmony in HM causes a serious side effect of premature convergence due to lack of diversity in HM.

1.7.1.3 Adaptive Harmony Search algorithm

A new adaptation for HS was proposed in (Hasançebi et al., 2009; Saka and Hasancebi, 2009) by changing HMCR and PAR dynamically during the execution of Harmony Search algorithm. Initially HMCR and PAR are respectively set to $HMCR^{(0)}$ and $PAR^{(0)}$ then the dynamic calculation of these parameters is adapted as follows:

$$HMCR^k = (1 + \frac{1 - HMCR^{'}}{HMCR^{'}} \times e^{-\gamma \cdot N(0,1)})^{-1} \tag{1.9}$$

$$PAR^k = (1 + \frac{1 - PAR^{'}}{PAR^{'}} \times e^{-\gamma \cdot N(0,1)})^{-1} \tag{1.10}$$

where $HMCR^k$ and PAR^k are the sampled values of the adapted parameters for a new harmony vector. N(0, 1) is a normally distributed random number in the range of 0 to 1, γ is called learning rate recommended to be in range of [0.25, 0.50]. $HMCR^{'}$ and $PAR^{'}$ are the average values of improvisation parameters obtained by averaging the corresponding values of all the solution vectors within the HM matrix i.e.

$$HMCR^{'} = \frac{\sum_{i=1}^{HMS} HMCR^i}{HMS} \tag{1.11}$$

$$PAR' = \frac{\sum_{i=1}^{HMS} PAR^i}{HMS} \tag{1.12}$$

1.7.1.4 Self-adaptive Harmony Search

Self adaptive Harmony Search was proposed by Wang et al. in (Wang and Huang, 2010). The authors of Self adaptive Harmony Search introduced three modifications in standard HS. In the initialization step a low discrepancy sequence is used to have more uniformly initialized harmonies in the HM. The second important alteration is to decrease PAR during execution rather than increasing it, to prevent overshooting and oscillation around the optimal solution. The third modification is the removal of BW parameter and changing the pitch adjustment step (step 12 of Algorithm 1) as shown below.

$$trial^i = (max(HM^i) - trial^i) \times ran[0, 1) \tag{1.13}$$

$$trial^i = (trial^i - min(HM^i)) \times ran[0, 1) \tag{1.14}$$

where ran[0, 1) is a uniformly distributed random number ranging from 0 to less than 1, $trial^i$ is the i^{th} variable selected from HM, and $max(HM^i)$ and $min(HM^i)$ are respectively highest and lowest values of the i^{th} variable in the HM. In each step one of the Equations (1.13) or (1.14) is randomly selected to adjust the current pitch.

1.7.1.5 Self-adaptive Global Best Harmony Search

Pan et al. modified the GHS algorithm to create Self-adaptive Global Best Harmony Search (SAGHS) algorithm (Pan et al., 2010b). The main difference between GHS and SAGHS is that pitch adjustment step (Step 12 of Algorithm 1) has been modified as Equation (1.15).
Thus to generate i^{th} component of the harmony H, in the pitch adjustment step i^{th} component of the best harmony in HM is assigned to it.

$$H_i = HM_i^{Best} \quad where\ i \in \{1, 2, ..., D\}\ ,$$
$$D\ is\ the\ dimension\ of\ harmony\ and\ Best\ is\ the\ index\ of\ best\ harmony\ in\ HM. \tag{1.15}$$

In order to increase diversity in HM, the harmony memory consideration step (Step 10 of Algorithm 1) is modified as Equation (1.16). Further the BW is dynamically updated in the algorithm as Equation (1.17).

$$H_i = HM_i^j \pm rand \times BW \quad where\ i \in \{1,2,...,D\},\ j \in \{1,2,...,HMS\}$$
$$D\ is\ the\ dimension\ of\ harmony\ and\ HMS\ is\ the\ size\ of\ HM. \tag{1.16}$$

$$BW_{gn} = \begin{cases} BW_{max} - \frac{BW_{max}-BW_{min}}{NI} \times 2(gn-1) & if\ gn < \frac{NI}{2} \\ BW_{min} \quad if\ gn \geq \frac{NI}{2} \end{cases} \tag{1.17}$$

In Equation (1.17) BW_{gn} represents bandwidth for each generation, BW_{min} and BW_{max} is the minimum and maximum bandwidth respectively.

1.7.1.6 Other variants of Harmony Search based on handling of parameters

Cheng et al. (Cheng et al., 2008) developed Modified Harmony Search (MHS), which is based on the idea of selecting better harmony with higher probability. Pan et al. proposed a local best Harmony Search algorithm with dynamic sub populations (DLHS) in (Pan et al., 2010a). In DLHS the HM is divided into many small sized sub HMs and then independent evolution is performed on each sub-HM. The sub HMs are regrouped frequently to exchange information and maintain diversity. Explorative Harmony Search (EHS) algorithm proposed by Das et. al. eliminates the limitation of tuning the BW parameter by making it proportional to the population variance in HM (Das et al., 2011). Another variant of HS based on the idea of increasing the PAR rather than decreasing it so as to favor exploration in the beginning of the algorithm has been proposed in (Taherinejad, 2009). An Intelligent Tuned Harmony Search algorithm has been proposed in (Yadav et al., 2012) and an Improved Global-best harmony search algorithm has been introduced in (El-Abd, 2013).

1.7.2 Variants based on hybridization of HS with other metaheuristic algorithms

Harmony Search has been successfully hybridized with other metaheuristic algorithms. The goal of hybridization is to improve the capabilities of the optimization algorithms to solve complex problems (Blum and Roli, 2008; Grosan and Abraham, 2007). The inception of the ability of HS algorithm to be integrated with other metaheuristic return to the relative ease and flexible structure of HS.
Geem (Geem, 2009) hybridized the Harmony search with Particle swarm optimization algorithm to

create an efficient algorithm for solving water network design problem. A modified HS by integrating a component from Dispersed particle swarm optimization (Cai et al., 2008) is proposed in (Dos Santos Coelho and De Andrade Bernert, 2009) and has been utilized for synchronization of discrete time chaotic systems. Wang et al. (Wang et al., 2009) improved the performance of HS by integrating it with Clonal Selection Algorithm (CSA) (De Castro and Von Zuben, 2000). All the harmonies in the HM were updated by using Clonal Selection Algorithm. Even though this approach resulted in increases of computational time, however it improved the effectiveness of HS to deal with the premature convergence. In (Lee and Zomaya, 2009) three metaheuristic algorithms namely Simulated Annealing (SA), GA and Artificial Immune System (AIS) were used to enhance the quality of the harmonies stored in HM and thus increase the convergence speed and at the same time preventing the HS from getting stuck in the local optima. To enhance exploitation capabilities of HS it has been hybridized with Sequential Quadratic Programming (SQP) that acts as a local search operator in (Fesanghary et al., 2008). SQP is applied with a probability of P_c to improve the quality of the new improvised vector. Also as a final step once the HS meets the stopping criteria SQP is applied to the best harmony so that its quality can further be improved. Jang et al. (Jang et al., 2008) proposed a hybrid framework that combined HS with Nelder Mead Simplex Algorithm a local search component to improve the quality of stored harmony memory vectors in HM. A hybrid HS and Differential Evolution has been proposed in (Gao et al., 2008).

1.7.3 Applications of Harmony Search Algorithm

The number of applications of HS is too diverse and large to allow for a complete enumeration. In this paragraph some of the recent applications of HS are listed as: Dynamic relocation of mobile base station in wireless sensor networks (Mohd Alia, 2017), Optimization of buttressed earth-retaining walls ((Molina-Moreno et al., 2017)), Supervised learning (Elola et al., 2016), Resource Leveling Problem with minimal lags (Ponz-Tienda et al., 2017), Optimization of renewable energy charging with energy storage system ((Geem and Yoon, 2017)), Water quality prediction (Jaddi and Abdullah, 2017), Water distribution system design(Jung et al., 2017), Gene selection for cancer classification (Elyasigomari et al., 2017), Integrated production and transportation scheduling in MTO manufacturing (Guo et al., 2017), Feature selection for high dimensional imbalanced class data (Moayedikia et al., 2017), Multi-objective optimization (Valaei and Behnamian, 2017), Kurdish character recognition (Zarro and Anwer, 2016), Day-ahead scheduling problem of a microgrid with consideration of power flow constraints (Zhang et al., 2016), Optimizing urban traffic light scheduling problem (Gao et al., 2016), Nearest Neighbor realization of quantum circuits on a 2-Dimensional grid (Alfailakawi et al., 2016), economic load dispatch problem in electrical engineering (Al-Betar et al., 2016; Wang and Li, 2013), Optimization of truss structure (Cheng et al., 2016), Remodularization for object-oriented software systems (Chhabra et al., 2017), Fine tuning of Deep Belief Networks (Papa et al., 2016), Image reconstruction from projections (Ouaddah and Boughaci, 2016), Epileptic seizure detection (Zainuddin et al., 2016), Project Portfolio Selection (Esfahani et al., 2016), Stock price prediction (Dash and Dash, 2016), Optimal power flow for power system security enhancement ((Pandiarajan and Babulal, 2016)), unit commitment problem (Kamboj et al., 2016), Energy-efficient routing algorithm for wireless sensor networks (Zeng and Dong, 2016), clustering (Hoang et al., 2014), Optimization of Hydropower Storage Projects (Mousavi et al., 2017),classification and feature selection (Diao and Shen, 2012; Fattahi et al., 2015). A detailed survey on applications of HS can be found in (Manjarres et al., 2013; Yoo et al., 2014).

1.8 Motivation and Objectives of the Thesis

The efficiency of a metaheuristic algorithms depend on the extent of balance between diversification and intensification during the course of the search. An ideal metaheuristic algorithm must have efficient exploration in the beginning of execution and enhanced exploitation towards the end. In order to strike a balance between the two contradictory properties of exploration and exploitation two variants of HS are proposed namely Two Phase Harmony Search and Shrinking Memory Harmony Search. The HS algorithm is hybridized with another established metaheuristic algorithm namely Simulated Annealing. The HS algorithm is hybridized with Hill Climbing operator to solve a combinatorial optimization problem.
This Thesis is computationally dominant and interdisciplinary in nature. The objectives of this Thesis in brief are:

1. To design efficient and reliable Harmony Search based algorithms.
2. To test the algorithms on benchmark problems appearing in literature.
3. To use the algorithms for solving real life optimization problems arising in various fields of science and engineering.

1.9 Organization of the Thesis

The chapter wise summary of the Thesis is given below:
Chapter 1 is introductory in nature. Besides stating the relevant definitions it gives an introduction to

Harmony Search Algorithm and existing literature review.

Chapter 2 introduces a novel algorithm based on hybridization of Harmony search and Simulated Annealing called HS-SA to inherit their advantages in a complementary way. The performance of HS-SA is analyzed and compared with Harmony Search algorithm and Simulated Annealing on IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed algorithm on multimodal benchmark functions.

Chapter 3 introduces Two phase Harmony search (TPHS) algorithm that attempts to strikes a balance between exploration and exploitation by concentrating on diversification in the first phase using catastrophic mutation and then switches to intensification using local search in the second phase. The performance of TPHS is analyzed and compared with 15 state-of-the-art metaheuristic algorithms on all the 30 IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed TPHS algorithm in terms of accuracy particularly on multimodal functions.

Chapter 4 introduces Shrinking Memory Harmony search (SMHS) algorithm, the SMHS attempts to strike a balance between exploration and exploitation by concentrating on diversification in the beginning using extended memory and broad Bandwidth and then gradually switching to intensification by shrinking harmony memory and utilizing local search operator. The performance of SMHS is compared with nineteen state-of-the-art metaheuristic algorithms (four HS variants, five PSO variants, eight DE variants, and one variant each of GA and ES) on all the 30 IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed SMHS algorithm on multimodal function and its performance is outstanding on composition functions.

In Chapter 5 the performance of the three proposed algorithms is compared on IEEE CEC 2014 benchmark suite and a real life problem called Camera Calibration (a highly non linear, 12 dimensional optimization problem from the field of computer vision). It is established SMHS is the better performing algorithm not only on all categories of benchmark functions but also on Camera Calibration problem.

Chapter 6 introduces a specialized Memetic algorithms created by hybridization of Harmony Search Algorithm and Hill Climbing operator to solve Sudoku puzzles, a well known NP-Complete combinatorial optimization problem.

In Chapter 7 Harmony search is utilized to solve Maximum Clique problem a well known NP-Hard combinatorial optimization problem.

In Chapter 8 extraction of lycopene from tomato pomace is formulated as a five dimensional nonlinear optimization problem and HS algorithm is used to determine the optimal setting of parameters for extraction of lycopene from tomato pomace.

The Thesis concludes with the Chapter 9. It derives the overall conclusions of this Thesis. It outlines the limitations and scope of the proposed algorithms. Later it suggests future scope and new directions of research in this area.

Chapter 2

A Hybrid Harmony Search and Simulated Annealing Algorithm for Continuous Optimization

2.1 Introduction

Harmony search is a powerful metaheuristic algorithm with excellent exploitation capabilities but suffers a limitation of premature convergence if one or more initially generated solutions/harmonies are in the vicinity of local optima. In order to take care of this limitation this chapter proposes a novel algorithm based on hybridization of Harmony Search and Simulated Annealing called HS-SA to inherit their advantages in a complementary way. Taking the inspiration from Simulated Annealing the proposed HS-SA algorithm accepts even the inferior harmonies, compared to the harmonies already stored in Harmony Memory, with probability determined by parameter called Temperature. The Temperature parameter is initially kept high to favor exploration of search space and is linearly decreased to gradually shift focus to exploitation of promising search areas. The performance of HS-SA is analyzed and compared with Harmony Search algorithm and Simulated Annealing on IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed algorithm on multimodal benchmark functions. The performance of HS-SA is particularly outstanding on composition problems which are generally very challenging test beds for meta-heuristic algorithms.

The organization of this chapter is as follows. Section 2.2 provides an introduction to Simulated Annealing algorithm, Section 2.3 provides a detailed description about the proposed HS-SA algorithm, Section 2.4 provides detail about numerical experimentation and analysis of results and the chapter concludes with Section 2.5.

2.2 Simulated Annealing

Simulated Annealing (SA) is an iterative meta-heuristic for solving nonlinear and non-convex optimization problems. It was introduced in (Kirkpatrick et al., 1983) based on the Metropolis algorithm (Metropolis et al., 1953) and different variations were introduced later in (Xinchao, 2011) and (García-Martínez et al., 2012). SA has been extensively used in problems such as Traveling Salesman Problem (Geng et al., 2011), packing problem (Hopper and Turton, 2001), supply chain management (Subramanian et al., 2013), vehicle routing (Tam and Ma, 2004), machine scheduling (Jin et al., 2009), timetabling (Schaerf, 1999) and Neural Networks (Wang and Smith, 1998).

A typical SA algorithm for a minimization problem is given as Algorithm 2. The algorithm starts from an initial randomly selected solution S (step 2) and temperature parameter denoted as T is set to its initial value T_0 (step 3). The cost of the initial solution S is calculated (step 4) and the following steps are repeated until the stopping criterion is satisfied. A neighbor solution S' of S is generated and its fitness is calculated (step 7). The Neighbor() function is used to generate a new solution of S by changing the value of one or more components. A better solution is always accepted and an inferior solution is also accepted with a probability determined by its fitness and current temperature T (step 12). The temperature is initially kept high so as to favor inferior moves and is gradually decreased by a factor α (step 16) to lower the probability of accepting inferior moves.

Algorithm 2 SIMULATED ANNEALING (SA)

```
 1: Let $f()$ be the given Objective function.
 2: Initialize $S, T_0\ and\ L$
 3: $T \leftarrow T_0$
 4: $Cost\_Current = f(S)$
 5: while (Stopping Criteria Not Reached) do
 6:     $S' \leftarrow Neighbor(S)$
 7:     $Cost\_New = f(S')$
 8:     $\triangle\ Cost = Cost\_New - Cost\_Current$
 9:     if $(\triangle\ Cost < 0)$ then
10:         $S = S' \quad and \quad Cost\_Current = Cost\_New$
11:     else
12:         if $(rand(0,1) < e^{\frac{-\triangle Cost}{T}})$ then
13:             $S \leftarrow S' \quad and \quad Cost\_Current = Cost\_New$
14:         end if
15:     end if
16:     After every L iterations Set $T = \alpha T$
17: end while
```

2.3 Proposed Hybrid Harmony Search and Simulated Annealing (HS-SA) algorithm

Harmony Search is a powerful metaheuristic algorithm with excellent exploitation capabilities, however it has serious limitation of getting stuck in local optimal usually referred as premature convergence if the initially selected harmonies are in the vicinity of local optima. In order to remove this limitation HS algorithm is hybridized with SA by proposing HS-SA algorithm so as to increase exploration particularly in the beginning of execution to escape local optima.

The success of a meta heuristic algorithm depends on the extent of balance between exploration and exploitation. An ideal meta heuristic algorithm must have greater exploration capabilities in the earlier generations and enhanced exploitation capabilities towards the later generations Yadav et al. (2012). In HS-SA an attempt has been made to achieve this goal. Taking the inspiration from SA, the HS-SA algorithm accepts even inferior harmonies with probability determined by a parameter called Temperature (T). The Temperature parameter is initially kept high to favor inferior moves and hence increase capability of escaping local optima and is linearly decreased to gradually shift focus to exploitation of good harmonies. The pseudo code of HS-SA for a minimization problem is shown as Algorithm 3. The HS-SA and standard HS share the same structure, with the exception that even inferior harmonies are accepted in HS-SA. Step 2 initializes the algorithmic parameters and it can be observed that three extra parameters of Simulated Annealing (T_0, α, L) are also required in HS-SA. Step number 7 to 16 generate a new harmony as in standard HS. In step 18 not only the superior harmonies (compared to worst harmony) are always accepted but the inferior harmonies are accepted with probability determined by the fitness of the new harmony and the current temperature. The temperature parameter is gradually decreased in step 21 so as to reduce the probability of accepting inferior harmonies and hence favour exploitation of good harmonies. In Algorithm 3 rand is a uniform random number generator generating random numbers between 0 and 1.

2.4 Numerical Experiments on CEC 2014 benchmark suite

In this section, the performance of the proposed HS-SA algorithm is evaluated on IEEE CEC 2014 Benchmark functions (Liang et al., 2013). The HS-SA algorithm is compared with standard HS and Simulated Annealing. The experimentation has been carried out on all the IEEE CEC 2014 Benchmark functions using 30 dimensions. As per the instructions of test suite every problem is tested with 51 independent runs.

The parameter setting adopted for standard HS has been taken from (El-Abd, 2013) and is shown in Table 2.1 along with the parameter setting of SA and HS-SA algorithm. Obtaining an optimal pa-

Algorithm 3 HYBRID HARMONY SEARCH SIMULATED ANNEALING (HS-SA)

```
1: Let $f()$ be the given Objective function
2: Initialize Parameters HMCR, PAR, BW, HMS, $T_0, \alpha, L$
3: Initialize Harmony Memory (HM)
4: Set $T = T_0, \quad L = 3 \times D$
5: while (Stopping Criteria Not Reached) do
6:    Find current Worst harmony and Best harmony in HM
7:    for $i = 1$ to $D$ do
8:       if (rand $\leq$ HMCR) then
9:          $H_i = HM_i^j$ where j=rand_int(1,HMS)
10:         if (rand $\leq$ PAR) then
11:            $H_i = H_i \pm rand \times BW$
12:         end if
13:      else
14:         Generate $H_i$ randomly within the allowed bounds
15:      end if
16:   end for
17:   $\triangle \, Cost = f(H) - f(Worst)$
18:   if $(\triangle \, Cost < 0 \quad OR \quad rand < e^{\frac{-\triangle Cost}{T}})$ then
19:      Update HM by replacing Worst harmony by H
20:   end if
21:   After every L iterations Set $T = \alpha T$
22: end while
23: print Best Harmony as obtained solution
```

rameter setting for a metaheuristic algorithm is a hyper optimization problem, however the parameter setting shown in Table 2.1 was found out to be appropriate for most, if not all the problem instances. Parameter T_0 for every function has been calculated by using Equation (2.1). Where Worst and Best in Equation (2.1) respectively refer to the worst and best function value obtained after evaluating the function for ten random vectors, β is the initial acceptance rate and is taken as 0.95.

$$T_0 = \frac{Worst - Best}{log(\beta)} \tag{2.1}$$

All the algorithms have been implemented in Dev C++ 5.0 and the experimentation has been carried out on a laptop with Windows 10 operating system, intel core i3 processor and 4GB of RAM.

Table 2.1: Parameter setting of algorithms used in this study.

Algorithm	HMS	HMCR	PAR	BW	T_0	α	L
HS (Geem et al., 2001)	5	.9	.3	.001	-	-	-
SA (Kirkpatrick et al., 1983)	-	-	-	-	*	0.99	3×D
HS-SA	5	.9	.3	.001	*	0.99	3×D

* indicates that T_0 has been calculated as explained in section titled Numerical Experiments and - indicated the parameter is not applicable.

2.4.1 IEEE CEC 2014 Benchmark suite

The IEEE CEC 2014 Benchmark suite is a collection of 30 unconstrained continuous optimization problems with varying difficulty levels. The functions 1 through 3 are unimodal, 4 through 16 are simple multimodal functions, 17 through 22 are hybrid functions and 23 through 30 are composition functions. The search range for each function is $[-100, 100]^D$ where D is the dimension of the problem.

Each problem is tested with 51 independent runs and the error values obtained in 51 runs are sorted from the smallest (best) to the largest (worst) and then best, worst, mean, median and standard variance of error values for each function is presented. MaxFES is the maximum number of function evaluations allowed and is equal to $10^4 \times D$.

2.4.2 Analysis of results

The results reported in this chapter are in the format as specified and required in IEEE CEC 2014 benchmark suite. Tables 2.2, 2.3 and 2.4 show the results on 30 dimensional problems, the best results are highlighted in bold font. The recorded results are the minimum, maximum, mean, median, and standard deviation of the error value obtained as specified in IEEE 2014 Benchmark suite. The error value is the absolute difference between obtained objective function value by the algorithm and the known function value.

For unimodal functions SA reports the best results and the best mean results in two instances and HS obtained the best result and best mean result in one instance. In case of simple multimodal functions HS-SA, HS and SA reported the best results in 5, 4 and 3 instances respectively, and reported the best mean results in 6, 4 and 2 instances respectively. In case of six instances of hybrid multimodal functions all the three algorithms produced best results in two instances; HS-SA, SA and HS respectively produced the best mean results in 7, 3, 2 instances. In case of composition multimodal functions HS-SA, HS and SA produced the best mean results in 5, 2, 1 instances respectively and the best results in 6, 1, 1 instances respectively.

From the above discussion it can be concluded that HS-SA algorithm outperforms its competitors on multimodal functions, the superior performance of HS-SA is particularly evident on composition functions, which are the shifted, rotated, expanded, and combined variants of the classical functions and hence offer the greatest complexity.

2.4.2.1 Convergence Behaviour

The convergence graphs of all the benchmark functions are plotted in Figures 2.1, 2.2, 2.3 and 2.4 to study the convergence behavior of algorithms. The horizontal axis represents the number of function evaluations and the vertical line represents the mean of absolute error of 51 runs in logarithmic scale. As is evident from most of the convergence graphs SA shows slow convergence in the beginning compared to both HS and SA, however it makes steep progress towards the end. Comparing HS and HS-SA they almost follow same trajectory however in most of the cases HS-SA produces better results compared to HS. Since HS-SA has enhances exploration capabilities compared to HS, thus making it capable to escape local optima and hence it shows better performance.

2.4.3 Wilcoxon rank test analysis

The paired Wilcoxons rank-sum test is conducted at the 5% significance level to judge if the difference in performance between the HS-SA and the competing algorithm is statistically significant, the results are reported in Table 2.5. The cases are respectively marked as '+', '-' and '=' when the performance of HS-SA is significantly better than, worse than, or similar to the competing algorithm. Table 2.5 reveals that in most of the cases the difference in performance is statistically significant and there are only a few instances where the difference is not significant.

For unimodal functions HS outperformed HS-SA on one instance whereas the difference is statistically insignificant in the remaining two instances. In case of simple multimodal functions HS-SA significantly outperformed HS on eight instances whereas latter outperformed former on three instances and the difference is statistically insignificant in the remaining two instances. On hybrid functions HS-SA significantly outperformed HS on five instances whereas latter outperformed former in the remaining one instances. In case of composition functions HS-SA significantly outperformed HS on five instances whereas latter outperformed former on two and in the remaining one instance the difference is not statistically significant.

For unimodal functions HS-SA outperformed SA on one instance whereas latter outperformed former in the remaining two instances. In case of simple multimodal functions HS-SA significantly outperformed SA on eight instances whereas latter outperformed former on four instances and the difference is statistically insignificant in the remaining one instance. On hybrid functions HS-SA and SA significantly outperformed each other on three instances. In case of composition functions HS-SA significantly outperformed HS on six instances whereas latter outperformed former on one instances and in the remaining one instance the difference is not statistically significant.

2.4.3.1 Algorithm Complexity

The time complexity of algorithms is computed as per the requirements laid down in IEEE CEC 2014 Benchmark suite. The complexity is represented in terms of three parameters T0, T1 and T2. T0 is the time complexity of a specified test program provided in IEEE CEC 2014 Benchmark suite reproduced as Algorithm 4. T1 is the computing time for 2×10^5 function evaluations of function 18 with dimension D and T2 is the computing time taken by the algorithm when the stopping criteria is 2×10^5 function evaluations of function 18 with dimension D. The time complexity for 30 dimensional problem is shown as Figure 2.5.

It is evident from Figure 2.5 the time complexity of the algorithms shows the following order.

$$SA < HS < HS - SA$$

HS-SA has to re adjust the temperature parameter during execution resulting in slightly higher time complexity compared to HS.

Algorithm 4 TEST PROGRAM (T0)

```
for $i = 1$ to $1000000$ do
    x=0.5+(double) i
    x=x + x; x=x/2; x=x*x; x=sqrt(x); x=log(x); x=exp(x); x=x/(x+2);
end for
```

2.5 Conclusion

This article introduces a hybrid variant of Harmony Search algorithm for continuous optimization problems with the aim to exhibited the desired behavior of exploring the search space at the earlier iterations and exploiting good solutions towards the later iterations. This was achieved by initially setting the parameter Temperature to a high value so as to favor inferior moves. The Temperature parameter is linearly reduced to gradually shift the focus from exploration of search space to exploitation of promising search areas. The performance of proposed HS-SA algorithm is evaluated as per the specifications laid down in IEEE CEC 2014 benchmark suite and it is established by statistical tests that the proposed algorithm is highly efficient on multimodal function.

Table 2.2: Error values obtained by HS, SA and HS-SA on Function number 1 through 10.

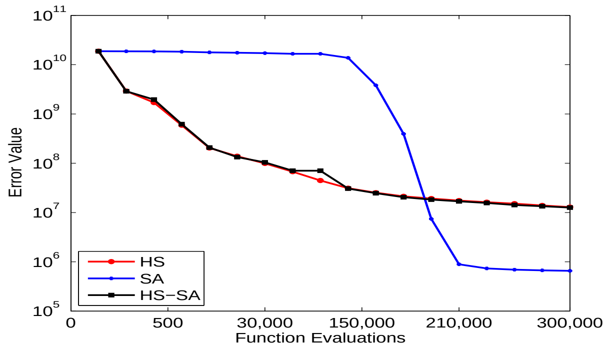
Function	Algorithm	Mean	Median	Best	Worst	Std Dev
1	HS	1.2898E+07	1.1256E+07	1.2757E+06	3.7050E+07	8.5380E+06
	SA	**6.5497E+05**	**5.6590E+05**	**1.1262E+05**	**1.9510E+06**	**4.1818E+05**
	HS-SA	1.2617E+07	8.5402E+06	1.5442E+06	4.0779E+07	1.0093E+07
2	HS	1.0748E+04	5.8189E+03	1.4352E+02	3.4211E+04	1.1490E+04
	SA	**6.8042E+03**	**4.3200E+03**	**1.6228E+01**	**2.9534E+04**	**7.5039E+03**
	HS-SA	1.2802E+04	1.2407E+04	1.2917E+02	3.3934E+04	1.0268E+04
3	HS	**4.6459E+03**	**3.9072E+03**	**8.6021E+00**	2.1796E+04	4.1211E+03
	SA	2.7893E+04	2.7115E+04	7.8582E+03	4.4693E+04	9.2701E+03
	HS-SA	6.1764E+03	4.4345E+03	2.4125E+02	**1.7541E+04**	4.8560E+03
4	HS	1.1612E+02	1.2484E+02	6.3859E+01	1.5181E+02	3.0220E+01
	SA	**2.3841E+00**	**4.4402E-01**	**6.6404E-02**	**6.9998E+01**	**1.0170E+01**
	HS-SA	1.1738E+02	1.2605E+02	6.0184E+00	2.0606E+02	4.1858E+01
5	HS	2.0000E+01	2.0000E+01	2.0000E+01	2.0000E+01	3.8090E-05
	SA	2.0000E+01	2.0000E+01	2.0000E+01	2.0007E+01	1.4842E-03
	HS-SA	2.0000E+01	2.0000E+01	2.0000E+01	2.0002E+01	2.6141E-04
6	HS	1.4421E+01	1.4553E+01	9.8893E+00	1.9776E+01	2.0702E+00
	SA	3.5211E+01	3.5751E+01	2.6362E+01	4.5260E+01	4.0480E+00
	HS-SA	**1.2055E+01**	**1.2013E+01**	**9.8626E+00**	**1.9544E+01**	2.1116E+00
7	HS	1.5451E-02	**7.8050E-03**	**1.5700E-04**	**8.3465E-02**	1.8883E-02
	SA	2.2008E-02	1.5028E-02	8.3000E-05	8.8494E-02	1.9567E-02
	HS-SA	**1.4292E-02**	1.2552E-02	1.9900E-04	9.7968E-02	2.0653E-02
8	HS	**4.3647E-05**	**4.4000E-05**	**2.6000E-05**	**5.6000E-05**	**6.6622E-06**
	SA	3.9893E+02	3.9599E+02	2.4277E+02	5.8702E+02	8.0111E+01
	HS-SA	4.6588E-05	4.7000E-05	2.6000E-05	6.4000E-05	8.0053E-06
9	HS	6.8566E+01	6.6662E+01	4.2633E+01	9.7506E+01	1.4704E+01
	SA	5.6210E+02	5.6711E+02	2.7950E+02	8.9842E+02	1.2471E+02
	HS-SA	**6.5356E+01**	**6.5647E+01**	**3.6814E+01**	**9.1536E+01**	**1.3080E+01**
10	HS	2.0355E-01	2.0977E-01	1.0515E-01	3.3868E-01	5.5092E-02
	SA	4.6142E+03	4.5796E+03	3.4108E+03	6.0320E+03	6.3234E+02
	HS-SA	**2.0148E-01**	**1.9047E-01**	**8.5307E-02**	**2.9387E-01**	**4.6566E-02**

Table 2.3: Error values obtained by HS, SA and HS-SA on Function number 11 through 20.

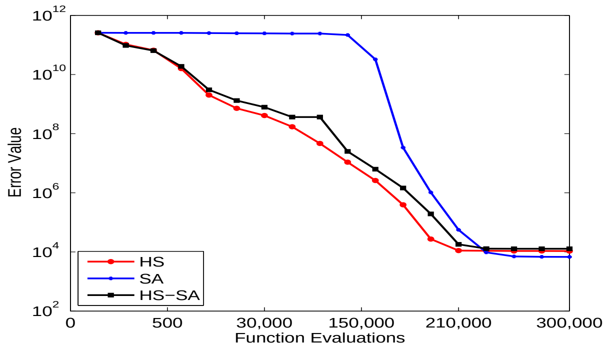
Function	Algorithm	Mean	Median	Best	Worst	Std Dev
11	HS	2.0271E+03	2.0486E+03	1.0901E+03	**2.9462E+03**	4.0440E+02
	SA	4.5791E+03	4.5506E+03	3.2777E+03	6.0769E+03	5.4029E+02
	HS-SA	**1.9899E+03**	**1.9559E+03**	**6.3470E+02**	3.3266E+03	4.9702E+02
12	HS	1.6681E-01	1.5897E-01	6.1356E-02	2.6858E-01	5.0019E-02
	SA	**2.4637E-02**	**2.0439E-02**	**8.4100E-03**	**6.4771E-02**	**1.2607E-02**
	HS-SA	1.6853E-01	1.6566E-01	**6.1270E-02**	3.0058E-01	5.2063E-02
13	HS	5.4085E-01	5.4299E-01	**3.0465E-01**	8.7141E-01	1.1932E-01
	SA	5.4220E-01	5.3973E-01	3.6364E-01	7.5985E-01	1.0655E-01
	HS-SA	**5.3161E-01**	**5.2862E-01**	3.0982E-01	**8.0228E-01**	1.1346E-01
14	HS	4.4240E-01	3.5203E-01	2.4030E-01	1.0571E+00	2.2474E-01
	SA	**2.9681E-01**	**2.8557E-01**	**2.0912E-01**	**3.9718E-01**	**4.6954E-02**
	HS-SA	4.0668E-01	3.4310E-01	2.0409E-01	1.0504E+00	2.0518E-01
15	HS	1.4252E+01	1.3387E+01	5.5428E+00	2.7132E+01	5.5849E+00
	SA	**9.1952E+00**	**8.4518E+00**	**3.9489E+00**	**1.5876E+01**	**2.6056E+00**
	HS-SA	1.4155E+01	1.2918E+01	4.8635E+00	2.9317E+01	5.7329E+00
16	HS	9.4670E+00	9.5518E+00	**7.8243E+00**	**1.1028E+01**	6.5864E-01
	SA	1.4192E+01	1.4272E+01	1.2597E+01	1.4848E+01	4.1428E-01
	HS-SA	**9.4563E+00**	**9.4030E+00**	8.0110E+00	1.1269E+01	7.6297E-01
17	HS	1.8725E+06	1.1507E+06	6.5245E+04	7.8273E+06	1.6072E+06
	SA	**3.7270E+05**	**3.2775E+05**	**5.0462E+04**	**9.4757E+05**	**1.9633E+05**
	HS-SA	2.0083E+06	1.6854E+06	1.3187E+05	6.8902E+06	1.4732E+06
18	HS	6.2863E+03	2.5298E+03	7.2457E+01	2.8773E+04	7.8468E+03
	SA	**3.2887E+03**	**1.8619E+03**	1.3694E+02	**2.1540E+04**	**3.9771E+03**
	HS-SA	6.1800E+03	4.6128E+03	**4.0631E+01**	2.4763E+04	6.2672E+03
19	HS	2.9814E+01	9.3142E+00	5.9647E+00	**1.1896E+02**	3.5770E+01
	SA	2.1447E+01	1.2027E+01	9.0321E+00	7.3727E+01	2.1834E+01
	HS-SA	**2.0454E+01**	**9.2266E+00**	**5.8494E+00**	1.3300E+02	3.3613E+01
20	HS	6.3997E+03	**5.6052E+03**	**1.7248E+02**	**2.4392E+04**	4.9723E+03
	SA	4.0537E+04	4.0253E+04	5.6433E+03	1.0063E+05	2.0853E+04
	HS-SA	**6.2076E+03**	6.1032E+03	8.0318E+02	2.6505E+04	5.5557E+03

Table 2.4: Error values obtained by HS, SA and HS-SA on Function number 21 through 30.

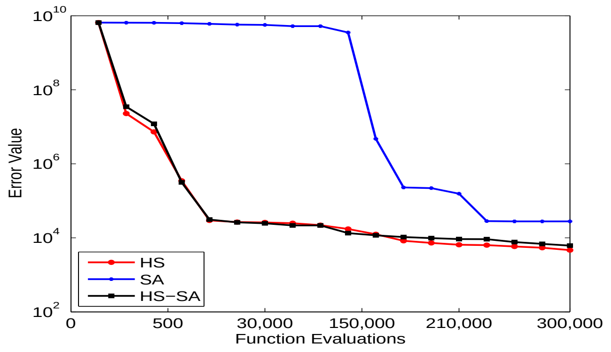
Function	Algorithm	Mean	Median	Best	Worst	Std Dev
21	HS	7.4351E+05	6.5937E+05	6.9129E+04	2.2859E+06	5.3016E+05
	SA	**4.5160E+05**	**4.5294E+05**	**2.0710E+04**	**1.0308E+06**	**2.3946E+05**
	HS-SA	6.5564E+05	5.0706E+05	8.1763E+04	2.1329E+06	5.2939E+05
22	HS	4.7430E+02	4.8986E+02	**4.0511E+01**	**7.6838E+02**	1.6864E+02
	SA	1.1004E+03	1.1054E+03	2.9535E+02	1.5805E+03	2.6886E+02
	HS-SA	**4.5885E+02**	**4.8636E+02**	1.6711E+02	9.4424E+02	1.8586E+02
23	HS	3.1542E+02	3.1534E+02	3.1493E+02	3.1666E+02	3.7407E-01
	SA	**3.1524E+02**	**3.1524E+02**	**3.1524E+02**	**3.1524E+02**	**1.7983E-04**
	HS-SA	3.1563E+02	3.1560E+02	3.1524E+02	3.1697E+02	3.6749E-01
24	HS	2.3340E+02	2.3168E+02	2.2666E+02	2.4891E+02	5.0111E+00
	SA	3.5520E+02	2.9313E+02	2.4826E+02	8.2182E+02	1.2077E+02
	HS-SA	**2.3032E+02**	**2.3009E+02**	**2.2663E+02**	**2.4624E+02**	**4.9981E+00**
25	HS	2.0729E+02	2.0699E+02	2.0427E+02	2.1326E+02	1.7382E+00
	SA	2.2085E+02	2.2064E+02	2.0322E+02	2.5340E+02	1.3047E+01
	HS-SA	**2.0503E+02**	**2.0595E+02**	**2.0425E+02**	**2.1227E+02**	**1.6913E+00**
26	HS	1.3069E+02	1.0064E+02	1.0034E+02	3.3391E+02	5.2147E+01
	SA	1.6399E+02	1.0070E+02	1.0030E+02	1.0479E+03	1.3431E+02
	HS-SA	**1.2486E+02**	**1.0063E+02**	**1.0033E+02**	**2.0154E+02**	**4.8889E+01**
27	HS	6.3111E+02	6.1789E+02	4.0264E+02	**8.1322E+02**	6.9457E+01
	SA	1.0175E+03	1.2241E+03	4.0137E+02	1.5153E+03	4.6734E+02
	HS-SA	**6.0663E+02**	**6.0815E+02**	**4.0136E+02**	8.8188E+02	**1.5391E+02**
28	HS	1.0206E+03	9.9931E+02	8.1043E+02	**1.4169E+03**	1.2232E+02
	SA	6.3822E+03	6.3662E+03	2.6346E+03	1.1300E+04	1.8660E+03
	HS-SA	**1.0009E+03**	**9.9930E+03**	**7.9611E+02**	2.2693E+03	2.6564E+02
29	HS	**1.4609E+03**	1.3822E+03	6.1623E+02	**2.7196E+03**	**4.3376E+02**
	SA	1.1862E+06	1.4408E+03	8.0681E+02	1.8308E+07	4.1591E+06
	HS-SA	1.7137E+05	**1.2378E+03**	**5.6384E+02**	8.6704E+06	1.2019E+06
30	HS	4.1742E+03	4.1078E+03	**1.5014E+03**	8.3453E+03	1.4848E+03
	SA	**3.0790E+03**	**2.9167E+03**	1.7383E+03	**8.0588E+03**	**1.0475E+03**
	HS-SA	5.2030E+03	4.8460E+03	1.5169E+03	1.3999E+04	2.3352E+03



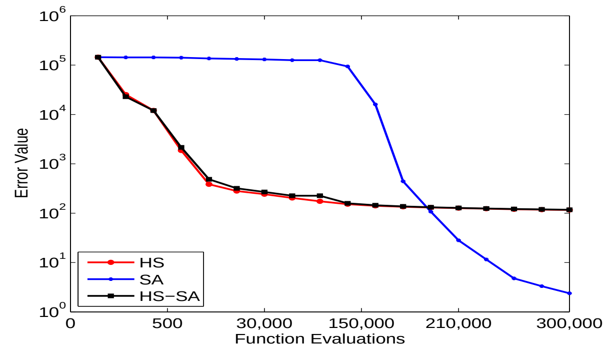
(a) Function 1



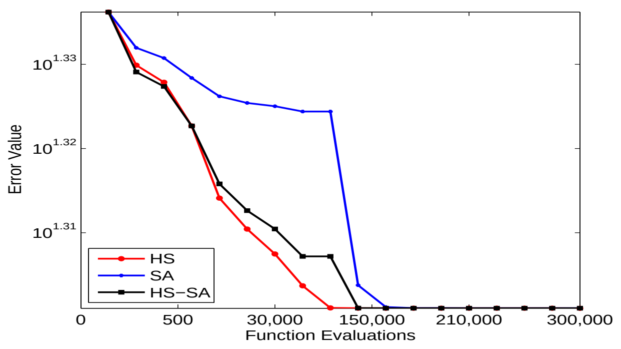
(b) Function 2



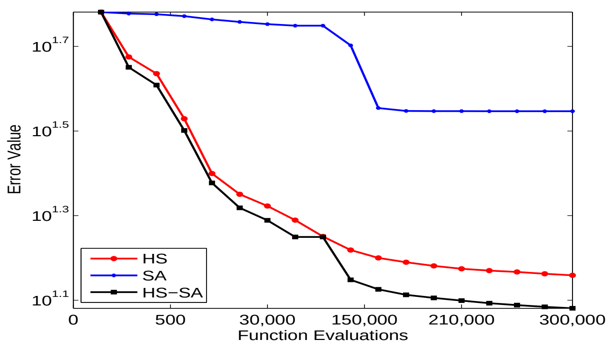
(c) Function 3



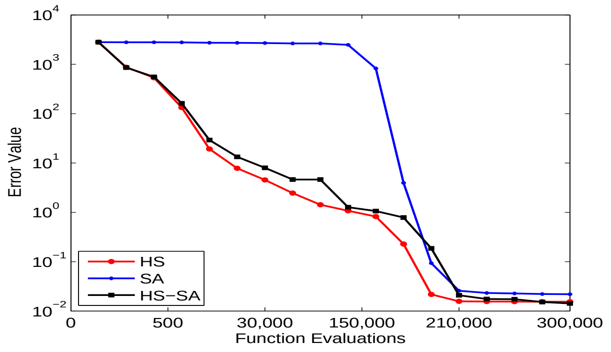
(d) Function 4



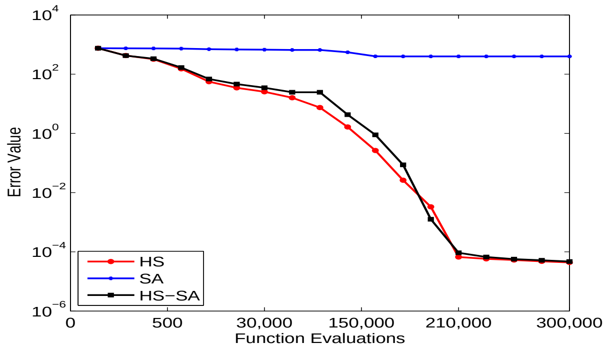
(e) Function 5



(f) Function 6

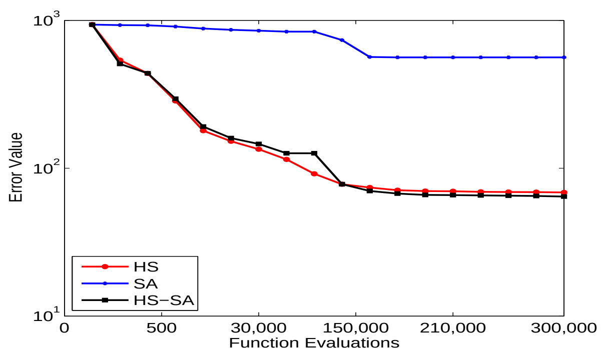


(g) Function 7

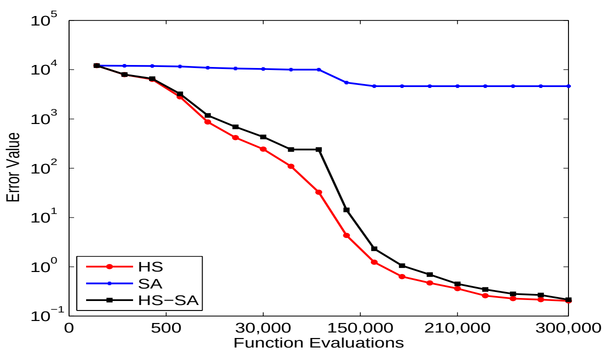


(h) Function 8

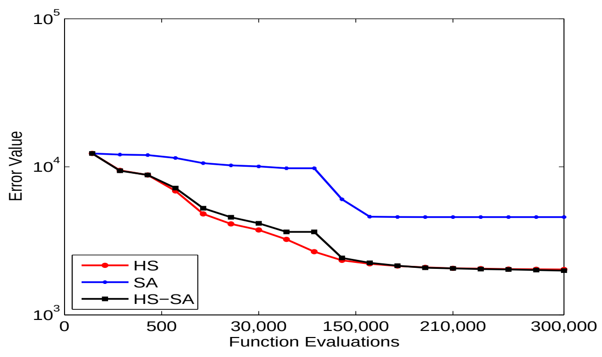
Figure 2.1: Convergence graphs of function number 1 through 8.



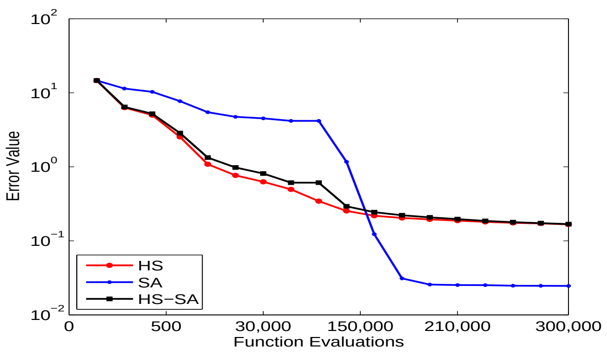
(a) Function 9



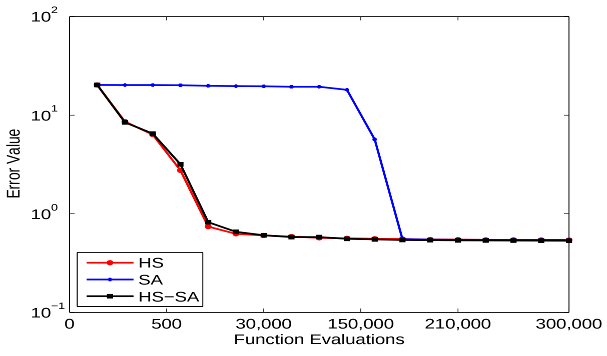
(b) Function 10



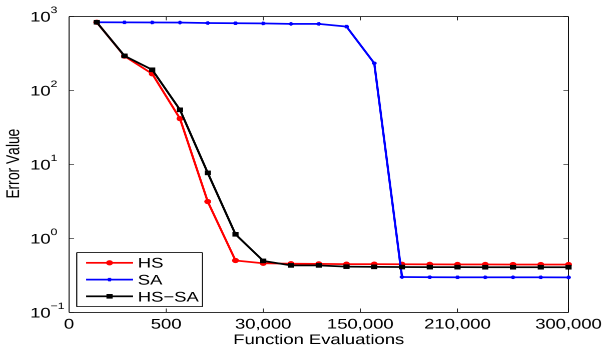
(c) Function 11



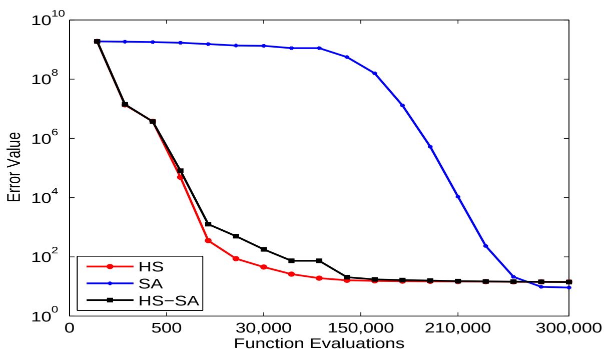
(d) Function 12



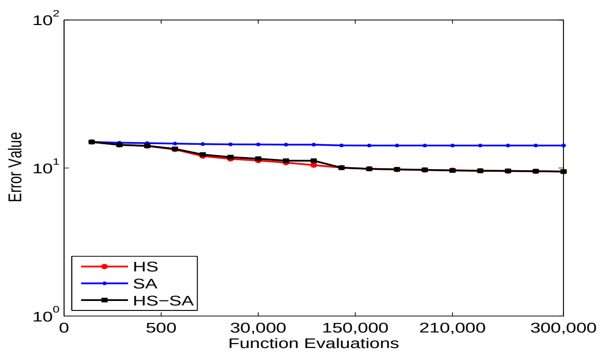
(e) Function 13



(f) Function 14

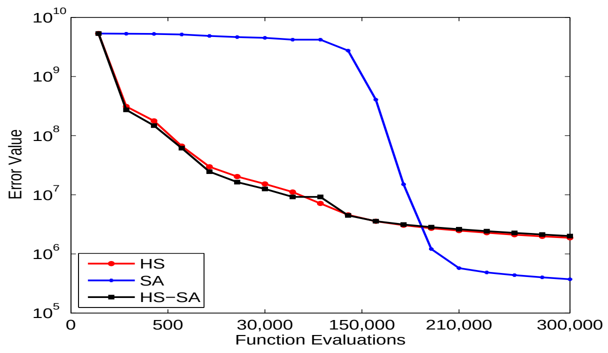


(g) Function 15

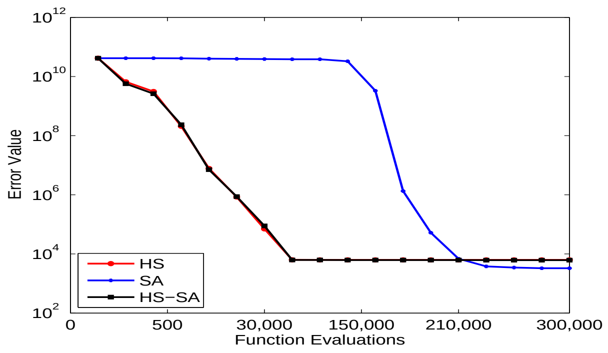


(h) Function 16

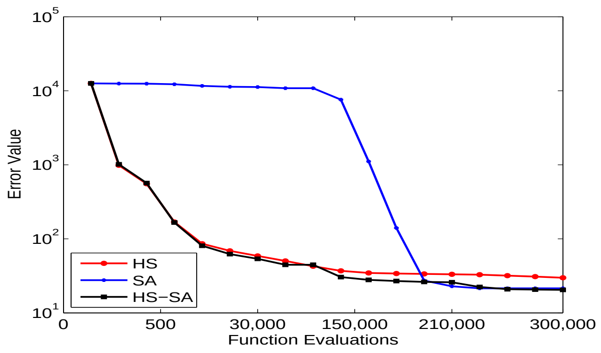
Figure 2.2: Convergence graphs of function number 9 through 16.



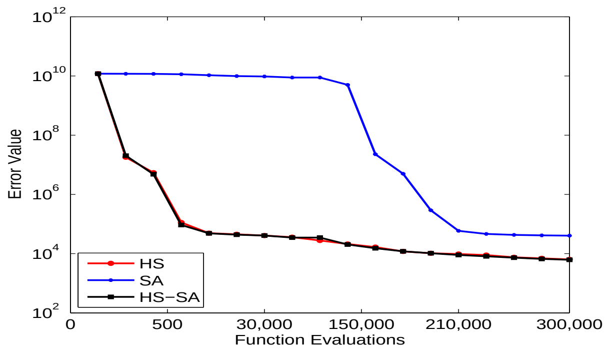
(a) Function 17



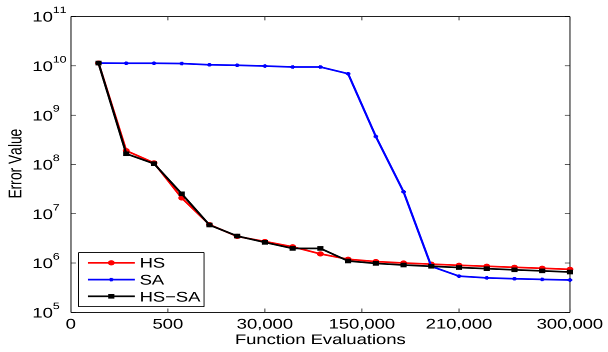
(b) Function 18



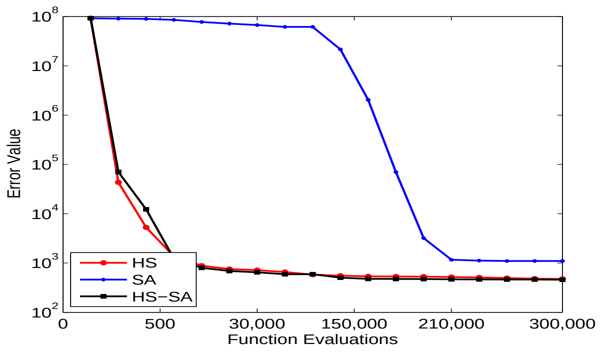
(c) Function 19



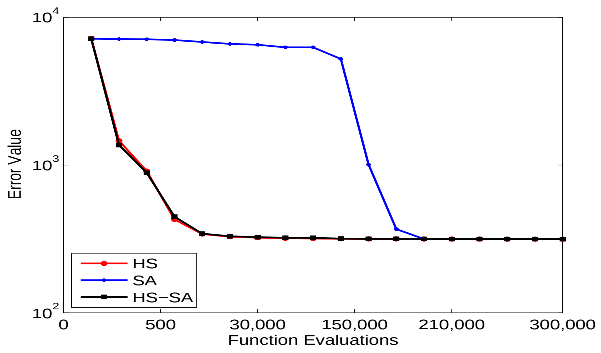
(d) Function 20



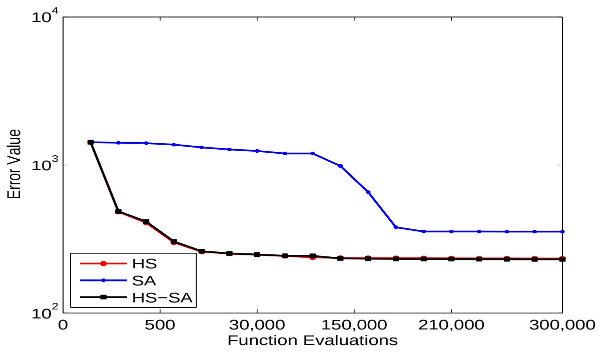
(e) Function 21



(f) Function 22

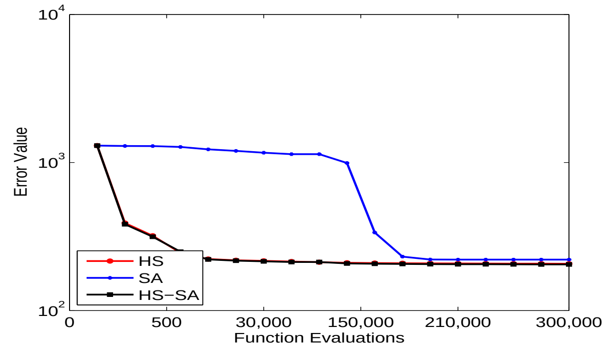


(g) Function 23

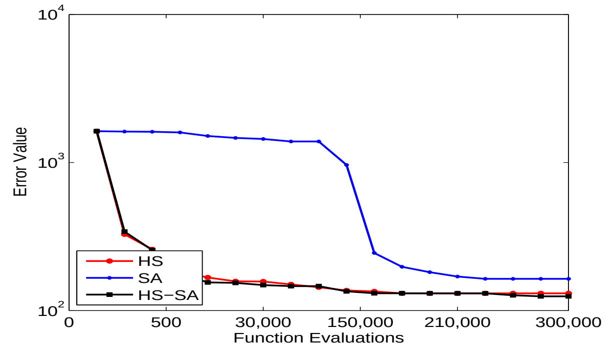


(h) Function 24

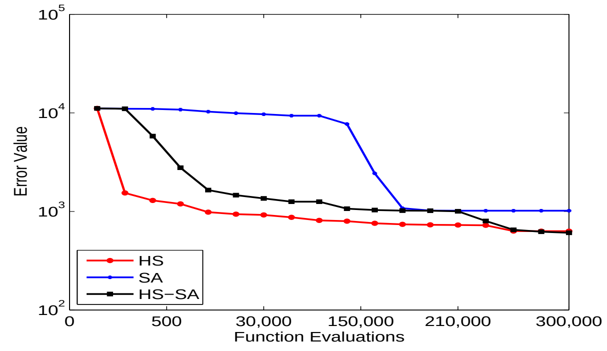
Figure 2.3: Convergence graphs of function number 17 through 24.



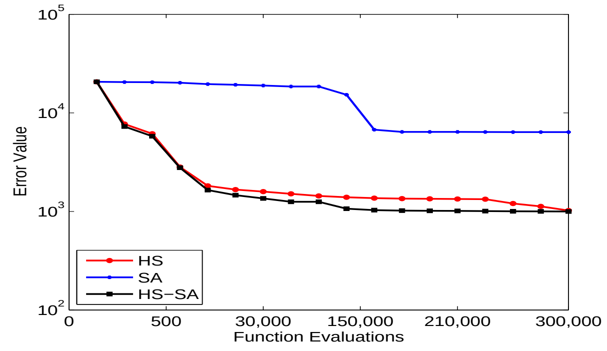
(a) Function 25



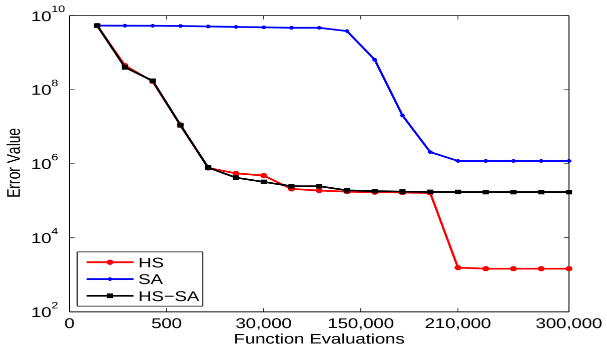
(b) Function 26



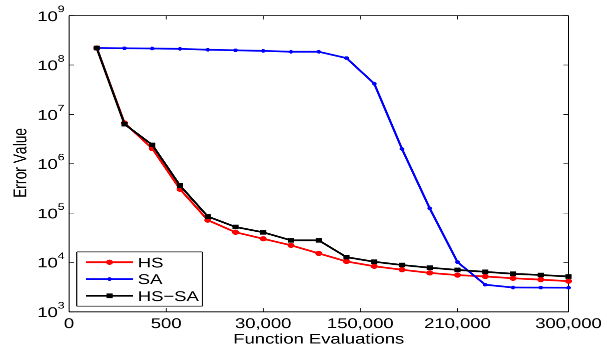
(c) Function 27



(d) Function 28



(e) Function 29



(f) Function 30

Figure 2.4: Convergence graphs of function number 25 through 30.

Table 2.5: Wilcoxon rank test results between the HS-SA and the competing algorithms.

FUNCTION NO.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HS-SA/HS	=	=	-	-	=	+	+	-	+	+	+	-	+	=	+
HS-SA/SA	-	-	+	-	=	+	+	+	+	+	+	-	+	-	-
FUNCTION NO.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
HS-SA/HS	+	-	+	+	+	+	+	=	+	+	+	+	+	-	-
HS-SA/SA	+	-	-	+	+	-	+	=	+	+	+	+	+	+	-

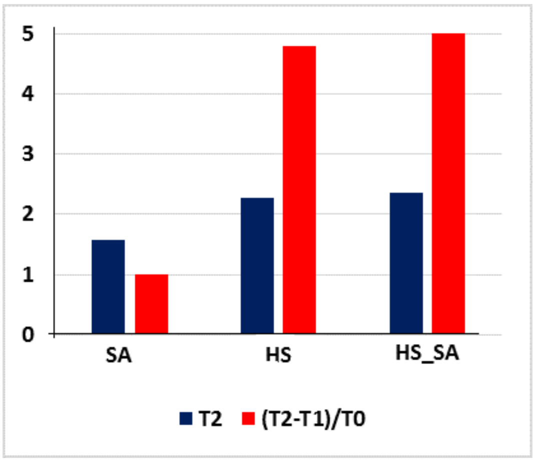


Figure 2.5: Time complexity of Algorithms in seconds for 30 Dimensional problem.

Chapter 3

A Two Phase Harmony Search Algorithm for Continuous Optimization

3.1 Introduction

The efficiency of a meta heuristic algorithms depend on the extent of balance between diversification and intensification during the course of the search. An ideal meta heuristic algorithm must have efficient exploration in the beginning and enhanced exploitation towards the end. In this Chapter a Two Phase Harmony Search (TPHS) algorithm is proposed that attempts to strikes a balance between exploration and exploitation by concentrating on diversification in the first phase using catastrophic mutation and then switches to intensification using local search in the second phase. The performance of TPHS is analyzed and compared with four state-of-the-art HS variants on all the 30 IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed TPHS algorithm in terms of accuracy particularly on multimodal functions when compared with other state-of-the-art HS variants, further comparison with state-of-the-art evolutionary algorithms reveal excellent performance of TPHS on composition functions. Composition functions are combined, rotated, shifted, and biased version of other unimodal and multi-modal test functions and mimic the difficulties of real search spaces by providing a massive number of local optima and different shapes for different regions of the search space.

The organization of this chapter is as follows. Section 3.2 provides a detailed description about the proposed TPHS algorithm, Section 3.3 provides detail about numerical experimentation, analysis of results and comparison with other state-of-the-art metaheuristic algorithms and finally the chapter concludes with Section 3.4.

3.2 Proposed Two Phase Harmony Search (TPHS) Algorithm for Continuous Optimization

The observation that the standard HS algorithm shows little or no progress particularly towards the end of execution, in most of the problems led us to the development of Two Phase harmony search (TPHS) algorithm. Since towards the end of program execution all the harmonies in HM are already of good quality, thus the probability of improving them using recombination is very less resulting in slow convergence of standard HS towards its end. In order to remove this limitation of standard

HS we propose TPHS algorithm in this chapter. The TPHS algorithm consists of two phases: Phase I concentrates on diversification by utilizing the concept of Catastrophic mutation from GA and Phase II shifts focus to intensification by using local search. The pseudo code of TPHS is shown as Algorithm 5 and from the pseudo code it is evident that during the initial $(3/4)^{th}$ of execution time the algorithm executes Phase I and during the last quarter of program execution the algorithm executes Phase II. Even though the break point between Phase I and Phase II has been arbitrary chosen as $(3/4)^{th}$ of execution time, however it can be adjusted depending on the problem in hand.

As pointed out in (Taherinejad, 2009; Wang and Huang, 2010) the value of the parameter PAR and BW should be decreased with time to prevent overshoot and oscillations. This would help the algorithm to diversify the search space of the solution vectors and prevent the solution from getting trapped in local optimal. Thus it seems reasonable that decreasing the value of the parameter PAR and BW with iterations could fine tune the final solutions. Therefore PAR and BW are linearly decreased (as per Equation (3.1) & (3.2) respectively) in step number 4 and 5 of Algorithm (5).

$$PAR_{gn} = PAR_{max} + \frac{(PAR_{max} - PAR_{min})}{NI} \times (gn - 1) \tag{3.1}$$

where

PAR_{gn} is pitch adjusting rate for each generation,

PAR_{min} and PAR_{max} is the minimum and maximum pitch adjusting rate respectively,

NI is the maximum number of generations,

gn is generation number.

$$BW_{gn} = BW_{max} \times exp(c \times (gn - 1)) \quad where \quad c = \frac{Ln(\frac{BW_{min}}{BW_{max}})}{NI} \tag{3.2}$$

In Equation (3.2) BW_{gn} represents bandwidth for each generation, BW_{min} and BW_{max} are respectively the minimum and maximum bandwidth.

Taking the inspiration from (Jin and Li, 1997) the concept of catastrophic mutation from GA has been used to increase the exploration capabilities of the TPHS algorithm in Phase I. Catastrophic mutation allows the evolutionary algorithm to increase diversity in the population on the cost of decreasing convergence speed so as to escape the local optimal. The Phase I of TPHS is the standard HS algorithm with the exception that all the harmonies (except the elite) in HM are re initialized (Catastrophic mutation) , if the best harmony is not updated in L iterations. This results in exploration of new search areas and therefore helps the algorithm to escape local optimal. The Phase II of TPHS is the standard HS algorithm with the addition that in each iteration of the algorithm one of the harmonies is

randomly selected from HM for local search (exploitation) and in case the harmony produced after local search (referred as TEMP in Algorithm 5) is better than parent harmony, the parent harmony is replaced by this new harmony (TEMP). The local search is performed in step 30 of Algorithm (5) by slightly altering each component of the selected harmony within the bounds specified by parameter BW2. It must be noted that the new harmony produced after local search (i.e. TEMP) is compared with the parent harmony and not the worst one so as to maintain the diversity in the HM. Towards the end of execution all the harmonies in the HM are of good quality, therefore it seems reasonable to concentrate on different promising search areas (Harmonies) for exploitation rather than a single area around the best harmony. Thus in step number 28 of Algorithm (5) one of the harmonies from HM is randomly selected for exploitation. Whenever any dimension of the harmony moves out of bounds it is randomly generated within the bounds (in step no. 12, 15 and 31 of Algorithm (5)) rather than assigning minimum/maximum value so as to maintain diversity in HM.
In Algorithm (5) $PAR_{min}, PAR_{max}, BW_{min}, BW_{max}$ respectively denote minimum pitch adjustment rare, maximum pitch adjustment rare, minimum bandwidth and maximum bandwidth. Bandwidth 2 (BW2) controls the degree of exploitation of TEMP. The other parameters are the same as already defined for Algorithm (1). The key differences between standard HS and TPHS are:

1. In TPHS both PAR and BW are linearly decreased as per Equation (3.1) and (3.2) respectively whereas both PAR and BW remain constant in standard HS.

2. The Harmony Memory is re initialized except the elite (catastrophic mutation) if the best harmony is not updated in L generations.

3. Towards the end (last quarter in this chapter) of the execution the algorithm shifts focus to exploitation, by performing local search around the best harmonies/solutions stored in HM.

3.3 Numerical Experiments

In this section performance of the proposed TPHS algorithm is evaluated on IEEE CEC 2014 Benchmark functions (Liang et al., 2013). The IEEE CEC 2014 Benchmark suite has been explained in Section (2.4.1) of Chapter 2.

3.3.1 Comparison of TPHS with some state-of-the-art HS variants

The TPHS algorithm has been compared with Standard HS (Geem et al., 2001) and its three state-of-the-art variants: Improved Harmony Search (Mahdavi et al., 2007), Global best Harmony Search (Omran and Mahdavi, 2008) and Self-adaptive Global Best Harmony Search (Pan et al., 2010b). The

experimentation has been carried out on all the IEEE CEC 2014 Benchmark functions using 30 dimension. As per the instructions of test suite every problem is tested with 51 independent runs.
The parameter setting adopted for standard HS, GHS, IHS has been taken from (El-Abd, 2013) and is shown in Table 3.1. Obtaining an optimal parameter setting for a metaheuristic algorithm is a hyper optimization problem, however the parameter setting shown in Table 3.1 for TPHS was found out to be optimal for most if not all the problem instances. All the algorithms have been implemented in Dev C++ 5.0 and the experimentation has been carried out on a laptop with Windows 10 operating system, intel core i3 processor and 4GB of RAM.

Table 3.1: Parameter setting of algorithms.

Algorithm	HMS	HMCR	PAR	BW
HS (Geem et al., 2001)	5	.9	.3	.001
GHS (Omran and Mahdavi, 2008)	5	.9	$PAR_{min} = 0.01,\ PAR_{max} = 0.99$	—
IHS (Mahdavi et al., 2007)	5	.95	$PAR_{min} = 0.01,\ PAR_{max} = 0.99$	$BW_{min} = .00001, BW_{max} = \frac{UB-LB}{20}$
SAGHS (Pan et al., 2010b)	5	$HMCR_m = 0.98$	$PAR_m = 0.9$	$BW_{min} = .00005,\ BW_{max} = \frac{UB-LB}{10}$
TPHS	5	.9	$PAR_{min} = 0.3,\ PAR_{max} = 0.99$	$BW_{min} = .001,\ BW_{max} = \frac{UB-LB}{20},\ BW2 = 1$ and $L = .05 \times MaxFES$

3.3.1.1 Analysis of results

The results reported in this chapter are in the format as specified and required in IEEE CEC 2014 benchmark suite. Tables 3.2, 3.3 and 3.4 show the results on 30 dimensional problems, the best results are highlighted by bold font. The recorded results are the minimum, maximum, mean, median, and standard deviation of the error value obtained as specified in IEEE 2014 Benchmark suite. The error is the absolute value of difference between obtained objective function value by the algorithm and the actual function value.
For unimodal functions HS, IHS and SAGHS reports the best results in one instances each. SAGHS, HS obtained the best mean results in two and one instances respectively. In case of simple multimodal functions HS, IHS, SAGHS and TPHS produced the best mean results in 2, 5, 4 and 4 instances respectively and produced the best results in 3, 6, 4 and 4 instances respectively. For Hybrid multimodal functions TPHS produced the best mean results in all the 6 instances and obtained the best results in 5 functions. For composition multimodal functions TPHS obtained the best mean results and the best re-

sults in 6 instances followed by SAGHS obtaining best mean results in 2 instances. Thus even though the performance of the TPHS algorithm is not very satisfactory on unimodal functions, however it has significantly outperformed its variants on multimodal functions particularly hybrid and composition functions which are considered to be harder than simple multimodal functions.

3.3.1.2 Convergence Behavior

The convergence graphs of all the benchmark functions are plotted in Figures 3.1, 3.2, 3.3 and 3.4 to study the convergence behavior of algorithms. The horizontal line represents the number of function evaluations and the vertical line represents the mean of absolute error of 51 runs in logarithmic scale. As is evident from most of the convergence graphs the TPHS algorithm exhibits slow convergence speed in the beginning of execution, the reason being frequent re initialization of Harmony Memory, however it helps TPHS to explore vast search space and hence increase exploration initially. Due to the exploitation of the harmonies by performing random search around the best harmony (Local search) in the second phase most of the graphs (particularly evident for Functions 6, 17, 18, 19, 20, 21, 24, 25, 26, 27 and 28) depict that TPHS makes steep progress even towards the end of execution. Thus dividing the TPHS into two phases enhances exploration in the beginning and exploitation towards the end. This fact can be verified from convergence graph of function 18 TPHS makes slow progress in the first phase ($\frac{3}{4}^{th}$ of execution time) compared to other algorithms, however it makes steep progress in the second phase ($\frac{1}{4}^{th}$ of execution time) and overtakes all other algorithms.
TPHS consists of two operators catastrophic mutation applied in the beginning of execution and local search applied towards the end of execution. In order to study the effect of these two operators individually we have selected function 24 with dimension 10. The mean error of 51 runs is respectively 123.6, 118.2 and 111.8 when only catastrophic mutation is applied, only local search is applied and when both operators are applied. The results obtained where found statistically significant using Wilcoxon rank test at 5% level of significance. Thus it can be concluded that catastrophic mutation has increased exploration in the beginning whereas local search operator has increased exploitation towards the end.

3.3.1.3 Algorithm Complexity

The complexity of algorithms is computed as per the requirements laid down in IEEE CEC 2014 Benchmark suite and has been explained in Section 2.4.3.1. The time complexity for 30 dimensional problem is shown as Figure 3.5 and it can be observed that the complexity of HS, IHS, GHS and TPHS is almost same however the time complexity of SAGHS is slightly higher than others.

3.3.2 Comparison of TPHS with some state-of-the-art meta heuristic algorithms

In this section the performance of TPHS is compared with some state-of-the-art meta heuristic algorithms for continuous optimization like Evolution Strategy with covariance matrix adaptation (CMA-ES) (Hansen and Ostermeier, 2001), Comprehensive Learning PSO (CL-PSO) (Liang et al., 2006), Adaptive particle swarm optimization (APSO) (Zhan et al., 2009), Dynamic Neighborhood Learning PSO (DNL-PSO) (Nasir et al., 2012), Heterogeneous Comprehensive Learning PSO (HCL-PSO) (Lynn and Suganthan, 2015), Social Learning PSO (SL-PSO) (Cheng and Jin, 2015) , Self Regulating PSO (SR-PSO) (Tanweer et al., 2015), Social Spider Optimization Algorithm (SSO) (Cuevas et al., 2013), Differential Evolution (DERAND1BIN) (Guo et al., 2015), Differential Evolution with successful parent selecting framework (DERAND1BIN-SPS) (Guo et al., 2015) and Dynamic multi-swarm particle swarm optimizer with harmony search (DMS-PSO-HS) (Zhao et al., 2011) on 30 dimensional IEEE CEC 2014 benchmark problems as per the guidelines laid down in the benchmark suite, the results are reported in Tables 3.5, 3.6, 3.7, 3.8. The results of competing algorithm are highlighted by bold if it performs better than TPHS. The parameter setting adopted for all competing algorithms is same as given in the respective reference.

CL-PSO outperformed TPHS on all the three unimodal functions. In case of simple multimodal functions, TPHS outperformed CL-PSO on three instances whereas latter outperformed former in the remaining ten instances. TPHS outperformed CL-PSO on four instances of hybrid functions whereas latter outperformed former in the remaining two instances. In case of composition function TPHS outperformed CL-PSO on five instances and the results are vice versa in the remaining three instances. Thus in unimodal and simple multimodal functions CL-PSO is the winner whereas in case of hybrid and composition functions proposed TPHS is the winner.

TPHS outperformed APSO on all the three instances of unimodal, on all the thirteen instances of simple multimodal and on all the six instances of hybrid functions. In case of composition functions TPHS outperformed APSO on two instances whereas latter outperformed former in the remaining six instances.

DNL-PSO outperformed TPHS on all the three instances of unimodal functions. DNL-PSO outperformed TPHS on seven instances of simple multimodal functions and the result is vice versa in the remaining remaining six instances. In four instances of hybrid functions TPHS is the winner whereas in the remaining two instances DNL-PSO is the winner. TPHS outperformed DNL-PSO on four instances of hybrid functions whereas latter outperformed former in the remaining two instances. In composition functions TPHS outperformed DNL-PSO on five instances whereas latter outperformed former in the remaining three instances. Thus in case of unimodal and simple multimodal functions DNL-PSO is the winner whereas in case of hybrid and composition functions proposed TPHS is the winner.

HCL-PSO outperformed TPHS on all the three instances of unimodal functions. HCL-PSO outperformed TPHS on eleven instances of simple multimodal functions and the result is vice versa in the remaining two instances. In two instances of hybrid functions TPHS is the winner whereas in the remaining four instances HCL-PSO is the winner. In composition functions TPHS outperformed HCL-PSO on six instances whereas latter outperformed former in the remaining two instances.

SL-PSO outperformed TPHS on two instances of unimodal functions and on the remaining one instance TPHS is the winner. SL-PSO outperformed TPHS on eight instances of simple multimodal functions and the result is vice versa in the remaining five instances. Both TPHS and SL-PSO has outperformed each other in three instances of hybrid functions. In composition functions TPHS outperformed SL-PSO on six instances whereas latter outperformed former on the remaining two instances.

SR-PSO outperformed TPHS on all the three instances of unimodal functions. SR-PSO outperformed TPHS on eight instances of simple multimodal functions and the result is vice versa in the remaining remaining five instances. In five instances of hybrid functions SR-PSO is the winner whereas in the remaining one instances TPHS is the winner. In composition functions TPHS outperformed SR-PSO on seven instances whereas latter outperformed former in the remaining one instances.

CMA-ES outperformed TPHS on two instances of unimodal functions whereas latter outperformed former in the remaining one instance. CMA-ES outperformed TPHS on six instances of simple multimodal functions and the result is vice versa in the remaining remaining seven instances. In three instances of hybrid functions TPHS is the winner whereas in the remaining three instances CMA-ES is the winner. In composition functions TPHS outperformed CMA-ES on five instances whereas latter outperformed former in the remaining three instances.

The proposed TPHS algorithm outperformed SSO on all the thirty instances.

The standard Differential Evolution (DERAND1BIN) outperformed TPHS on two instances of unimodal functions and on the remaining one instance TPHS is the winner. TPHS outperformed DERAND1BIN on ten instances of simple multimodal functions whereas latter is the winner in the remaining three instances. TPHS outperformed DERAND1BIN on five instances of hybrid functions whereas latter is the winner in just one instance. In composition functions TPHS outperformed DERAND1BIN on six instances whereas latter outperformed former on the remaining two instances.

DERAND1BIN-SPS outperformed TPHS on two instances of unimodal functions and on the remaining one instance TPHS is the winner. TPHS outperformed DERAND1BIN-SPS on eight instances of simple multimodal functions whereas latter is the winner in the remaining five instances. DERAND1BIN-SPS outperformed TPHS on five instances of hybrid functions whereas latter is the winner in just one instance. In composition functions TPHS outperformed DERAND1BIN-SPS on four instances whereas latter outperformed former on the remaining four instances.

DMS-PSO-HS outperformed TPHS on two instances of unimodal functions and on the remaining one instance TPHS is the winner. DMS-PSO-HS outperformed TPHS on ten instances of simple multimodal functions whereas latter is the winner in the remaining three instances. TPHS outperformed DMS-PSO-HS on all the instances of hybrid and composition functions.

In this section TPHS was compared with eleven state-of-the-art metaheuristic algorithms for continuous optimization. TPHS showed excellent performance on composition functions, out of the eleven competing algorithms only one algorithm namely APSO managed to outperform TPHS on composition functions. It is worth to mention even though the performance of APSO is slightly better than TPHS on composition functions, however TPHS outperformed APSO on all the instances of unimodal, simple multimodal and hybrid functions. Note that Composition functions are combined, rotated, shifted, and biased version of other unimodal and multi-modal test functions and mimic the difficulties of search spaces offerer by real life problems by providing a massive number of local optima and different shapes for different regions of the search space, an algorithm must exhibit a proper balance between exploration and exploitation to tackle such problems.

3.3.3 Wilcoxon rank test analysis

The paired Wilcoxons rank-sum test is conducted at the 5% significance level to judge if the difference in performance between the TPHS and the competing algorithm is statistically significant, the results are reported in Table 3.9. The cases are respectively marked as '+', '-' and '=' when the performance of TPHS is significantly better than, worse than, or similar to the competing algorithm. Table 3.9 reveals that in most of the cases the difference in performance is statistically significant and there are only a instances where the difference is not significant.

3.4 Conclusion

This chapter introduced a new variant of Harmony Search algorithm for continuous optimization problems. The proposed algorithm called TPHS exhibited the desired behavior of exploring the search space at the beginning and then exploiting good solutions towards the end. This was achieved by dividing TPHS into two phases. The first phase concentrates mainly on exploration using catastrophic mutation and in the second phase the focus shifts to exploitation using local search. The search area was gradually decreased by decreasing the BW and PAR with time.

TPHS algorithm has been compared with fifteen state-of-the-art metaheuristic algorithms. The bases of performance is the set of 30 benchmark functions proposed in IEEE CEC 2014. All the criterion laid down in the benchmark suite are adopted to produce the required analysis metrics. Based on the numerical results it is established even though performance of TPHS is not very satisfactory on unimodal functions however it outperforms HS variants on multimodal functions. The performance of TPHS is particularly outstanding on composition functions when compared to state-of-the-art metaheurisctic algorithms for continuous optimization. Note that Composition functions are combined, rotated, shifted, and biased version of other unimodal and multi-modal test functions and mimic the difficulties of search spaces offerer by real life problems by providing a massive number of local optima and different shapes for different regions of the search space, an algorithm must exhibit a proper balance between exploration and exploitation to approximate the global optimal of such problems.

Algorithm 5 TWO PHASE HARMONY SEARCH ALGORITHM (TPHS)

```
1: Define $HMCR, PAR_{min}, PAR_{max}, BW_{min}, BW_{max}, HMS, BW2$.
2: Initialize Harmony Memory (HM).
3: while NOF $\leq$ Max_FES do
4:   Adjust PAR as per Equation (3.1).
5:   Adjust BW as per Equation (1.17).
6:   Find Worst and Best harmonies in HM.
7:   for $i = 1$ to $D$ do
8:     if (rand $\leq$ HMCR) then
9:       $H_i = HM_i^j$ where j=rand_int(1,HMS)
10:      if (rand $\leq$ PAR) then
11:        $H_i = H_i \pm rand \times BW$
12:        Generate $H_i$ randomly within the allowed bounds if it moves out side bound after Pitch
           adjustment.
13:      end if
14:    else
15:      Generate $H_i$ randomly within the allowed bounds.
16:    end if
17:  end for
18:  Evaluate harmony H
19:  NOF++ /* NOF is a counter representing number of function evaluations performed*/
20:  if (H is better than Worst Harmony in HM) then
21:    Update HM by replacing Worst harmony by H.
22:  end if
23:  if NOF $\leq .75\times$ Max_FES then
24:    {Phase I for exploration}
25:    If the BEST harmony does not change in L iterations re initialize all the harmonies in the
       HM except the elite. /* Catastrophic mutation for diversification*/
26:  else
27:    {Phase II for exploitation}
28:    TEMP=$HM^K$ where K=rand_int(1,HMS) /* Select random harmony from HM for intensifi-
       cation */
29:    for $i = 1$ to $D$ do
30:      $TEMP_i = TEMP_i \pm BW2 \times rand$ /* Local search for intensification */
31:      Generate $TEMP_i$ randomly within the allowed bounds if it moves out side bound after
         adjustment.
32:    end for
33:    Evaluate harmony TEMP
34:    NOF++
35:    if TEMP is better than $HM^K$ then
36:      $HM^K = TEMP$ /*Replace parent harmony by new one*/
37:    end if
38:  end if
39: end while
40: print Best Harmony obtained by the algorithm as solution.
```

Table 3.2: Error values obtained by HS, GHS, IHS, SAGHS and TPHS on Function number 1 through 10.

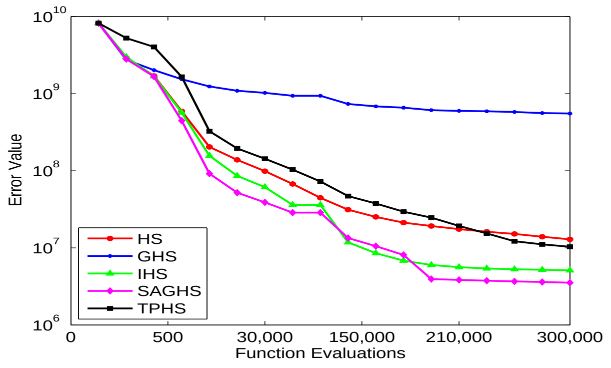
Function	Algorithm	Mean	Median	Best	Worst	Std Dev
1	HS	1.2898E+07	1.1256E+07	1.2757E+06	3.7050E+07	8.5380E+06
	GHS	5.5304E+08	5.5251E+08	2.2774E+08	8.9384E+08	1.3331E+08
	IHS	5.1117E+06	3.8814E+06	7.5670E+05	2.2954E+07	4.1204E+06
	SAGHS	**3.5401E+06**	**3.1267E+06**	**6.7670E+05**	**9.0978E+06**	**1.7926E+06**
	TPHS	1.0342E+07	9.5660E+06	4.3612E+06	3.0861E+07	4.9993E+06
2	HS	**1.0748E+04**	**5.8189E+03**	1.4352E+02	**3.4211E+04**	**1.1490E+04**
	GHS	5.1074E+10	5.0526E+10	4.1518E+10	6.2548E+10	4.6701E+09
	IHS	1.3405E+04	1.0105E+04	**1.2369E+00**	3.6883E+04	1.2150E+04
	SAGHS	1.3070E+04	9.5449E+03	5.5077E+00	5.1221E+04	1.2816E+04
	TPHS	1.0658E+06	1.0937E+06	6.0514E+05	1.4416E+06	1.8124E+05
3	HS	4.6459E+03	3.9072E+03	**8.6021E+00**	2.1796E+04	4.1211E+03
	GHS	7.7427E+04	7.3782E+04	4.7596E+04	1.5279E+05	1.8602E+04
	IHS	1.0165E+04	6.5060E+03	5.0759E+02	5.5072E+04	1.1198E+04
	SAGHS	**2.2834E+03**	**1.7112E+03**	2.3510E+01	7.9129E+03	1.9046E+03
	TPHS	2.4343E+03	2.1490E+03	2.8063E+02	**6.9590E+03**	1.5892E+03
4	HS	1.1612E+02	1.2484E+02	6.3859E+01	1.5181E+02	3.0220E+01
	GHS	6.8740E+03	6.9255E+03	3.9398E+03	9.7943E+03	1.1909E+03
	IHS	6.9072E+01	7.5569E+01	**1.0632E+00**	1.4873E+02	3.7522E+01
	SAGHS	**4.6127E+01**	**2.8905E+01**	5.8500E+00	**9.8818E+01**	2.6781E+01
	TPHS	1.1659E+02	1.2044E+02	6.8637E+01	1.6753E+02	2.4942E+01
5	HS	2.0000E+01	2.0000E+01	2.0000E+01	2.0000E+01	3.8090E-05
	GHS	2.0936E+01	2.0946E+01	2.0789E+01	2.1046E+01	5.6890E-02
	IHS	2.0000E+01	2.0000E+01	2.0000E+01	2.0000E+01	1.1113E-05
	SAGHS	2.0000E+01	2.0000E+01	2.0000E+01	2.0000E+01	1.2778E-05
	TPHS	2.0931E+01	2.0939E+01	2.0709E+01	2.1033E+01	6.3582E-02
6	HS	1.4421E+01	1.4553E+01	9.8893E+00	1.9776E+01	2.0702E+00
	GHS	3.8602E+01	3.9094E+01	3.3271E+01	4.1335E+01	1.5711E+00
	IHS	1.3630E+01	1.3630E+01	7.5007E+00	1.8956E+01	2.5875E+00
	SAGHS	1.3963E+01	1.3942E+01	7.3736E+00	1.9469E+01	2.3539E+00
	TPHS	**2.8907E+00**	**2.9377E+00**	**1.7516E+00**	**4.9279E+00**	**8.1642E-01**
7	HS	1.5451E-02	7.8050E-03	1.5700E-04	8.3465E-02	1.8883E-02
	GHS	3.1877E+02	3.0622E+02	2.3989E+02	4.3520E+02	5.1026E+01
	IHS	**8.5438E-03**	**7.3960E-03**	**0.0000E+00**	**3.4336E-02**	**9.1937E-03**
	SAGHS	3.3846E-02	1.9720E-02	0.0000E+00	1.6434E-01	3.4615E-02
	TPHS	9.0903E-01	9.1778E-01	7.4936E-01	9.7793E-01	5.1068E-02
8	HS	4.3647E-05	4.4000E-05	2.6000E-05	5.6000E-05	6.6622E-06
	GHS	2.9940E+02	3.0089E+02	2.3775E+02	3.4806E+02	2.4578E+01
	IHS	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	SAGHS	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	TPHS	1.9839E+00	1.7699E+00	3.5319E-01	5.5341E+00	1.2307E+00
9	HS	**6.8566E+01**	**6.6662E+01**	**4.2633E+01**	**9.7506E+01**	**1.4704E+01**
	GHS	3.3998E+02	3.3731E+02	3.0224E+02	4.0221E+02	2.1684E+01
	IHS	7.7371E+01	7.4674E+01	4.3778E+01	1.1243E+02	1.7341E+01
	SAGHS	9.6145E+01	9.4521E+01	5.9697E+01	1.5585E+02	2.1113E+01
	TPHS	1.4776E+02	1.2890E+02	8.9750E+01	2.0689E+02	3.4546E+01
10	HS	2.0355E-01	2.0977E-01	**1.0515E-01**	**3.3868E-01**	5.5092E-02
	GHS	6.3799E+03	6.3880E+03	5.7062E+03	7.0003E+03	3.2841E+02
	IHS	3.2345E-01	2.5943E-01	1.5125E-01	1.2848E+00	2.3837E-01
	SAGHS	**1.9548E-01**	**1.8737E-01**	1.2492E-01	3.5393E-01	**5.2196E-02**
	TPHS	1.9780E+01	2.0051E+01	9.9087E+00	2.7334E+01	4.2670E+00

Table 3.3: Error value obtained by HS, GHS, IHS, SAGHS and TPHS on Function number 11 through 20.

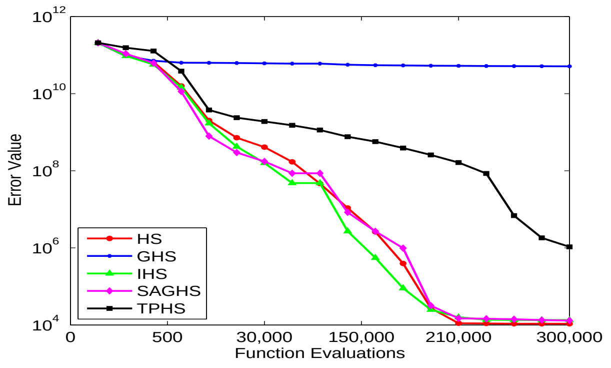
Function	Algorithm	Mean	Median	Best	Worst	Std Dev
11	HS	2.0271E+03	2.0486E+03	1.0901E+03	**2.9462E+03**	4.0440E+02
	GHS	7.0700E+03	7.1401E+03	6.0754E+03	7.6881E+03	3.4849E+02
	IHS	**2.0043E+03**	**2.0085E+03**	8.3726E+02	3.3345E+03	5.0644E+02
	SAGHS	2.3526E+03	2.4471E+03	1.2628E+03	3.3160E+03	5.0402E+02
	TPHS	4.1312E+03	4.1517E+03	**5.7415E+02**	6.7373E+03	1.6932E+03
12	HS	1.6681E-01	1.5897E-01	6.1356E-02	2.6858E-01	5.0019E-02
	GHS	2.3749E+00	2.4272E+00	1.3879E+00	3.0444E+00	3.2126E-01
	IHS	**8.2523E-02**	7.9840E-02	**2.5872E-02**	**1.8604E-01**	**3.7921E-02**
	SAGHS	1.2286E-01	**1.1364E-01**	6.0261E-02	2.4778E-01	4.1331E-02
	TPHS	2.4943E+00	2.4910E+00	1.8035E+00	3.0691E+00	2.6679E-01
13	HS	5.4085E-01	5.4299E-01	3.0465E-01	8.7141E-01	1.1932E-01
	GHS	5.8505E+00	5.8931E+00	5.0164E+00	6.8080E+00	4.1574E-01
	IHS	5.2125E-01	5.0641E-01	2.5694E-01	8.0457E-01	1.2618E-01
	SAGHS	5.1663E-01	5.0089E-01	3.0362E-01	8.0925E-01	1.1867E-01
	TPHS	**3.7631E-01**	**3.5977E-01**	**2.1373E-01**	**5.6539E-01**	**9.1385E-02**
14	HS	4.4240E-01	3.5203E-01	2.4030E-01	1.0571E+00	2.2474E-01
	GHS	1.4829E+02	1.4912E+02	1.2060E+02	1.7370E+02	1.4049E+01
	IHS	4.4064E-01	3.4738E-01	2.0361E-01	9.8340E-01	2.3195E-01
	SAGHS	4.3648E-01	3.1858E-01	**1.3886E-01**	1.0572E+00	2.5262E-01
	TPHS	**3.1817E-01**	**3.1202E-01**	1.6756E-01	**4.2938E-01**	**5.5921E-02**
15	HS	1.4252E+01	1.3387E+01	5.5428E+00	2.7132E+01	5.5849E+00
	GHS	5.3442E+04	3.7607E+04	1.2241E+04	4.9159E+05	7.0130E+04
	IHS	1.1973E+01	1.0533E+01	**3.5509E+00**	3.1311E+01	5.0716E+00
	SAGHS	**6.7818E+00**	**6.2194E+00**	3.7810E+00	**1.3669E+01**	2.0976E+00
	TPHS	1.7448E+01	1.7476E+01	1.4544E+01	1.9963E+01	1.0298E+00
16	HS	9.4670E+00	9.5518E+00	7.8243E+00	1.1028E+01	6.5864E-01
	GHS	1.2894E+01	1.2950E+01	1.2030E+01	1.3255E+01	2.7413E-01
	IHS	9.7692E+00	9.6238E+00	7.8164E+00	1.1303E+01	6.9112E-01
	SAGHS	1.0035E+01	1.0063E+01	8.3375E+00	1.1544E+01	7.2495E-01
	TPHS	**8.4200E+00**	**8.3470E+00**	**7.2800E+00**	**9.5521E+00**	4.8655E-01
17	HS	1.8725E+06	1.1507E+06	6.5245E+04	7.8273E+06	1.6072E+06
	GHS	1.7787E+07	1.7106E+07	4.1156E+06	3.7052E+07	6.2113E+06
	IHS	3.2222E+05	2.7595E+05	4.9077E+04	1.3522E+06	2.2660E+05
	SAGHS	5.9517E+05	5.7166E+05	6.3973E+04	1.8089E+06	3.7610E+05
	TPHS	**3.1185E+05**	**2.7201E+05**	**8.5087E+03**	**1.0751E+06**	2.4092E+05
18	HS	6.2863E+03	2.5298E+03	7.2457E+01	2.8773E+04	7.8468E+03
	GHS	8.6656E+08	8.8799E+08	1.1087E+08	1.5049E+09	3.0575E+08
	IHS	4.8757E+03	2.5367E+03	**5.0858E+01**	1.8914E+04	4.8932E+03
	SAGHS	5.1710E+03	2.5884E+03	6.9119E+01	3.1256E+04	5.9942E+03
	TPHS	**2.9688E+03**	**1.9882E+03**	3.8090E+02	**1.4099E+04**	**2.8869E+03**
19	HS	2.9814E+01	9.3142E+00	5.9647E+00	1.1896E+02	3.5770E+01
	GHS	2.2127E+02	2.2418E+02	1.3347E+02	2.7965E+02	3.0520E+01
	IHS	1.3346E+01	1.0597E+01	**3.6172E+00**	8.7776E+01	1.4399E+01
	SAGHS	2.0974E+01	1.3931E+01	5.5005E+00	8.3078E+01	2.1603E+01
	TPHS	**6.7812E+00**	**6.7391E+00**	4.8215E+00	**9.4500E+00**	**9.8160E-01**
20	HS	6.3997E+03	5.6052E+03	1.7248E+02	2.4392E+04	4.9723E+03
	GHS	3.8663E+04	3.2773E+04	8.3942E+03	1.0930E+05	2.3295E+04
	IHS	6.6755E+03	5.6384E+03	4.1518E+02	2.6303E+04	5.7149E+03
	SAGHS	1.2179E+03	**6.1255E+02**	**6.5699E+01**	6.5438E+03	1.4115E+03
	TPHS	**9.5134E+02**	6.5674E+02	2.6539E+02	**3.2450E+03**	**7.0946E+02**

Table 3.4: Error value obtained by HS, GHS, IHS, SAGHS and TPHS on Function number 21 through 30.

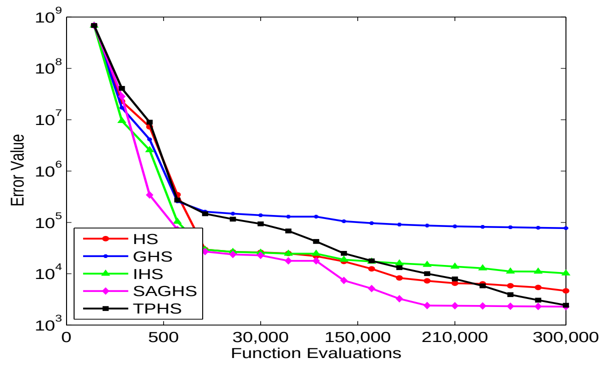
Function	Algorithm	Mean	Median	Best	Worst	Std Dev
21	HS	7.4351E+05	6.5937E+05	6.9129E+04	2.2859E+06	5.3016E+05
	GHS	4.8848E+06	4.2238E+06	7.8191E+05	1.0705E+07	2.3543E+06
	IHS	1.8529E+05	1.2740E+05	1.0835E+04	7.7702E+05	1.6187E+05
	SAGHS	3.2397E+05	2.3415E+05	2.8262E+04	1.3063E+06	2.4493E+05
	TPHS	**1.5138E+05**	**1.0916E+05**	**1.0296E+04**	**5.1142E+05**	**1.2235E+05**
22	HS	4.7430E+02	4.8986E+02	4.0511E+01	7.6838E+02	1.6864E+02
	GHS	1.2795E+03	1.3099E+03	8.0160E+02	1.5477E+03	1.8188E+02
	IHS	4.9181E+02	4.8962E+02	2.9915E+01	8.6093E+02	1.9143E+02
	SAGHS	4.7914E+02	4.9418E+02	2.7009E+01	8.5830E+02	2.0031E+02
	TPHS	**1.5578E+02**	**1.4972E+02**	**2.3576E+01**	**5.0357E+02**	**1.0842E+02**
23	HS	3.1542E+02	3.1534E+02	3.1493E+02	3.1666E+02	3.7407E-01
	GHS	5.8552E+02	5.6517E+02	4.7610E+02	7.9607E+02	7.2078E+01
	IHS	3.1421E+02	3.1415E+02	3.1402E+02	3.1485E+02	1.6504E-01
	SAGHS	3.1524E+02	3.1524E+02	3.1524E+02	3.1524E+02	9.6367E-06
	TPHS	**3.1400E+02**	**3.1398E+02**	**3.1393E+02**	**3.1417E+02**	5.1837E-02
24	HS	2.3340E+02	2.3168E+02	2.2666E+02	2.4891E+02	5.0111E+00
	GHS	2.1671E+02	2.0701E+02	2.0010E+02	2.9787E+02	2.3504E+01
	IHS	2.3147E+02	2.2916E+02	2.2512E+02	2.4749E+02	6.1227E+00
	SAGHS	2.3099E+02	2.2923E+02	2.2451E+02	2.4408E+02	5.6198E+00
	TPHS	**2.1155E+02**	**2.1286E+02**	**2.0009E+02**	**2.1609E+02**	**3.9617E+00**
25	HS	2.0729E+02	2.0699E+02	2.0427E+02	2.1326E+02	1.7382E+00
	GHS	2.2494E+02	2.3057E+02	2.0001E+02	2.6928E+02	2.3910E+01
	IHS	2.0549E+02	2.0485E+02	2.0300E+02	2.1384E+02	2.0823E+00
	SAGHS	2.0112E+02	2.0115E+02	2.0042E+02	**2.0163E+02**	2.3373E-01
	TPHS	**2.0040E+02**	**2.0013E+02**	**2.0000E+02**	2.0200E+02	4.4645E-01
26	HS	1.3069E+02	1.0064E+02	1.0034E+02	3.3391E+02	5.2147E+01
	GHS	1.1176E+02	1.0619E+02	1.0459E+02	2.0330E+02	2.2759E+01
	IHS	1.4743E+02	1.0068E+02	1.0029E+02	3.5485E+02	6.2448E+01
	SAGHS	1.3764E+02	1.0063E+02	1.0025E+02	3.4782E+02	6.1227E+01
	TPHS	**1.1019E+02**	**1.0038E+02**	**1.0012E+02**	**2.0053E+02**	2.9747E+01
27	HS	6.3111E+02	6.1789E+02	4.0264E+02	8.1322E+02	6.9457E+01
	GHS	7.9858E+02	7.9513E+02	6.1348E+02	9.2713E+02	4.2917E+01
	IHS	7.4407E+02	7.5930E+02	4.0189E+02	8.4764E+02	7.3896E+01
	SAGHS	4.1657E+02	4.0420E+02	4.0266E+02	7.7062E+02	6.2092E+01
	TPHS	**3.8579E+02**	**3.8998E+02**	**3.4221E+02**	**5.5816E+02**	**3.8756E+01**
28	HS	1.0206E+03	9.9931E+02	8.1043E+02	1.4169E+03	1.2232E+02
	GHS	3.0923E+03	3.0154E+03	2.1470E+03	4.5924E+03	4.6842E+02
	IHS	1.0521E+03	1.0045E+03	8.3999E+02	2.1851E+03	2.3466E+02
	SAGHS	7.1793E+02	**4.8252E+02**	**4.0378E+02**	2.6649E+03	5.2086E+02
	TPHS	**7.1194E+02**	7.1797E+02	5.6402E+02	**7.7076E+02**	**3.6876E+01**
29	HS	1.4609E+03	1.3822E+03	6.1623E+02	2.7196E+03	4.3376E+02
	GHS	7.3746E+07	6.7510E+07	2.7276E+07	1.5847E+08	3.0567E+07
	IHS	1.5692E+03	1.2843E+03	4.3703E+02	5.7513E+03	8.1797E+02
	SAGHS	**2.1350E+02**	**2.1325E+02**	**2.0940E+02**	**2.1840E+02**	**2.0596E+00**
	TPHS	2.0231E+03	1.9218E+03	1.4897E+03	2.8963E+03	3.3445E+02
30	HS	4.1742E+03	4.1078E+03	1.5014E+03	8.3453E+03	1.4848E+03
	GHS	7.9341E+05	8.0434E+05	3.2141E+05	1.3284E+06	2.4945E+05
	IHS	1.6444E+03	1.5555E+03	1.0291E+03	2.5541E+03	3.6630E+02
	SAGHS	**7.7142E+02**	**7.6262E+02**	**4.2766E+02**	**1.1588E+03**	**1.7935E+02**
	TPHS	3.4871E+03	3.5835E+03	1.2875E+03	5.3605E+03	1.0256E+03



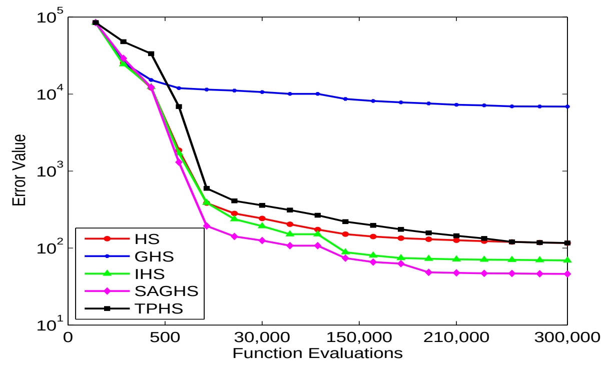
(a) Function 1



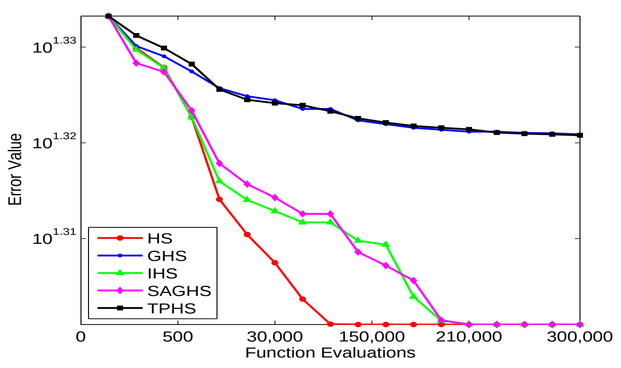
(b) Function 2



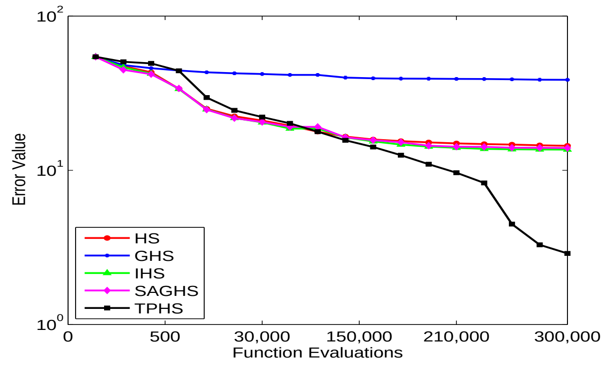
(c) Function 3



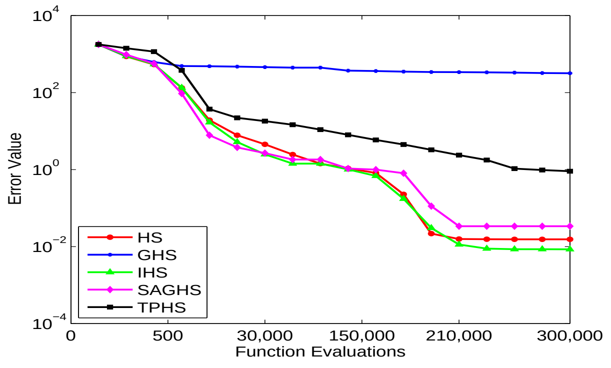
(d) Function 4



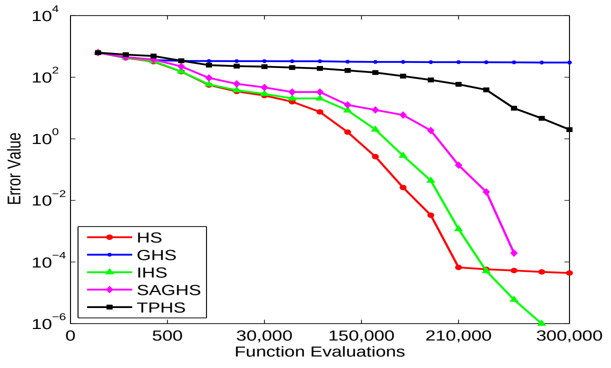
(e) Function 5



(f) Function 6

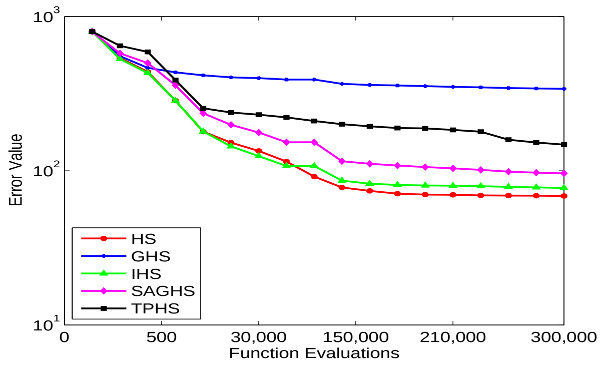


(g) Function 7

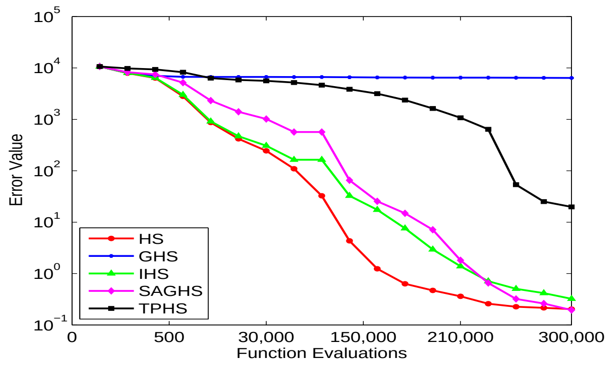


(h) Function 8

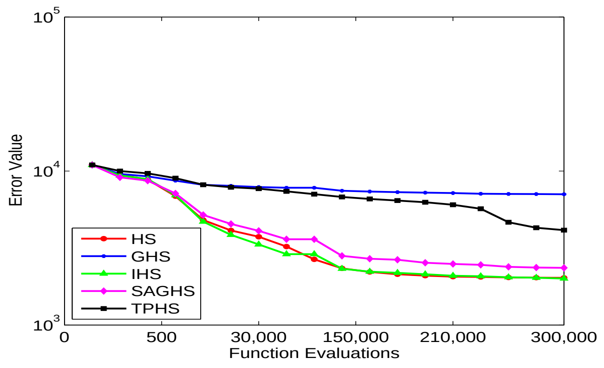
Figure 3.1: Convergence graphs of function number 1 through 8.



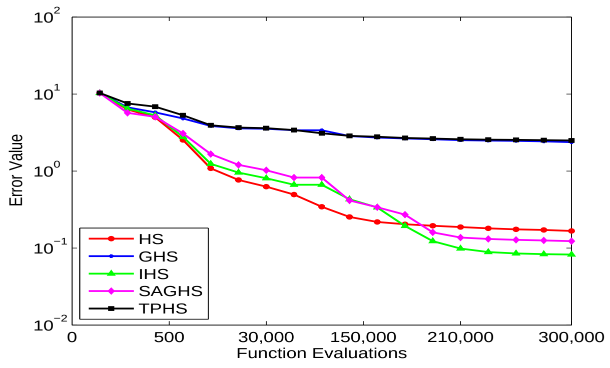
(a) Function 9



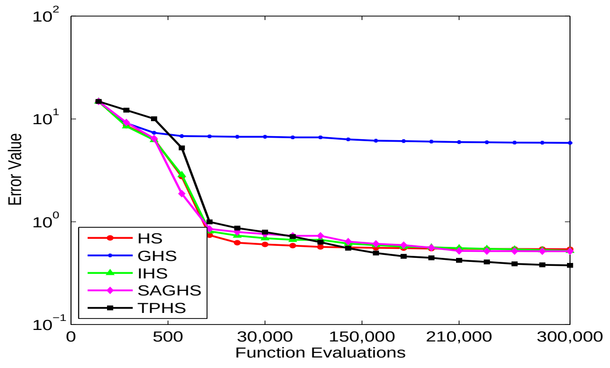
(b) Function 10



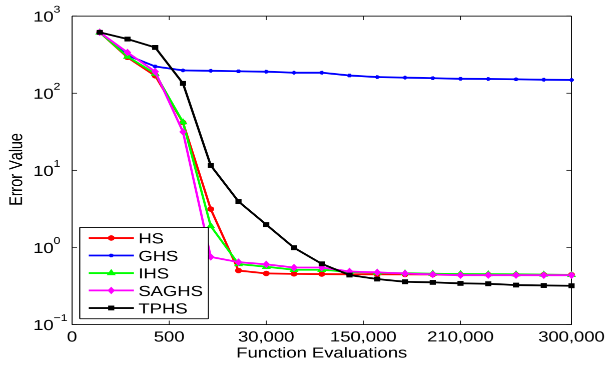
(c) Function 11



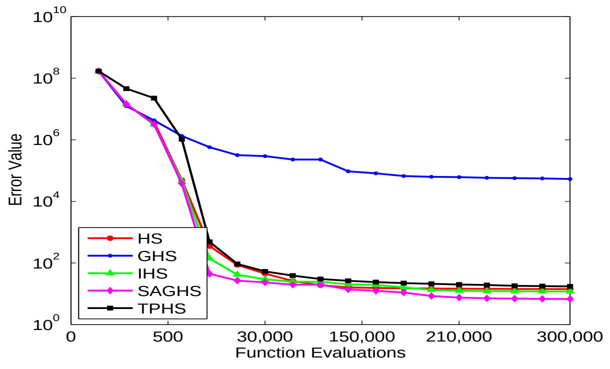
(d) Function 12



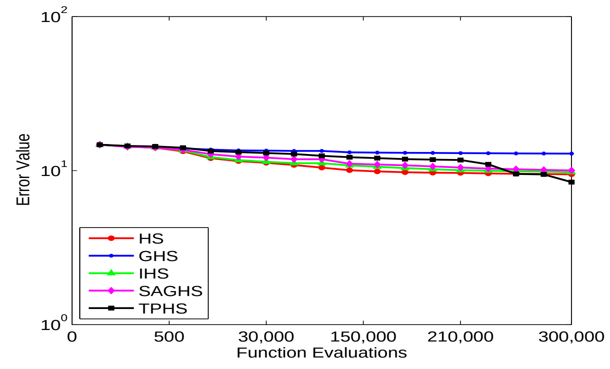
(e) Function 13



(f) Function 14

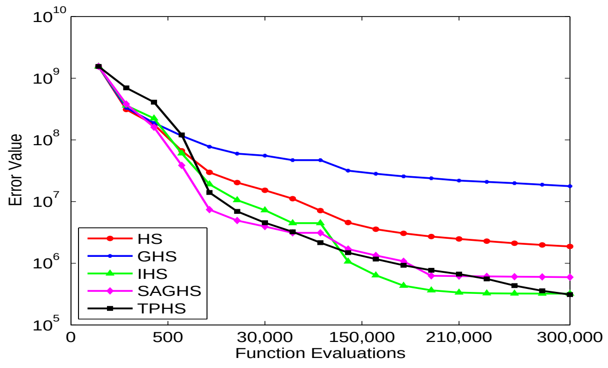


(g) Function 15

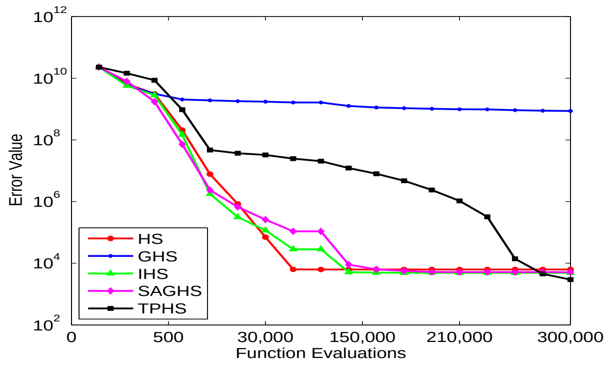


(h) Function 16

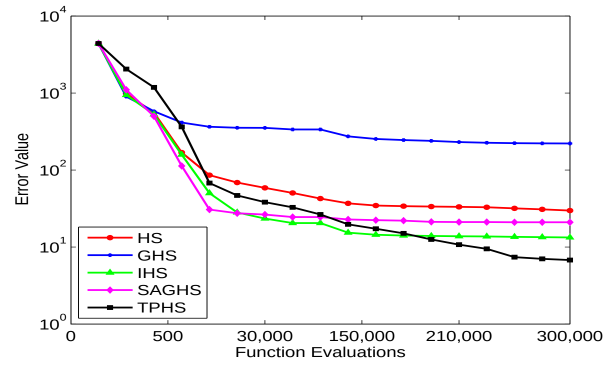
Figure 3.2: Convergence graphs of function number 9 through 16.



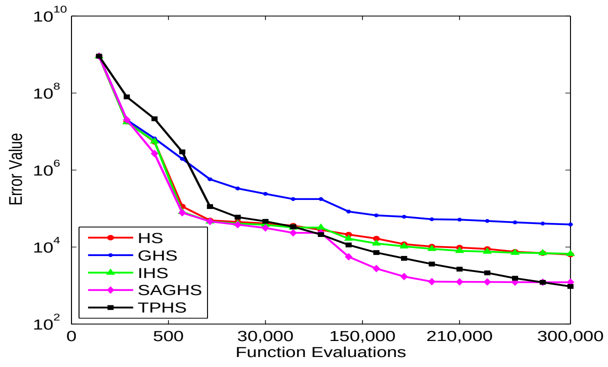
(a) Function 17



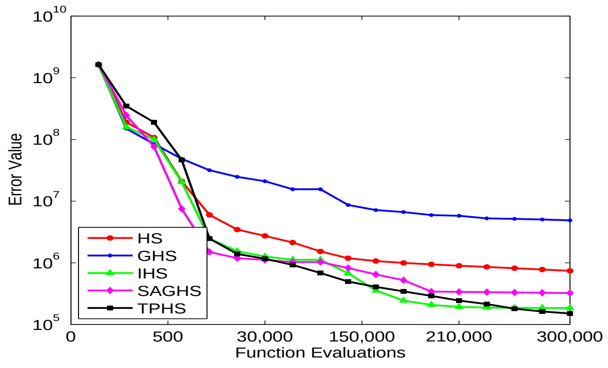
(b) Function 18



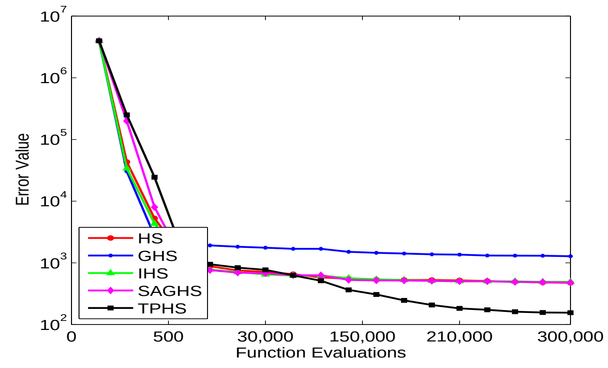
(c) Function 19



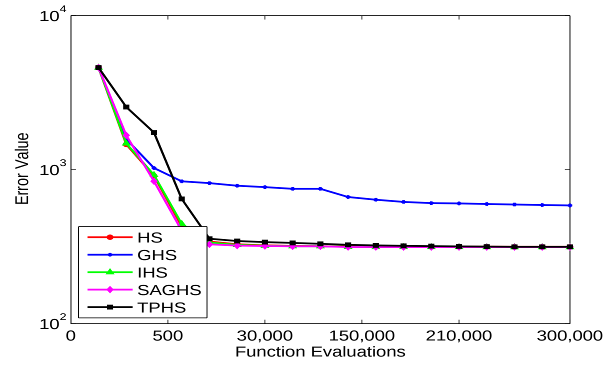
(d) Function 20



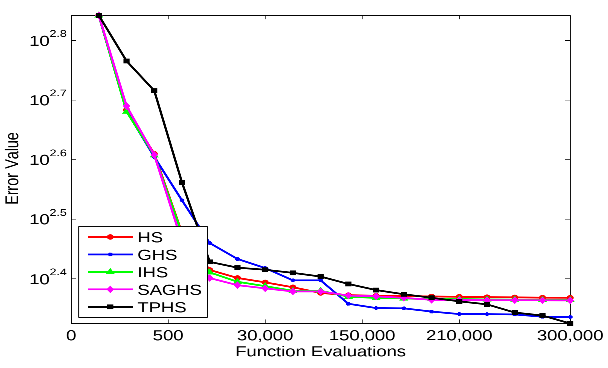
(e) Function 21



(f) Function 22

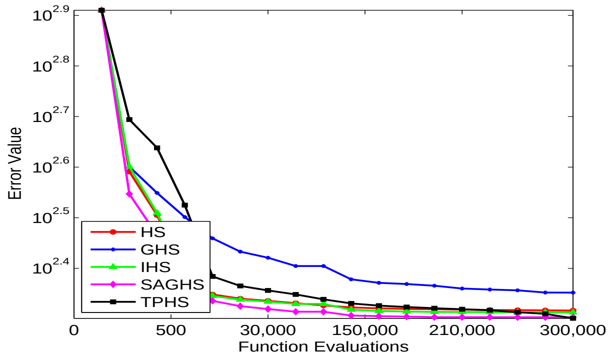


(g) Function 23

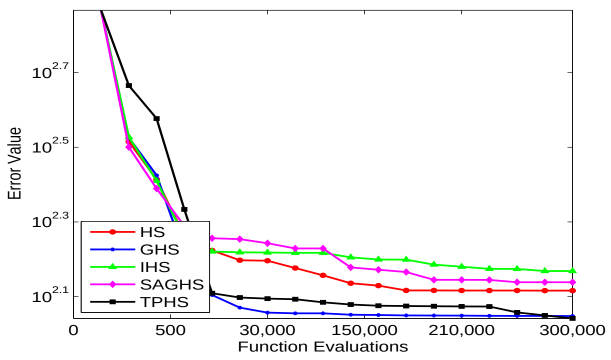


(h) Function 24

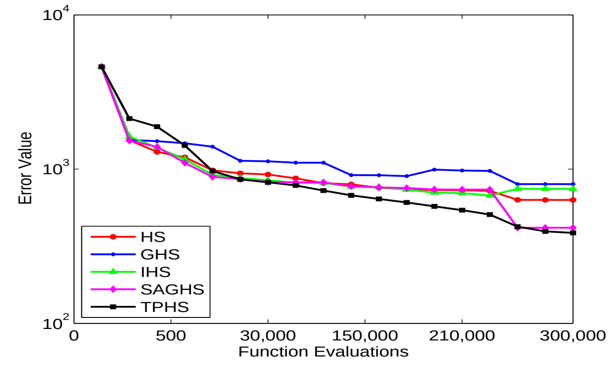
Figure 3.3: Convergence graphs of function number 17 through 24.



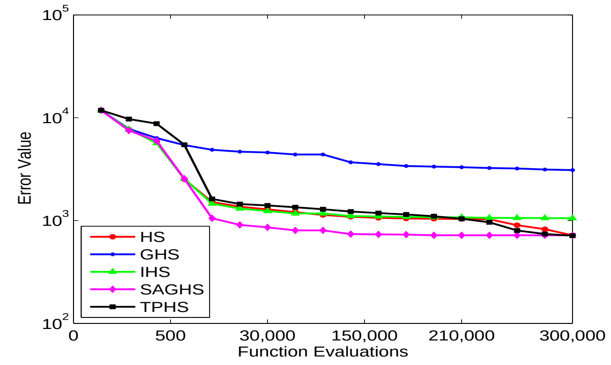
(a) Function 25



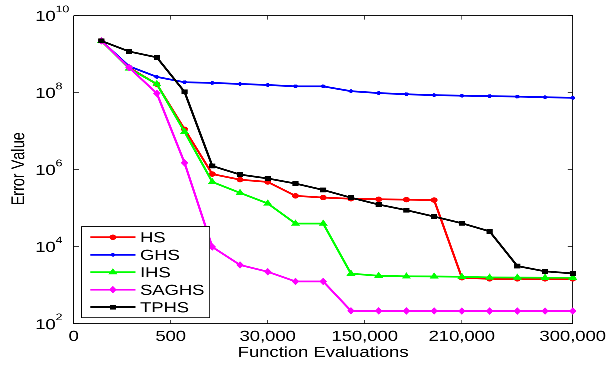
(b) Function 26



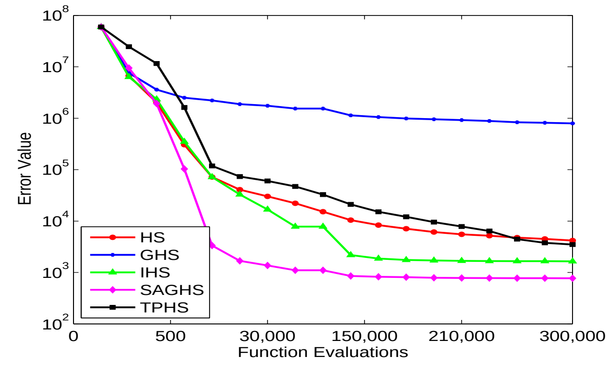
(c) Function 27



(d) Function 28



(e) Function 29



(f) Function 30

Figure 3.4: Convergence graphs of function number 25 through 30.

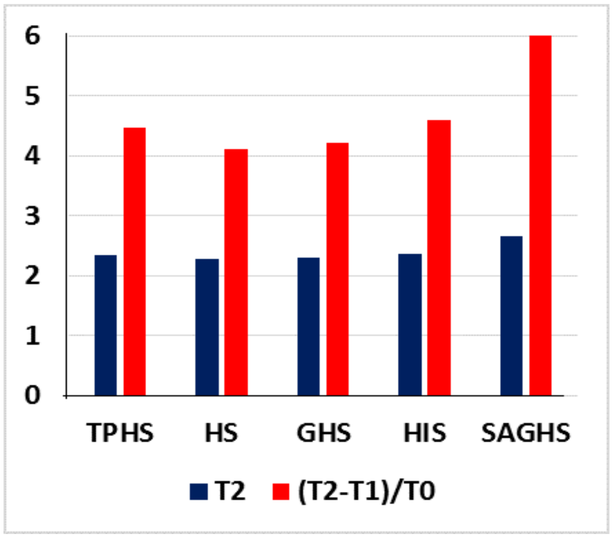


Figure 3.5: Complexity of Algorithms in seconds for 30 Dimensional problem.

Table 3.5: Mean and Standard Deviation of error value obtained by TPHS, CL-PSO, APSO and DNL-PSO on IEEE CEC 2014 benchmark functions.

	TPHS		CL-PSO		APSO		DNL-PSO	
Function	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	1.03E+07	5.00E+06	**2.62E+06**	1.21E+06	2.69E+09	3.28E+08	**1.47E+06**	1.16E+06
2	1.07E+06	1.81E+05	**2.34E+02**	1.01E+03	1.02E+11	2.29E+09	**1.78E+00**	4.18E+00
3	2.43E+03	1.59E+03	**5.86E+01**	8.21E+01	1.19E+06	1.25E+06	**9.40E+01**	1.35E+02
4	1.17E+02	2.49E+01	**5.20E+01**	3.44E+01	2.49E+04	1.54E+03	**7.31E+00**	1.01E+01
5	2.09E+01	6.36E-02	**2.00E+01**	7.44E-03	2.13E+01	5.61E-02	2.09E+01	5.72E-02
6	2.89E+00	8.16E-01	9.72E+00	1.98E+00	4.80E+01	1.79E+00	4.59E+00	2.05E+00
7	9.09E-01	5.11E-02	**1.45E-04**	1.03E-03	1.06E+03	3.85E+01	**1.06E-02**	1.10E-02
8	1.98E+00	1.23E+00	**7.22E-01**	1.12E+00	5.03E+02	3.02E+01	4.36E+01	1.16E+01
9	1.48E+02	3.45E+01	**4.52E+01**	1.00E+01	4.78E+02	6.30E+00	**5.07E+01**	1.39E+01
10	1.98E+01	4.27E+00	3.24E+01	7.35E+01	9.30E+03	5.68E+02	1.50E+03	4.26E+02
11	4.13E+03	1.69E+03	**1.85E+03**	3.22E+02	9.24E+03	4.86E+02	**2.98E+03**	7.52E+02
12	2.49E+00	2.67E-01	**1.32E-01**	3.35E-02	5.91E+00	1.32E+00	**2.14E+00**	3.91E-01
13	3.76E-01	9.14E-02	**3.00E-01**	3.93E-02	1.03E+01	7.53E-01	**3.32E-01**	8.99E-02
14	3.18E-01	5.59E-02	**2.40E-01**	3.30E-02	3.95E+02	2.22E+01	4.58E-01	2.39E-01
15	1.74E+01	1.03E+00	**4.68E+00**	1.03E+00	1.05E+06	0.00E+00	**3.71E+00**	1.15E+00
16	8.42E+00	4.87E-01	9.58E+00	5.33E-01	1.42E+01	2.37E-01	1.18E+01	7.26E-01
17	3.12E+05	2.43E+05	5.51E+05	3.82E+05	2.86E+08	1.28E+08	**1.81E+05**	1.24E+05
18	2.97E+03	2.89E+03	**2.46E+02**	2.55E+02	8.75E+09	3.11E+09	4.21E+04	7.76E+04
19	6.78E+00	9.82E-01	**6.61E+00**	8.84E-01	8.45E+02	1.15E+02	**5.67E+00**	1.91E+00
20	9.51E+02	7.09E+02	1.78E+03	2.09E+03	1.59E+07	1.37E+07	1.68E+03	1.26E+03
21	1.51E+05	1.22E+05	1.56E+05	1.26E+05	1.33E+08	7.50E+07	1.55E+05	1.89E+05
22	1.56E+02	1.08E+02	2.43E+02	1.06E+02	1.31E+04	9.38E+03	3.88E+02	1.93E+02
23	3.14E+02	5.18E-02	3.15E+02	1.71E-13	**2.00E+02**	0.00E+00	3.14E+02	1.71E-13
24	2.11E+02	4.58E+00	2.23E+02	5.96E+00	**2.00E+02**	0.00E+00	2.35E+02	8.78E+00
25	2.00E+02	7.12E-01	2.06E+02	1.02E+00	2.00E+02	0.00E+00	2.01E+02	2.63E-01
26	1.11E+02	4.34E+01	**1.04E+02**	1.94E+01	1.86E+02	2.68E+01	**1.00E+02**	1.05E-01
27	3.86E+02	3.88E+01	4.37E+02	8.14E+01	**2.00E+02**	0.00E+00	4.13E+02	4.68E+01
28	7.12E+02	3.69E+01	9.46E+02	1.42E+02	**2.00E+02**	0.00E+00	**4.04E+02**	2.19E+01
29	2.02E+03	3.34E+02	**1.08E+03**	1.79E+02	**2.00E+02**	0.00E+00	2.06E+02	1.71E+00
30	3.49E+03	1.03E+03	**2.38E+03**	5.69E+02	**2.00E+02**	0.00E+00	**9.24E+02**	2.33E+02

Table 3.6: Mean and Standard Deviation of error value obtained by TPHS, HCL-PSO, SL-PSO and SR-PSO on IEEE CEC 2014 benchmark functions.

Function	TPHS		HCL-PSO		SL-PSO		SR-PSO	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	1.03E+07	5.00E+06	**2.61E+05**	1.95E+05	**3.67E+05**	2.36E+05	**1.44E+06**	1.89E+06
2	1.07E+06	1.81E+05	**8.73E+01**	1.92E+02	**1.03E+04**	1.02E+04	**2.18E+02**	4.08E+02
3	2.43E+03	1.59E+03	**1.45E+02**	1.63E+02	6.86E+03	5.96E+03	**1.75E+01**	2.66E+01
4	1.17E+02	2.49E+01	**4.04E+01**	3.26E+01	**3.26E+01**	2.98E+01	**1.03E+02**	3.70E+01
5	2.09E+01	6.36E-02	**2.02E+01**	4.98E-02	2.09E+01	4.22E-02	2.09E+01	4.30E-02
6	2.89E+00	8.16E-01	3.82E+00	1.70E+00	**9.03E-01**	1.01E+00	3.16E+00	1.62E+00
7	9.09E-01	5.11E-02	**3.16E-04**	7.79E-04	**9.18E-04**	3.20E-03	**1.06E-02**	1.41E-02
8	1.98E+00	1.23E+00	**2.47E-13**	8.33E-14	1.64E+01	4.26E+00	3.51E+01	9.10E+00
9	1.48E+02	3.45E+01	**4.20E+01**	9.10E+00	**2.47E+01**	2.06E+01	**4.25E+01**	1.15E+01
10	1.98E+01	4.27E+00	**4.96E+00**	2.32E+01	3.79E+02	2.23E+02	8.02E+02	3.24E+02
11	4.13E+03	1.69E+03	**1.84E+03**	3.01E+02	**8.81E+02**	4.82E+02	**2.12E+03**	5.49E+02
12	2.49E+00	2.67E-01	**1.80E-01**	5.07E-02	**2.30E+00**	5.34E-01	**1.82E+00**	7.15E-01
13	3.76E-01	9.14E-02	**2.29E-01**	5.60E-02	**1.78E-01**	3.21E-02	**2.06E-01**	4.06E-02
14	3.18E-01	5.59E-02	**2.19E-01**	2.90E-02	4.01E-01	7.93E-02	**2.01E-01**	3.69E-02
15	1.74E+01	1.03E+00	**3.95E+00**	1.19E+00	**5.12E+00**	3.87E+00	**3.83E+00**	9.48E-01
16	8.42E+00	4.87E-01	9.53E+00	6.81E-01	1.21E+01	2.51E-01	1.05E+01	6.60E-01
17	3.12E+05	2.43E+05	**7.99E+04**	7.15E+04	**1.11E+05**	7.38E+04	**1.52E+05**	1.17E+05
18	2.97E+03	2.89E+03	**1.23E+02**	7.32E+01	**2.05E+03**	3.47E+03	**2.71E+03**	3.06E+03
19	6.78E+00	9.82E-01	**5.61E+00**	1.28E+00	7.16E+00	1.26E+00	**6.35E+00**	1.57E+00
20	9.51E+02	7.09E+02	9.33E+02	8.87E+02	2.13E+04	1.20E+04	**4.13E+02**	2.84E+02
21	1.51E+05	1.22E+05	**2.51E+04**	1.73E+04	**7.90E+04**	6.01E+04	**5.22E+04**	4.17E+04
22	1.56E+02	1.08E+02	2.13E+02	8.40E+01	1.69E+02	9.92E+01	3.10E+02	1.21E+02
23	3.14E+02	5.18E-02	3.15E+02	3.46E-12	3.15E+02	1.71E-13	3.15E+02	6.47E-02
24	2.11E+02	4.58E+00	2.25E+02	1.15E+00	2.30E+02	5.76E+00	2.07E+02	1.02E+01
25	2.00E+02	7.12E-01	2.05E+02	1.53E+00	2.06E+02	1.80E+00	2.06E+02	1.06E+00
26	1.11E+02	4.34E+01	**1.00E+02**	5.93E-02	1.12E+02	3.22E+01	1.32E+02	4.48E+01
27	3.86E+02	3.88E+01	4.02E+02	1.46E+00	**3.66E+02**	6.09E+01	4.40E+02	7.96E+01
28	7.12E+02	3.69E+01	8.83E+02	5.38E+01	9.16E+02	9.19E+01	1.13E+03	3.06E+02
29	2.02E+03	3.34E+02	9.11E+02	9.63E+01	1.68E+03	5.53E+02	5.27E+05	2.61E+06
30	3.49E+03	1.03E+03	**1.82E+03**	3.55E+02	**3.03E+03**	8.60E+02	**2.22E+03**	7.96E+02

Table 3.7: Mean and Standard Deviation of error value obtained by TPHS, DERAND1BIN, DERAND1BIN-SPS, and DMS-PSO-HS on IEEE CEC 2014 benchmark functions.

	TPHS		DERAND1BIN		DERAND1BIN-SPS		DMS-PSO-HS	
Function	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	1.03E+07	5.00E+06	8.81E+07	1.88E+07	3.28E+07	9.55E+06	**4.61E+06**	3.09E+06
2	1.07E+06	1.81E+05	**1.72E+03**	4.93E+02	**1.55E+03**	4.18E+02	**8.44E-01**	1.99E+00
3	2.43E+03	1.59E+03	**2.55E+01**	5.04E+00	**1.90E+01**	2.97E+00	3.74E+03	4.73E+03
4	1.17E+02	2.49E+01	1.23E+02	5.77E+00	**8.97E+01**	6.70E+00	**3.58E+01**	3.63E+01
5	2.09E+01	6.36E-02	2.09E+01	5.77E-02	2.09E+01	5.51E-02	**2.01E+01**	1.00E-02
6	2.89E+00	8.16E-01	3.03E+01	1.08E+00	4.59E+00	1.40E+00	1.05E+01	5.97E+02
7	9.09E-01	5.11E-02	**4.34E-02**	7.93E-02	**1.05E-03**	7.41E-04	**3.13E-02**	1.20E-02
8	1.98E+00	1.23E+00	1.20E+02	6.53E+00	8.65E+01	1.13E+01	**0.00E+00**	0.00E+00
9	1.48E+02	3.45E+01	1.95E+02	8.96E+00	1.78E+02	1.16E+01	**3.16E+01**	8.92E+02
10	1.98E+01	4.27E+00	3.91E+03	2.38E+02	2.42E+03	2.66E+02	**8.09E-01**	9.99E+02
11	4.13E+03	1.69E+03	6.55E+03	2.48E+02	6.19E+03	2.71E+02	**1.69E+03**	8.95E+02
12	2.49E+00	2.67E-01	**2.08E+00**	2.05E-01	**9.36E-01**	3.52E-01	**1.93E-01**	1.20E+03
13	3.76E-01	9.14E-02	4.88E-01	4.56E-02	4.15E-01	5.06E-02	**2.49E-01**	1.30E+03
14	3.18E-01	5.59E-02	**2.94E-01**	3.77E-02	**2.68E-01**	2.98E-02	6.42E-01	1.40E+03
15	1.74E+01	1.03E+00	1.90E+01	1.13E+00	**1.73E+01**	9.07E-01	**5.95E+00**	1.50E+03
16	8.42E+00	4.87E-01	1.25E+01	2.23E-01	1.20E+01	2.59E-01	8.82E+00	1.60E+03
17	3.12E+05	2.43E+05	2.40E+06	5.69E+05	7.71E+05	2.91E+05	9.31E+05	9.47E+05
18	2.97E+03	2.89E+03	2.88E+04	1.53E+04	**1.51E+03**	1.66E+03	9.73E+03	9.30E+03
19	6.78E+00	9.82E-01	1.06E+01	5.81E-01	**5.77E+00**	2.43E-01	8.34E+00	1.90E+03
20	9.51E+02	7.09E+02	**4.58E+02**	8.33E+01	**2.01E+02**	2.85E+01	3.79E+03	5.95E+02
21	1.51E+05	1.22E+05	1.89E+05	6.88E+04	**3.93E+04**	1.74E+04	1.93E+05	1.07E+05
22	1.56E+02	1.08E+02	2.10E+02	7.23E+01	**1.00E+02**	8.53E+01	3.13E+02	2.09E+03
23	3.14E+02	5.18E-02	3.15E+02	8.08E-05	3.15E+02	8.31E-05	3.15E+02	2.30E+03
24	2.11E+02	4.58E+00	**2.09E+02**	3.63E+00	**2.04E+02**	4.93E-01	2.26E+02	2.40E+03
25	2.00E+02	7.12E-01	2.23E+02	2.80E+00	2.10E+02	1.69E+00	2.07E+02	2.50E+00
26	1.11E+02	4.34E+01	**1.00E+02**	4.77E-02	**1.00E+02**	3.87E-02	1.58E+02	2.48E+01
27	3.86E+02	3.88E+01	3.89E+02	3.73E+01	**3.33E+02**	3.74E+00	5.75E+02	2.64E+01
28	7.12E+02	3.69E+01	9.77E+02	2.78E+01	7.99E+02	1.87E+01	9.25E+02	2.77E+03
29	2.02E+03	3.34E+02	1.18E+04	3.05E+03	2.46E+03	2.82E+02	4.20E+06	4.85E+06
30	3.49E+03	1.03E+03	5.59E+03	8.73E+02	**2.55E+03**	5.23E+02	1.09E+04	1.07E+04

Table 3.8: Mean and Standard Deviation of error value obtained by TPHS, CMA-ES and SSO on IEEE CEC 2014 benchmark functions.

	TPHS		CMA-ES		SSO	
Function	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	1.03E+07	5.00E+06	**0.00E+00**	0.00E+00	3.87E+09	1.21E+09
2	1.07E+06	1.81E+05	**0.00E+00**	0.00E+00	1.42E+11	2.21E+10
3	2.43E+03	1.59E+03	5.30E+04	1.60E+05	1.12E+07	1.57E+07
4	1.17E+02	2.49E+01	**3.10E-01**	1.00E+00	4.40E+04	1.03E+04
5	2.09E+01	6.36E-02	**2.00E+01**	7.20E+01	2.14E+01	7.98E-02
6	2.89E+00	8.16E-01	3.20E+01	8.30E+00	4.92E+01	2.77E+00
7	9.09E-01	5.11E-02	**1.20E-03**	3.40E+03	1.29E+03	1.89E+02
8	1.98E+00	1.23E+00	4.40E+02	8.20E+01	5.22E+02	3.21E+01
9	1.48E+02	3.45E+01	6.50E+02	1.50E+02	6.41E+02	3.84E+01
10	1.98E+01	4.27E+00	5.10E+03	6.20E+02	9.42E+03	4.27E+02
11	4.13E+03	1.69E+03	5.00E+03	7.60E+02	9.60E+03	5.11E+02
12	2.49E+00	2.67E-01	**4.60E-02**	2.90E+02	6.74E+00	1.42E+00
13	3.76E-01	9.14E-02	**3.30E-01**	2.00E+01	1.11E+01	1.44E+00
14	3.18E-01	5.59E-02	5.30E-01	2.20E+01	4.38E+02	6.62E+01
15	1.74E+01	1.03E+00	**3.60E+00**	9.10E+01	3.74E+07	2.26E+07
16	8.42E+00	4.87E-01	1.40E+01	4.20E+01	1.43E+01	2.13E-01
17	3.12E+05	2.43E+05	**2.00E+03**	4.70E+02	4.15E+08	2.10E+08
18	2.97E+03	2.89E+03	**2.80E+02**	1.30E+02	1.04E+10	3.43E+09
19	6.78E+00	9.82E-01	9.40E+00	1.90E+00	1.53E+03	5.75E+02
20	9.51E+02	7.09E+02	2.00E+04	9.50E+04	2.42E+07	2.53E+07
21	1.51E+05	1.22E+05	**1.00E+03**	3.10E+02	2.22E+08	1.13E+08
22	1.56E+02	1.08E+02	4.20E+02	2.30E+02	4.26E+04	4.90E+04
23	3.14E+02	5.18E-02	3.15E+02	3.50E+13	2.27E+03	5.91E+02
24	2.11E+02	4.58E+00	3.10E+02	2.40E+02	5.59E+02	3.63E+01
25	2.00E+02	7.12E-01	2.00E+02	2.30E+03	4.97E+02	7.14E+01
26	1.11E+02	4.34E+01	**1.10E+02**	5.90E+01	3.67E+02	1.41E+02
27	3.86E+02	3.88E+01	4.50E+02	1.20E+02	2.09E+03	3.65E+02
28	7.12E+02	3.69E+01	4.60E+03	3.20E+03	8.79E+03	7.61E+02
29	2.02E+03	3.34E+02	**2.00E+02**	1.70E+00	8.96E+08	2.53E+08
30	3.49E+03	1.03E+03	**9.50E+02**	3.30E+02	1.44E+07	8.93E+06

Table 3.9: Wilcoxon rank test results between the TPHS and 15 compared algorithms.

FUNCTION NO.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TPHS/HS	=	-	+	=	-	+	-	-	-	-	-	-	+	+	-
TPHS/GHS	+	+	+	+	=	+	+	+	+	+	+	=	+	+	+
TPHS/IHS	-	-	+	-	-	+	-	-	-	-	-	-	+	+	-
TPHS/SAGHS	-	-	=	-	-	+	-	-	-	-	-	-	+	=	-
TPHS/CL-PSO	-	-	-	-	-	+	-	-	-	+	-	-	-	-	-
TPHS/APSO	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
TPHS/DNL-PSO	-	-	-	-	=	+	-	+	-	+	-	-	-	+	-
TPHS/HCL-PSO	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-
TPHS/SL-PSO	-	-	+	-	=	-	-	+	-	+	+	-	-	+	-
TPHS/SR-PSO	-	-	-	-	=	=	-	+	-	+	-	-	-	-	-
TPHS/CMA-ES	-	-	+	-	-	+	=	+	+	+	+	-	=	+	-
TPHS/SSO	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
TPHS/DERAND1BIN	+	-	-	=	+	+	-	+	+	+	+	-	+	-	+
TPHS/DERAND1BIN-SPS	-	+	+	-	+	+	-	+	-	+	+	-	+	-	-
TPHS/DMS-PSO-HS	-	-	+	-	-	+	-	-	-	-	-	-	-	+	-

FUNCTION NO.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
TPHS/HS	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+
TPHS/GHS	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
TPHS/IHS	+	=	=	+	+	=	+	+	+	+	+	+	+	-	-
TPHS/SAGHS	+	+	+	+	=	+	+	+	+	+	+	+	+	-	-
TPHS/CL-PSO	+	+	-	=	+	=	+	+	+	=	=	+	+	-	-
TPHS/APSO	+	+	+	+	+	+	+	-	-	=	+	-	-	-	-
TPHS/DNL-PSO	+	-	+	-	+	=	+	+	+	+	-	+	+	+	-
TPHS/HCL-PSO	+	-	-	-	=	-	+	+	+	+	-	+	+	+	-
TPHS/SL-PSO	+	-	-	=	+	-	=	+	+	+	-	=	+	+	-
TPHS/SR-PSO	+	-	=	=	-	-	+	=	+	=	=	+	+	+	-
TPHS/CMA-ES	+	-	-	+	+	-	+	+	+	+	=	+	+	-	-
TPHS/SSO	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
TPHS/DERAND1BIN	+	+	+	+	-	+	+	+	-	+	-	+	+	+	+
TPHS/DERAND1BIN-SPS	+	+	-	-	-	-	-	+	+	+	=	-	+	+	-
TPHS/DMS-PSO-HS	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Chapter 4
Shrinking Memory Harmony Search Algorithm for Continuous Optimization

4.1 Introduction

The efficiency of meta heuristic algorithms depend on the extent of balance between diversification and intensification during the course of the search. Intensification also called exploitation is the ability of an algorithm to exploit the search space in the vicinity of the current good solution while diversification also called exploration is the process of exploring the new regions of a large search space thus allows dissemination of the new information into the population. Proper balance between these two contradicting characteristics is a must to enhance the performance of the algorithm.

A large Harmony Memory size although increases the exploration of the HS algorithm however reduces its exploitation capabilities particularly towards the end of execution and on the other hand small size of Harmony memory reduces its capabilities to escape local optimal. Based on the idea of balanced intensification and diversification a new Harmony Search variant called Shrinking Memory Harmony Search algorithm (SMHS) is proposed in this chapter. SMHS attempts to strike a balance between exploration and exploitation by concentrating on diversification in the beginning using extended memory and broad Bandwidth and then gradually switching to intensification by shrinking harmony memory and utilizing local search operator. The performance of SMHS is compared with nineteen state-of-the-art meta heuristic algorithms (four Harmony Search, five Particle Swarm optimization , eight Differential evolution and one variant each of Genetic Algorithm and Evolutionary Strategies) on all the 30 IEEE CEC 2014 benchmark functions (Liang et al., 2013). The numerical results demonstrate the superiority of the proposed SMHS algorithm on multimodal function and its performance is outstanding on composition functions. Composition functions are combined, rotated, shifted, and biased version of other unimodal and multi-modal test functions and mimic the difficulties of real search spaces by providing a massive number of local optima and different shapes for different regions of the search space. The proposed SMHS has not only outperformed several state-of-the-art algorithms like EPSDE, SaDE, CL-PSO, DNL-PSO, CoDE in terms of accuracy of results but is also 4900%, 4300%, 4000%, 2000%, 1088% respectively computationally efficient than them.

The organization of this chapter is as follows. Section 4.2 provides a detailed description about the proposed SMHS algorithm, Section 4.3 provides detail about numerical experimentation, analysis of results and comparison with other state-of-the-art metaheuristic algorithms and finally the chapter concludes with Section 4.4.

4.2 Proposed Shrinking Memory harmony search (SMHS) algorithm

The observation that small value of Harmony Memory Size (HMS) seizes standard HS to escape local optimal and on the other hand a large size of Harmony memory reduces its exploitation capabilities led us to the development of SMHS. The SMHS dynamically shrinks the size of Harmony Memory, uses dynamically adjusting BW/PAR, a new mechanism of generating harmony and a simple local search operator to create a balance between exploration and exploitation.

The pseudo code of SMHS is shown as Algorithm 6. Line no. 9 to 25 generate a new harmony using HM or randomization however there is a significant difference adopted by standard HS and the SMHS in generating new harmony. if the i^{th} component of the new harmony is generated using HM in standard HS, i^{th} component of some random harmony from HM is assigned to it however in SMHS either the i^{th} component of some random harmony from HM is assigned to it (step 14) or mean of i^{th} component of two randomly selected harmonies from HM (step 12) is assigned to it. The i^{th} component of harmony if generated using HM is slightly altered (determined by BW parameter) with probability PAR in step 17 (Pitch adjustment). Incase the Pitch adjustment causes the harmony to move outside the specified bounds, it is set at bounds in step 18 and 19. The newly generated harmony (H) replaces the worst harmony in HM if H is better than it (step 28 and 29). The step 32 to 45 constitutes the local search operator. The i^{th} component of the best harmony is slightly altered (within the range determined by parameter BW2) with probability P. The new harmony produced after local search (refereed as TEMP) replaces the parent harmony in case it is better than it. Even though in this algorithm the local search operator is executed towards the last quarter of algorithm execution however it can be adjusted depending on the problem in hand. After every L iterations the size of Harmony Memory shrinks by deleting the worst harmony from it and adjusting the HMS parameter accordingly.

Wang and Huang (Wang and Huang, 2010) suggested that the value of the parameter PAR and BW should be decreased with time to prevent overshoot and oscillations. This would help the algorithm to diversify the search space of the solution vectors and prevent the solution from getting trapped in local optimal. Thus it seems reasonable that decreasing the value of the parameter PAR and BW with iterations could fine tune the final solutions. Therefore PAR and BW are linearly decreased in step number 6 and 7 of Algorithm (6).

It is easy to verify the asymptotic time complexity of the proposed SMHS algorithm remains same as that of standard HS i.e. $O(N.D)$ where D is the dimension of the problem in hand and N is the number of function evaluations allowed.

In Algorithm (6) PAR_{min}, PAR_{max}, BW_{min}, BW_{max}, HMS_{min} respectively denote minimum pitch adjustment rate, maximum pitch adjustment rate, minimum bandwidth, maximum bandwidth minimum size of Harmony memory. P is the probability of altering i^{th} component of harmony selected for local search and BW2 controls the degree of exploitation around it. The other parameters are the same as already defined for Algorithm (1). The key differences between standard HS and proposed SMHS are:

1. In SMHS both PAR and BW are linearly decreased as per Equation (4.1) and (4.2) respectively whereas both PAR and BW remain constant in standard HS.

   $$PAR_{gn} = PAR_{max} + \frac{(PAR_{max} - PAR_{min})}{NI} \times (gn - 1) \tag{4.1}$$

   where PAR_{gn} is pitch adjusting rate for each generation, PAR_{min} and PAR_{max} is the minimum and maximum pitch adjusting rate respectively, NI is the maximum number of generations, gn is generation number.

   $$BW_{gn} = BW_{max} - \frac{BW_{max} - BW_{min}}{NI} \times (gn - 1) \tag{4.2}$$

   where BW_{gn} is band width for each generation, BW_{min} and BW_{max} is the minimum and maximum bandwidth respectively, NI is the maximum number of generations, gn is generation number.

2. The Harmony Memory is gradually shrieked by deleting worst harmony from HM after every L generations.

3. The i^{th} component of new harmony is either generated by using i^{th} component of some random harmony from HM or by taking mean of i^{th} component of two random harmonies from HM.

4. Towards the end (last quarter in this chapter) of the execution the algorithm shifts focus to exploitation by parallely performing local search around the best harmony.

4.3 Numerical Experiments

In this section performance of the proposed SMHS algorithm is evaluated on 30 dimensional IEEE CEC 2014 Benchmark functions (Liang et al., 2013). The IEEE CEC 2014 Benchmark suite has already been explained in Chapter 2, Section 2.4.1.

The SMHS algorithm has been compared with nineteen state-of-the-art algorithms including four

Algorithm 6 SHRINKING MEMORY HARMONY SEARCH ALGORITHM (SMHS)

```
1: Define $HMCR, PAR_{min}, PAR_{max}, BW_{min}, BW_{max}, BW2$.
2: Set $HMS = 100 \times D, \quad HMS_{min} = 5, \quad L = D, \quad P = 0.3$
3: Initialize Harmony Memory (HM).
4: NOF=HMS /* NOF is a counter representing number of function evaluations performed*/
5: **while** NOF $\leq$ Max_FES **do**
6:    Adjust PAR as per Equation (4.1).
7:    Adjust BW as per Equation (4.2).
8:    Find **Worst** harmonies in HM.
9:    **for** $i = 1$ to $D$ **do**
10:      **if** (rand $\leq$ HMCR) **then**
11:         **if** (rand $\leq$ 0.5) **then**
12:            $H_i = (HM_i^j + HM_i^k)/2$ where j=rand_int(1,HMS) and k=rand_int(1,HMS)
13:         **else**
14:            $H_i = HM_i^j$ where j=rand_int(1,HMS)
15:         **end if**
16:         **if** (rand $\leq$ PAR) **then**
17:            $H_i = H_i \pm rand \times BW$
18:            Set $H_i = LB_i \quad if \quad H_i < LB_i$
19:            Set $H_i = UB_i \quad if \quad H_i > UB_i$
20:            /* $LB_i \ and \ UB_i$ are respectively upper and lower bounds of dimension i*/
21:         **end if**
22:      **else**
23:         Generate $H_i$ randomly within the allowed bounds.
24:      **end if**
25:   **end for**
26:   Evaluate harmony H
27:   NOF++
28:   **if** (H is better than **Worst** Harmony in HM) **then**
29:      Update HM by replacing **Worst** harmony by H.
30:   **end if**
31:   **if** NOF $\geq .75\times$ Max_FES **then**
32:      /* Perform local search in the last quarter of execution */
33:      TEMP=$HM^{Best}$ /* Select Best harmony for intensification */
34:      **for** $i = 1$ to $D$ **do**
35:         **if** $rand \leq P$ **then**
36:            $TEMP_i = TEMP_i \pm BW2 \times rand$ /* Local search by slightly altering best harmony
               */
37:         **end if**
38:         Set $TEMP_i = LB_i \quad if \quad TEMP_i < LB_i$
39:         Set $TEMP_i = UB_i \quad if \quad TEMP_i > UB_i$
40:      **end for**
41:      Evaluate harmony TEMP
42:      NOF++
43:      **if** TEMP is better than $HM^{Best}$ **then**
44:         $HM^{Best} = TEMP$ /*Update Best harmony*/
45:      **end if**
46:   **end if**
47:   **if** $HMS \geq HMS_{min}$ **then**
48:      Reduce size of HM by deleting **Worst** harmony and Set HMS=HMS-1 after every L iterations
49:   **end if**
50: **end while**
51: print **Best** Harmony in HM as solution.
```

variants of Harmony search, five variants of Particle Swarm optimization, Eight Differential Evolution variants, one variant each of Genetic Algorithm and Evolutionary strategies. The experimentation has been carried out as per the instructions laid down in IEEE CEC 2014 Benchmark suite.

4.3.1 Comparison of SMHS with some state-of-the-art Harmony Search variants

In this section SMHS is compared with four variants of HS algorithm namely standard Harmony search (HS) (Geem et al., 2001), Global Best Harmony Search (GHS) (Omran and Mahdavi, 2008) , Improved Harmony Search (IHS) (Mahdavi et al., 2007) and Self-adaptive Global Best Harmony Search (SAGHS) (Pan et al., 2010b). The parameter setting adopted for standard HS, GHS, IHS has been taken from (El-Abd, 2013) and is shown in Table 4.1. Obtaining an optimal parameter setting for a metaheuristic algorithm is a hyper optimization problem, however the parameter setting shown in Table 4.1 for SMHS was found out to be optimal for most if not all the problem instances.

Table 4.1: Parameter setting of algorithms.

Algorithm	HMS	HMCR	PAR	BW
HS	5	.9	.3	.001
GHS	5	.9	$PAR_{min} = 0.01,\ PAR_{max} = 0.99$	—
IHS	5	.95	$PAR_{min} = 0.01,\ PAR_{max} = 0.99$	$BW_{min} = .00001, BW_{max} = \frac{UB-LB}{20}$
SAGHS	5	$HMCR_m = 0.98$	$PAR_m = 0.9$	$BW_{min} = .00005,\ BW_{max} = \frac{UB-LB}{10}$
SMHS	-	.9	$PAR_{min} = 0.3,\ PAR_{max} = 0.99$	$BW_{min} = .00001,\ BW_{max} = \frac{UB-LB}{20},\ BW2 = 1$ $HMS = 100 \times D, \quad HMS_{min} = 5, \quad L = D, \quad P = 0.3$

The results reported in this chapter are in the format as specified and required in IEEE CEC 2014 benchmark suite. Tables 4.2, 4.3 and 4.4 show the results obtained by the five algorithms on 30 dimensional IEEE 2014 Benchmark functions. The recorded results are the minimum, maximum, mean, median, and standard deviation of the error value obtained as specified in IEEE 2014 Benchmark suite. The error is the absolute value of difference between obtained objective function value by the algorithm and the actual function value. In all the Tables (4.2 through 4.4) the best result obtained are highlighted by bold font.

A pair wise Wilcoxons rank-sum test at 5% level of significance is used to statistically compare the performance of the five competing algorithms to confirm if the difference in results is statistically significant. The sampling data used for applying the statistical test has been obtained by performing 51 independent runs of each algorithm. The mean results of the best performing algorithm has been superscripted by $\star$ if the results produced are statistically significant comparing with other algorithms. The presence of $\star$ on two or more algorithms for a function indicates a tie when compared using statistical testing e.g. The presence of $\star$ on SAGHS and IHS for function 5 (Table 4.2) indicates SAGHS and IHS have performed significantly better than all other algorithms on function 5 however there is no significant difference between the two.

For unimodal functions (1 through 3) SMHS produced the best results in two instances and best mean results with minimum standard deviation in two instances (statistically significant in both cases). The worst results produced by SMHS are better than worst results of all other algorithms. Thus SMHS has outperformed other algorithms both in terms of efficiency and reliability on unimodal functions. In case of 13 simple unimodal functions SMHS, IHS, SAGHS produced the best mean results in 8 (statistically significant in 7 cases), 3, 3 instances respectively and produced the best results in 7, 5, 2 instances respectively. SMHS produced better worst results compared to others in 6 instances followed by HS, SAGHS and IHS in 3,2,4 instances respectively. SMHS produced the best mean results with minimum standard deviation in 6 instances followed by IHS, SAGHS in 3, 2 instances respectively. Thus proposed SMHS has significantly outperformed its variants both in terms of accuracy and stability on simple multimodal functions. In case of hybrid multimodal functions proposed SMHS algorithm produced significantly best mean results with minimum standard deviation in all the six instances. In composition functions SMHS has obtained significant best mean results in 6 instances followed by SAGHS and GHS in 1 instance each. SMHS produced the best mean results with minimum standard deviation in 4 instances followed by SAGHS in 2 instances.

Table 4.5 displays the rank obtained by SMHS when compared with HS variants, it further shows the average rank on unimodal, multimodal and composition functions. The SMHS algorithm outperformed all other HS variants on both unimodal and multimodal functions with average rank 1.66 and 1.73 respectively followed by SAGHS with average rank 2 on unimodal functions and 2.57 on multimodal functions. On composition functions the performance of SMHS is outstanding with average rank of 1.57 followed by SAGHS with average rank 2.28. Friedman test conducted at 5% level of significance confirms that the obtained ranking is statistically significant on multimodal and composition functions with P value of 4.148E-11 and 4.1E-3 respectively.

Thus the proposed SMHS algorithm outperformed other competing algorithms on all classes of functions. The performance of SMHS is particularly outstanding on hybrid and composition functions wherein it obtained the average rank of 1 and 1.57 respectively. The hybrid and composition functions are considered to be harder than simple multimodal functions.

4.3.1.1 Convergence Behavior

The convergence graphs of all the benchmark functions are plotted in Figures 4.1, 4.2, 4.3 and 4.4 to study the convergence behavior of algorithms. The horizontal line represents the number of function evaluations and the vertical line represents the mean of absolute error of 51 runs in logarithmic scale. As is evident from most of the convergence graphs the SMHS algorithm exhibits slow convergence speed in the beginning of execution, the reason being large size of Harmony Memory used, however it helps SMHS to explore vast search space and hence increase exploration initially. Due to the exploitation of the harmonies by performing random search around the best harmony (Local search) towards the end of execution most of the graphs (particularly evident for Functions 1, 2, 3, 4, 5, 6, 7, 9 10, 11, 12, 15, 17, 18, 20, 28, 29, 30) depict that SMHS makes steep progress even towards the end of execution. Thus gradually decreasing the size of harmony memory and using the local search operator creates a balance between exploration and exploitation and hence increases the performance of the proposed SMHS on most of the benchmark functions.

4.3.2 Comparison of SMHS with some state-of-the-art metaheuristic Algorithms

In this section proposed SMHS algorithm is compared with five state-of-the-art variants of PSO algorithm, Eight Differential Evolution variants, one GA variant and one ES variant on 30 dimensional IEEE CEC 2014 benchmark problems. The mean and standard deviation of 51 runs is reported in Tables 4.6, 4.7, 4.9, 4.10, 4.11, 4.13 . The results of all the algorithms has been obtained by coding them as per CEC 2014 benchmark specifications and adopting the parameter setting as given in respective papers. In all the six Table (4.6, 4.7, 4.9, 4.10, 4.11, 4.13) the mean results of all algorithm are in bold face if the corresponding algorithm performs better than SMHS algorithm. The paired Wilcoxons rank-sum test is conducted at 5% significance level to judge the significant difference of performance between the proposed SMHS algorithm and competing algorithm. The cases are marked with +, -, and = when the performance of SMHS is significantly better than, worse than, or similar to the competing algorithm. The Wilcoxons rank-sum test value is represented as column W.

4.3.2.1 Comparison of SMHS with PSO variants

In this section proposed SMHS algorithm is compared with state-of-the-art PSO variants namely : Comprehensive Learning PSO (CL-PSO) (Liang et al., 2006), Dynamic Neighborhood Learning PSO(DNL-PSO) (Nasir et al., 2012), Heterogeneous Comprehensive Learning PSO (HCL-PSO) (Lynn and Suganthan, 2015), Social Learning PSO (SL-PSO) (Cheng and Jin, 2015), Self Regulating PSO (SR-PSO) (Tanweer et al., 2015) on 30 dimensional IEEE CEC 2014 benchmark problems and the results are reported in Tables 4.6 and 4.7 .

On unimodal functions CL-PSO obtained significantly better results on two instances whereas SMHS obtained significantly better results on one instance. For simple multimodal functions SMHS obtained better results on eight instances compared to CL-PSO that managed to obtain better results on three instances and on the remaining one instance the difference is not statistically significant. In case of hybrid multimodal functions out of six instances SMHS obtained better results on five instances whereas on the remaining one instance the difference is not statistically significant. In case of composition functions SMHS outperformed CL-PSO on seven instances and on the remaining one instances the difference is not significant.

On unimodal functions both SMHS and DNL-PSO outperformed each other on one instance whereas on the remaining one instance the difference is not statistically significant. On simple multimodal functions SMHS outperformed DNL-PSO on ten instances whereas latter outperformed former on just one instance and in remaining two cases the difference is not statistically significant. In case of hybrid multimodal functions SMHS significantly outperformed DNL-PSO on all the six instances. In case of composition functions SMHS outperformed DNL-PSO on six instances whereas DNL-PSO outperformed SMHS on two instances.

On unimodal functions both SMHS and HCL-PSO outperformed each other on one instance whereas on the remaining one instance the difference is not statistically significant. On simple multimodal functions SMHS outperformed HCL-PSO on five instances whereas HCL-PSO outperformed SMHS on five instances and on remaining three instances the difference is not statistically significant. In case of hybrid multimodal functions SMHS outperformed HCL-PSO on three instances and on the remaining three instances the difference is not statistically significant. In case of composition functions SMHS outperformed HCL-PSO on seven instances and on remaining one instance the difference is not statistically significant.

On unimodal functions SMHS outperformed SL-PSO on two instance whereas on the remaining one instance the difference is not statistically significant. On simple multimodal functions SMHS outperformed SL-PSO on eight instances whereas the latter outperformed former on three instance and on remaining two cases the difference is not statistically significant. In case of six hybrid multimodal functions SMHS outperformed SL-PSO on five instances and on the remaining one instance the difference is not statistically significant. In case of composition functions SMHS outperformed SL-PSO on seven instances and on the remaining one instance the difference is not statistically significant.

On unimodal functions SR-PSO outperformed SMHS on two instance whereas latter outperformed former on one instance. On simple multimodal functions SMHS outperformed SR-PSO on nine instances whereas SR-PSO outperformed SMHS on two instances and on remaining two cases the difference is not statistically significant. In case of hybrid multimodal functions SMHS significantly outperformed SR-PSO on all the six instances. In case of composition functions SMHS outperformed SR-PSO on seven instances whereas SR-PSO outperformed SMHS on just one instance.

Table 4.8 displays the rank obtained by SMHS when compared with PSO variants, it further shows the average rank on unimodal, multimodal and composition functions. In terms of average ranking on unimodal functions SR-PSO and HCL-PSO performed better than SMHS; SL-PSO and CL-PSO performed inferior whereas performance of DNL-PSO and SMHS is same. SMHS with average rank 1.96 outperformed all variants of PSO on multimodal functions followed by HCL-PSO with average rank of 2.67. The performance of SMHS is particulary impressive on composition functions where its average rank is 1.625 followed by DNL-PSO with average rank 2.625. Friedman test conducted at 5% level of significance confirms that the obtained ranking is statistically significant on multimodal and composition functions with P value 1.5568E-6 and 2.6E-3 respectively.

It can be concluded even though the performance of SMHS is not very impressive on unimodal functions however it has significantly outperformed all variants of PSO on multimodal functions. SMHS with average rank of 1.96 has outstanding performance on composition functions.

4.3.2.2 Comparison of SMHS with DE variants

In this section proposed SMHS is compared with eight state-of-the-art variants of Differential algorithm namely: Standard DE (derand1bin) (Guo et al., 2015), jDE (Brest et al., 2006), JADE (Zhang and Sanderson, 2009), SaDE (Qin et al., 2009), EPSDE (Mallipeddi et al., 2011), CoDE (Wang et al., 2011), derand1bin-SPS (Guo et al., 2015), MPEDE (Wu et al., 2016) and the results are reported in Tables 4.7, 4.9 and 4.11.

On unimodal functions derand1bin significantly outperformed SMHS on two instances whereas SMHS obtained better results on one instance. For simple multimodal functions SMHS outperformed derand1bin on twelve instances and on the remaining one instance the difference is not statistically significant. In case of hybrid multimodal functions SMHS obtained better results on five instances whereas on the remaining one instance the difference is not statistically significant. In case of composition functions SMHS outperformed derand1bin on seven instances whereas derand1bin outperformed SMHS on just one instance.

jDE significantly outperformed SMHS on all the three instances of unimodal functions. On simple multimodal functions SMHS outperformed jDE on eight instances whereas jDE outperformed SMHS on five instances. jDE significantly outperformed SMHS on five out of six instances of hybrid multimodal functions. In case of composition functions SMHS significantly outperformed jDE on all the eight instances .

JADE significantly outperformed SMHS on all the three instances of unimodal functions. On simple multimodal functions SMHS outperformed JADE on four instances whereas JADE outperformed SMHS on six instances and on the remaining three instances the difference is not statistically significant. JADE outperformed SMHS on three instances of hybrid multimodal functions whereas latter outperformed former on two instances and in the remaining two cases the difference is not statistically significant. In case of composition functions SMHS outperformed JADE on seven instances and on the remaining one instance the difference is not statistically significant.

In case of unimodal functions SaDE outperformed SMHS on two instances whereas SMHS outperformed SaDA on one instance. On simple multimodal functions SMHS outperformed SaDE on seven instances whereas SaDE outperformed SMHS on three instances and on the remaining two instances the difference is not statistically significant. SaDE outperformed SMHS on one instances of hybrid multimodal functions whereas latter outperformed former on three instances and in the remaining two instances the difference is not statistically significant. In case of composition functions SMHS significantly outperformed SaDE on all the eight instances.

In case of unimodal functions EPSDE outperformed SMHS on all the three instances. On simple multimodal functions SMHS outperformed EPSDE on eight instances whereas EPSDE outperformed SMHS on five instances. In case of hybrid multimodal functions both EPSDE and SMHS outperformed each other on two instances whereas on remaining two instances the difference is not statistically significant. In case of composition functions SMHS outperformed EPSDE on four instances whereas EPSDE outperformed SMHS on three instance and in the remaining one instances the difference is not statistically significant.

In case of unimodal functions CoDE outperformed SMHS on all the three instances. On simple multimodal functions SMHS outperformed CoDE on four instances whereas CoDE outperformed SMHS on six instances and on remaining three instances the difference is not statistically significant. In case of hybrid multimodal functions CoDE outperformed SMHS on five instances and on the remaining one instance the difference is not statistically significant. In case of composition functions SMHS significantly outperformed CoDE on seven and on the remaining one instances the difference is not statistically significant.

In case of unimodal functions derand1bin-sps outperformed SMHS on two instances whereas latter outperformed former on one instance. On simple multimodal functions SMHS outperformed derand1bin-sps on eleven instances whereas derand1bin-sps outperformed SMHS on one instances and on remaining one instances the difference is not statistically significant. In case of hybrid multimodal functions SMHS outperformed derand1bin-sps on five instances whereas latter outperformed former on just one instance. In case of composition functions SMHS significantly outperformed derand1bin-sps on seven instances whereas derand1bin-sps outperformed SMHS on just one instance. In case of unimodal functions MPEDE outperformed SMHS on all the three instances. On simple multimodal functions SMHS outperformed MPEDE on four instances whereas MPEDE outperformed SMHS on six instances and on the remaining two instances the difference is not statistically significant. In case of hybrid multimodal functions MPEDE outperformed SMHS on all the six instances. In case of composition functions SMHS significantly outperformed MPEDE on six instances whereas MPEDE outperformed SMHS on two instance.

Table 4.12 displays the rank obtained by SMHS and DE variants on benchmark functions along with average rank on unimodal, multimodal and composition functions. The performance of SMHS is very poor as it ranks last but one on unimodal functions. However on multimodal functions with an average rank of 3.48 the algorithm outperforms all variants of DE except MPEDE having average rank 3.19 whereas CoDE ranks third. In terms of time complexity SMHS is 169% faster than MPEDE and is 1088% faster than CoDE. On composition functions the performance of SMHS is outstanding as it ranks first with an average rank of 1.88 followed by MPEDE with average rank 3.63. Friedman test conducted at 5% level of significance confirms that the obtained ranking is statistically significant on unimodal, multimodal and composition functions with P value 0.0152, 4.711E-16 and 0.0005 respectively.

4.3.2.3 Comparison of SMHS with GA and ES variant

In this section the proposed SMHS algorithm is compared with state-of-the-art variant of Genetic Algorithm (GL-25) (García-Martínez et al., 2008) and Evolutionary Strategies (CMA-ES) (Hansen and Ostermeier, 2001) and the results are reported in Table 4.13.

In case of unimodal functions GL-25 significantly outperformed SMHS on two instances whereas latter outperformed former on one instance. On simple multimodal functions SMHS outperformed GL-25 on eleven instances whereas GL-25 outperformed SMHS on just one instances and in remaining one instance the difference is not statistically significant. In case of hybrid multimodal functions SMHS significantly outperformed GL-25 on four instances and on the remaining two cases the difference in not statistically significant. In case of composition functions SMHS outperformed GL-25 on five instances whereas GL-25 outperformed SMHS on two instance and on the remaining one instances the difference is not statistically significant.

In case of unimodal functions CMA-ES significantly outperformed SMHS on all the three instance. On simple multimodal functions SMHS outperformed CMA-ES on nine instances whereas CMA-ES outperformed SMHS on three instances and on the remaining one instance the difference is not statistically significant. In case of hybrid multimodal functions SMHS outperformed CMA-ES on five instances whereas latter outperformed former on one instance and on the remaining one instances the difference in not statistically significant. In case of composition functions SMHS significantly outperformed CMA-ES on all the eight instances.

It is evident from Table 4.14 that in terms of average ranking the proposed SMHS algorithm ranked first with average rank of 1.41 followed by CMA-ES (with average rank 2.26) and GL-25 (with average rank 2.3) respectively on multimodal functions. On composition functions the performance of SMHS is outstanding as it ranks first with an average rank of 1.25 followed by CMA-ES with average rank 2.0. Friedman test conducted at 5% level of significance confirms that the obtained ranking on multimodal and composition functions is statistically significant with P value of 0.0008 and 0.0137 respectively.

4.3.3 Algorithm Complexity

The time complexity of all the algorithms is computed as per the requirements laid down in IEEE CEC 2014 Benchmark suite as explained in Section 2.4.3.1 and is given in Table 4.15.

Comparing with PSO variants SR-PSO is computationally less expensive (29%)than SMHS whereas all the other algorithms are computationally much expensive than SMHS. SMHS is 4000% computationally efficient than CL-PSO and is 2000%, 1800% and 280% faster than DNL-PSO, HCL-PSO and SL-PSO respectively.

Comparing with DE all the variants except jDE are computationally expensive than SMHS algorithm. SMHS is 4900%, 4300%, 1088%, 696%, 671%, 169%, 46% respectively faster than EPSDE, SaDE, CoDE, derand1bin-sps, derand1bin, MPEDE, JADE.

SMHS outperforms both CMA-ES and GL-25 in terms of time complexity as it is 687% faster than CMA-ES and is 2947% faster than GL-25.

4.4 Conclusion

This chapter introduced a new variant of Harmony Search algorithm for continuous optimization problems. The proposed algorithm called SMHS exhibited the desired behavior of exploring the search space at the beginning and exploiting good solutions towards the end. This was achieved by gradually shrinking the size of harmony memory, modifying the method of new harmony generation and performing local search towards the end. The search area was gradually decreased by decreasing the BW and PAR with time.

The proposed SMHS algorithm is compared with nineteen state-of-the-art evolutionary algorithms. The bases of performance is the set of 30 benchmark functions proposed in IEEE CEC 2014. All the criterion laid down in the benchmark suite are adopted to produce the required analysis metrics. Based on the numerical results and statistical tests it is established that SMHS outperforms all variants of HS algorithm on all categories of benchmark functions. SMHS outperformed all PSO variants on multimodal functions. Except MPEDE proposed SMHS has outperformed all variants of DE algorithm along with GL-25 and CMA-ES on multimodal functions. SMHS has shown outstanding performance on composition functions by outperforming all the nineteen competing algorithms.

Baring two HS variants (HS & GHS) the proposed SMHS algorithm has outperformed all the algorithms except jDE and SR-PSO in terms of time complexity. The highlights are EPSDE, SaDE, CL-PSO, DNL-PSO, CoDE which are not only inefficient in terms of accuracy when compared with SMHS but are 4900%, 4300%, 4000%, 2000%, 1088% respectively slower than the proposed SMHS algorithm.

Table 4.2: Error values obtained by HS, GHS, IHS, SAGHS and SMHS on Function number 1 through 10.

Function	Algorithm	Mean	Median	Best	Worst	Std Dev
1	HS	12898393.24	11255923.52	1275742.852	37049559.42	8537988.248
	GHS	553041042.4	552511218	227743276.5	893837505.7	133305128.1
	IHS	5111743.106	3881430.548	756695.3035	22954475.51	4120366.238
	SAGHS	3540053.152	3126725.561	676698.0242	9097762.17	1792631.959
	SMHS	**213023.5073**⋆	**208385.3916**	**80011.32734**	**435815.8459**	**64339.08784**
2	HS	**10747.65256**⋆	**5818.946263**	143.523898	34210.70203	11490.37171
	GHS	51074437927	50525661022	41517688491	62548437082	4670086400
	IHS	13404.55454⋆	10104.81313	**1.23685**	36883.301	12149.68661
	SAGHS	13069.93494⋆	**9544.947033**	5.507739	51220.7409	12816.30864
	SMHS	11738.06579⋆	12230.94873	2662.934099	**17669.45523**	**2701.945614**
3	HS	4645.883677	3907.232285	8.602066	21795.92305	4121.066791
	GHS	77427.47655	73782.12306	47596.07342	152794.6573	18602.35023
	IHS	10165.06983	6506.020454	507.586263	55071.91822	11197.86017
	SAGHS	2283.423074	1711.209265	23.510212	7912.869444	1904.61988
	SMHS	**73.74878692**⋆	**54.557035**	**1.594621**	**330.902643**	**64.08100845**
4	HS	116.1209097	124.843786	63.858588	151.810694	30.22002157
	GHS	6873.958106	6925.479665	3939.804568	9794.289035	1190.891016
	IHS	69.07221918	75.569381	1.063157	148.728518	37.52153128
	SAGHS	46.12697376	28.905198	5.849951	98.817704	26.78086338
	SMHS	**5.498342922**⋆	**1.497448**	**0.956326**	**69.181075**	**15.77162381**
5	HS	20.0001781	20.000181	20.0001	20.000263	3.80903E-05
	GHS	20.93591163	20.945665	20.789115	21.045615	0.056889952
	IHS	19.99999631⋆	20	**19.999924**	**20**	1.11135E-05
	SAGHS	**19.99999478**⋆	**19.999999**	19.999939	**20**	1.27776E-05
	SMHS	20.01340376	20.012851	20.006312	20.060855	**0.007176906**
6	HS	14.42079782	14.553111	9.889294	19.775678	2.070182661
	GHS	38.601612	39.094041	33.27127	41.334606	1.571050892
	IHS	13.62951998	13.63003	7.500701	18.956444	2.587514228
	SAGHS	13.9625838	13.942381	7.373615	19.468567	2.353850259
	SMHS	**4.416854647**⋆	**4.324486**	**1.952183**	**6.937839**	**1.15386537**
7	HS	0.015450569	0.007805	0.000157	0.083465	0.018882591
	GHS	318.7682459	306.216442	239.893687	435.19595	51.02624576
	IHS	**0.008543765**⋆	**0.007396**	**0**	**0.034336**	**0.009193675**
	SAGHS	0.033846412	0.01972	**0**	0.164343	0.034615109
	SMHS	0.018747784	0.014623	0.002915	0.064336	0.014264143
8	HS	4.36471E-05	0.000044	0.000026	0.000056	6.66222E-06
	GHS	299.3966623	300.891308	237.747273	348.061379	24.57763546
	IHS	**0**⋆	**0**	**0**	**0**	**0**
	SAGHS	**0**⋆	**0**	**0**	**0**	**0**
	SMHS	1.922145961	1.991083	0.000407	4.981983	1.332264855
9	HS	68.56553465	66.66222	42.63296	97.505583	14.70353103
	GHS	339.9753958	337.307034	302.235313	402.211249	21.68426804
	IHS	77.37083616	74.674224	43.778077	112.429944	17.34141875
	SAGHS	96.14477645	94.520848	59.697402	155.852986	21.11349873
	SMHS	**26.90473286**⋆	**26.866657**	**16.915199**	**38.804829**	**5.683187425**
10	HS	0.203551608⋆	0.209773	**0.105149**	**0.33868**	0.055092209
	GHS	6379.864173	6388.030002	5706.171649	7000.342841	328.4144327
	IHS	0.323453412	0.259432	0.151246	1.28479	0.238371758
	SAGHS	**0.195483373**⋆	**0.187373**	0.124916	0.353928	**0.052196404**
	SMHS	6.989729157	6.45606	2.606772	11.669273	2.357008632

Table 4.3: Error values obtained by HS, GHS, IHS, SAGHS and SMHS on Function number 11 through 20.

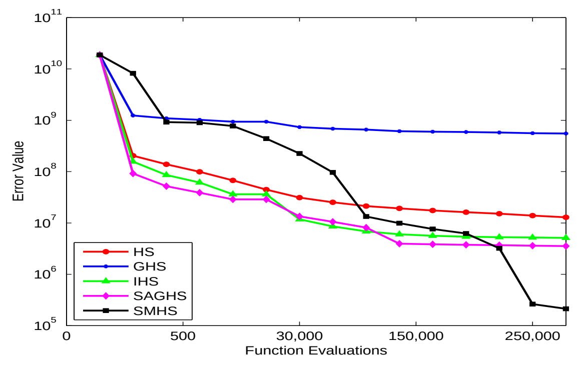
Function	Algorithm	Mean	Median	Best	Worst	Std Dev
11	HS	2027.080108	2048.614204	1090.134144	**2946.196146**	404.4026456
	GHS	7070.026547	7140.110076	6075.412077	7688.125781	**348.489945**
	IHS	2004.250884	2008.451785	**837.262591**	3334.502214	506.4423186
	SAGHS	2352.570445	2447.141552	1262.78901	3315.981056	504.0230796
	SMHS	**1805.600182**⋆	**1573.515363**	865.926792	4753.75715	857.7162758
12	HS	0.16681349	0.158972	0.061356	0.268577	0.050018689
	GHS	2.374949922	2.427165	1.387925	3.04436	0.321263496
	IHS	**0.082523078**⋆	**0.07984**	**0.025872**	**0.186043**	**0.037921337**
	SAGHS	0.122863667	0.113641	0.060261	0.247777	0.041330693
	SMHS	0.625223745	0.612016	0.156168	1.09193	0.190783827
13	HS	0.540846059	0.542993	0.304645	0.871414	0.119321937
	GHS	5.850542824	5.893115	5.016382	6.807992	0.415742818
	IHS	0.521251902	0.50641	0.25694	0.804571	0.126179498
	SAGHS	0.516630176	0.500885	0.303623	0.809245	0.118666333
	SMHS	**0.18662198**⋆	**0.183684**	**0.10202**	**0.298234**	**0.039571128**
14	HS	0.442398745	0.352029	0.240295	1.057127	0.224739131
	GHS	148.2893225	149.122914	120.60311	173.699911	14.04855252
	IHS	0.440639039	0.347378	0.203606	0.983402	0.231951897
	SAGHS	0.436484627	0.318582	0.138859	1.057207	0.252619023
	SMHS	**0.20902598**⋆	**0.210331**	**0.12922**	**0.290887**	**0.037121302**
15	HS	14.25183516	13.386853	5.542791	27.13197	5.584896209
	GHS	53442.27954	37607.24159	12241.04067	491585.2736	70129.66734
	IHS	11.97256022	10.532876	3.550865	31.3114	5.071632477
	SAGHS	6.781837863	6.219424	3.780967	13.668984	2.097646724
	SMHS	**3.901825804**⋆	**3.770978**	**2.079053**	**7.706382**	**1.160102074**
16	HS	9.466967745⋆	9.551849	7.824325	**11.027997**	0.658644472
	GHS	12.89389518	12.950327	12.0298	13.255343	**0.274134003**
	IHS	9.769183725⋆	9.623785	7.816398	11.302884	0.691121273
	SAGHS	10.03466557⋆	10.062951	8.337515	11.543517	0.724948835
	SMHS	**9.416664588**⋆	**9.542452**	**7.577889**	11.17894	0.719343406
17	HS	1872455.054	1150714.85	65245.29172	7827324.942	1607209.857
	GHS	17786962.93	17106287.63	4115632.923	37051843.72	6211276.262
	IHS	322217.286	275952.662	49076.7676	1352171.361	226598.5551
	SAGHS	595172.2906	571663.3701	63973.37375	1808868.527	376098.5264
	SMHS	**29252.04161**⋆	**29949.78783**	**12225.04603**	**44817.02614**	**7461.684854**
18	HS	6286.288995	2529.802758	72.45676	28773.28667	7846.793695
	GHS	866563213.4	887991906.2	110874540.9	1504905850	305745139.9
	IHS	4875.709344	2536.737544	50.857977	18914.12268	4893.203143
	SAGHS	5170.971233	2588.407522	69.118786	31255.83507	5994.194634
	SMHS	**164.9463745**⋆	**137.047333**	**46.878659**	**517.083324**	**100.9456214**
19	HS	29.81384718	9.314233	5.964728	118.961093	35.77007601
	GHS	221.2664664	224.180508	133.467578	279.652268	30.51975
	IHS	13.3460691	10.596594	3.617235	87.776488	14.39915309
	SAGHS	20.9736938	13.931164	5.500482	83.077851	21.60309993
	SMHS	**4.40011**⋆	**4.402149**	**4.122914**	**7.39868**	**1.253168106**
20	HS	6399.684901	5605.206108	172.482507	24391.98067	4972.295895
	GHS	38662.83332	32772.73234	8394.155391	109300.9111	23294.68409
	IHS	6675.48685	5638.377426	415.183357	26302.77468	5714.862651
	SAGHS	1217.869568	612.552866	65.698978	6543.778255	1411.543191
	SMHS	**133.9134672**⋆	**123.79357**	**55.901024**	**298.428824**	**53.74761146**

Table 4.4: Error value obtained by HS, GHS, IHS, SAGHS and SMHS on Function number 21 through 30.

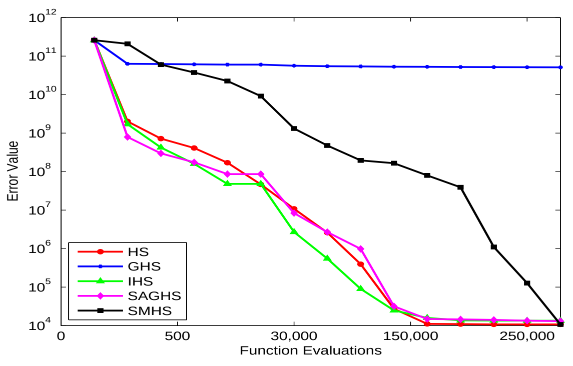
Function	Algorithm	Mean	Median	Best	Worst	Std Dev
21	HS	743513.9241	659366.3933	69129.07749	2285912.755	530164.7687
	GHS	4884804.492	4223776.471	781910.5889	10704996.07	2354320.695
	IHS	185293.2851	127396.1153	10835.18363	777016.9621	161869.8105
	SAGHS	323971.8709	234153.4984	28262.18029	1306293.626	244930.6569
	SMHS	**27589.97734**⋆	**27898.50361**	**13913.97038**	**45909.00805**	**5849.117544**
22	HS	474.3012613	489.861004	40.51064	768.378022	168.6369286
	GHS	1279.455167	1309.890051	801.598218	1547.729524	181.8802168
	IHS	491.8082253	489.619135	29.915167	860.929376	191.4266893
	SAGHS	479.1369292	494.178812	**27.008538**	858.303043	200.3122931
	SMHS	**201.0627432**⋆	**153.189069**	140.924194	**500.815824**	**88.44076938**
23	HS	315.4197806	315.341833	314.932119	316.66243	0.374068396
	GHS	585.5150125	565.169431	476.102695	796.066091	72.07819599
	IHS	315.2441	315.2441	315.2441	315.2441	**0**
	SAGHS	315.2441233	315.244121	315.244109	315.244149	9.63669E-06
	SMHS	**314.2139646**⋆	**314.151317**	**314.023479**	**314.850163**	0.165042007
24	HS	233.402739	231.678696	226.661674	248.910545	5.011128284
	GHS	**216.7086231**⋆	**207.012698**	**200.096649**	297.866707	23.50419288
	IHS	231.4708485	229.160944	225.115663	247.485571	6.122655755
	SAGHS	230.9868523	229.231748	224.505952	244.083832	5.61981002
	SMHS	223.5529364	223.76708	221.359778	**224.863774**	**0.875821903**
25	HS	207.2921108	206.994387	204.271347	213.259391	1.738213126
	GHS	224.9444395	230.56587	**200.006862**	269.27625	23.91023736
	IHS	205.492629	204.852315	203.000073	213.835901	2.082258816
	SAGHS	201.1192975	201.14822	200.417256	201.629896	0.233726723
	SMHS	**200.0149112**⋆	**200.0146**	200.010982	**200.020977**	**0.002094303**
26	HS	130.689059	100.640222	100.344593	333.907232	52.14705699
	GHS	111.7605673	106.188353	104.590445	203.301147	22.75896703
	IHS	147.4301095	100.677159	100.289159	354.850807	62.44768405
	SAGHS	137.6414507	100.631273	100.248868	347.820504	61.22686766
	SMHS	**100.2432175**⋆	**100.235868**	**100.160609**	**100.362343**	**0.045565105**
27	HS	631.1097115	617.887475	402.635132	813.217812	69.4571261
	GHS	798.584023	795.134636	613.477271	927.126278	42.91659415
	IHS	744.0677306	759.300689	401.885097	847.637132	73.89595062
	SAGHS	416.5739448	404.200142	402.663585	770.616986	62.09240929
	SMHS	**325.703752**⋆	**323.46806**	**311.802915**	**376.054565**	**12.33567324**
28	HS	1020.619369	999.30626	810.427072	1416.928016	122.3162145
	GHS	3092.27368	3015.439564	2147.014276	4592.394606	468.4237471
	IHS	1052.110489	1004.450247	839.985652	2185.129095	234.6577668
	SAGHS	717.9256325	482.517605	403.780821	2664.908141	520.8586495
	SMHS	**320.4685211**⋆	**301.325745**	**300.848013**	**831.068543**	**95.16088826**
29	HS	1460.934144	1382.166198	616.225827	2719.629464	433.7608047
	GHS	73746205.69	67509668.12	27276490.96	158474932.1	30566978.14
	IHS	1569.225437	1284.334529	437.029571	5751.302264	817.9743332
	SAGHS	**213.5011533**⋆	**213.246657**	**209.403796**	**218.398893**	**2.059574806**
	SMHS	640.3656	645.356707	550.042563	1241.214908	100.0886119
30	HS	4174.195026	4107.780848	1501.448255	8345.342409	1484.76458
	GHS	793405.7188	804339.4469	321407.5252	1328381.927	249451.853
	IHS	1644.421618	1555.504891	1029.097929	2554.079213	366.3027769
	SAGHS	771.4193602	762.619792	**427.661795**	**1158.771434**	**179.3532807**
	SMHS	**683.2017**⋆	**690.420883**	617.843155	1234.492046	265.5031526

Table 4.5: Rank obtained by HS, GHS, IHS, SAGHS, SMHS on IEEE CEC 2014 benchmark problems.

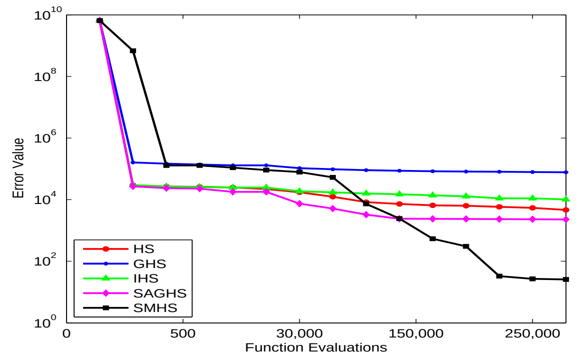
Function	HS	GHS	IHS	SAGHS	SMHS
1	4	5	3	2	1
2	4	5	1	2	3
3	3	5	4	2	1
4	4	5	3	2	1
5	3	5	1	2	4
6	4	5	2	3	1
7	2	5	1	4	3
8	3	5	1	1	4
9	2	5	3	4	1
10	2	5	3	1	4
11	2	5	3	4	1
12	3	5	1	2	4
13	4	5	3	2	1
14	2	5	3	4	1
15	4	5	3	2	1
16	1	5	3	4	2
17	4	5	2	3	1
18	3	5	2	4	1
19	4	5	3	2	1
20	4	5	3	2	1
21	4	5	2	3	1
22	3	5	4	2	1
23	4	5	2	1	3
24	5	1	4	3	2
25	4	5	3	2	1
26	4	2	5	3	1
27	3	5	2	4	1
28	3	5	4	2	1
29	3	5	4	1	2
30	4	5	3	2	1
Avg. Rank Unimodal	3.66	5	2.66	2	**1.66**
Avg. Rank Multimodal	3.23	4.73	2.69	2.57	**1.73**
Avg. Rank Composition	3.71	4	3.42	2.28	**1.57**



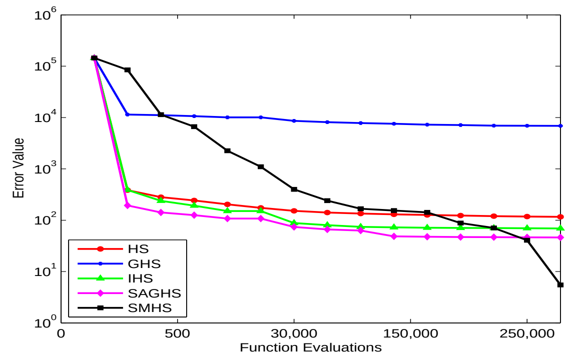
(a) Function 1



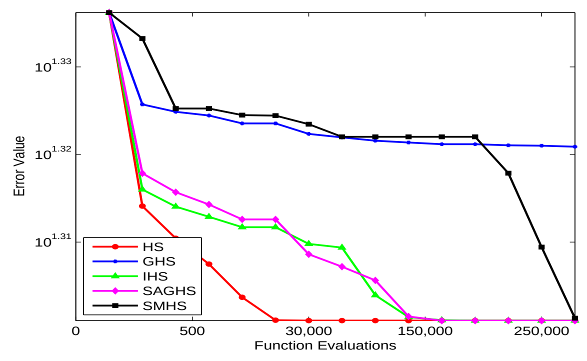
(b) Function 2



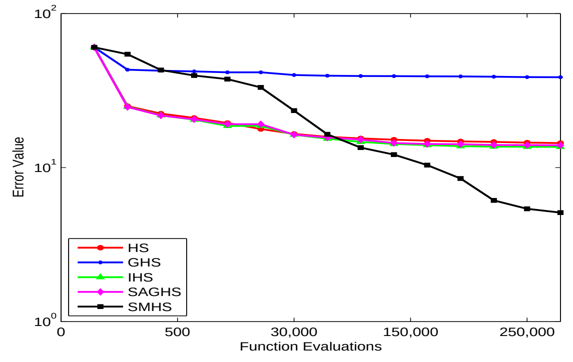
(c) Function 3



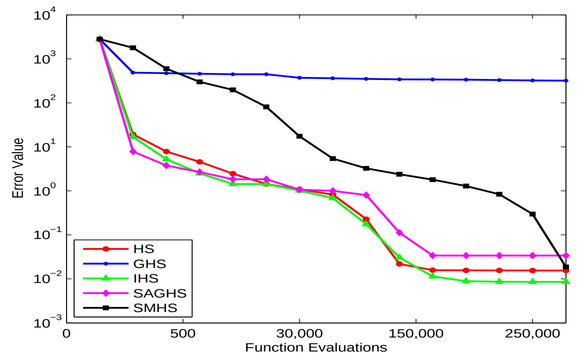
(d) Function 4



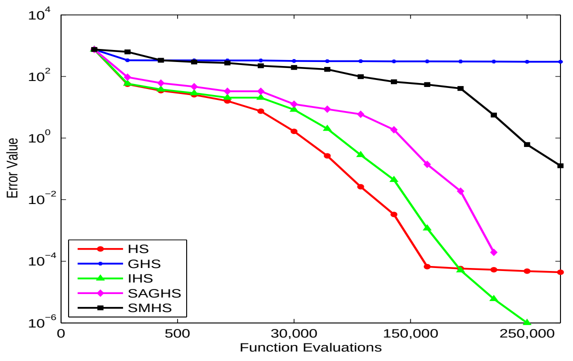
(e) Function 5



(f) Function 6

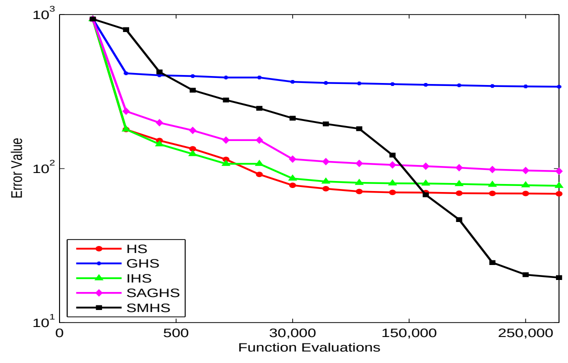


(g) Function 7

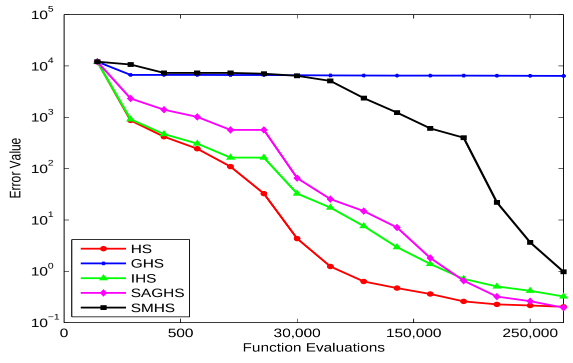


(h) Function 8

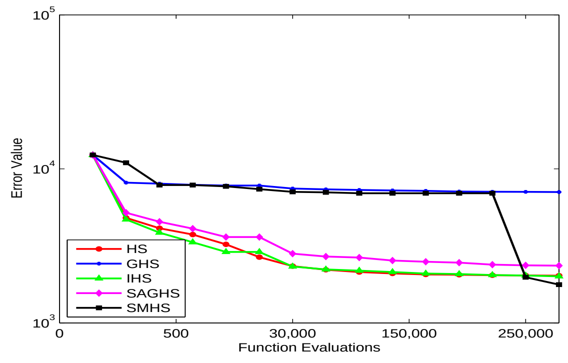
Figure 4.1: Convergence graphs of function number 1 through 8.



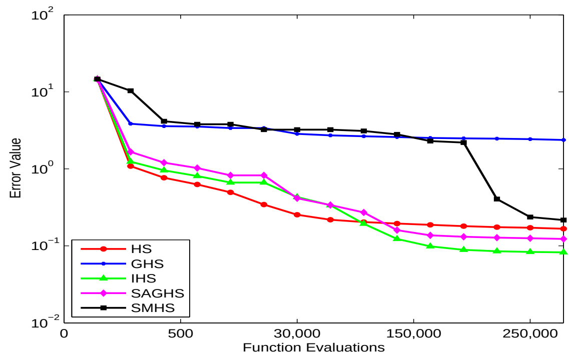
(a) Function 9



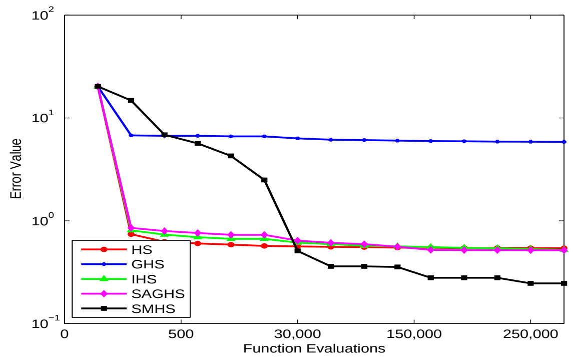
(b) Function 10



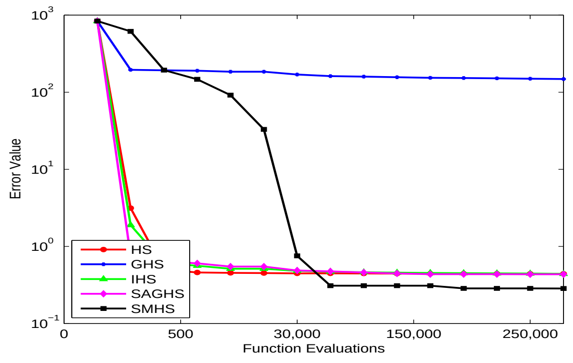
(c) Function 11



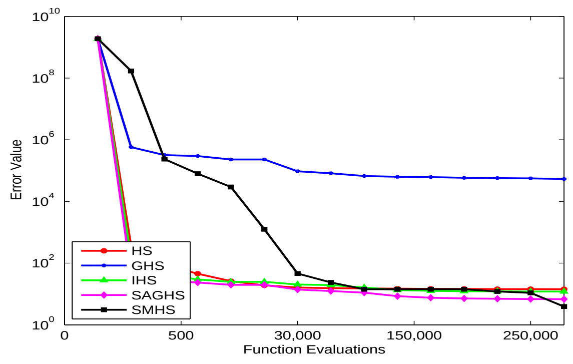
(d) Function 12



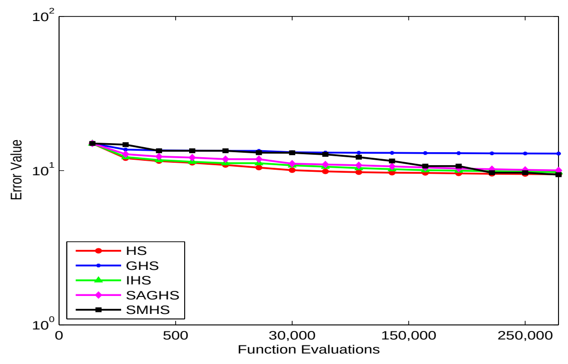
(e) Function 13



(f) Function 14

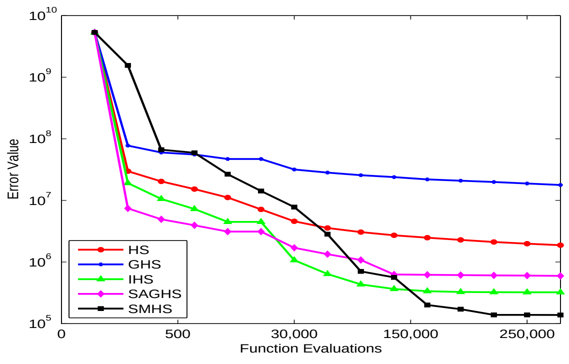


(g) Function 15

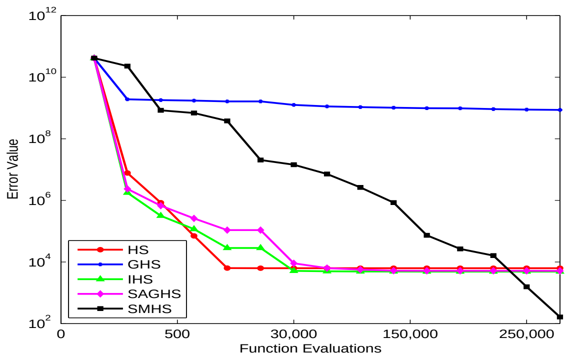


(h) Function 16

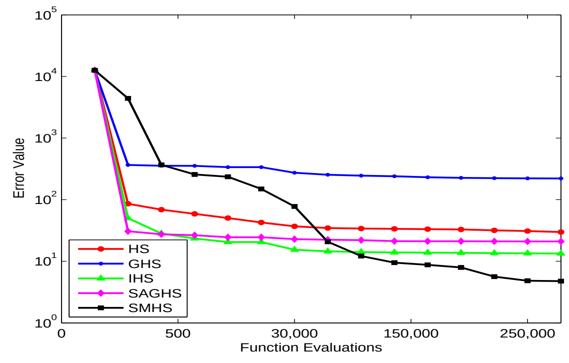
Figure 4.2: Convergence graphs of function number 9 through 16.



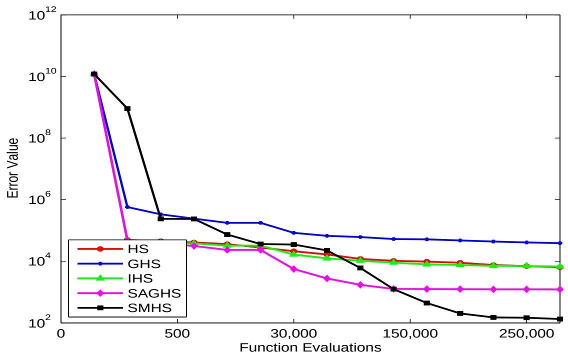
(a) Function 17



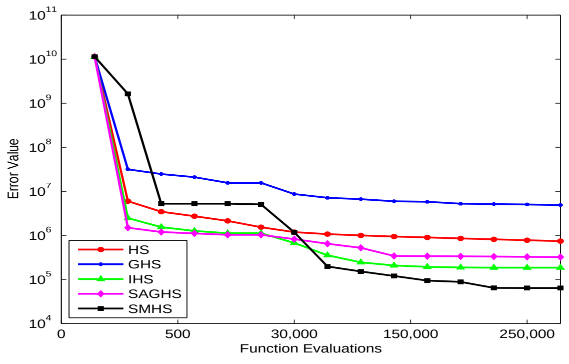
(b) Function 18



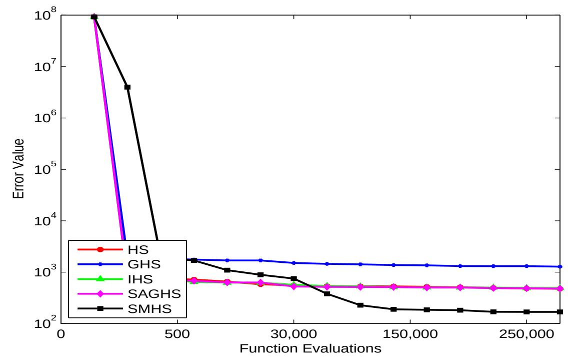
(c) Function 19



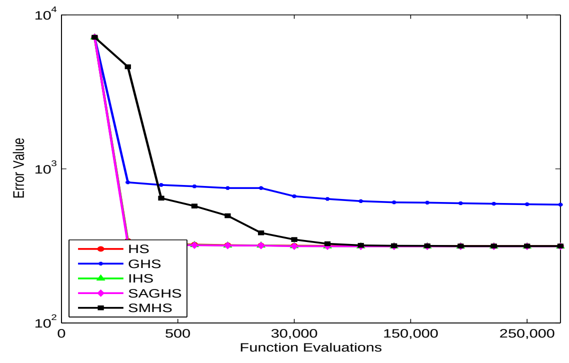
(d) Function 20



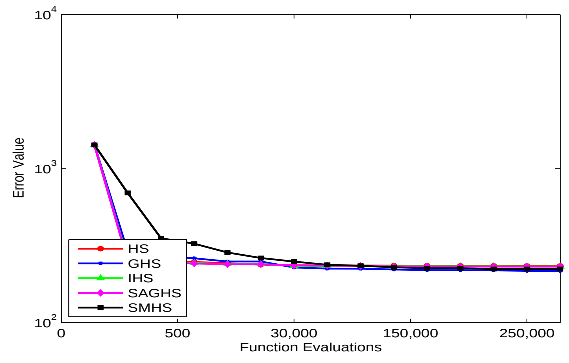
(e) Function 21



(f) Function 22

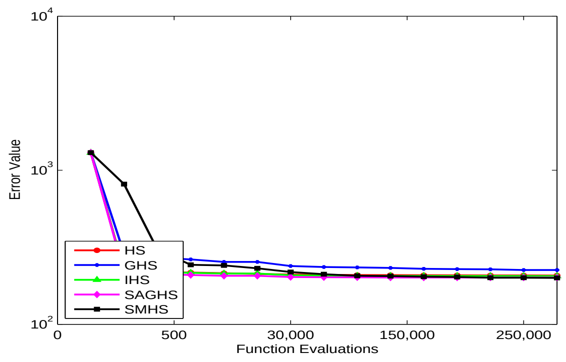


(g) Function 23

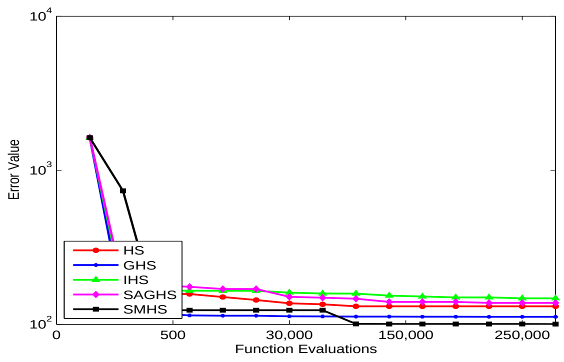


(h) Function 24

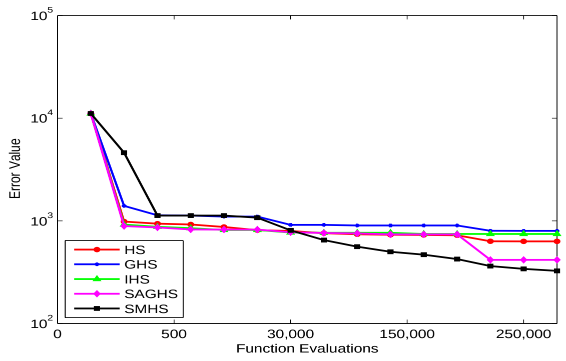
Figure 4.3: Convergence graphs of function number 17 through 24.



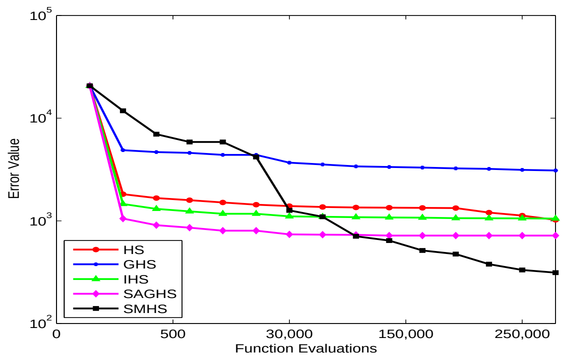
(a) Function 25



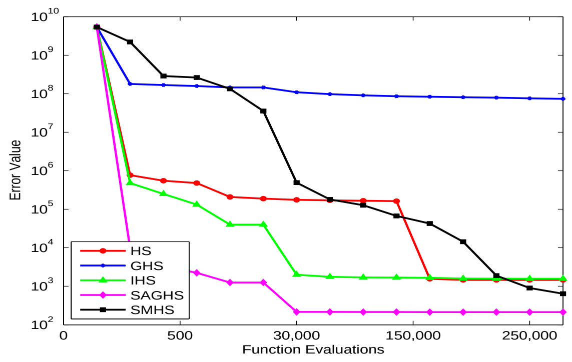
(b) Function 26



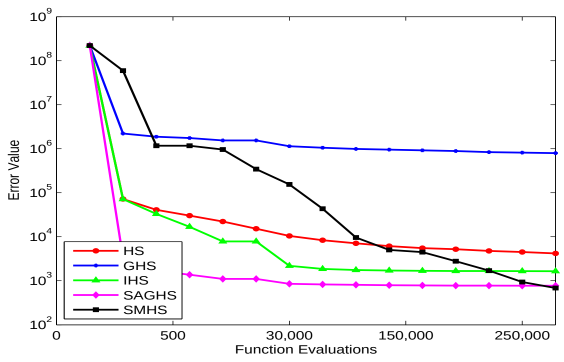
(c) Function 27



(d) Function 28



(e) Function 29



(f) Function 30

Figure 4.4: Convergence graphs of function number 25 through 30.

Table 4.6: Mean and Standard Deviation of error value obtained by SMHS, CL-PSO, DNL-PSO and HCL-PSO on IEEE CEC 2014 benchmark problems.

	SMHS		CL-PSO			DNL-PSO			HCL-PSO		
Function	Mean	Std Dev	Mean	Std Dev	W	Mean	Std Dev	W	Mean	Std Dev	W
1	2.13E+05	6.43E+04	2.62E+06	1.21E+06	+	1.47E+06	1.16E+06	+	2.61E+05	1.95E+05	=
2	1.17E+04	2.70E+03	**2.34E+02**	1.01E+03	-	**1.78E+00**	4.18E+00	-	**8.73E+01**	1.92E+02	-
3	7.37E+01	6.41E+01	**5.86E+01**	8.21E+01	-	9.40E+01	1.35E+02	=	1.45E+02	1.63E+02	+
4	5.50E+00	1.58E+01	5.20E+01	3.44E+01	+	7.31E+00	1.01E+01	+	4.04E+01	3.26E+01	+
5	2.00E+01	7.18E-03	2.00E+01	7.44E-03	+	2.09E+01	5.72E-02	+	2.02E+01	4.98E-02	+
6	4.42E+00	1.15E+00	9.72E+00	1.98E+00	+	4.59E+00	2.05E+00	=	**3.82E+00**	1.70E+00	-
7	1.87E-02	1.43E-02	**1.45E-04**	1.03E-03	-	**1.06E-02**	1.10E-02	-	**3.16E-04**	7.79E-04	-
8	1.92E+00	1.33E+00	**7.22E-01**	1.12E+00	-	4.36E+01	1.16E+01	+	**2.47E-13**	8.33E-14	-
9	2.69E+01	5.68E+00	4.52E+01	1.00E+01	+	5.07E+01	1.39E+01	+	4.20E+01	9.10E+00	+
10	6.99E+00	2.36E+00	3.24E+01	7.35E+01	=	1.50E+03	4.26E+02	+	**4.96E+00**	2.32E+01	-
11	1.81E+03	8.58E+02	1.85E+03	3.22E+02	+	2.98E+03	7.52E+02	+	1.84E+03	3.01E+02	+
12	6.25E-01	1.91E-01	**1.32E-01**	3.35E-02	-	2.14E+00	3.91E-01	+	**1.80E-01**	5.07E-02	-
13	1.87E-01	3.96E-02	3.00E-01	3.93E-02	+	3.32E-01	8.99E-02	+	2.29E-01	5.60E-02	+
14	2.09E-01	3.71E-02	2.40E-01	3.30E-02	+	4.58E-01	2.39E-01	+	2.19E-01	2.90E-02	=
15	3.90E+00	1.16E+00	4.68E+00	1.03E+00	+	**3.71E+00**	1.15E+00	=	3.95E+00	1.19E+00	=
16	9.42E+00	7.19E-01	9.58E+00	5.33E-01	=	1.18E+01	7.26E-01	+	9.53E+00	6.81E-01	=
17	2.93E+04	7.46E+03	5.51E+05	3.82E+05	+	1.81E+05	1.24E+05	+	7.99E+04	7.15E+04	+
18	1.65E+02	1.01E+02	2.46E+02	2.55E+02	=	4.21E+04	7.76E+04	+	**1.23E+02**	7.32E+01	=
19	4.40E+00	1.25E+00	6.61E+00	8.84E-01	+	5.67E+00	1.91E+00	+	5.61E+00	1.28E+00	+
20	1.34E+02	5.37E+01	1.78E+03	2.09E+03	+	1.68E+03	1.26E+03	+	9.33E+02	8.87E+02	+
21	2.76E+04	5.85E+03	1.56E+05	1.26E+05	+	1.55E+05	1.89E+05	+	**2.51E+04**	1.73E+04	=
22	2.01E+02	8.84E+01	2.43E+02	1.06E+02	+	3.88E+02	1.93E+02	+	2.13E+02	8.40E+01	=
23	3.14E+02	9.64E-06	3.15E+02	1.71E-13	+	**3.14E+02**	1.71E-13	-	3.15E+02	3.46E-12	+
24	2.24E+02	8.76E-01	**2.23E+02**	5.96E+00	=	2.35E+02	8.78E+00	+	2.25E+02	1.15E+00	+
25	2.00E+02	2.09E-03	2.06E+02	1.02E+00	+	2.01E+02	2.63E-01	+	2.05E+02	1.53E+00	+
26	1.00E+02	4.56E-02	1.04E+02	1.94E+01	+	1.00E+02	1.05E-01	+	1.00E+02	5.93E-02	=
27	3.26E+02	1.23E+01	4.37E+02	8.14E+01	+	4.13E+02	4.68E+01	+	4.02E+02	1.46E+00	+
28	3.20E+02	9.52E+01	9.46E+02	1.42E+02	+	4.04E+02	2.19E+01	+	8.83E+02	5.38E+01	+
29	6.40E+02	1.00E+02	1.08E+03	1.79E+02	+	**2.06E+02**	1.71E+00	-	9.11E+02	9.63E+01	+
30	6.83E+02	3.66E+02	2.38E+03	5.69E+02	+	9.24E+02	2.33E+02	+	1.82E+03	3.55E+02	+

Table 4.7: Mean and Standard Deviation of error value obtained by SMHS, SL-PSO and SR-PSO on IEEE CEC 2014 benchmark problems.

	SMHS		SL-PSO			SR-PSO		
Function	Mean	Std Dev	Mean	Std Dev	W	Mean	Std Dev	W
1	2.13E+05	6.43E+04	3.67E+05	2.36E+05	+	1.44E+06	1.89E+06	+
2	1.17E+04	2.70E+03	**1.03E+04**	1.02E+04	=	**2.18E+02**	4.08E+02	-
3	7.37E+01	6.41E+01	6.86E+03	5.96E+03	+	**1.75E+01**	2.66E+01	-
4	5.50E+00	1.58E+01	3.26E+01	2.98E+01	+	1.03E+02	3.70E+01	+
5	2.00E+01	7.18E-03	2.09E+01	4.22E-02	+	2.09E+01	4.30E-02	+
6	4.42E+00	1.15E+00	**9.03E-01**	1.01E+00	-	**3.16E+00**	1.62E+00	-
7	1.87E-02	1.43E-02	**9.18E-04**	3.20E-03	-	**1.06E-02**	1.41E-02	-
8	1.92E+00	1.33E+00	1.64E+01	4.26E+00	+	3.51E+01	9.10E+00	+
9	2.69E+01	5.68E+00	2.47E+01	2.06E+01	+	4.25E+01	1.15E+01	+
10	6.99E+00	2.36E+00	3.79E+02	2.23E+02	+	8.02E+02	3.24E+02	+
11	1.81E+03	8.58E+02	**8.81E+02**	4.82E+02	-	2.12E+03	5.49E+02	+
12	6.25E-01	1.91E-01	2.30E+00	5.34E-01	+	1.82E+00	7.15E-01	+
13	1.87E-01	3.96E-02	**1.78E-01**	3.21E-02	=	2.06E-01	4.06E-02	+
14	2.09E-01	3.71E-02	4.01E-01	7.93E-02	+	**2.01E-01**	3.69E-02	=
15	3.90E+00	1.16E+00	5.12E+00	3.87E+00	=	**3.83E+00**	9.48E-01	=
16	9.42E+00	7.19E-01	1.21E+01	2.51E-01	+	1.05E+01	6.60E-01	+
17	2.93E+04	7.46E+03	1.11E+05	7.38E+04	+	1.52E+05	1.17E+05	+
18	1.65E+02	1.01E+02	2.05E+03	3.47E+03	+	2.71E+03	3.06E+03	+
19	4.40E+00	1.25E+00	7.16E+00	1.26E+00	+	6.35E+00	1.57E+00	+
20	1.34E+02	5.37E+01	2.13E+04	1.20E+04	+	4.13E+02	2.84E+02	+
21	2.76E+04	5.85E+03	7.90E+04	6.01E+04	+	5.22E+04	4.17E+04	+
22	2.01E+02	8.84E+01	**1.69E+02**	9.92E+01	=	3.10E+02	1.21E+02	+
23	3.14E+02	9.64E-06	3.15E+02	1.71E-13	+	3.15E+02	6.47E-02	+
24	2.24E+02	8.76E-01	2.30E+02	5.76E+00	+	**2.07E+02**	1.02E+01	-
25	2.00E+02	2.09E-03	2.06E+02	1.80E+00	+	2.06E+02	1.06E+00	+
26	1.00E+02	4.56E-02	1.12E+02	3.22E+01	=	1.32E+02	4.48E+01	+
27	3.26E+02	1.23E+01	3.66E+02	6.09E+01	+	4.40E+02	7.96E+01	+
28	3.20E+02	9.52E+01	9.16E+02	9.19E+01	+	1.13E+03	3.06E+02	+
29	6.40E+02	1.00E+02	1.68E+03	5.53E+02	+	5.27E+05	2.61E+06	+
30	6.83E+02	3.66E+02	3.03E+03	8.60E+02	+	2.22E+03	7.96E+02	+

Table 4.8: Rank obtained by SMHS, CL-PSO, DNL-PSO, HCL-PSO, SL-PSO and SR-PSO on IEEE CEC 2014 benchmark problems.

Function	SMHS	CL-PSO	DNL-PSO	HCL-PSO	SL-PSO	SR-PSO
1	1	6	5	2	3	4
2	6	4	1	2	5	3
3	3	2	4	5	6	1
4	1	5	2	4	3	6
5	1	2	4	3	5	6
6	4	6	5	3	1	2
7	6	1	5	2	3	4
8	3	2	6	1	4	5
9	2	5	6	3	1	4
10	2	3	6	1	4	5
11	2	4	6	3	1	5
12	3	1	5	2	6	4
13	2	5	6	4	1	3
14	2	4	6	3	5	1
15	3	5	1	4	6	2
16	1	3	5	2	6	4
17	1	6	5	2	3	4
18	2	3	6	1	4	5
19	1	5	3	2	6	4
20	1	5	4	3	6	2
21	2	6	5	1	4	3
22	1	4	6	3	2	5
23	2	3	1	5	3	6
24	3	2	6	4	5	1
25	1	6	2	3	4	5
26	2	4	3	1	5	6
27	1	5	4	3	2	6
28	1	5	2	3	4	6
29	2	4	1	3	5	6
30	1	5	2	3	6	4
Avg. Rank Unimodal	3.33	4.00	3.33	3.00	4.67	**2.67**
Avg. Rank Multimodal	**1.96**	4.04	4.19	2.67	3.89	4.22
Avg. Rank Composition	**1.63**	4.25	2.63	3.13	4.25	5.00

Table 4.9: Mean and Standard Deviation of error value obtained by SMHS, Standard-DE, jDE and JADE on IEEE CEC 2014 benchmark problems.

FUNCTION	SMHS		derand1bin			jDE			JADE		
	Mean	Std Dev	Mean	Std Dev	W	Mean	Std Dev	W	Mean	Std Dev	W
1	2.13E+05	6.43E+04	8.81E+07	1.88E+07	+	**1.07E+05**	1.08E+05	-	**4.18E+02**	1.07E+03	-
2	1.17E+04	2.70E+03	**1.72E+03**	4.93E+02	-	**2.23E-15**	7.64E-15	-	**2.01E-14**	1.30E-14	-
3	7.37E+01	6.41E+01	**2.55E+01**	5.04E+00	-	**1.56E-14**	2.54E-14	-	**4.14E-04**	2.01E-03	-
4	5.50E+00	1.58E+01	1.23E+02	5.77E+00	+	**3.78E+00**	1.25E+01	-	**8.36E-14**	4.26E-14	-
5	2.00E+01	7.18E-03	2.09E+01	5.77E-02	+	2.03E+01	3.50E-02	+	2.03E+01	2.78E-02	+
6	4.42E+00	1.15E+00	3.03E+01	1.08E+00	+	1.05E+01	4.36E+00	+	8.92E+00	2.46E+00	+
7	1.87E-02	1.43E-02	4.34E-02	7.93E-02	=	**6.46E-14**	5.63E-14	-	**1.11E-14**	3.38E-14	-
8	1.92E+00	1.33E+00	1.20E+02	6.53E+00	+	**0.00E+00**	0.00E+00	-	**0.00E+00**	0.00E+00	-
9	2.69E+01	5.68E+00	1.95E+02	8.96E+00	+	4.61E+01	7.29E+00	+	**2.52E+01**	3.92E+00	=
10	6.99E+00	2.36E+00	3.91E+03	2.38E+02	+	**1.22E-03**	4.90E-03	-	**6.53E-03**	1.13E-02	-
11	1.81E+03	8.58E+02	6.55E+03	2.48E+02	+	2.48E+03	2.75E+02	+	**1.58E+03**	2.63E+02	=
12	6.25E-01	1.91E-01	2.08E+00	2.05E-01	+	**4.42E-01**	5.58E-02	-	**2.63E-01**	3.62E-02	-
13	1.87E-01	3.96E-02	4.88E-01	4.56E-02	+	2.96E-01	4.01E-02	+	2.25E-01	3.53E-02	+
14	2.09E-01	3.71E-02	2.94E-01	3.77E-02	+	2.87E-01	2.77E-02	+	2.35E-01	4.13E-02	+
15	3.90E+00	1.16E+00	1.90E+01	1.13E+00	+	5.72E+00	5.86E-01	+	**3.13E+00**	4.03E-01	-
16	9.52E+00	7.19E-01	1.25E+01	2.23E-01	+	9.98E+00	2.98E-01	+	**9.41E+00**	3.36E-01	=
17	2.93E+04	7.46E+03	2.40E+06	5.69E+05	+	**2.39E+03**	2.43E+03	-	**1.15E+03**	3.82E+02	-
18	1.65E+02	1.01E+02	2.88E+04	1.53E+04	+	**1.75E+01**	7.11E+00	-	2.44E+02	1.16E+03	+
19	4.40E+00	1.25E+00	1.06E+01	5.81E-01	+	4.65E+00	6.81E-01	+	**4.42E+00**	7.04E-01	-
20	1.34E+02	5.37E+01	4.58E+02	8.33E+01	+	**1.12E+01**	3.16E+00	-	3.45E+03	2.55E+03	+
21	2.76E+04	5.85E+03	1.89E+05	6.88E+04	+	**3.00E+02**	2.37E+02	-	**1.68E+04**	4.33E+04	-
22	2.01E+02	8.84E+01	2.10E+02	7.23E+01	=	**1.29E+02**	5.92E+01	-	**1.69E+02**	5.60E+01	=
23	3.15E+02	9.64E-06	3.15E+02	8.08E-05	+	3.15E+02	1.71E-13	+	3.15E+02	1.71E-13	+
24	2.24E+02	8.76E-01	**2.09E+02**	3.63E+00	-	2.24E+02	1.32E+00	+	2.26E+02	3.47E+00	+
25	2.00E+02	2.09E-03	2.23E+02	2.80E+00	+	2.03E+02	6.53E-01	+	2.04E+02	1.29E+00	+
26	1.00E+02	4.56E-02	1.00E+02	4.77E-02	+	1.00E+02	3.69E-02	+	1.02E+02	1.38E+01	=
27	3.26E+02	1.23E+01	3.89E+02	3.73E+01	+	3.68E+02	6.54E+01	+	3.48E+02	4.83E+01	+
28	3.20E+02	9.52E+01	9.77E+02	2.78E+01	+	7.88E+02	2.35E+01	+	7.95E+02	3.94E+01	+
29	6.40E+02	1.00E+02	1.18E+04	3.05E+03	+	8.04E+02	7.61E+01	+	7.34E+02	1.18E+02	+
30	6.83E+02	3.66E+02	5.59E+03	8.73E+02	+	1.46E+03	6.48E+02	+	1.51E+03	6.65E+02	+

Table 4.10: Mean and Standard Deviation of error value obtained by SMHS, SADE and EPSDE on IEEE CEC 2014 benchmark problems.

FUNCTION	SMHS		SADE			EPSDE		
	Mean	Std Dev	Mean	Std Dev	W	Mean	Std Dev	W
1	2.13E+05	6.43E+04	3.78E+05	2.57E+05	+	**2.39E+04**	8.41E+04	-
2	1.17E+04	2.70E+03	**1.73E-14**	5.46E-14	-	**5.68E-14**	4.61E-14	-
3	7.37E+01	6.41E+01	**2.15E+01**	8.74E+01	-	**2.37E-11**	5.30E-11	-
4	5.50E+00	1.58E+01	3.17E+01	3.95E+01	+	**3.72E+00**	2.29E+00	-
5	2.00E+01	7.18E-03	2.05E+01	4.22E-02	+	2.03E+01	4.93E-02	+
6	4.42E+00	1.15E+00	4.96E+00	1.90E+00	=	1.89E+01	2.28E+00	+
7	1.87E-02	1.43E-02	**8.53E-03**	1.32E-02	-	**2.22E-03**	6.13E-03	-
8	1.92E+00	1.33E+00	**1.17E-01**	3.21E-01	-	**1.95E-02**	1.38E-01	-
9	2.69E+01	5.68E+00	3.71E+01	8.14E+00	+	4.31E+01	7.53E+00	+
10	6.99E+00	2.36E+00	**4.04E-01**	7.22E-01	-	**2.45E-01**	3.32E-01	-
11	1.81E+03	8.58E+02	3.20E+03	7.20E+02	+	3.58E+03	6.25E+02	+
12	6.25E-01	1.91E-01	8.10E-01	1.09E-01	+	**5.12E-01**	5.81E-02	-
13	1.87E-01	3.96E-02	2.68E-01	4.70E-02	+	2.65E-01	4.38E-02	+
14	2.09E-01	3.71E-02	2.34E-01	3.83E-02	+	2.86E-01	8.40E-02	+
15	3.90E+00	1.16E+00	4.07E+00	1.18E+00	=	5.54E+00	7.22E-01	+
16	9.52E+00	7.19E-01	1.10E+01	2.56E-01	+	1.13E+01	3.95E-01	+
17	2.93E+04	7.46E+03	**1.27E+04**	1.15E+04	-	4.86E+04	5.96E+04	=
18	1.65E+02	1.01E+02	4.02E+02	6.86E+02	+	1.03E+03	4.36E+03	=
19	4.40E+00	1.25E+00	**4.11E+00**	8.31E-01	-	1.30E+01	1.23E+00	+
20	1.34E+02	5.37E+01	1.69E+02	4.04E+02	=	**5.70E+01**	6.85E+01	-
21	2.76E+04	5.85E+03	**4.01E+03**	7.75E+03	-	**1.32E+04**	2.39E+04	-
22	2.01E+02	8.84E+01	**1.46E+02**	6.42E+01	=	2.64E+02	8.63E+01	+
23	3.15E+02	9.64E-06	3.15E+02	1.71E-13	+	**3.14E+02**	1.71E-13	-
24	2.24E+02	8.76E-01	2.27E+02	3.23E+00	+	2.28E+02	5.50E+00	+
25	2.00E+02	2.09E-03	2.09E+02	2.84E+00	+	2.00E+02	3.72E-01	+
26	1.00E+02	4.56E-02	1.12E+02	3.21E+01	+	1.00E+02	3.90E-02	=
27	3.26E+02	1.23E+01	4.14E+02	3.79E+01	+	8.57E+02	8.96E+01	+
28	3.20E+02	9.52E+01	8.94E+02	3.67E+01	+	3.93E+02	1.23E+01	+
29	6.40E+02	1.00E+02	1.12E+03	2.42E+02	+	**2.14E+02**	1.34E+00	-
30	6.83E+02	3.66E+02	1.61E+03	5.23E+02	+	**5.58E+02**	1.27E+02	-

Table 4.11: Mean and Standard Deviation of error value obtained by SMHS, CoDE, derand1bin-sps and MPEDE on IEEE CEC 2014 benchmark problems.

FUNCTION	SMHS		CoDE			derand1bin-sps			MPEDE		
	Mean	Std Dev	Mean	Std Dev	W	Mean	Std Dev	W	Mean	Std Dev	W
1	2.13E+05	6.43E+04	**2.97E+04**	2.73E+04	-	3.28E+07	9.55E+06	+	**1.10E-03**	7.79E-03	-
2	1.17E+04	2.70E+03	**0.00E+00**	0.00E+00	-	**1.55E+03**	4.18E+02	-	**1.67E-15**	6.69E-15	-
3	7.37E+01	6.41E+01	**0.00E+00**	0.00E+00	-	**1.90E+01**	2.97E+00	-	**4.01E-14**	2.59E-14	-
4	5.50E+00	1.58E+01	**1.27E+00**	9.01E+00	-	8.97E+01	6.70E+00	+	**1.24E+00**	8.79E+00	-
5	2.00E+01	7.18E-03	2.00E+01	7.43E-02	=	2.09E+01	5.51E-02	+	2.04E+01	4.88E-02	+
6	4.42E+00	1.15E+00	**2.13E+00**	1.90E+00	-	4.59E+00	1.40E+00	=	**6.01E-01**	7.83E-01	-
7	1.87E-02	1.43E-02	**2.90E-04**	1.44E-03	-	**1.05E-03**	7.41E-04	-	**3.38E-04**	1.69E-03	-
8	1.92E+00	1.33E+00	**3.90E-02**	1.93E-01	-	8.65E+01	1.13E+01	+	**8.92E-15**	3.06E-14	-
9	2.69E+01	5.68E+00	3.98E+01	1.14E+01	+	1.78E+02	1.16E+01	+	2.81E+01	7.93E+00	=
10	6.99E+00	2.36E+00	**4.51E-01**	3.81E-01	-	2.42E+03	2.66E+02	+	**1.19E+00**	5.56E-01	-
11	1.81E+03	8.58E+02	**1.74E+03**	4.51E+02	=	6.19E+03	2.71E+02	+	2.36E+03	3.84E+02	+
12	6.25E-01	1.91E-01	**5.58E-02**	2.89E-02	-	9.36E-01	3.52E-01	+	**4.98E-01**	8.12E-02	-
13	1.87E-01	3.96E-02	2.46E-01	5.43E-02	+	4.15E-01	5.06E-02	+	2.11E-01	2.98E-02	+
14	2.09E-01	3.71E-02	2.47E-01	3.70E-02	+	2.68E-01	2.98E-02	+	2.40E-01	2.86E-02	+
15	3.90E+00	1.16E+00	3.10E+00	7.50E-01	+	1.73E+01	9.07E-01	+	3.91E+00	6.84E-01	=
16	9.52E+00	7.19E-01	**9.23E+00**	7.47E-01	=	1.20E+01	2.59E-01	+	1.00E+01	4.37E-01	+
17	2.93E+04	7.46E+03	**1.62E+03**	1.56E+03	-	7.71E+05	2.91E+05	+	**1.96E+02**	1.43E+02	-
18	1.65E+02	1.01E+02	**1.60E+01**	5.61E+00	-	1.51E+03	1.66E+03	+	**1.45E+01**	6.25E+00	-
19	4.40E+00	1.25E+00	**2.68E+00**	4.38E-01	-	5.77E+00	2.43E-01	+	**3.73E+00**	5.26E-01	-
20	1.34E+02	5.37E+01	**1.14E+01**	5.52E+00	-	2.01E+02	2.85E+01	+	**9.33E+00**	3.17E+00	-
21	2.76E+04	5.85E+03	**2.11E+02**	1.75E+02	-	3.93E+04	1.74E+04	+	**1.20E+02**	8.92E+01	-
22	2.01E+02	8.84E+01	**1.88E+02**	1.06E+02	=	**1.00E+02**	8.53E+01	-	**1.06E+02**	7.17E+01	-
23	3.15E+02	9.64E-06	3.15E+02	2.27E-13	+	3.15E+02	8.31E-05	+	3.15E+02	1.71E-13	+
24	2.24E+02	8.76E-01	2.25E+02	2.20E+00	+	**2.04E+02**	4.93E-01	-	2.25E+02	2.18E+00	+
25	2.00E+02	2.09E-03	2.03E+02	5.25E-01	+	2.10E+02	1.69E+00	+	2.00E+02	4.11E-03	+
26	1.00E+02	4.56E-02	1.00E+02	4.58E-02	=	1.00E+02	3.87E-02	+	1.00E+02	2.99E-02	+
27	3.26E+02	1.23E+01	3.80E+02	3.87E+01	+	3.33E+02	3.74E+00	+	3.65E+02	4.68E+01	+
28	3.20E+02	9.52E+01	8.28E+02	3.74E+01	+	7.99E+02	1.87E+01	+	8.32E+02	4.23E+01	+
29	6.40E+02	1.00E+02	7.86E+02	1.25E+02	+	2.46E+03	2.82E+02	+	**6.38E+02**	1.90E+02	-
30	6.83E+02	3.66E+02	9.35E+02	3.94E+02	+	2.55E+03	5.23E+02	+	**6.80E+02**	3.15E+02	-

Table 4.12: Rank obtained by SMHS, Standard-DE,jDE, JADE, SaDE, EPSDE, CoDE, derand1bin-sps and MPEDE on IEEE CEC 2014 benchmark problems.

Function	SMHS	Standard-DE	jDE	JADE	SaDE	EPSDE	CoDE	derand1bin-sps	MPEDE
1	6	9	5	2	7	3	4	8	1
2	9	8	3	5	4	6	1	7	2
3	9	8	2	5	7	4	1	6	3
4	6	9	5	1	7	4	3	8	2
5	1	9	4	3	7	5	2	8	6
6	3	9	7	6	5	8	2	4	1
7	8	9	2	1	7	6	3	5	4
8	7	9	1	1	6	4	5	8	3
9	2	9	7	1	4	6	5	8	3
10	7	9	1	2	4	3	5	8	6
11	3	9	5	1	6	7	2	8	4
12	6	9	3	2	7	5	1	8	4
13	1	9	7	3	6	5	4	8	2
14	1	9	8	3	2	7	5	6	4
15	3	9	7	2	5	6	1	8	4
16	3	9	4	2	6	7	1	8	5
17	6	9	4	2	5	7	3	8	1
18	4	9	3	5	6	7	2	8	1
19	4	8	6	5	3	9	1	7	2
20	5	8	2	9	6	4	3	7	1
21	7	9	3	6	4	5	2	8	1
22	2	8	4	6	5	9	7	1	3
23	2	9	3	3	3	1	7	8	3
24	3	2	4	7	8	9	5	1	6
25	1	9	5	6	7	3	4	8	2
26	3	7	5	8	9	4	2	6	1
27	1	7	5	3	8	9	6	2	4
28	1	9	3	4	8	2	6	5	7
29	2	9	6	4	7	1	5	8	3
30	2	9	5	6	7	1	4	8	3
Avg. Rank Unimodal	8.00	8.33	3.33	4.00	6.00	4.33	**2.00**	7.00	**2.00**
Avg. Rank Multimodal	3.48	8.48	4.41	3.78	5.85	5.33	3.56	6.67	**3.19**
Avg. Rank Composition	**1.88**	7.63	4.50	5.13	7.13	3.75	4.88	5.75	3.63

Table 4.13: Mean and Standard Deviation of error value obtained by SMHS, GL-25, and CMA-ES on IEEE CEC 2014 benchmark problems.

FUNCTION	SMHS		GL-25			CMA-ES		
	Mean	Std Dev	Mean	Std Dev	W	Mean	Std Dev	W
1	2.13E+05	6.43E+04	9.35E+05	7.80E+05	+	**2.70E-14**	1.13E-14	-
2	1.17E+04	2.70E+03	**1.77E+03**	2.65E+03	-	**5.29E-14**	2.11E-14	-
3	7.37E+01	6.41E+01	**2.09E-01**	6.04E-01	-	**1.07E-13**	4.31E-14	-
4	5.50E+00	1.58E+01	9.59E+01	1.42E+01	+	**1.07E-13**	5.11E-14	-
5	2.00E+01	7.18E-03	2.10E+01	3.63E-02	+	2.00E+01	1.57E-02	+
6	4.42E+00	1.15E+00	4.91E+00	2.99E+00	=	4.20E+01	1.06E+01	+
7	1.87E-02	1.43E-02	**1.04E-10**	4.88E-10	-	**2.66E-03**	4.65E-03	-
8	1.92E+00	1.33E+00	2.44E+01	5.95E+00	+	4.29E+02	9.03E+01	+
9	2.69E+01	5.68E+00	6.04E+01	5.89E+01	+	6.37E+02	1.59E+02	+
10	6.99E+00	2.36E+00	1.00E+03	6.15E+02	+	5.15E+03	8.13E+02	+
11	1.81E+03	8.58E+02	5.96E+03	1.40E+03	+	5.14E+03	7.61E+02	+
12	6.25E-01	1.91E-01	2.49E+00	2.80E-01	+	**2.82E-01**	2.61E-01	-
13	1.87E-01	3.96E-02	2.72E-01	3.89E-02	+	2.35E-01	5.87E-02	+
14	2.09E-01	3.71E-02	2.58E-01	3.39E-02	+	3.60E-01	7.16E-02	+
15	3.90E+00	1.16E+00	1.21E+01	4.87E+00	+	**3.52E+00**	1.06E+00	=
16	9.52E+00	7.19E-01	1.16E+01	7.00E-01	+	1.43E+01	**3.67E-01**	+
17	2.93E+04	7.46E+03	1.57E+05	8.01E+04	+	**1.65E+03**	3.68E+02	-
18	1.65E+02	1.01E+02	1.65E+02	1.65E+02	=	**1.35E+02**	4.10E+01	=
19	4.40E+00	1.25E+00	5.11E+00	6.67E-01	+	9.88E+00	1.95E+00	+
20	1.34E+02	5.37E+01	1.98E+02	1.60E+02	+	2.63E+02	1.11E+02	+
21	2.76E+04	5.85E+03	6.02E+04	3.23E+04	+	1.10E+03	3.28E+02	+
22	2.01E+02	8.84E+01	**1.59E+02**	3.04E+01	=	3.10E+02	1.79E+02	+
23	3.15E+02	9.64E-06	3.15E+02	1.71E-13	+	3.15E+02	1.71E-13	+
24	2.24E+02	8.76E-01	**2.22E+02**	4.95E-01	-	2.98E+02	3.80E+02	+
25	2.00E+02	2.09E-03	2.08E+02	1.74E+00	+	2.04E+02	2.99E+00	+
26	1.00E+02	4.56E-02	1.00E+02	4.32E-02	=	1.02E+02	1.38E+01	+
27	3.26E+02	1.23E+01	**3.03E+02**	9.65E-01	-	4.17E+02	1.75E+02	+
28	3.20E+02	9.52E+01	8.81E+02	3.21E+01	+	3.98E+03	3.09E+03	+
29	6.40E+02	1.00E+02	9.99E+02	8.73E+01	+	8.01E+02	9.49E+01	+
30	6.83E+02	3.66E+02	1.20E+03	3.01E+02	+	2.55E+03	6.60E+02	+

Table 4.14: Rank obtained by SMHS, GL-25 and CMA-ES on IEEE CEC 2014 benchmark problems.

Function	SMHS	GL-25	CMA-ES
1	2	3	1
2	3	2	1
3	3	2	1
4	2	3	1
5	2	3	1
6	1	2	3
7	3	1	2
8	1	2	3
9	1	2	3
10	1	2	3
11	1	3	2
12	2	3	1
13	1	3	2
14	1	2	3
15	2	3	1
16	1	2	3
17	2	3	1
18	2	3	1
19	1	2	3
20	1	2	3
21	2	3	1
22	1	2	3
23	1	2	2
24	2	1	3
25	1	3	2
26	1	2	3
27	2	1	3
28	1	2	3
29	1	3	2
30	1	2	3
Avg. Rank Unimodal	2.67	2.33	**1.00**
Avg. Rank Multimodal	**1.41**	2.30	2.26
Avg. Rank Composition	**1.25**	2.00	2.63

Table 4.15: Time complexity in seconds (Dimension 10).

Algorithm	T_0	T_1	$\hat{T}_2$	$\frac{\hat{T}_2 - T_1}{T_0}$
HS	0.187	0.296	0.5954	1.601069519
GHS	0.187	0.296	0.5936	1.59144385
IHS	0.187	0.296	0.6562	1.926203209
SAGHS	0.187	0.296	0.7718	2.544385027
SMHS	0.187	0.296	0.61	1.679144385
CL-PSO	0.121886	0.603636	9.102916	69.73138835
DNL-PSO	0.121886	0.603636	5.4368482	39.65354676
SR-PSO	0.121886	0.603636	0.7489106	1.191889142
SL-PSO	0.121886	0.603636	1.3932988	6.478699769
HCL-PSO	0.121886	0.603636	4.5066256	32.02163989
derand1bin	0.121886	0.603636	1.4222552	6.716269301
MPEDE	0.121886	0.603636	1.1551446	4.524790378
CODE	0.121886	0.603636	3.0349978	19.94783486
EPSDE	0.121886	0.603636	10.9371618	84.78025204
JADE	0.121886	0.603636	0.904352	2.467190654
jDE	0.121886	0.603636	0.6612136	0.472388954
SaDE	0.121886	0.603636	9.7029524	74.65431961
derand1bin-sps	0.121886	0.603636	1.4530082	6.968578836
CMA-ES	0.121886	0.603636	2.2149718	13.22002363
GL-25	0.121886	0.603636	6.8400168	51.16568597

Chapter 5
Comparison of Proposed HS-SA, TPHS and SMHS Algorithms

In this chapter the algorithms proposed in previous three chapters namely HS-SA, TPHS and SMHS are analyzed and compared on IEEE CEC 2014 benchmark functions and a real life problem called Camera Calibration problem, a highly non linear twelve dimensional optimization problem from the field of computer vision.

The organization of this chapter is as follows. Section 5.1 compares the performance of the proposed algorithms on IEEE CEC 2014 benchmark suite. Section 5.2 introduces the camera calibration problem. Section 5.3 provides description about solving Camera Calibration problem using HS algorithm. Section 5.4 describes numerical experimentation on Camera Calibration problem. Section 5.5 compares the performance of proposed algorithms on Camera Calibration problem and finally the chapter concludes with Section 5.6.

5.1 Comparison on IEEE CEC 2014 benchmark suite

Tables 5.1, 5.2 and 5.3 compare the performance of the three proposed algorithms (HS-SA, TPHS and SMHS) on 30 dimensional IEEE CEC 2014 benchmark functions. The best results are highlighted in bold font. The recorded results are the minimum, maximum, mean, median, and standard deviation of the error value obtained as specified in IEEE 2014 Benchmark suite. The error value is the absolute difference between obtained objective function value by the algorithm and the known function value.

In case of unimodal functions SMHS reports the best mean results with minimum standard deviation on all the three instances and reported the best results in two instances followed by HS-SA on one instance.

In case of simple multimodal functions SMHS, HS-SA and TPHS reports the best mean results in 6, 5 and 2 instances respectively and reported the best results in 5, 7 and 1 instances respectively.

In case of hybrid multimodal functions SMHS and TPHS reports the best mean results in 5, 1 instances respectively (whereas HS-SA failed to produce best mean results).

In case of composition multimodal functions SMHS reported the best mean results in 6 instances followed by TPHS in 2 instances whereas HS-SA failed to produce best mean results. SMHS reported the best results in 5 instances followed by TPHS in 3 instances.

Thus SMHS has outperformed the other two competing algorithms on all categories of problems viz a viz unimodal, simple multimodal, hybrid multimodal and composition multimodal functions.

5.1.1 Algorithm Complexity

The time complexity of all the three competing algorithms is computed as per the requirements laid down in IEEE CEC 2014 Benchmark suite, as explained in Section 2.4.3.1 and is given in Table 5.4. All the three proposed algorithms have slightly higher time complexity than the standard HS and hence the order of the four algorithms in terms of time complexity is:

$$\text{SMHS} > \text{HS-SA} > \text{TPHS} > \text{HS}$$

5.2 Camera Calibration

The problem of camera calibration has been studied extensively in photogrammetry and computer vision community because of its important applications such as vehicle guidance, robotic navigation and 3D-reconstruction. Camera calibration problem deals with finding the geometrical relationship between the 3D scene and its 2D images taken by two cameras. It defines exactly how the scene has been projected by the camera to result in the given image(s). Camera calibration involves:

1. Determination of the orientation and position of the camera with respect to the scene which is specified by extrinsic parameters.

2. Determination of the internal geometric and optimal characteristics of the camera, which is specified by intrinsic parameters.

Given a point in a scene extrinsic parameters transform its 3-D world coordinates into 2-D camera coordinates, which are then transformed to 2-D image coordinates using intrinsic parameters.
The existing methods for camera calibration can be broadly categorized as linear (Abdel-Aziz, 1971; Duane, 1971; Faig, 1975; Gennery, 1979; Okamoto, 1981; Paquette et al., 1990; Wong, 1975) and non-linear (Paquette et al., 1990; Sobel, 1974) approaches. In linear methods lens distortion is not taken into consideration, thus calibration accuracy is very low to meet the requirements of commercial machine vision application. Even though nonlinear approaches provide more precise solution, however they have the limitation of being computationally extortionate and require precise initial estimates. Other standard procedure for camera calibration is ”two-step” method (Weng et al., 1992), in the first step linear approach is used to generates an approximate solution, which is improved by using a non linear iterative process in the second step. The first step (linear approach) is key to the success of this procedure. Approximate solution provided by the linear technique must be precise enough for the subsequent nonlinear techniques to converge correctly . The existing linear techniques are notorious for their lack of accuracy and robustness because of being susceptible to noise in image

coordinates (Wan and Xu, 1996). Haralick et al. (Haralick et al., 1989) showed that when the noise level exceeds a threshold or the number of control points is low these methods become extremely unstable and are error prone. This problem can be alleviated by use of more control points, however fabrication of more control points is an expensive, difficult and time consuming process. In case of applications with small number of control points (close to minimum required), it is not possible for linear procedures to consistently provide good initial solution for the subsequent non linear methods to find the optimal solution.

Another limitation is that virtually all nonlinear techniques use variants of conventional optimization techniques like conjugate gradient, gradient descent or Newton method. They therefore all inherent well known problems associated with these traditional methods of optimization namely sensitivity of getting trapped in local extrema and slow convergence. The problem is more prominent if objective function landscape contains isolated valleys or broken ergodicity. Camera calibration objective function involves 12 parameters and thus leads to complex error surface with the global minimum hidden among numerous local extrema. Therefore the risk of obtaining local rather than global optimum is very high with conventional methods.

To alleviate the problem with the conventional camera calibration techniques many metaheuristc algorithms have been proposed. Hati et.al. (Hati and Sengupta, 2001) has used a binary coded genetic algorithm for solving camera calibration, whereas real coded genetic algorithm has been used in (Ji and Zhang, 2001; Kumar et al., 2008). Even though impressive results have been shown in (Ji and Zhang, 2001) both in terms of parameter values and pixel error, however the authors have not taken lenes distortion into consideration.

In this chapter music inspired Harmony Search (HS) algorithm (Geem et al., 2001) is used for camera calibration. The reason for choosing Harmony Search is: it requires only few free parameters to adjust, converges quickly and has been proven to be robust in solving a variety of non linear optimization problems. The algorithm proposed in this chapter takes lenes distortion into consideration and hence results in 12 dimensional highly non linear search space with a number of local optima.

Researchers have used Artificial neural networks to solve the problem of calibration in (Ahmed et al., 1999; Xing et al., 2007), however the main drawback of employing neural networks to solve this problem is estimation of good initial value of weight vector of network. Particle swarm algorithm has been utilized to solve camera calibration problem in (Wang et al., 2008a; Ze-Tao et al., 2008; Zhang et al., 2012). A hybrid technique based on differential evolution and PSO has been utilized in (Deng et al., 2016) to solve the same problem and recently Biogeography based optimization algorithm has been

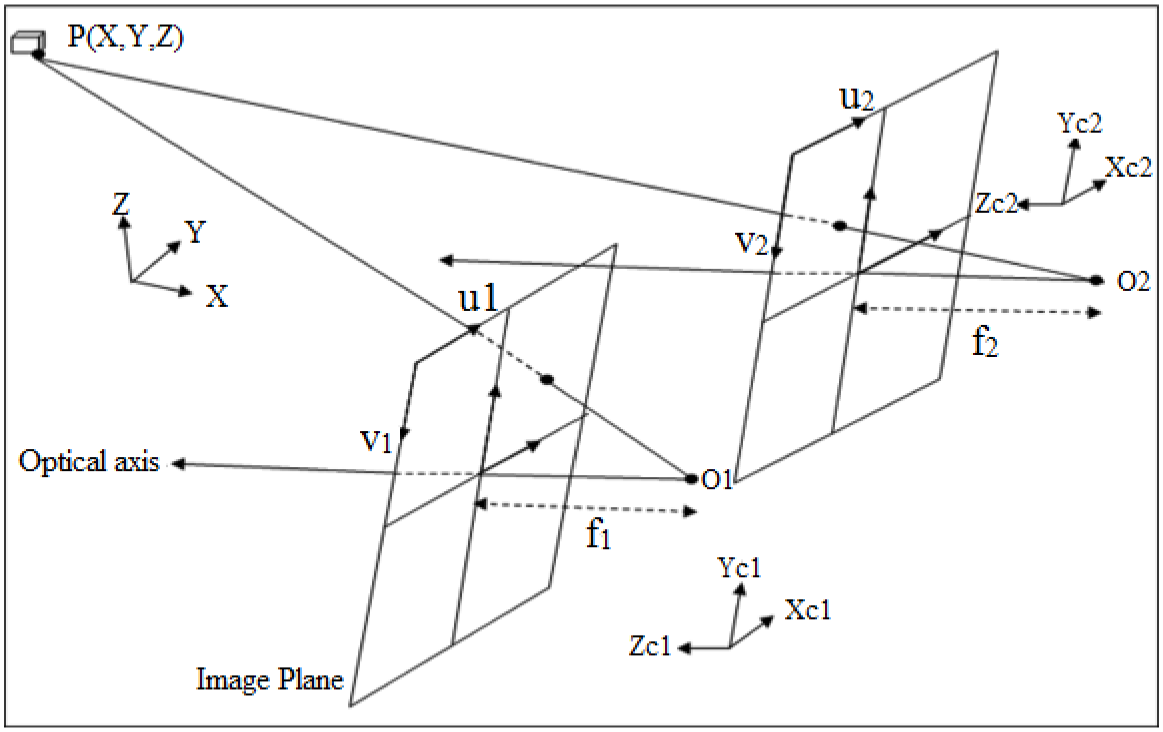


Figure 5.1: Stereo camera calibration model.

used to address the problem of camera calibration in (Garg and Deep, 2016).

5.2.1 Perspective Geometry

In this section the camera model defining the geometry of the image formation is presented. Figure 1 shows the pinhole model for two cameras. The stereo camera system consists of five coordinate frames: a world reference frame (X, Y, Z), two camera frames (X_{ci}, Y_{ci}, Z_{ci}) and two image frames $(u_i, v_i), i = 1, 2$. The origins O_1 and O_2 of the cameras coordinate systems coincide with their corresponding optical centers and their Z coordinate axes are collinear with corresponding optical axes, which are perpendicular to corresponding image planes and intersect them in their principal points. The image plane of each camera is at a distance $f_i,\ i = 1, 2$ (its focal length) apart from the corresponding optical center. The transformation of 3-D world coordinates into 2-D image coordinates by a camera involves following four steps .

1. Equation (5.1) transforms the 3-D world coordinates to 2-D camera coordinates:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = R. \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + T, \tag{5.1}$$

R is a 3×3 rotation matrix used to represents the orientation of the camera relative to the world coordinate system and T is the translation vector used to represent position of the camera relative to world coordinate system. Further, R and T may be expressed as

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \quad T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$

The elements r_{ij} of the matrix R can further be expressed in terms of swing, tilt and pan angle (α, β, γ) as

$$\begin{aligned}
r_{11} &= cos\alpha\ cos\beta \\
r_{12} &= sin\alpha\ cos\gamma + cos\alpha\ sin\beta\ sin\gamma \\
r_{13} &= sin\alpha\ sin\gamma - cos\alpha\ sin\beta\ cos\gamma \\
r_{21} &= -sin\alpha\ cos\beta \\
r_{22} &= cos\alpha\ cos\gamma - cos\alpha\ sin\beta\ sin\gamma \\
r_{23} &= cos\alpha\ sin\gamma - sin\alpha\ sin\beta\ cos\gamma \\
r_{31} &= sin\beta \\
r_{32} &= -cos\beta\ sin\gamma \\
r_{33} &= cos\beta\ cos\gamma
\end{aligned}$$

2. The 3-D camera coordinates (X_c, Y_c, Z_c) are then transformed into ideal retinal coordinates (X_u, Y_u) of the camera using perspective projection equations:

$$X_u = f\frac{X_c}{Z_c}, \quad Y_u = f\frac{Y_c}{Z_c} \tag{5.2}$$

3. The ideal retinal coordinates (X_u, Y_u) are then transformed into actual retinal coordinates (X_d, Y_d) by considering lens radial distortion coefficient k:

$$X_d = X_u(1 + kr^2)^{-1}, \quad Y_d = Y_u(1 + kr^2)^{-1} \tag{5.3}$$

The distance between the image center and the actual retinal coordinates is denoted by r.

4. Then the actual retinal coordinates (X_d, Y_d) are transformed into pixel coordinates (u,v):

$$u = X_d N_x + u_0, \quad v = Y_d N_y + v_0 \tag{5.4}$$

where (u_0, v_0) represents the pixel coordinates of the image center; N_x and N_y is the number of pixels that are contained in the unit distance of the image plane along the X and Y axes, respectively.

Combining the above four steps the image formation process of a 3-D point P(X,Y,Z) can be written as:

$$N_s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = Q \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{5.5}$$

where N_s is a 3×3 matrix representing scaling factor and matrix Q is decomposable into two matrices as: $Q = AD$ with $D = [RT]$, and A being a 3×3 matrix called the intrinsic matrix and can be expressed fully in terms of the intrinsic parameters of camera $(f, k, N_x, N_y, u_0, v_0)$. The matrix Q can be formulated as follows:

$$Q = [q_1^T \; q_2^T \; q_3^T] \tag{5.6}$$

where $q_i^T, \; (i = 1, 2, 3)$ represent the rows of Q. Using Equations (5.5) and (5.6),the pixel coordinates (u,v) can be represented as

$$u = \frac{q_1^T P}{q_3^T P}, \quad v = \frac{q_2^T P}{q_3^T P} \tag{5.7}$$

Using Equation (5.7), the 3-D world coordinates of a point P(X, Y, Z) can be transformed into pixel coordinates (u, v) for a single camera. This process being irreversible i.e. the 3-D position of point P cannot be uniquely determined from its pixel coordinates (u, v). However, from its stereo images, $p_l(u_l, v_l)$ and $p_r(u_r, v_r)$, taken by two cameras (or a single camera in two different positions) a point P(X, Y, Z) in the 3-D world can be uniquely determined .

5.2.2 Mathematical formation of Camera calibration problem

The camera calibration problem is mathematically expressed as a nonlinear optimization problem in which the objective is to minimize the square root of the sum of squares of Euclidean distances between the observed pixel positions and their corresponding calculated pixel positions. Thus, the camera calibration problem can be stated as follows:

Given N number of control points whose world coordinates $P_i(X_i, Y_i, Z_i)$ and observed pixel positions are known with high precision, the problem is to determine the optimal values of 12 decision variables $(u_o, v_o, N_x, N_y, f, k, \alpha, \beta, \gamma, T_x, T_y, T_z)$ that minimize the objective function given by equation 5.8:

$$F = \sqrt{\sum_{i=1}^{N} \left[\left(\frac{q_1^T P}{q_3^T P} - u \right)^2 + \left(\frac{q_2^T P}{q_3^T P} - v \right)^2 \right.} \tag{5.8}$$

Based on the knowledge of camera any suitable range for decision variables can be chosen, as an example we can choose $\beta \in [-\pi, \pi]$.

5.3 Solving Camera calibration problem using HS algorithm

For solving the camera calibration problem with HS algorithm, let H denote a vector consisting of the unknown intrinsic and extrinsic parameters of the camera, that is

$$H = (u_o, v_o, N_x, N_y, f, k, \alpha, \beta, \gamma, T_x, T_y, T_z) \tag{5.9}$$

For convenience we may write $H = (h_1, h_2, h_3, \ldots, h_{12})$, with $h_i,\ i = 1, 2, \ldots, 12$ representing the camera parameters in the same order as given in Equation (5.9). Then vector H with all its components within their bounds represents a potential solution to the problem.
The HS algorithm starts by initializing all the harmonies in the Harmony memory, each harmony in the HM represents a potential solution and has the same structure as defined for H in Equation (5.9). Every component of the harmony is randomly initialized within the allowed bounds. Once the initialization phase is over potential solutions in HM are evaluated using fitness function defined as Equation (5.8). Equation (5.8) gives the error value (i.e. difference between the actual and obtained coordinated) of the control points and thus the objective of the algorithm is to minimize Equation (5.8).
While generating the i^{th} component of the new harmony say H i.e. $H_i, 1 \leq i \leq 12$ either the i^{th} component of some existing harmony from HM is selected with probability HMCR (step 9 to 13 of Algorithm 1) or it is randomly generated within the allowed bounds (step 15 of Algorithm 1). In case the i^{th} component of new harmony is selected from HM, it is further adjusted within the allowed Band width (BW) range with probability PAR (step 11, 12 of Algorithm 1).The new harmony H is evaluated using Equation (5.8) and replaces the worst harmony of HM, if it is better than it (step 18, 19 of Algorithm 1).

5.4 Experimentation on Camera Calibration

This section presents extensive experiments with HS algorithm so as to evaluates its performance on camera calibration problem. The simulations has been carried out using varying number of control points. The accuracy is evaluated by calculating camera errors and pixel errors. Pixel error is defined as the root mean square Euclidean distance between the observed pixel positions and the corresponding re projected pixel positions calculated using estimated parameters values, whereas camera error is defined as the mean Euclidean distance between the ground truth and the estimated value of that parameter.

The stopping criteria for all the algorithm is 100000 function evaluations of objective function (Equation 5.8). The algorithms has been implemented in Dev C++ 5.0 and the experimentation has been carried out on a laptop with Windows 10 operating system, intel core i3 processor and 4GB of RAM.

5.4.1 Generation of synthetic data

In order to generate the synthetic data a calibration chart shown as Figure 5.2, having 8×8 grids in X and Y directions has been utilized to obtain 64 grid points. This calibration chart is shifted on eleven different positions along Z direction to get a total of 704 points $P_i(X_i, Y_i, Z_i)$ in 3-D space. The values of intrinsic and extrinsic parameters that were used for calculating the 2-D image coordinates from the 3-D grid points are called the ground truth values. The ground truth values for camera along with the parameter bounds used in the experimentation is shown as Table 5.5.

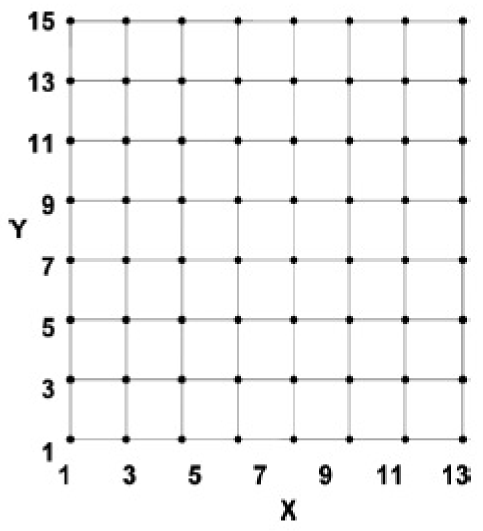


Figure 5.2: Calibration chart

5.5 Comparison on HA-SA, TPHS, SMHS on Camera Calibration problem

The simulations are performed for 5, 10, 50, 100, 200, 300, 400, 500 control points (represented as N) and the experiment is repeated 50 times for each control point. It is not possible to show the results of parameter values of all the experiments in tabular form or graphically, so the result of best run in terms of parameter values of the three competing algorithms HS-SA, TPHS, SMHS is respectively shown in Tables 5.6, 5.7, 5.8. Most of the parameter values obtained in Tables 5.6, 5.7, 5.8 are close to ground values. However no specific observation can be made regarding the effect of number of control points on accuracy of parameter values, further no concrete conclusion can be drawn about the performance of competing algorithms by just compering parameter values because on some parameters one algorithm is better whereas on other parameters some other algorithm is better. Thus the algorithms have been compared in terms of pixel error. Tables 5.9, 5.10, 5.11, 5.12 respectively show the results in terms of pixel error for HS, HS-SA, TPHS, SMHS . The reported results are mean, maximum, minimum and standard deviation of pixel error for various number of control points along with average calibration time in seconds. For all the four algorithms it can be observed that the increase in number of control points results in the decreases of average pixel error .

Comparing HS with the three proposed algorithms (HS-SA, TPHS, SMHS) in terms of average pixel error it can be observed from Tables 5.9, 5.10, 5.11 & 5.12 that all the three proposed algorithms outperformed standard HS irrespective of the number of control points used for calibration. In order to create a balance between exploration and exploitation the proposed algorithms inherit some additional complexity compared to standard HS & hence results in slightly higher calibration time for these algorithms.

Irrespective of the number of control points used, SMHS is the best performer amongst the proposed algorithms. Tables 5.9, 5.10, 5.11 & 5.12 reveals with the increase in number of control points the average pixel error decreases. Calibrating with control points as few as 5, all the algorithms shows very small pixel error and thus are effective even at smaller number of control points. This is an important achievement because creating the control points is difficult and time consuming job in practice. Increase in number of control points results in decrease of pixel error.

For the best performing SMHS algorithm, the minimum and mean of pixel error is respectively 0.102 and 2.213 when the number of control points is 5 with the increase in number of control points it gets reduced and finally becomes 0.015 and 0.602 when the number of control points is 500. With the increase in number of control points the complexity of objective function increases (Equation 5.8) and hence the execution time also increases.

Since HS-SA is the hybridized version of Harmony Search and Simulated Annealing, thus it has also been compared with SA algorithm. From Tables 5.9 and 5.13 it can be observed that HS-SA outperforms SA in terms of mean pixel error irrespective of the number of control points used. The simpler structure of SA provides it slight advantage over HS-SA in terms of time complexity.

5.6 Conclusion

This chapter compared the proposed three algorithms namely HS-SA, TPHS & SMHS on IEEE CEC 2014 benchmark suit and camera calibration problem, a highly nonlinear twelve dimensional optimization problem from the field of computer vision. SMHS outperformed the other two competing algorithms on all categories of problems viz a viz unimodal, simple multimodal, hybrid multimodal and composition multimodal functions. The comparison on camera calibration problem revels superior performance of all the proposed HS variants compared to standard HS algorithm irrespective of the number of control points used for calibration. Amongst the three proposed variants SMHS revealed the superiority in solving camera calibration problem.

Table 5.1: Error value obtained by HS-SA, TPHS and SMHS on Function number 1 through 10.

Function	Algorithm	Mean	Median	Best	Worst	Std Dev
1	HS-SA	1.2600E+07	8.5400E+06	1.5400E+06	4.0800E+07	1.0100E+07
	TPHS	1.0300E+07	9.5700E+06	4.3600E+06	3.0900E+07	5.0000E+06
	SMHS	**2.1302E+05**	**2.0839E+05**	**8.0011E+04**	**4.3582E+05**	**6.4339E+04**
2	HS-SA	1.2800E+04	1.2400E+04	**1.2900E+02**	3.3900E+04	1.0300E+04
	TPHS	1.0700E+06	1.0900E+06	6.0500E+05	1.4400E+06	1.8100E+05
	SMHS	**1.1738E+04**	**1.2231E+04**	2.6629E+03	**1.7669E+04**	**2.7019E+03**
3	HS-SA	6.1800E+03	4.4300E+03	2.4100E+02	1.7500E+04	4.8600E+03
	TPHS	2.4300E+03	2.1500E+03	2.8100E+02	6.9600E+03	1.5900E+03
	SMHS	**7.3749E+01**	**5.4557E+01**	**1.5946E+00**	**3.3090E+02**	**6.4081E+01**
4	HS-SA	1.1700E+02	1.2600E+02	6.0200E+00	2.0600E+02	4.1900E+01
	TPHS	1.1700E+02	1.2000E+02	6.8600E+01	1.6800E+02	2.4900E+01
	SMHS	**5.4983E+00**	**1.4974E+00**	**9.5633E-01**	**6.9181E+01**	**1.5772E+01**
5	HS-SA	**2.0000E+01**	**2.0000E+01**	**2.0000E+01**	**2.0000E+01**	**2.6100E-04**
	TPHS	2.0900E+01	2.0900E+01	2.0700E+01	2.1000E+01	6.3600E-02
	SMHS	2.0013E+01	2.0013E+01	2.0006E+01	2.0061E+01	7.1769E-03
6	HS-SA	1.2100E+01	1.2000E+01	9.8600E+00	1.9500E+01	2.1100E+00
	TPHS	**2.8900E+00**	**2.9400E+00**	**1.7500E+00**	**4.9300E+00**	**8.1600E-01**
	SMHS	4.4169E+00	4.3245E+00	1.9522E+00	6.9378E+00	1.1539E+00
7	HS-SA	**1.4300E-02**	**1.2600E-02**	**1.9900E-04**	9.8000E-02	2.0700E-02
	TPHS	9.0900E-01	9.1800E-01	7.4900E-01	9.7800E-01	5.1100E-02
	SMHS	1.8748E-02	1.4623E-02	2.9150E-03	**6.4336E-02**	1.4264E-02
8	HS-SA	**4.6600E-05**	**4.7000E-05**	**2.6000E-05**	**6.4000E-05**	**8.0100E-06**
	TPHS	1.9800E+00	1.7700E+00	3.5300E-01	5.5300E+00	1.2300E+00
	SMHS	1.9221E+00	1.9911E+00	4.0700E-04	4.9820E+00	1.3323E+00
9	HS-SA	6.5400E+01	6.5600E+01	3.6800E+01	9.1500E+01	1.3100E+01
	TPHS	1.4800E+02	1.2900E+02	8.9800E+01	2.0700E+02	3.4500E+01
	SMHS	**2.6905E+01**	**2.6867E+01**	**1.6915E+01**	**3.8805E+01**	**5.6832E+00**
10	HS-SA	**2.0100E-01**	**1.9000E-01**	**8.5300E-02**	**2.9400E-01**	**4.6600E-02**
	TPHS	1.9800E+01	2.0100E+01	9.9100E+00	2.7300E+01	4.2700E+00
	SMHS	6.9897E+00	6.4561E+00	2.6068E+00	1.1669E+01	2.3570E+00

Table 5.2: Error value obtained by HS-SA, TPHS and SMHS on Function number 11 through 30.

Function	Algorithm	Mean	Median	Best	Worst	Std Dev
11	HS-SA	1.9899E+03	1.9559E+03	**6.3470E+02**	**3.3266E+03**	4.9702E+02
	TPHS	4.1312E+03	4.1517E+03	5.7415E+02	6.7373E+03	1.6932E+03
	SMHS	**1.8056E+03**	**1.5735E+03**	8.6593E+02	4.7538E+03	8.5772E+02
12	HS-SA	**1.6853E-01**	**1.6566E-01**	**6.1270E-02**	**3.0058E-01**	**5.2063E-02**
	TPHS	2.4943E+00	2.4910E+00	1.8035E+00	3.0691E+00	2.6679E-01
	SMHS	6.2522E-01	6.1202E-01	1.5617E-01	1.0919E+00	1.9078E-01
13	HS-SA	5.3161E-01	5.2862E-01	3.0982E-01	8.0228E-01	1.1346E-01
	TPHS	3.7631E-01	3.5977E-01	2.1373E-01	5.6539E-01	9.1385E-02
	SMHS	**1.8662E-01**	**1.8368E-01**	**1.0202E-01**	**2.9823E-01**	**3.9571E-02**
14	HS-SA	4.0668E-01	3.4310E-01	2.0409E-01	1.0504E+00	2.0518E-01
	TPHS	3.1817E-01	3.1202E-01	1.6756E-01	4.2938E-01	5.5921E-02
	SMHS	**2.0903E-01**	**2.1033E-01**	**1.2922E-01**	**2.9089E-01**	**3.7121E-02**
15	HS-SA	1.4155E+01	1.2918E+01	4.8635E+00	2.9317E+01	5.7329E+00
	TPHS	1.7448E+01	1.7476E+01	1.4544E+01	1.9963E+01	1.0298E+00
	SMHS	**3.9018E+00**	**3.7710E+00**	**2.0791E+00**	**7.7064E+00**	**1.1601E+00**
16	HS-SA	9.4563E+00	9.4030E+00	8.0110E+00	1.1269E+01	7.6297E-01
	TPHS	**8.4200E+00**	**8.3470E+00**	**7.2800E+00**	**9.5521E+00**	**4.8655E-01**
	SMHS	9.4167E+00	9.5425E+00	7.5779E+00	1.1179E+01	7.1934E-01
17	HS-SA	2.0083E+06	1.6854E+06	1.3187E+05	6.8902E+06	1.4732E+06
	TPHS	3.1185E+05	2.7201E+05	**8.5087E+03**	1.0751E+06	2.4092E+05
	SMHS	**2.9252E+04**	**2.9950E+04**	1.2225E+04	**4.4817E+04**	**7.4617E+03**
18	HS-SA	6.1800E+03	4.6128E+03	**4.0631E+01**	2.4763E+04	6.2672E+03
	TPHS	2.9688E+03	1.9882E+03	3.8090E+02	1.4099E+04	2.8869E+03
	SMHS	**1.6495E+02**	**1.3705E+02**	4.6879E+01	**5.1708E+02**	**1.0095E+02**
19	HS-SA	2.0454E+01	9.2266E+00	5.8494E+00	1.3300E+02	3.3613E+01
	TPHS	6.7812E+00	6.7391E+00	4.8215E+00	9.4500E+00	9.8160E-01
	SMHS	**4.4001E+00**	**4.4021E+00**	**4.1229E+00**	**7.3987E+00**	1.2532E+00
20	HS-SA	6.2076E+03	6.1032E+03	8.0318E+02	2.6505E+04	5.5557E+03
	TPHS	9.5134E+02	6.5674E+02	2.6539E+02	3.2450E+03	7.0946E+02
	SMHS	**1.3391E+02**	**1.2379E+02**	**5.5901E+01**	**2.9843E+02**	**5.3748E+01**

Table 5.3: Error value obtained by HS-SA, TPHS and SMHS on Function number 21 through 30.

Function	Algorithm	Mean	Median	Best	Worst	Std Dev
21	HS-SA	6.556E+05	5.071E+05	8.176E+04	2.133E+06	5.294E+05
	TPHS	1.514E+05	1.092E+05	**1.030E+04**	5.114E+05	1.224E+05
	SMHS	**2.759E+04**	**2.790E+04**	1.391E+04	**4.591E+04**	**5.849E+03**
22	HS-SA	4.589E+02	4.864E+02	1.671E+02	9.442E+02	1.859E+02
	TPHS	**1.558E+02**	**1.497E+02**	**2.358E+01**	5.036E+02	1.084E+02
	SMHS	2.011E+02	1.532E+02	1.409E+02	**5.008E+02**	8.844E+01
23	HS-SA	3.156E+02	3.156E+02	3.152E+02	3.170E+02	3.675E-01
	TPHS	**3.140E+02**	**3.140E+02**	**3.139E+02**	**3.142E+02**	**5.184E-02**
	SMHS	3.142E+02	3.142E+02	3.140E+02	3.149E+02	1.650E-01
24	HS-SA	2.303E+02	2.301E+02	2.266E+02	2.462E+02	4.998E+00
	TPHS	**2.116E+02**	**2.129E+02**	**2.001E+02**	**2.161E+02**	**3.962E+00**
	SMHS	2.236E+02	2.238E+02	2.214E+02	2.249E+02	8.758E-01
25	HS-SA	2.050E+02	2.060E+02	2.043E+02	2.123E+02	1.691E+00
	TPHS	2.004E+02	2.001E+02	2.000E+02	2.020E+02	4.465E-01
	SMHS	**2.000E+02**	**2.000E+02**	**2.000E+02**	**2.000E+02**	**2.094E-03**
26	HS-SA	1.249E+02	1.006E+02	1.003E+02	2.015E+02	4.889E+01
	TPHS	1.102E+02	1.004E+02	**1.001E+02**	2.005E+02	2.975E+01
	SMHS	**1.002E+02**	**1.002E+02**	1.002E+02	**1.004E+02**	**4.557E-02**
27	HS-SA	6.066E+02	6.082E+02	4.014E+02	8.819E+02	1.539E+02
	TPHS	3.858E+02	3.900E+02	3.422E+02	5.582E+02	3.876E+01
	SMHS	**3.257E+02**	**3.235E+02**	**3.118E+02**	**3.761E+02**	**1.234E+01**
28	HS-SA	1.001E+03	9.993E+03	7.961E+02	2.269E+03	2.656E+02
	TPHS	7.119E+02	7.180E+02	5.640E+02	**7.708E+02**	3.688E+01
	SMHS	**3.205E+02**	**3.013E+02**	**3.008E+02**	8.311E+02	9.516E+01
29	HS-SA	1.714E+05	1.238E+03	5.638E+02	8.670E+06	1.202E+06
	TPHS	2.023E+03	1.922E+03	1.490E+03	2.896E+03	3.345E+02
	SMHS	**6.404E+02**	**6.454E+02**	**5.500E+02**	**1.241E+03**	**1.001E+02**
30	HS-SA	5.203E+03	4.846E+03	1.517E+03	1.400E+04	2.335E+03
	TPHS	3.487E+03	3.584E+03	1.288E+03	5.361E+03	1.026E+03
	SMHS	**6.832E+02**	**6.904E+02**	**6.178E+02**	**1.234E+03**	**2.655E+02**

Table 5.4: Time complexity in seconds (Dimension 30)

Algorithm	$\hat{T}_2$	$\frac{\hat{T}_2 - T_1}{T_0}$
HS	2.28	4.11
TPHS	2.35	4.46
HS-SA	2.37	4.8
SMHS	2.372	4.83

Table 5.5: Camera parameter bounds and Ground Truth values for camera.

Parameter	Ground Truth	Bounds
f	10	[5, 15]
N_x	200	[170, 230]
N_y	200	[170, 230]
u_0	20	[15, 25]
v_0	19	[15, 25]
K	0.15152	[0, 0.5]
T_x	60	[20, 80]
T_y	35	[25, 45]
T_z	1210	[1000, 1400]
α	0	$[-\pi/2,\ \pi/2]$
β	0	$[-\pi/2,\ \pi/2]$
γ	0	$[-\pi/2,\ \pi/2]$

Table 5.6: Best Parameter values obtained in 50 runs (HS-SA).

Parameters	N=5	N=10	N=50	N=100	N=200	N=300	N=400	N=500
f	8.253	7.06	9.924	12.397	15	12.314	11.861	7.643
N_x	217.986	209.671	214.286	212.658	184.347	188.408	220.348	221.912
N_y	230	206.564	216.72	213.266	186.513	188.791	225.93	222.688
u_0	20.927	15.656	15.812	19.831	24.769	21.664	22.64	15.313
v_0	22.181	23.051	24.212	18.475	21.495	17.959	18.754	19.42
K	0.174	0.024	0.191	0.379	0.374	0.214	0.297	0.103
T_x	48.173	79.541	76.334	66.077	79.399	65.158	60.469	61.69
T_y	39.664	31.662	41.447	25	27.465	41.67	40.733	31.985
T_z	1002.058	1000.33	1226.186	1004.9	1157.501	1331.794	1335.615	1004.411
α	0.00104	0.0008	-0.00065	-0.00152	-0.00193	0.00056	0.0016	-0.00082
β	-0.00934	0.00836	0.00525	0.00352	0.01813	0.00533	0.00283	-0.00453
γ	-0.00663	-0.0026	-0.01049	0.00731	0.002	-0.00317	-0.00372	0.00139

Table 5.7: Best Parameter values obtained in 50 runs (TPHS).

Parameters	N=5	N=10	N=50	N=100	N=200	N=300	N=400	N=500
f	13.77	13.65	15	15	13.884	15	11.995	15
N_x	203.92	176.069	175.918	178.143	224.194	212.93	215.02	192.42
N_y	174.86	203.005	173.941	214.136	193.235	210.607	214.082	171.825
u_0	20.667	23.882	25	22.053	16.111	20.485	15.391	23.692
v_0	15.295	24.077	22.436	24.998	23.214	22.225	19.981	23.964
K	0.282	0.477	0.273	0.366	0.366	0.496	0.282	0.331
T_x	39.412	41.965	21.179	63.31	34.083	63.943	66.378	39.169
T_y	37.029	35.954	39.865	38.947	32.622	40.759	37.435	39.057
T_z	1144.108	1098.257	1246.541	1181.174	1374.711	1052.596	1237.953	1254.143
α	-0.13499	-0.05958	-0.01421	0.01034	-0.06361	-0.0159	0.11631	-0.011
β	-0.00808	-0.01868	-0.0172	0.0029	-0.02019	0.00375	0.00046	-0.00699
γ	0.00205	-0.00494	-0.00911	-0.01246	-0.00201	-0.00757	-0.00303	-0.00728

Table 5.8: Best Parameter values obtained in 50 runs (SMHS).

Parameters	N=5	N=10	N=50	N=100	N=200	N=300	N=400	N=500
f	9.231	8.126	9.724	10.173	12.121	11.231	11.112	8.612
N_x	216.816	207.162	212.223	212.489	192.321	190.189	215.381	223.126
N_y	223	208.749	217.722	211.231	192.314	189.145	223.913	213.814
u_0	21.974	17.216	19.890	19.457	23.914	22.147	23.464	17.934
v_0	22.881	21.475	22.247	18.8875	20.478	19.59	19.746	21.242
K	0.123	0.074	0.182	0.315	0.324	0.217	0.247	0.112
T_x	55.45	71.465	72.469	63.485	77.449	62.785	61.475	62.973
T_y	34.567	33.478	40.157	27.98	26.425	42.167	41.123	33.784
T_z	1012.18	1002.133	1206.26	1011.19	1137.321	1311.214	1315.757	1014.221
α	0.00024	0.00015	-0.000215	-0.00132	-0.0018	0.000416	0.0012	-0.00062
β	-0.00412	0.00816	0.00542	0.00312	0.013	0.003	0.00312	-0.00412
γ	-0.00643	-0.00116	-0.0102	0.00842	0.0031	-0.00312	-0.00362	0.00143

Table 5.9: Statistics of Pixel Error for varying number of control points (HS).

Control Points (N)	Mean Pixel Error	Min. Pixel Error	Max. Pixel Error	SD Pixel Error	Calibration Time(s)
5	2.81	0.499	3.997	0.756	0.519
10	1.995	0.233	3.471	0.801	0.554
50	1.761	0.369	3.216	0.711	0.746
100	1.51	0.341	3.05	0.73	1.006
200	1.32	0.192	3.04	0.709	1.507
300	1.213	0.164	3.09	0.742	2.014
400	0.917	0.018	2.76	0.795	2.52
500	0.876	0.03	2.788	0.729	3.028

Table 5.10: Statistics of Pixel Error for varying number of control points (HS-SA).

Control Points (N)	Mean Pixel Error	Min. Pixel Error	Max. Pixel Error	SD Pixel Error	Calibration Time(s)
5	2.428	0.106	3.553	0.83	0.549
10	1.752	0.327	3.473	0.885	0.58
50	1.45	0.216	3.104	0.639	0.779
100	1.347	0.07	3.104	0.642	1.056
200	1.198	0.153	2.921	0.525	1.585
300	1.093	0.223	3.105	0.716	2.111
400	0.85	0.066	2.559	0.71	2.652
500	0.802	0.023	2.592	0.679	3.179

Table 5.11: Statistics of Pixel Error for varying number of control points (TPHS).

Control Points (N)	Mean Pixel Error	Min. Pixel Error	Max. Pixel Error	SD Pixel Error	Calibration Time(s)
5	2.368	0.572	4.068	0.819	0.7
10	1.71	0.568	3.602	0.762	0.718
50	1.435	0.647	2.517	0.432	0.973
100	1.39	0.567	2.471	0.459	1.292
200	1.30	0.701	2.605	0.427	1.929
300	1.20	0.516	2.335	0.378	2.657
400	0.901	0.565	2.336	0.425	3.355
500	0.861	0.501	2.534	0.498	4.069

Table 5.12: Statistics of Pixel Error for varying number of control points (SMHS).

Control Points (N)	Mean Pixel Error	Min. Pixel Error	Max. Pixel Error	SD Pixel Error	Calibration Time(s)
5	2.213	0.102	3.013	0.72	0.552
10	1.602	0.315	3.10	0.785	0.590
50	1.40	0.204	3.002	0.612	0.782
100	1.322	0.05	2.804	0.635	1.071
200	1.175	0.146	2.721	0.516	1.592
300	1.064	0.215	2.505	0.519	2.120
400	0.78	0.051	2.449	0.72	2.663
500	0.602	0.015	2.122	0.549	3.213

Table 5.13: Statistics of Pixel Error for varying number of control points (SA).

Control Points (N)	Mean Pixel Error	Min. Pixel Error	Max. Pixel Error	SD Pixel Error	Calibration Time(s)
5	2.658	0.158	4.062	0.921	0.217
10	1.775	0.221	3.512	0.67	0.229
50	1.652	0.011	3.076	0.788	0.333
100	1.38	0.095	3.299	0.728	0.46
200	1.252	0.17	2.887	0.713	0.717
300	1.116	0.032	2.822	0.768	0.973
400	0.945	0.104	2.548	0.717	1.233
500	1.016	0.101	2.663	0.647	1.531

Chapter 6
Harmony Search based Memetic Algorithms for Solving Sudoku

The development of hybrid procedures for optimization focuses on enhancing the strength and compensating for the weakness of two or more complementary approaches. The goal is to intelligently combine the key elements of the competing methodologies to create a superior solution procedure. The objective of this chapter is to explore the hybridization between Harmony Search (HS) and Hill Climbing (HC) algorithm by utilizing the exploration power of the former and exploitation power of the latter in the context of solving Sudoku which is a well-known hard Combinatorial Optimization problem. We call this hybrid algorithm Harmony Search Hill Climber (HSHC). In order to extend the exploration capabilities of HSHC it is further modified to create three different algorithms namely Retrievable Harmony Search Hill Climber (RHSHC), Global Best Retrievable Harmony Search Hill Climber (GB-RHSHC) and Random Best Retrievable Harmony Search Hill Climber (RB-RHSHC). Comparing the four algorithms proposed in this chapter RHSHC outperforms its three variations in terms of effectiveness. Experimental results demonstrate that RHSHC perform significantly better than standard Harmony Search algorithm and standard Hill climber algorithm. On comparing RHSHC with the genetic algorithm it has been concluded that former outperforms latter both in terms of effectiveness and efficiency particularly on Hard and Expert level puzzles. Comparing RHSHC and hybrid AC3-tabu search algorithm it has been concluded that RHSHC is very competent to hybrid AC3-tabu search algorithm.

The organization of this chapter is as follows. Section 6.1 is introductory in nature. Section 6.2 introduces the related work in the context of solving Sudoku. Section 6.3 describes the proposed algorithms. Section 6.5 compares the proposed HSHC with other metaheuristic algorithms and finally the chapter concludes with Section 6.6.

6.1 Introduction

Sudoku is a Japanese puzzle which consists of an N $\times$ N square that is divided into $\sqrt{N}$ sub-squares, each of size $\sqrt{N} \times \sqrt{N}$. Here N is a perfect square and is known as the order of the Sudoku. In the beginning, there are some static numbers (called givens or fixed) in the puzzle. The game is to fill all non givens such that each row, column and sub-square contains integers from 1 to N exactly once. Sudoku is a well-known NP complete problem (Takayuki and Takahiro, 2003). The difficulty level of the Sudoku puzzle is determined by around twenty factors and the number of initial givens has no or little role in it(Mantere and Koljonen, 2006). Figure 6.1 is an example of a Sudoku puzzle of order 9 along with its solution. In the solution, each row, column and sub-square contains integers from 1 to 9 exactly once, further the givens remain intact. Sudoku is linked to real world applications including conflict free wavelength routing in wide band optical networks, statistical design and error correcting codes, as well as timetabling and experimental design(Jones et al., 2008). Another closely related problem to sudoku is generating threshold matrix for halftoning grayscale images (Mantere and Koljonen, 2006).

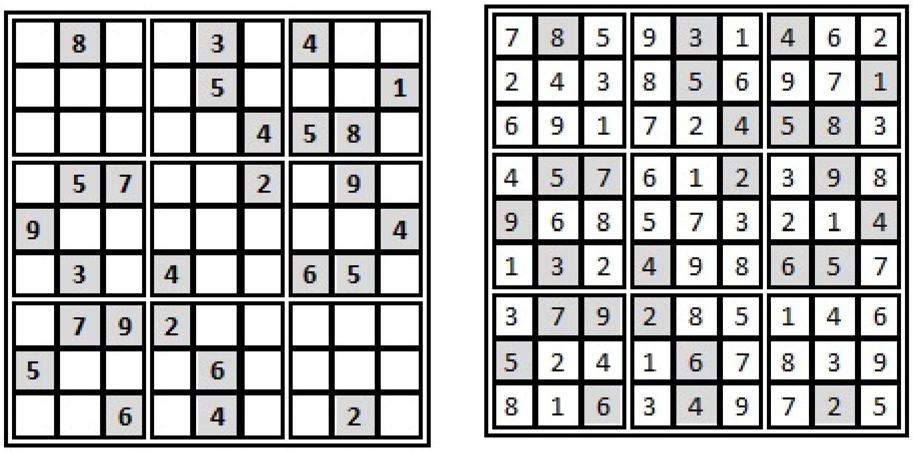
HS has demonstrated several advantages like simplicity, flexibility, adaptability, generality,and scalability over traditional optimization techniques (Al-Betar et al., 2012) and has been particularly successful on combinatorial optimization problems where it has outperformed other meta heuristic algorithms like genetic algorithm as well as the traditional branch and bound method (Geem, 2005) . HS works by generating a new vector that encodes a candidate solution, after considering the selection of all existing quality vectors. This forms a contrast with conventional evolutionary approaches such as GAs that consider only two (parent) vectors in order to produce a new (child) vector. It increases the exploration capabilities of HS (Diao and Shen, 2012) and hence has been preferred over other global search techniques like GA in this chapter.

Hill climbing is a local search optimization operator. It is an iterative algorithm that starts with an arbitrary solution of a problem, then explores to find a better solution by incrementally changing a single component of the solution. If the change produces a better solution, the new solution is accepted, repeating until no further improvements can be found.

Memetic Algorithms (MA) are a class of stochastic global search heuristics in which population based Evolutionary algorithms are hybridized with problem specific solvers, typically local search heuristic techniques to improve the quality of the solution (Ong et al., 2006a). The name is inspired by Richard Dawkinś concept of *meme*, which represents a unit of cultural evolution that can represent local refinement (Dawkins, 2006). MAs have arisen as a reciprocation to the problem encountered in the conventional evolutionary algorithms which are good at global exploration of the search space however can take relatively large time to find the optimal with sufficient precision (Ong et al., 2006b). This often limits the practicality of evolutionary algorithms in many real world problems where computational time is of paramount importance. This hybridization between global and local search methods refereed to as MA has been shown to be more efficient (i.e., requiring orders of magnitude fewer evaluations to converge) and effective (i.e., identifying high quality solutions that would otherwise be unreachable by evolutionary algorithm or local search alone) than traditional evolutionary algorithms on several problem domains (Al-Betar et al., 2012). MA are successful and popular for solving optimization problems in many contexts (Acampora et al., 2015; Al-Betar et al., 2012; Bansal et al., 2013; Chan et al., 2012; Ishibuchi et al., 2003; Jadon et al., 2015; Lin et al., 2016; Sharma et al., 2013, 2016). Hart et. al. (Hart et al., 2004) gives an elaborate review of MAs.

Many attempts have been made in literature to solve Sudoku puzzles these include exact search methods such as constraint programming (Moon and Gunther, 2006) and Boolean satisfiability (Lynce and Ouaknine, 2006) to heuristic and metaheuristic based algorithms including Simulated Annealing (SA) (Lewis, 2007), GA (Mantere and Koljonen, 2006), Ant Systems (Mullaney, 2006), Differential Evolution(DE) (Boryczka and Juszczuk, 2012) to name a few. Other less traditional techniques in this context such as Sinkhorn balancing (Moon et al., 2009), rewriting rules (Moon et al., 2009) and entropy minimization (Gunther and Moon, 2012) has also been proposed to tackle this problem.

The objective of this chapter is to create an algorithm namely Harmony Search Hill Climber (HSHC) by hybridizing HS and Hill Climbing operator with an aim to solve Sudoku. In order to increase the exploration capabilities of HSHC, three variations of HSHC have been proposed. Since the proposed algorithms use hill climbing operator they can be termed to fall in the category of memetic algorithms.



	8			3		4		
				5				1
					4	5	8	
	5	7			2		9	
9								4
	3		4			6	5	
	7	9	2					
5				6				
		6		4			2	

(a) Sudoku Puzzle

7	8	5	9	3	1	4	6	2
2	4	3	8	5	6	9	7	1
6	9	1	7	2	4	5	8	3
4	5	7	6	1	2	3	9	8
9	6	8	5	7	3	2	1	4
1	3	2	4	9	8	6	5	7
3	7	9	2	8	5	1	4	6
5	2	4	1	6	7	8	3	9
8	1	6	3	4	9	7	2	5

(b) Solution

Figure 6.1: A Sudoku Puzzle with 26 givens and its solution.

6.2 Related work

Deterministic algorithms based on branch and bound strategy have been used to solve Sudoku, since Sudoku is an NP-complete problem, thus we cannot find a polynomial time algorithm for all possible problem instances, unless P = NP (Garey, 1979). Thus researchers have made efforts to solve Sudoku puzzles using various meta heuristic algorithms.

For instance Mantere and Koljonen have used Genetic Algorithm (GA) to solve Sudoku puzzle in (Mantere and Koljonen, 2006). The algorithm is extension of the one devoted to solve magic square problems. In the algorithm each chromosome is represented as an integer array with size 81. Each chunk of 9 digits starting from left corrosponds to 3×3 sub block of Sudoku. Each sub block is initialized with numbers from 1 to 9 such that there is no repetition of digits and givens remain intact. The crossover site is only at the boundary of sub blocks, the mutation used is: swap mutation, 3-swap mutation and insertion mutation and is allowed only within the sub block. The algorithm has been tested on different categories of Sudoku puzzles although good performance have been exhibited in solving easy and medium type Sudoku however the success rate for challenging, difficult and super difficult puzzles is 30%, 4% and 6% respectively.

Das et.al.(Das et al., 2012) have proposed Retrievable GA algorithm by modifying the GA algorithm proposed in (Mantere and Koljonen, 2006). The main difference between Retrievable GA and the one proposed in (Mantere and Koljonen, 2006) is that in former population is re initialized after certain number of generations (which depend on difficulty level of puzzle). Experimental results demonstrate that Retrievable GA performs better than the one proposed in (Mantere and Koljonen, 2006) both in terms of effectiveness and efficiency (Das et al., 2012). The success rate shown by Retrievable GA in solving easy and medium puzzles is 100% and for difficult and superdifficult puzzles it is 16% and 9% respectively.

Nicolau and Ryan (Nicolau and Ryan, 2006) have used Genetic algorithm using Grammatical Evolution (GAuGE) for solving sudoku, GAuGE uses position independent representation. Each phenotype variable is encoded as a genotype string along with an associated phenotype position to learn linear relationships between variables. The GAuGE algorithm has shown promising results for majority of test puzzles, however as reported in (Nicolau and Ryan, 2006) there were some test puzzles on which the algorithm failed completely.

Li and Deng (Li and Deng, 2011) have modified the various important operators of GA in a bold way so that the algorithm has higher reliability, quicker convergence and better stability. Sato and Inoue (Sato and Inoue, 2010) have proposed a hybrid GA local search algorithm to tackle the sudoku puzzles

and have shown good performance particularly for solving difficult puzzles. In (Deng and Li, 2013) a hybrid GA has been utilized to solve Sudoku. The proposed algorithm was able to solve majority of easy puzzles however the success rate for difficult and superdifficult was 17% and 0% respectively. In order to accelerate the speed of Genetic algorithms for sudoku solving a parallel GA has been proposed in (Sato et al., 2013).
Hill Climbers have been tested to solve sudoku puzzle in (Moraglio et al., 2006). In order to restrict the search space explored by Hill Climber the concept of Smart Square Mutation has been applied. This mutation applies the most obvious constraint to the possible values Sudoku can take. For example if a row has a fixed '9' then '9' is removed from the set of possible values of that row, the same concept is extended to columns and sub squares. Though the proposed Hill climbing algorithm powered by Smart Square Mutation succeeded in solving easy type puzzles however it completely failed in solving medium and hard ones.
In (Lewis, 2007) a simulating annealing algorithm has been presented to solve sudoku, however the approach is mainly centered on creating solvable problems than solving hard Sudoku puzzles.
A hybrid tabu search algorithm has been proposed to solve the Sudoku puzzle in (Soto et al., 2015) and the algorithm has shown very promising results for solving difficult and superdifficult puzzles.
In (Boryczka and Juszczuk, 2012) Differential Evolution(DE) has been tested on sudoku puzzles, the authors have further tested the algorithm on classifying the Sudoku puzzles depending on their difficulty level. Though encouraging results have been reported however the algorithm has been tested on only few puzzles.
An evolutionary algorithm employing filtered mutations have been proposed in (Wang et al., 2015) to tackle Sudoku puzzle, the algorithm has been tested on 6 puzzles (2 each of type easy, medium, difficult and super difficult) and has shown good results particularly in solving difficult and super difficult puzzles.
A hybrid AC3-tabu search algorithm for solving sudoku has been proposed in (Soto et al., 2013). The algorithm has been created by combining tabu search with an arc-consistency 3 (AC3) algorithm that acts as a domain reducer. This integration reduces the number of tabu search iterations thus increases the convergence speed of the algorithm. As illustrated in (Simonis, 2005) Sudoku can be represented as constraint network, thus consequence techniques from constraint satisfaction can be applied on them. Arc-consistency is one of the most utilized filtration technique in constraint satisfaction for reducing the search space of combinatorial problems. Arc-consistency is formally defined as local consistency within the constraint programming field (Rossi et al., 2006). A local consistency defines

properties that the constraint problem must satisfy after constraint propagation. The hybrid AC3-tabu search algorithm has shown excellent performance on all types of Sudoku puzzles and has been compared with genetic algorithm proposed in (Mantere and Koljonen, 2007). The simulation results show that former is significantly effective than later particularly on hard Sudoku puzzles.
In (Soto et al., 2014) a Cuckoo Search Algorithm with Geometric Operators has been utilized for solving Sudoku Problems. The algorithm was able to solve easy and medium sudoku with approximately 100% success rate and hard puzzles with approximately 65% success rate in 10,000 iterations and if allowed to run for unlimited iterations the success rate for all types of puzzles was 100%.
HS algorithm has been used to solve Sudoku puzzle in (Geem, 2007) however there are three main limitations in that article.

1. In order to carry out experimentation only two Sudoku puzzles one of type Easy another of type Hard has been used to conclude that the algorithm solves easy problem very efficiently and fails to solve the hard problem, however in randomized algorithms one can't jump on conclusion by taking such a small example set.

2. There are different types of Sudoku puzzles like Beginner, Easy, Medium, Hard, and Expert however only Easy and Hard problem has been attempted.

3. The fitness function used is

$$Minimize \quad Z = \sum_{i=1}^{9}\left|\sum_{j=1}^{9} x_{ij} - 45\right| + \sum_{j=1}^{9}\left|\sum_{i=1}^{9} x_{ij} - 45\right| + \sum_{k=1}^{9}\left|\sum_{(l,m)\in B_k} x_{lm} - 45\right|$$

$$where\ X_{ij} \in \{1, 2 \ldots 9\}\ is\ the\ (i,j)^{th}\ element\ of\ Sudoku \quad (6.1)$$

The first term in equation (6.1) represents the penalty function for each horizontal row; the second term for each vertical column; and the third term for each block. The solution is reached when there is no violation (i.e. repeating number) in rows, columns and blocks thus for solution equation (6.1) evaluates to Zero.

There are two main disadvantages with this fitness function (equation 6.1).

1. Even if the fitness function evaluates to zero it is not guaranteed that the solution has been found. A detailed discussion on the limitations of the above mentioned fitness function (equation 6.1) can be found in (Weyland, 2015).

2. It gives more penalty to repetition of higher face value digits than lower face value digits. Let us assume there are two potential solutions A and B such that in solution A, the digit '9' occurs twice in some row and in solution B, the digit '1' occurs thrice in some row. Then according to the fitness function (equation 6.1) solution A is better than solution B, although solution A has more violation than solution B.

Thus there is a scope to modify the basic HS algorithm proposed in (Geem, 2007), so that it can be effective on all categories of Sudoku puzzles viz. Beginner, Easy, Medium, Hard and Expert.

6.3 Proposed HSHC Algorithm

In this chapter a hybrid algorithm of Harmony search and Hill Climbing has been proposed, namely Harmony Search Hill Climber (HSHC). In order to increase the exploration potentiality of HSHC three variants of HSHS have been proposed, namely Retrievable Harmony Search Hill Climber (RHSHC), Global Best Retrievable Harmony Search Hill Climber (GB-RHSHC) and Random Best Retrievable Harmony Search Hill Climber (RB-RHSHC).

Algorithm 7 is the Pseudo code of HSHC algorithm. In Algorithm 7 the parameters HMS, HMCR, NH_SIZE respectively denote Harmony Memory size, Harmony Memory Consideration Rate and Size of neighborhood to be explored by Hill Climbing operator.

6.3.1 Detailed Description of the proposed HSHC Algorithm

The working of HSHC (Algorithm 7) is described in the following steps.

Step 1

In this step the free parameters of algorithm like Harmony Memory Size (HMS), Harmony Memory Consideration Rate (HMCR) and Neighborhood Size (NH_SIZE) to be explored by Hill Climber operator are initialized.

Step 2

Initialization of harmony memory (HM) is performed in this step. So far as the structure of the harmony is concerned, each harmony represents a complete Sudoku. Thus harmony is represented by a vector of dimension N $\times$ N where N is the order of the Sudoku puzzle. Harmony Memory (HM) is an array of such vectors with dimension HMS. Mathematically the structure of HM is

$$H_1 = \begin{bmatrix} h_{11}^1 & h_{12}^1 \dots h_{1N}^1 \\ h_{21}^1 & h_{22}^1 \dots h_{2N}^1 \\ \multicolumn{2}{c}{\dots\dots\dots} \\ h_{N1}^1 & h_{N2}^1 \dots h_{NN}^1 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} h_{11}^2 & h_{12}^2 \dots h_{1N}^2 \\ h_{21}^2 & h_{22}^2 \dots h_{2N}^2 \\ \multicolumn{2}{c}{\dots\dots\dots} \\ h_{N1}^2 & h_{N2}^2 \dots h_{NN}^2 \end{bmatrix}$$

$$\vdots$$

$$H_{HMS} = \begin{bmatrix} h_{11}^{HMS} & h_{12}^{HMS} \dots h_{1N}^{HMS} \\ h_{21}^{HMS} & h_{22}^{HMS} \dots h_{2N}^{HMS} \\ \multicolumn{2}{c}{\dots\dots\dots\dots} \\ h_{N1}^{HMS} & h_{N2}^{HMS} \dots h_{NN}^{HMS} \end{bmatrix}$$

Where H_n is the n^{th} harmony, $f(H_n)$ is the fitness value of the n^{th} harmony and h_{ij}^n is cell at row i and column j in the n^{th} harmony of HM. While initializing the harmonies a constraint is defined such that each row in the harmony contain numbers from 1 to N exactly once without repetition. Mathematically $(i, j)^{th}$ element of harmony n denoted by h_{ij}^n must satisfy following constraints

$$for\ each\ i^{th}\ row, (1 \le i \le N), h_{ij}^n \ne h_{ik}^n,\ \ k \ne j, 1 \le j, k \le N, n \in \{1, 2, \dots HMS\} \tag{6.2}$$

If all the elements in a harmony satisfy above constraints (equation 6.2) it is guaranteed that the frequency of occurrence of each number in the harmony is N. This is an important achievement because in a valid Sudoku solution frequency of occurrence of each number must be exactly N. Line number 15 through 27 of Algorithm 7 constitute the hill climbing operator, where neighbors of a given harmony are generated by exchanging cell values and this operator will never converge to solution if the above mentioned constraint on frequency of occurrence of digits is not obeyed by harmony. Therefore during random initialization of HM it is made sure that each row in the harmony must obey the constraint

represented by equation (6.2). Though there are many ways to achieve it, let us consider following two ways.
Consider the pseudo code given as Algorithm 8. In Algorithm 8 line number 3* generates random number from 1 to N and assigns it to r, line number 5* compare already assigned numbers in row i of H with r if any of the numbers happen to be same as that of r otherwise r gets regenerated. H[i][j] is assigned r where H is the harmony with dimension N $\times$ N and H[i][j] is the $(i, j)^{th}$ element of H . It is easy to verify that the worst case time complexity of Algorithm 8 is atleast O(N^3) where N is the order of the Sudoku.
Another algorithm to obtain randomization in an array is called Richard Durstenfeld algorithm (Durstenfeld, 1964). Pseudo code of the algorithm for random initialization of Sudoku using Richard Durstenfeld algorithm is given as Algorithm 9. In Algorithm 9 line number 2* and 3* sequentially assigns numbers from 1 to N to row i of H, then in line number 6* and 7* one of the numbers is randomly picked and shifted to end, the loop in line number 5* repeats this process for all the N numbers. Since the complexity of Algorithm 9 is O(N^2) so it is considered for random initialization of harmony in HM. Further it must be noted during initialization no special care is provided to givens/fixed cells, rather they are also initialized randomly and for any violation of givens the harmony will be penalized by fitness function.

Step 3

In this step each Harmony is evaluated to determine its fitness. The limitation of fitness function used in (Geem, 2007) has already been discussed in section 6.2, so fitness function proposed in (Das et al., 2012) has been used with slight modification. The fitness function proposed in this chapter consists of four parts, namely row-fitness, column-fitness, sub-square-fitness and givens violation fitness. The first three fitness terms have equal weightage, however the last one has W times more weightage than first three terms. Thus the overall fitness function becomes

$$Fitness\ function = Row\ fitness + Column\ fitness + Subsquare\ fitness - W \times Givens\ violation \quad (6.3)$$

The fitness function (equation 6.3) attains the maximum possible value when the solution is reached. Each row entry is compared with all the remaining entries to its right, If the two entries are not equal row fitness value is incremented by one otherwise it remains same. Thus in the solution the contribution from each row is $N(N-1)/2$ (sum of first N-1 natural numbers). Hence for N rows it is $N^2(N-1)/2$. Similar results hold for columns and sub squares. Hence for an $N \times N$ Sudoku puzzle the maximum possible fitness value is $3N^3(N-1)/2$. It must be noted that the contribution of penalty for violating givens will be zero in the solution. The fitness value for the solution of a 9 $\times$ 9 Sudoku puzzle is 972. Thus when the fitness function attains its maximum possible value it is guaranteed that solution has been obtained. Expressing this concept in mathematical form

$$f(i,j,k,l) = \begin{cases} 0 & \text{if } (i,j) = (k,l) \\ 1 & otherwise \end{cases}$$

Where (i,j) and (k,l) refer to two entries of N $\times$ N Sudoku puzzle. The fitness function for each row is defined as

$$Row\ fitness = \sum_{i=1}^{N} \sum_{j=1}^{N-1} \sum_{l=j+1}^{N} f(i,j,i,l) \tag{6.4}$$

The fitness function for each column is defined as

$$Column\ fitness = \sum_{j=1}^{N} \sum_{i=1}^{N-1} \sum_{l=i+1}^{N} f(i,j,l,j) \tag{6.5}$$

The fitness function for each sub square is defined as

$$Sub\ Square\ fitness =$$

$$\sum_{i=1}^{N} \left[\sum_{q=1}^{\sqrt{N}} \left\{ \sum_{\substack{j=1+\\(q-1)\sqrt{N}}}^{q\sqrt{N}-1} \sum_{l=j+1}^{q\sqrt{N}} f(i,j,i,l) + \sum_{\substack{r=1+(q-1)\sqrt{N}\\ i\neq t\sqrt{N}, t\in Z^+}} \sum_{k=i+1}^{\substack{i+\sqrt{N}-\\ i(mod\sqrt{N})}} \sum_{\substack{s=1+\\(q-1)\sqrt{N}}}^{q\sqrt{N}} f(i,r,k,s) \right\} \right] \tag{6.6}$$

The fitness function for violating fixed/givens is defined as

$$Givens\ violation = \sum_{i=1}^{N} \sum_{j=1}^{N} l(i,j) \tag{6.7}$$

$$where \quad l(i,j) = \begin{cases} 1 & \text{if } flag(i,j) \neq 0\ and\ (i,j) \neq flag(i,j)) \\ 0 & otherwise \end{cases}$$

Where flag is an N $\times$ N matrix whose $(i,j)^{th}$ element is 0 if $(i,j)^{th}$ element is not given otherwise it is equal to given/fixed $(i,j)^{th}$ element and Z^+ is the set of all positive integers.

Hence from equation (6.3) the fitness function is the sum of equations (6.4), (6.5), (6.6) minus W times equation (6.7). After thorough experimentation the value for W in equation (6.3) was fixed as 8.

Step 7

This step is executed with probability HMCR and in this step a new harmony say H^{new} is produced either by selecting an existing harmony from HM with probability P or by combing the harmonies in HM with probability 1-P. While producing a new harmony using existing harmonies a simple mechanism is used. While generating i^{th} $(1 \leq i \leq N)$ row of H^{new}, one of the harmonies from HM is randomly selected and its i^{th} row is added to H^{new}. The above procedure is repeated for all the N rows of H^{new}. The optimum value of P was found out to be .95 after thorough experimentation.

Step 9

The aim of this step is to speedup convergence by incorporating problem specific knowledge in harmony creation. This step must be executed with very low probability because it increases the probability of being stuck in local optimal. In this step a harmony is randomly generated however during random harmony creation two facts are kept in mind one the givens must remain intact another there must be no repetition of numbers in rows and columns. However repetition of numbers in blocks is allowed. Mathematically h_{ij} the $(i, j)^{th}$ element of H^{new} must satisfy following constraints if it is not fixed.

$$for\ each\ element\ \ h_{ij},\ h_{ij} \neq h_{ik}\ (j \neq k)\ and\ h_{ij} \neq h_{lj}\ (l \neq i), (i, j, k, l) \in \{1, 2 \ldots N\} \quad (6.8)$$

Steps 15 through 27 of Algorithm 7 constitute the Hill Climbing operator where a neighbor of a Harmony is generated by exchanging contents of two cells having different values.

Step 20

In this step NBR is compared with the best Harmony and in case NBR is better than or slightly inferior than BEST, NBR becomes ROOT. The reason for this move is that since NBR seems to be very promising as it can compete with the best Harmony it is reasonable to concentrate on this new harmony by exploring it further. If the difference between the fitness value of BEST and NBR is less than 3 it is considered to be marginally inferior to BEST and thus eligible to be explored further. After through experimentation the optimal value of T was found out to be 3.

Steps 23 and 24

If any harmony say NBR produced during hill climbing is better than the WORST Harmony in Harmony Memory, the WORST harmony in HM is replaced by that harmony.

One of the main disadvantages of evolutionary algorithms (including HS) is premature convergence due to lack of diversity in population, so in order to overcome this issue the concept of catastrophic mutation form GA (Jin and Li, 1997) has been adopted in HSHC and three new algorithms namely RHSHC, GB-RHSHC and RB-RHSHC are proposed depending on how Catastrophic mutation is ap-

plied. In Catastrophic mutation the population is unsettled by very high mutation rate (typically greater than 0.8) so as to recover the population diversity and thus avoid premature convergence in GA.
In this chapter catastrophic mutation is applied by re initializing the harmony memory (HM) if the best Harmony does update in 150,000 function evaluations. In GB-RHSHC catastrophic mutation is performed by re initializing all the harmonies in the HM except the best one, whereas in RB-RHSHC catastrophic mutation is performed by re initializing all the harmonies in the HM except one randomly selected harmony. In RHSHC catastrophic mutation is performed by re initializing the entire HM.
The time complexity of fitness function (Equation 6.3) is $O(N^3)$ and hence the overall time complexity of HSHC is $O(KN^3)$ where N is the order of sudoku and K is the allowed number of fitness function evaluations. It should be noted that the time complexity of GB-RHSHC, RB-RHSHC and RHSHC remains same as that of HSHC.

6.4 Computational Experiment

In order to check the effectiveness and efficiency of the proposed algorithms a set of 25 Sudoku puzzles (of order 9) five each of type Beginner, Easy, Medium, Hard and Expert have been taken from the web site www.sudoku.com and each problem has been tested 30 times. Determining the optimal setting for free parameters in a meta heuristic algorithm is a hyper optimization problem, however after thorough experimentation following parameter setting was found out to be effective for most if not all the cases: HMS=40, HMCR=0.99 and NH_SIZE=120. The stopping criteria for a run is either the solution is found or maximum execution time of 35 Seconds is attained. The experimentation was carried out on a laptop having specification- Intel CORE i3 processor, 4GB of RAM and Windows 8.1 Operating System.
Tables 6.1, 6.2, 6.3, 6.4 demonstrate the performance of the proposed algorithms HSHC, RHSHC, GB-RHSHC and RB-RHSHC respectively. The columns of all the four tables from left to right represent: Puzzle type (PUZZLE TYPE), Percentage of runs that are able to find solution of a given puzzle (SP), Minimum execution time (MINT), Maximum execution time (MAXT), Average execution time (AVGT), Standard deviation of execution time (SDT), Minimum number of fitness Function Evaluations performed (MINFE), Maximum number of fitness Function Evaluations performed (MAXFE), Average number of fitness Function Evaluations performed(AVGFE) and Standard Deviation of number of fitness Function Evaluations performed (SDFE).The above statistics in terms of execution time and number of fitness function evaluations required have been obtained for successful runs only.
Figure 6.2a compare the performance of proposed four algorithms in terms of success rate on different

types of Sudoku puzzles. Figure 6.3a, 6.3b and 6.3c respectively compare the performance of the four algorithms in terms of Minimum, Maximum and Average execution time required for successful runs . Figure 6.3d, 6.3e and 6.3f respectively compare the performance of the four algorithms in terms of Minimum, Maximum and Average number fitness function evaluations required to find the optimal. Note that success rate is the percentage of runs able to solve the Sudoku puzzle. It is very much evident from Figure 6.2a that the order followed by four algorithms in terms of success rate is:

$$\text{RHSHC} > \text{RB-RHSHC} > \text{GB-RHSHC} > \text{HSHC}$$

Except for Beginner type puzzles the algorithms follow the same order in terms of execution time of successful runs. Now let us analyse the behavior of these algorithms. The reason for highest success rate and execution time of RHSHC is while trying to figure out global optima the algorithm extensively reinitialize its HM thus increasing the probability of escaping from local optima but at the same time increasing its execution time because the algorithm is not using its previous experience while further exploring the search space. While trying to escape from local optima by reinitializing HM the GB-RHSHC algorithm preserves the best harmony obtained so far and thus utilizes the best of its experience while as RB-RHSHC preserves one of the good harmonies and not necessarily the best thus former increases the probability of fast convergence than latter on the cost of increasing the probability of being stuck in local optima. HSHC never performs catastrophic mutation (i.e., does not unsettle its HM by reinitialization) thus increasing its convergence speed but at the same time decreasing the probability of escaping from local optima. Since beginner type Sudoku puzzles comparatively take less number of function evaluations as a result frequency of HM reinitialization is decreased, so all the four algorithms have almost same execution time for such type of problems.

6.5 Comparison with other Heuristic Algorithms

The best performing RHSHC has been compared with the standard Harmony Search algorithm (Geem, 2007), Hill Climber algorithm (Moraglio et al., 2006), Retrievable Genetic algorithm (Das et al., 2012) and hybrid AC3-tabu search algorithm (Soto et al., 2013) for solving sudoku. A brief introduction of the above mentioned algorithms has already been provided in Section 6.2. Retrievable Genetic algorithm has been compared with Genetic algorithm proposed in (Mantere and Koljonen, 2006) and it has been established that former is superior to latter both in terms of effectiveness and efficiency so we have not compared our results with latter.

Since the fitness function proposed in this chapter is different as used in (Geem, 2007) further the mechanism for generating new Harmony whether by memory consideration or by randomization is also different here, so in order to keep the comparison fair all the programs have been run for equal amount of time (35 seconds). It has been found that the HS algorithm to solve Sudoku proposed in (Geem, 2007) was not able to solve a single problem from the set of 25 Sudoku puzzles, thus all the four algorithms proposed in this chapter (HSHC, RHSHC, GB-RHSHC, RB-RHSHC) significantly outperformed it.

The Hill climber algorithm proposed in (Moraglio et al., 2006)is able to solve Easy sudoku puzzles with 100% success rate however completely failed in solving Medium and Hard puzzles. Thus RHSHC is very effective than standard Hill climber algorithm proposed in (Moraglio et al., 2006) particularly for Medium, Hard and Expert level puzzles.

In order to keep the comparison fair between RHSHC and Retrievable GA (Das et al., 2012), Retrievable GA have been tested on same set of 25 sudoku puzzles, with the same stopping criteria and on the same machine on which RHSHC was executed. Further Retrievable GA algorithm is tested 30 time on each puzzle as was done with RHSHC algorithm.

Figures 6.2b and 6.3g compare the two algorithms (RHSHC & Retrievable GA) in terms of success rate and execution time respectively. As is evident from Figure 6.2b the success rate of both algorithms is almost same for Beginner and Easy level puzzles however for Medium, Hard and Expert level puzzles RHSHC significantly out performs Retrievable GA. The difference is more evident in Expert level puzzles where the success rate of RHSHC is approximately 80% and that of Retrievable GA is only 7%. In terms of execution time Retrievable GA outperforms RHSHC for Beginner and Easy type puzzles, however for Medium, Hard and Expert level puzzles RHSHC outperforms Retrievable GA (Figure 6.3g). Thus RHSHC is the better performing algorithm (in terms of effectiveness as well as efficiency) than Retrievable GA particularly for Medium, Hard and Expert level puzzles whereas for Beginner and Easy level puzzles there performance is almost same.

Hybrid AC3-tabu search algorithm and RHSHC algorithm has been tested on the same set of 25 Sudoku puzzles, run on same machine and with same stopping criteria. Figures 6.2c and 6.3h compare the two algorithms (RHSHC & Hybrid AC3-tabu search) in terms of success rate and execution time respectively. As is evident from Figure 6.2c the success rate of both algorithms is almost same for Beginner level puzzles. RHSHC is slightly better than Hybrid AC3-tabu algorithm on Easy level puzzles whereas on Medium, Hard and Expert level puzzles hybrid AC3-tabu search algorithm has slight advantage over RHSHC. In terms of time complexity (Figure 6.3h) hybrid AC3-tabu search outperforms RHSHC on all category of puzzles, except for Easy type puzzles were both take approximately same time .

6.5.1 Wilcoxon's rank-sum test

A pair wise Wilcoxon's rank-sum test at 5% level of significance is used to statistically compare the performance of the competing algorithms. The sampling data used for applying the test has been obtained by performing 30 independent runs of each algorithm. The Wilcoxon's rank-sum test revealed that there is no statistically significant difference between RHSHC and Retrievable GA on Beginner and Easy type puzzles, however the superior performance of RHSHC over Retrievable GA is statistically significant on Medium, Hard and Expert level puzzles. Comparing the performance of RHSHC and Hybrid AC3-tabu search algorithm Wilcoxon's rank-sum test revealed that there is no statistically significant difference on Beginner, Medium and Hard type puzzles, however the better performance of RHSHC over AC3-tabu search algorithm is statistically significant on Easy type puzzles similarly the better performance of AC3-tabu search over RHSHC is statistically significant on Expert level puzzles.

6.6 Conclusion

This chapter introduces a specialized Memetic algorithm namely HSHC created by hybridization of Harmony Search Algorithm and Hill Climbing operator to solve Sudoku puzzles. In order to increase exploration capabilities of HSHC algorithm three variations of basic HSHC are introduced. All the four proposed algorithms performed significantly better than the standard Harmony Search algorithm and standard Hill Climber algorithm. RHSHC outperformed its three variations (HSHC, GB-RHSHC, RB-RHSHC) in terms of success rate and was able to solve Beginner and Easy type problems with 100% success, further it also performed extremely well in solving Medium, Hard and Expert level puzzles. Comparing RHSHC and Retrievable GA, it was established that former significantly outperformed latter in terms of effectiveness and efficiency particularly for Medium, Hard and Expert level puzzles. Experimental results demonstrate that RHSHC is competent (if not better) to recently proposed hybrid AC3-tabu search algorithm.

Table 6.1: HSHC Performance

PUZZLE TYPE	SP	MINT	MAXT	AVGT	SDT	MINFE	MAXFE	AVGFE	SDFE
BEGINNER	70.67	0.251	32.911	4.892	2.734	40041	4979069	703649.7	445102.9
EASY	66	0.436	28.805	4.221	2.847	67853	4645532	657033.3	433648.7
MEDIUM	33.33	0.796	32.674	5.073	2.022	126055	4803732	792300.5	290206.2
HARD	33.33	0.719	29.014	6.895	2.399	116955	4759383	1114436	392600.2
EXPERT	30.67	0.945	28.009	5.804	2.921	144262	3647316	830308.3	355427.8

Table 6.2: RHSHC Performance

PUZZLE TYPE	SP	MINT	MAXT	AVGT	SDT	MINFE	MAXFE	AVGFE	SDFE
BEGINNER	100.00	0.260	29.121	3.103	2.301	41435	4358130	672271.0	635898.0
EASY	100.00	0.340	32.653	6.034	2.575	53750	4751663	634465.2	608455.6
MEDIUM	93.33	0.691	31.370	10.196	2.582	107590	4700497	1926207.5	1152346.3
HARD	88.67	1.029	31.267	12.771	1.798	157017	4954494	2025149.5	303393.9
EXPERT	80.00	1.025	33.828	13.579	1.301	159260	4868901	1911467.8	252190.1

Table 6.3: GB-RHSHC Performance

PUZZLE TYPE	SP	MINT	MAXT	AVGT	SDT	MINFE	MAXFE	AVGFE	SDFE
BEGINNER	81.33	0.296	22.399	3.289	3.186	47244	3581089	505804.7	477320.1
EASY	66.00	0.396	34.037	4.945	4.196	58969	4858880	766863.1	622109.7
MEDIUM	57.33	0.750	33.158	7.861	5.297	121566	4857597	1196316.0	772401.0
HARD	49.33	0.780	28.082	8.222	1.739	109090	4170662	1237255.9	248678.5
EXPERT	42.67	0.846	34.154	11.091	2.914	120754	4975036	1678574.1	450827.2

Algorithm 7 Harmony Search Hill Climber (HSHC)

```
1: Initialize Algorithm parameters HMS, HMCR, NH_SIZE.
2: Initialize Harmony Memory.
3: Evaluate each harmony of harmony memory using fitness function (Equation 6.3).
4: while Stopping Condition Not Reached do
5:     Find BEST and WORST Harmony in HM.
6:     if rand$(0, 1)$ is less than HMCR then
7:         With probability P select randomly one of the Harmony from HM otherwise generate a new
           harmony using the harmonies in HM, name it ROOT.
8:     else
9:         Generate a Harmony Randomly, name it ROOT.
10:    end if
11:    Evaluate ROOT using fitness function (Equation 6.3).
12:    if ROOT is solution then
13:        STOP.
14:    end if
15:    for $i = 1$ TO NH_SIZE do
16:        Select randomly two cells of ROOT with different values, swap their contents to obtain a
           neighbor of ROOT, name it NBR.
17:        if NBR is Solution then
18:            STOP.
19:        end if
20:        if NBR is better than or marginally inferior to BEST then
21:            set ROOT=NBR.
22:        end if
23:        if NBR is better than worst Harmony then
24:            update Harmony Memory by replacing WORST Harmony by NBR.
25:        end if
26:    end for
27: end while
```

Algorithm 8 Random Initialization of Sudoku

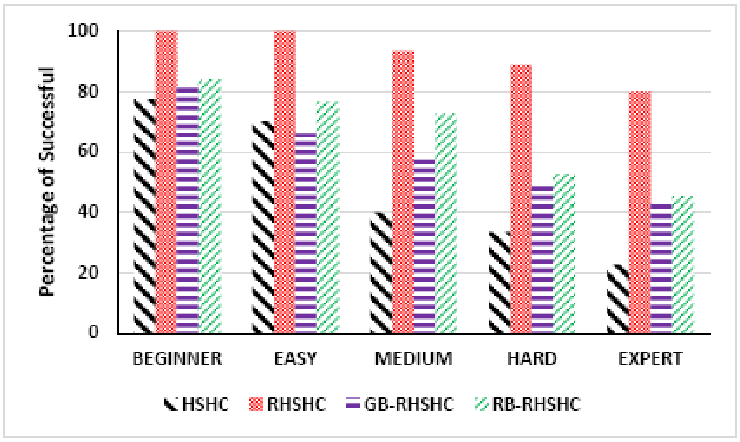
```
1*  for i=1 to N do
2*      for j=1 to N do
3*          r=rand(1,N)
4*          for l=1 to j-1 do
5*              if H[i][j]=r then
6*                  GOTO Line 3*
7*                  H[i][j]=r
8*              end if
9*          end for
10*     end for
11* end for
```

Algorithm 9 Richard Durstenfeld Algorithm for Random Initialization of Sudoku

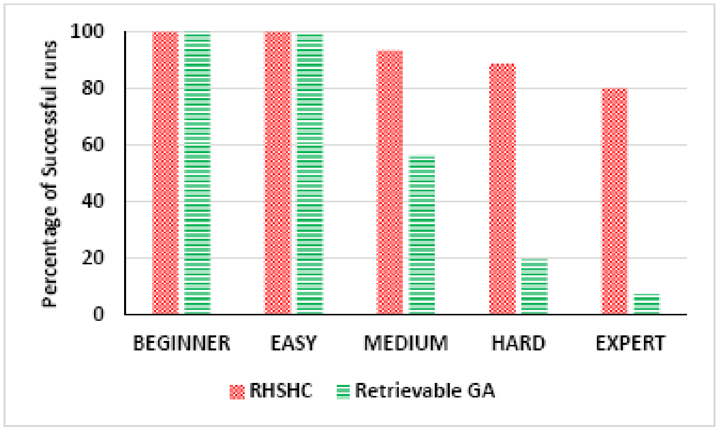
```
1*  for i=1 to N do
2*      for j=1 to N do
3*          H[i][j]=j
4*      end for
5*      for k=N down to 1 do
6*          r=rand(1,k)
7*          swap H[i][r] and H[i][k]
8*      end for
9*  end for
```

Table 6.4: RB-RHSHC Performance

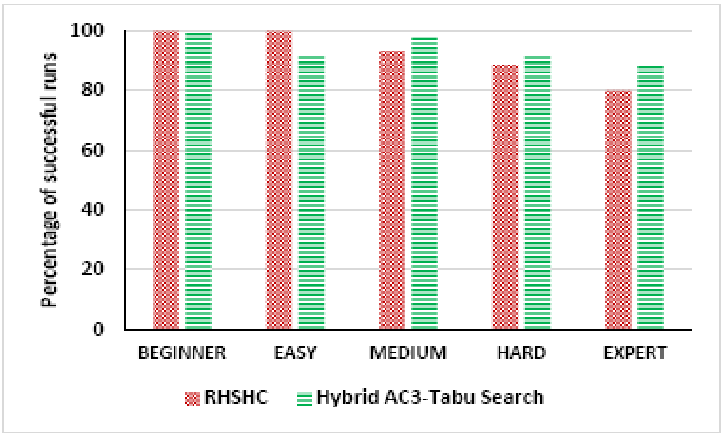
PUZZLE TYPE	SP	MINT	MAXT	AVGT	SDT	MINFE	MAXFE	AVGFE	SDFE
BEGINNER	84.00	0.287	32.235	3.052	1.402	45830	4000409	665515.9	415526.8
EASY	76.67	0.331	26.951	5.445	1.580	51823	4522048	828088.2	695742.7
MEDIUM	72.67	0.682	31.621	8.002	2.614	106805	5767232	1345171.3	538997.8
HARD	52.67	0.904	32.141	10.374	3.333	143274	5832768	1826020.3	490699.8
EXPERT	45.33	0.895	31.804	11.539	2.699	123038	4916839	1886465.3	307100.5



(a) HSHC and Variants.

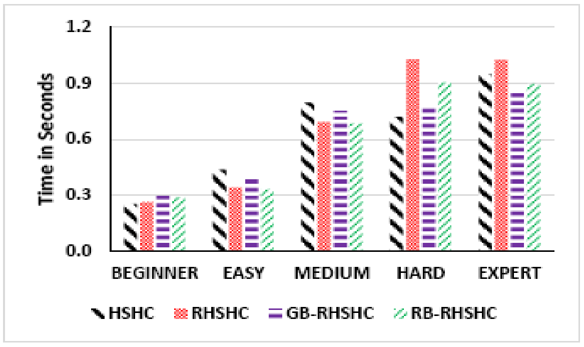


(b) RHSHC and Retrievable GA

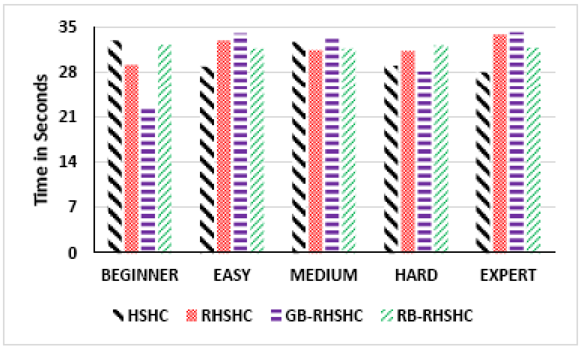


(c) RHSHC and Hybrid AC3-Tabu Search

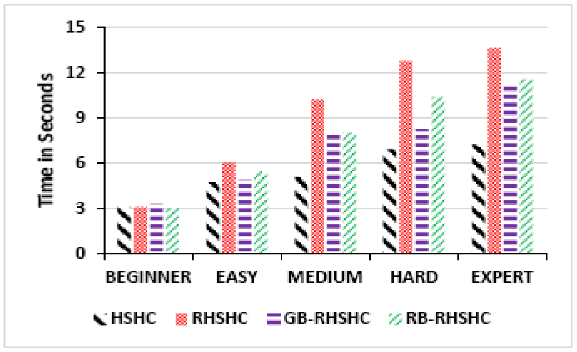
Figure 6.2: Success rate.



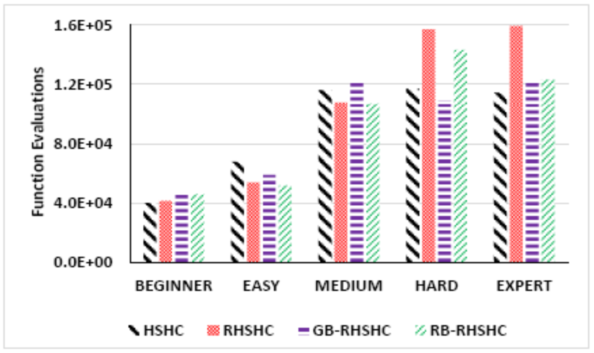
(a) Minimum Execution Time.



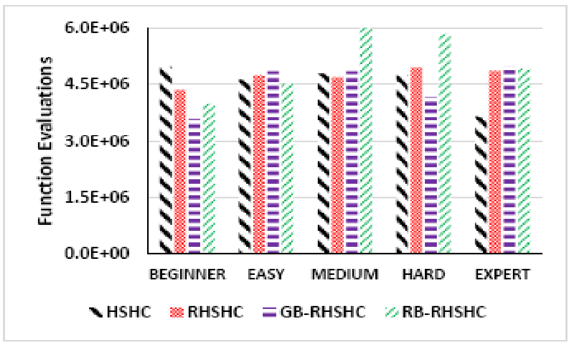
(b) Maximum Execution Time.



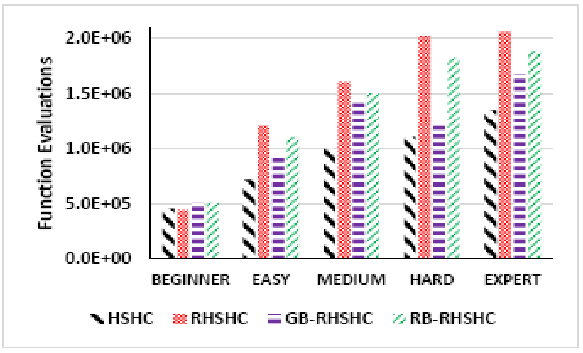
(c) Average Execution Time.



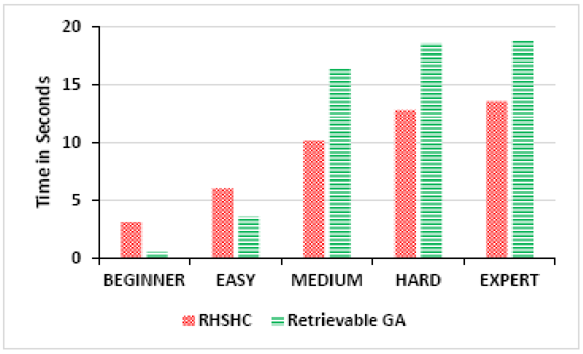
(d) Minimum Function Evaluations.



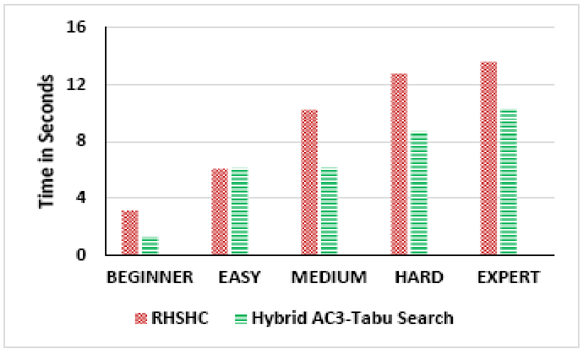
(e) Maximum Function Evaluations.



(f) Average Function Evaluations.



(g) Average Execution Time.



(h) Average Execution Time.

Figure 6.3: Execution Time and Function Evaluations

Chapter 7

A Heuristic Based Harmony Search Algorithm for Maximum Clique Problem

The maximum clique problem (MCP) is to determine a complete subgraph (clique) of maximum cardinality in a given graph. MCP is conspicuous for having real world applications and for its potentiality of modeling other combinatorial problems and is one of the most studied NP-hard problems. This chapter investigates the capabilities of Harmony Search (HS) algorithm, a music inspired meta heuristic for solving maximum clique problem. We propose and compare two different instantiations of a generic HS algorithm namely Harmony Search for MCP (HS_MCP) and Harmony Search with idiosyncratic harmonies for MCP (HSI_MCP) for this problem. HS_MCP has better exploitation and inferior exploration capabilities than HSI_MCP whereas HSI_MCP has better exploration and inferior exploitation capabilities than HSI_MCP, it has been concluded that former performs better than latter by testing them on all the instances of DIMACS benchmark graphs. HS_MCP has been compared with a recently proposed Harmony search based algorithm for MCP called Binary Harmony search (BHS) and the simulation results show that HS_MCP significantly outperforms BHS in terms of solution quality. The asymptotic time complexity of HS_MCP is $O(G \times N^3)$ where G is the number of generations and N is the number of nodes in the graph. A glimpse of effectiveness of some state-of-the-art exact algorithms on MCP has also been provided.
The organization of this chapter is as follows. Section 7.1 is introductory in nature. Section 7.2 describes the proposed algorithms. Section 7.3 describes the numerical experimentation and compares the proposed algorithm with other algorithms and finally the chapter concludes with Section 7.4.

7.1 Introduction

Given a simple undirected graph $G = (V, E)$ where $V = \{1, 2, \ldots, N\}$ is the vertex set and $E = V \times V$, is the edge set. A clique C is the complete sub graph of G i.e. all the vertices of C are pairwise adjacent. The size of clique is the number of vertices in it and the maximum clique is the one having maximum cardinality.
MCP is one of the most studied combinatorial optimization problems because it has wide range of practical applications in numerous fields like coding theory (Etzion and Ostergard, 1998), bioinfor-

matics and chemo informatics (Gomez Ravetti and Moscato, 2008; Malod-Dognin et al., 2010), examination planning (Carter and Johnson, 2001; Carter et al., 1996), scheduling (Dorndorf et al., 2008), financial networks (Boginski et al., 2006), social network analysis (Balasundaram et al., 2011; Pattillo et al., 2012), signal transmission analysis (Chen et al., 2010), telecommunication and wireless networks (Balasundaram and Butenko, 2006; Jain et al., 2005), combinatorial auctions (Wu and Hao, 2015b), visual feature matching (San Segundo and Artieda, 2015), community detection (Pattabiraman et al., 2015), computer vision and information retrieval (Pardalos and Xue, 1994).
In addition to these applications, the MCP is tightly related to some important combinatorial optimization problems such as clique partitioning, graph clustering, graph vertex coloring, max-min diversity, optimal winner determination, set packing and sum coloring (Wu and Hao, 2015a). These problems can either be directly formulated as a MCP or has a sub problem which requires to find a maximum clique.
MCP is highly untractable and its decision version is among the first 21 NP-complete problems presented in Karp's seminal paper on computational complexity (Karp, 1972) . A problem that is NP-complete has the property that it can be solved in polynomial time if and only if all other NP-complete problems can be solved in polynomial time. If an NP-Hard problem can be solved in polynomial time, then all NP-complete problems can be solved in polynomial time. All NP-complete problems are NP-Hard, but some NP-Hard problems are not known to be NP-complete Horowitz and Sahni (1978).
Even the approximation of MCP within a constant factor are NP-Hard (Feige et al., 1991). There is no polynomial time algorithm for approximating the MCP within a factor of $n^{1/4-\epsilon}$ for any $\epsilon > 0$ unless P = NP and it is not approximable within $n^{1-\epsilon}$ for any $\epsilon > 0$ unless co-RP=NP (Håstad, 1996), where n is the number of nodes of the graph. As pointed out in (Wu and Hao, 2015a) the current best-known polynomial-time approximation algorithm achieves only an approximation guarantee of $O(n(loglogn)^2/(logn)^3)$ (Feige, 2004). On the other hand, the study in (Engebretsen and Holmerin, 2003) shows that the MCP is not approximable within a factor of $n/2^{O(log(n)/\sqrt{loglogn}})$ under the assumption that NP $\subseteq ZPTIME(2^{O(logn(loglogn)^{3/2})}$. An improved result shows that MCP is not approximable within $n^{1-\epsilon}$ for any $\epsilon > 0$ unless NP=P (Zuckerman, 2006). A detailed survey on MCP can be found in (Bomze et al., 1999; Pardalos and Xue, 1994; Wu and Hao, 2015a).
Given the practical relevance and theoretical importance of MCP, considerable efforts have been devoted for the development of various algorithms for this problem. On the one hand, effective exact methods have been developed mainly based on the general branch and bound framework (Batsyn et al., 2014; Carraghan and Pardalos, 1990; Li and Quan, 2010; Maslov et al., 2014; McCreesh and Prosser,

2015; Östergård, 2002; San Segundo et al., 2016). These methods have the theoretical advantage of guaranteeing the optimality of the solution found. However, due to the inherent computational complexity of the MCP, exact methods can require a prohibitive computing time in general case and are often applicable to problems of small sizes. On the other hand, to handle problems whose optimal solutions cannot be reached within reasonable time using exact methods various heuristic and meta heuristic algorithms have been developed. The most effective heuristic algorithms for MCP are local search based, some of them worth mentioning are Reactive Local search (Battiti and Protasi, 2001), k-opt local search (Katayama et al., 2005), Phased local search (Pullan, 2006), Cooperating local search (Pullan et al., 2011), breakout local search (Benlic and Hao, 2013) and a very recent general swap based tabu search (Jin and Hao, 2015). A hybrid heuristic methods for MCP has been used in (Singh and Gupta, 2006). Early attempts of using Genetic algorithms (GA) for MCP date back to 1990s. However as concluded in (Carter and Park, 1993) pure GA are not effective for MCP, so this approach is often enhanced by incorporating other techniques like local search optimization. An example is the Heuristic based genetic algorithm proposed in (Marchiori, 1998) which uses GA operators (crossover and mutation) and a local search technique to enhance the quality of obtained solution. A hybrid evolutionary algorithm with guided mutation for MCP is introduced in (Zhang et al., 2005) and a Reactive evolutionary algorithm that used concept of guided mutation and Reactive local search can be found in (Brunato and Battiti, 2011). Other popular heuristics for the MCP include Simulated Annealing (Geng et al., 2007), Artificial Neural Networks (Yang et al., 2009), Ant Colony Optimization (Solnon and Fenet, 2006) and Biomolecular Filtering (Ordóñez-Guillén and Martínez-Pérez, 2015).
It must be understood that exact and heuristic methods constitute two complementing (rather than competing) solution approaches which can be applied in different situations to fulfill different objectives or combined in a complementary way to enhance the quality of the solution.
Since the second DIMACS implementation challenge dedicated to MCP, Graph Coloring and Satisfiability organized during 1992-1993 studies on these NP-hard problems are becoming more and more intense. In this chapter an attempt has been made to solve MCP using Harmony search (HS) algorithm. HS is a musicians inspired meta heuristic algorithm developed in 2001 (Geem et al., 2001). It is worth mentioning that to the best of our knowledge there is only one article available in literature in which an attempt has been made to solve MCP using HS based algorithm namely Binary Harmony Search (BHS) (Afkhami et al., 2013), however BHS has been tested on only 18 out of 80 instances of DIMACS benchmark graphs for MCP and further the results obtained are not very satisfactory.

7.2 Proposed Algorithms

This section gives a detailed description of the proposed algorithm namely Harmony Search Algorithm for Maximum Clique Problem (HS_MCP). The pseudo code of the proposed algorithm is given as Algorithm 10.

7.2.1 Representation and Fitness Evaluation

Let $G = \{V, E\}$ be an undirected graph of N vertices where V={1, 2, 3, ... N} be the set of nodes and $E \subseteq V \times V$ is a set of edges. Each harmony $H \subseteq V$ is represented as a string $H = \{V_1, V_2, V_3, ...V_N\} \in \{0, 1\}^N$ where $V_i = 1$ if and only if V_i is included in H and $V_i = 0$ otherwise. Therefore the search space is $\Omega = \{0, 1\}^N$. The fitness of H is defined as the number of vertices included in H if it represents a valid clique. Since the new harmony generated by HS Algorithm may not be a clique it is repaired so that it becomes a clique and therefore H always represents a valid clique, thus there is no need to define fitness for infeasible solutions. Harmony memory (HM) is a vector of harmonies having size HMS.

7.2.2 Harmony Memory initialization

With a randomly initialized Harmony Memory, the probability of each harmony representing a valid clique is very less due to very large search space of the problem compared to its solution space. So a heuristic approach is utilized to initialize the harmony memory. The algorithm generates each harmony in the HM as follows. Initially a vertex $V_i \in V$ is randomly picked and put in a subset named C. Then a vertex V_j is randomly selected from C's adjacency list P. V_j is added to C and all the nodes in P that are not adjacent to V_j are deleted. Thus P contains only those vertices that are adjacent to all nodes of C. The process is repeated until P becomes empty. Therefore set C represents a clique. Finally harmony H is initialized by assigning 1 to the position corresponding to the vertices in C and otherwise 0. All the harmonies in the HM are initialized following the above procedure. The pseudo code for HM initialization is given as Algorithm 11.

Step number 2 and 3 are executed M times and hence have the time complexity of $O(M)$ whereas the time complexity of step 4 if $O(MN)$. The complexity of step 6 and 7 is $O(MN)$ whereas the worst case complexity of step 8 is $O(MN^2)$. In the worst scenario, step 10 is executed $O(MN)$ times. Thus the time complexity of Harmony Memory initialization is determined by step 8 of algorithm and is $O(MN^2)$ in worst case. Note that M is equal to HMS the size of harmony memory and N is the number of nodes in the graph.

Algorithm 10 Harmony Search Algorithm for MCP (HS_MCP)

```
1:  Initialize Algorithm parameters HMS and HMCR.
2:  Initialize Harmony Memory.
3:  Evaluate each harmony of harmony memory using fitness function.
4:  while Stopping Condition Not Reached do
5:      Find Best and Worst Harmony in HM.
6:      for $i = 1$ TO N do
7:          if rand$(0, 1)$ less than HMCR then
8:              r=rand_int$(1, N)$.
9:              New_Harmony[i]=HM[r][i].
10:         else
11:             if rand$(0, 1) \leq P_i$ then
12:                 New_Harmony[i]=1.
13:             else
14:                 New_Harmony[i]=0.
15:             end if
16:         end if
17:     end for
18:     New_Harmony=Heuristic_Repair(New_Harmony)
19:     Evaluate New_Harmony.
20:     if New_Harmony is better than WORST harmony in HM, update HM by replacing WORST
        harmony by New_Harmony.
21:     if New_Harmony is better than BEST harmony in HM, filter all nodes having degree less than
        size of clique represented by New_Harmony.
22:     /* Step 23 is a Local Search Operator to be executed towards the end of program execution*/
23:     Randomly select a harmony say k from HM ( i.e. HM[k] ), Find a node $V_i$ in HM[k] such that
        deletion of $V_i$ enables us to add two new nodes to HM[k] such that it still remains a clique.
        Repeat this step until no such $V_i$ can be found.
24: end while
25: Print BEST harmony as obtained clique.
```

Algorithm 11 Harmony Memory Initialization Pseudo code

```
1: for $k = 1$ TO HMS do
2:     $C = \emptyset, \quad P = \emptyset$
3:     Randomly select $V_i$ from $V$, $C = C \cup V_i$
4:     $P =$ Vertices that are adjacent to $V_i$
5:     while $P \sim \emptyset$ do
6:         Randomly select a vertex $V_j$ from $P$
7:         $C = C \cup V_j$
8:         $P =$ vertices in $P$ that are adjacent to $V_j$
9:     end while
10:    HM[k] $= C$
11: end for
```

7.2.3 New Harmony Generation

While generating the i^{th} component of the new harmony (represented as New_ harmony) either the i^{th} component of some existing harmony stored in HM is used with probability HMCR or a randomly generated 1 or 0 (with probability P_i and $1 - P_i$ respectively) is assigned to the i^{th} component of New_ harmony. P_i is the probability of vertex $V_i \in HM$. The above procedure is depicted by step number 6 through 17 of Algorithm 10. Once the new harmony (New_Harmony) is generated it is very much likely that New_Harmony won't represent a clique so it is passed as an argument to Heuristic_Repair procedure (step 18 of Algorithm 10) and its job is to convert an arbitrary subgraph into a clique. The pseudo code of Heuristic_Repair procedure is shown as Algorithm 12. Note that in the pseudo code rand ∈ [0, 1] is a uniformly distributed random number generator and rand_int ∈ [L, U] generates an integer random number between L and U following uniform distribution.

7.2.4 Heuristic Repair

The heuristic repair algorithm used in this chapter is proposed in (Marchiori, 1998) and is presented as Algorithm 12. The algorithm has three phases- Enlarge, Extraction and Extension. During Enlarge phase some vertices are randomly added to New_Harmony and hence its time complexity is $O(N)$. In the Extraction phase for all nodes in the New_Harmony either delete the node with probability α or delete all the nodes in the New_Harmony that are not adjacent to it. Thus the time complexity of Extraction phase is $O(N^2)$ in worst case. Generally speaking α must be very small otherwise the clique obtained will be very small due to removal of lot of nodes so we have chosen $\alpha = .1$. Finally in the extension phase a node that is not in New_Harmony is added to New_Harmony if it is adjacent to all nodes of New_Harmony. The above process is repeated for all the nodes of the graph that are not in New_Harmony. The time complexity of Extension phase is $O(N^2)$ in worst case. Thus it is guaranteed that New_Harmony is a clique once returned from the Heuristic_Repair procedure. The overall time complexity of the procedure Heuristic Repair is $O(N^2)$.

7.2.5 Filter Vertices & Local Search operator

Degree of a vertex provides an upper bound on size of the clique the particular vertex can belong to. If the algorithm has already found clique of size say L then all the vertices having degree less than L can't be the part of the clique having size greater than or equal to L. So whenever the size of the largest clique is improved all the vertices having degree less than the size of best clique obtained are filtered out in step number 21 of Algorithm 10 and thus are not considered further while exploring the search space. The performance of this step is not very visible in dense graphs however it can show significant performance in sparse graphs.
The degree of all the nodes in the graph can be calculated with time complexity $O(N^2)$ in the beginning of algorithm using adjacency matrix and hence the time complexity of filtration step is $O(N)$.

The step 23 of Algorithm 10 essentially performs local search around a randomly selected harmony from HM by finding a node whose deletion can enable us to add two nodes thus increasing the size of the clique by 1. Once a node is deleted from the harmony in order to determine if a particular node can be added (so that the harmony still remains a valid clique) requires $O(N)$ comparisons since there are N nodes in the graph thus the complexity becomes $O(N^2)$. In the worst scenario we may have to check for deletion of all the nodes in the harmony and thus resulting in time complexity of $O(N^3)$.
The start of execution of the local search operator depends on the problem in hand, however throughout the experimentation the operator has been executed toward the last five percent of execution time. It was found executing this step from the beginning increases the convergence speed initially but causes the algorithm to get stuck in local optimal and hence deteriorate the quality of solution.

Pitch adjustment in a harmony can be achieved by adding or deleting few nodes from it. However the same thing is done in the Heuristic_Repair procedure so performing pitch adjustment is a redundant step and has been avoided.

7.2.6 Time complexity of proposed algorithm

The worst case asymptotic time complexity of the proposed algorithm HS_MCP has already been discussed in respective section and is summarized in Table 7.1. In Table 7.1 N, M and G respectively denote the number of nodes in the graph, size of harmony memory and maximum number of generations allowed. As is evident from Table 7.1 the worst case time complexity of the proposed algorithm is $O(G \times N^3)$. In order to increasing exploration capabilities of HS_MCP (by reducing exploitation

Table 7.1: Time Complexity of HS_MCP (Algorithm 7).

Step Number	**Time Complexity**
Step 1	$O(1)$
Step 2 (Initialization of Harmony Memory)	$O(MN^2)$
Step 3	$O(MN)$
Step 4	$O(G)$
Step 5	$O(GM)$
Step 6 through 17 (Generating New Harmony)	$O(GN)$
Step 18 (Heuristic Repair)	$O(G \times O(N^2))$
Step 19 (Evaluate Harmony)	$O(G \times N)$
Step 20 (Compare and Update HM)	$O(G \times N)$
Step 21 (Filter Nodes)	$O(G \times N)$
Step 23 (Local Search)	$O(G \times N^3)$

potentiality) it has been modified to create another algorithm namely Harmony Search Algorithm with idiosyncratic harmonies for MCP (HSI_MCP). The only difference between HS_MCP and HSI_MCP is that in HSI_MCP redundant harmonies are not allowed in Harmony Memory, so during initialization step Harmony memory is initialized with idiosyncratic harmonies only and also a New_Harmony is allowed to be added to HM if it is not already present in it. Thus enhancing the exploration capabilities of the algorithm at the cost of reduction in exploitation. Generating HMS number of idiosyncratic harmonies is a time consuming step particulary for graphs having large cliques with respect to their size so during initialization of HM in HMI_MCP, at most N attempts are made to fill HM and incase the HM is not completely filled in N attempts what ever the number of harmonies in HM, is finalized as HMS.

7.3 Experiments Results

In this section proposed algorithms namely HS_MCP and HSI_MCP has been studied experimentally on DIMACS benchmark graphs. These graphs provide a valuable source for evaluating the performance of algorithms for MCP as they arise from different areas of application. More specifically, eighty different graph instances belonging to eleven different families are provided as DIMACS benchmark instances and are available at http://dimacs.rutgers.edu/Challenges. The Brock graphs are generated in such a way that the expected maximum clique is much smaller than the real one. The Cx.y and DSJCx.y are random graphs of size x and density y. The CFat graphs arise from fault diagnosis problems and Gen are artificially generated graphs with large known embedded cliques. The Johnson and Hamming graphs are from coding theory problems and Keller graphs are based on Keller's conjecture on tilings using hypercubes. The Mann family of graphs is from set covering problems and PHat graphs are randomly generated graphs with large variation in the node degree distribution and large cliques than the random graphs, the San and Sanr graphs are random graphs with known cliques. The proposed algorithms HS_MCP and HSI_MCP have been implemented in C programming language and run on a laptop with specifications- windows 10 operating system, core i3 processor and 4GB of RAM. In order to check the efficiency and reliability of the proposed algorithms each problem has been tested 50 times. Finding the optimal setting for free parameter in a meta heuristic algorithm is in itself a challenging task however after through experimentation the following parameter setting for both the algorithms was found out to be effective in majority if not all the cases: HMCR=.95, HMS=15 and the stopping criteria for both the algorithms is 20,000 function evaluations to determine size of the clique.

Figure 7.1 displays the average of the largest clique obtained in fifty independent runs for different settings of HMCR when HMS is set as 15 on the DIMACS benchmark problem C1000.9 (a random graph with 1000 nodes and 0.9 density) and Figure 7.2 displays the average of the largest clique obtained in fifty independent runs for different settings of HMS when HMCR is set as 0.95 on the same graph. It can be observed in Figure 7.1 the average clique size obtained increases with increase in HMCR and once the HMCR touches the value of 0.95 it then starts gradually decreasing. We can observe in Figure 7.2 that the average clique size obtained increases with increase in HMS value and once the HMS touches the value of 15 it then starts gradually decreasing.

In most of the graphs the size of maximum clique contained in it is much smaller than the size of the graph itself and thus the probability of a random node of the graph belonging to maximum clique is very low, if the value of HMCR is low many nodes are randomly added to harmony and the clique obtained during Repair phase of Heuristic_Repair() procedure (Algorithm 12) is suboptimal. Increasing the value of HMS (beyond 15) causes reduction in exploitation of promising areas of search space and thus HS_MCP performs better for higher value of HMCR (0.95) and lower value of HMS (15).

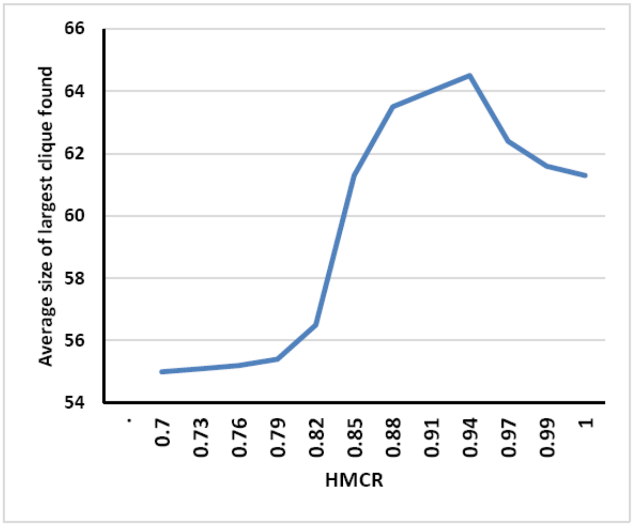


Figure 7.1: Effect of HMCR on average clique size (HS_MCP).

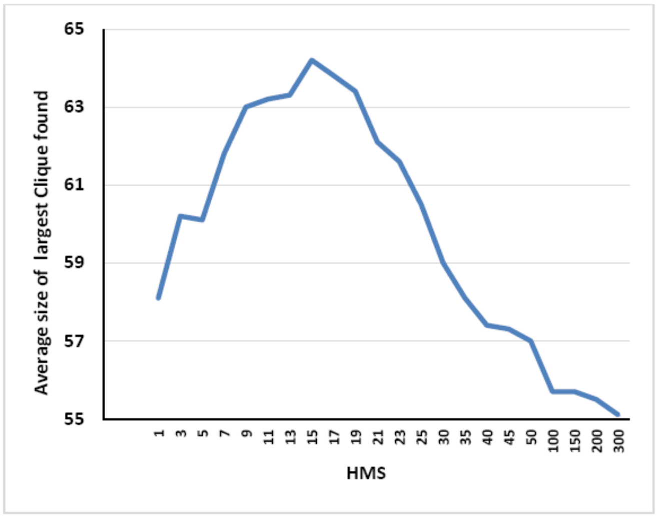


Figure 7.2: Effect of HMS on average clique size (HS_MCP).

7.3.1 Performance of HS_MCP Algorithm

Tables 7.2 & 7.3 summarizes the the performance of the two proposed algorithms. The first three columns labeled as Instance, Order and Density indicate the graph name, Number of nodes and Density of graph respectively and and the fourth column with label ω(G) contains the exact or best known solution. Columns labeled as HS Best, HS Avg and HS SD indicate the best, average and standard deviation of the clique size obtained by HS_MCP algorithm in fifty runs and the column with label HS Time is the average execution time in seconds. The entries in column HS Best are bold where the algorithm was able to produce exact or best known results.

As depicted in Tables 7.2 & 7.3 the performance of HS_MCP algorithm is very satisfactory on CFat and Johnson graphs as it was able to find the best known solution with 100% success rate for all the instances in less than 0.2 and .005 seconds respectively. Similarly the algorithm performed very satisfactory on Hamming graphs where in five out of six instances it was able to find the best solution with 100% success rate and in the sixth instance though the success rate was not 100% but it was able to find the best known solution. Note that success rate is the percentage of runs that were able to find global optimal or best known results.

On four out of five instances the algorithm was able to hit the best known solution in less than 2 seconds on Gen type graphs and on two instances out of three the algorithm was able to find the best known solution in Keller graphs, however on third instance (keller6 which is considered to be a very difficult graph) the algorithm was not able to find the best known solution (58) but still the solution (56) found is close to it.

For Dsjc type graphs the algorithm was able to hit the best known solution for both instances. For Mann family of graphs on two instances out of four the algorithm was able to find the best known solution and for remaining two instances the obtained solution is very close to global optimal.

For Brock family of graphs the performance of the algorithm is satisfactory on instances with 200 nodes, where in all the four instances the algorithm was able to hit the global optimal and for instances with 400 and 800 nodes it is unsatisfactory.

The performance is satisfactory on graphs of type Cx.y as on four out of seven instances the algorithm was able to hit best known solution and in the remaining three cases the solution obtained is close to best known solution. For Phat family of graphs with 300, 700 and 1500 nodes the algorithm performed satisfactory as it was able to find best known results on all the 12 instances of these graphs, however for Phat graphs with 500 and 1000 nodes the performance was unsatisfactory as there is significant difference between the best known results and the results obtained.

For San graphs with 200 nodes the algorithm performance is satisfactory where in four out of five instances the best known results were obtained and for San graphs with 400 nodes the performance is unsatisfactory because in only one out of five instances the best known solution was found and in the remaining four instances there is significant difference between the best known results and the obtained results.

For Sanr graphs in only one instance out of four best solution was found and in remaining three instanced there is significant difference between the best known and obtained solution.

In summary the algorithm is able to find the best known results in 50 out of 80 instances of DIMACS graphs, further in terms of time complexity the only time consuming graph is MANN_a81 taking around 6.5 minutes to converge and for 79 remaining instances the algorithm always converges in less than 36 seconds.

7.3.2 Performance of HSI_MCP Algorithm

Tables 7.4 & 7.5 summarize the performance of HSI_MCP algorithm. The columns with label HSI Best, HSI Avg, HSI SD respectively denote the Best, Average and Standard deviation of the clique size obtained by HSI_MCP algorithm in fifty runs and the column with label HSI Time indicated the average execution time in seconds. The entries in column HSI Best are bold where the algorithm was able to produce exact or best known results.

As depicted in Tables 7.4 & 7.5 the performance of HSI_MCP algorithm is very satisfactory on Johnson and Cfat graphs where the best known results were obtained in all the eleven instances with 100% success rate, and on Hamming graphs the performance is also satisfactory where in 5 out of 6 instances the global optimal was obtained with 100% success rate.

For Mann type graphs the performance is rather satisfactory where in 2 out of 4 instances best known results were obtained and for remaining 2 instances the obtained results are close to best known results.

For Dsjc graphs the performance is rather satisfactory where in one instance out of two best known results are obtained and in another instance the difference between the exact and obtained solution is just 2.

For Cx.y and Gen graphs the performance is unsatisfactory as in both the families the global optimal was hit just in one instance and in other instances the difference between the obtained and exact solution is significantly large.

For Keller type graphs the global optimal is found in one instance out of three and in second instance the difference between the actual and the obtained result is just 1, however for third instance (keller3) the difference between the best known result (58) and the obtained result (52) is 6.

For Brock family of graphs the performance is unsatisfactory as in just one instance best known results are obtained and in remaining eleven instances the obtained results are very inferior to actual results, similarly for Phat graphs even though best known results are obtained in four instances however in

remaining 11 instances the performance is unsatisfactory.
For San type of graphs the exact solution was hit in just one instance and for remaining instances the results obtained are not satisfactory and in Sanr type of graphs the exact solution was never hit and the results obtained are very inferior to actual results.

7.3.3 Comparison of HS_MCP and HSI_MCP

Comparing the performance of the two proposed algorithms it is very much evident from Table 7.4 & 7.5 that HS_MCP has out performed HSI_MCP in all families of graphs both in terms of solution quality and time complexity. In HS_MCP there is no constraint on uniqueness of harmonies as a result multiple copies of best harmonies are kept in HM and thus increasing the exploitation of the promising search areas on the other hand due to uniqueness constraint on harmonies in HM the exploration of the search space is increased on the cost of reducing exploitation of the promising search areas.
As is evident from the obtained results the algorithm having better exploitation capabilities rather than exploration has produces better results for this particular problem.
Generating idiosyncratic harmonies to fill HM is a time consuming process particularly for graphs having large cliques with respect to their size and that is the reason for higher time complexity of HSI_MCP comparing to HS_MCP e.g MANN_a81 graph has the biggest maximum clique with respect to its size and HSI_MCP approximately took 242 times more time than HS_MCP to converge for this problem.

7.3.4 Comparison of HS_MCP and Binary Harmony Search

Since HS_MCP is the better performing algorithm than HSI_MCP, it has been compared with Binary Harmony Search (BHS) algorithm proposed in (Afkhami et al., 2013). The stopping criteria for both BHS and HS_MCP is 20,000 function evaluations to determine size of the clique. Table 7.6 summarizes the comparison between HS_MCP and BHS on all the 18 graph instances on which BHS has been tested in (Afkhami et al., 2013). The columns labeled as instance and ω(G) respectively indicate name of the graph and exact or best known result, the column with label BHS Best and BHS Avg, respectively denote the best and average of the clique size obtained by BHS algorithm. The columns with label HS Best and HS Avg respectively denote the best and average of the clique size obtained by HS_MCP algorithm in fifty independent runs. The columns with label BHS Time and HS Time denote the average time taken in seconds by BHS and HS_MCP algorithm respectively. The entries in the column HS Best and BHS Best have been made bold where the corresponding algorithms have produced the exact or best known results, further the entries in HS Best and HS Avg have been marked by * where the results are better than corresponding BHS Best and BHS Avg results respectively similarly the entries in BHS Best and BHS Avg have been marked with * where the results are better than corresponding HS Best and HS Avg results.

Comparing the best results (HS Best, BHS Best) obtained by the two algorithms HS_MCP has outperformed BHS in 10 instances and in the remaining 6 instances both algorithms has produced same results, thus there is not a single instance in which HS_MCP has been outperformed by BHS, further HS_MCP was able to find the best known results in 16 out of 18 instances whereas BHS was able to find the best known results in only 8 instances. Comparing the average results in 10 instances HS_MCP has outperformed BHS and in 6 instances BHS has outperformed HS_MCP. Even though BHS has been run on a machine with better specifications (i7 processor, 4GB RAM) than the one on which HS_MCP was run still in 7 instances HS_MCP has out performed BHS algorithm in terms of time complexity. Thus it can be concluded that HS_MCP has significantly outperformed BHS particularly in terms of solution quality.

7.3.5 Comparison of HS_MCP with exact algorithms

In order to provide a glimpse of effectiveness of exact algorithms on MCP a recently developed state-of-the-art exact algorithm proposed by McCreesh and Prosser (McCreesh and Prosser, 2013) has been chosen and we call it Multi threading MCS (MTMCS). The Multi threading MCS is based on series of branch and bound algorithm proposed by Tomita (Tomita and Kameda, 2007; Tomita and Seki, 2003; Tomita et al., 2010) and is an efficient parallel algorithm that has reported super linear speed up for most of the DIMACS benchmark graphs. To the best of our knowledge MTMCS is the only exact algorithm that has been tested on large DIMACS graphs having nodes greater than 2,000 and has shown promising results. Table 7.7 shows the performance of MTMCS algorithm on selected DIMACS graphs. The first column in Table 7.7 represents the graph instance, second column is the maximum clique and the third column is the time taken by MTMCS algorithm along with the specifications of the machine on which it was run. As already reported MTMCS has shown super linear speed up for most of the benchmark graphs (and at least near linear speedup otherwise) the algorithm will defiantly take much more time if run on a sequential machine.

Our HS_MCP algorithm though run on a rather less powerful machine is able to find global optimal in gen400_p0.9_65 in only few seconds compared to 4.93 hours taken by MTMCS algorithm, similarly it is able to find global optimal in p_hat1500-3 in only few seconds compared to 128 days taken by MTMCS algorithm. HS_MCP is able to find very competitive results in C4000.5 (16 rather than 18) in few seconds only compared to MTMCS taking 19 days similarly HS_MCP is able to find competitive results in mann_a81 (1096 rather than 1100) in few minutes whereas MTMCS took 128 days.

In (Tomita et al., 2010) execution time reported for solving gen400_p0.9_65 using exact algorithms MCR (Tomita and Kameda, 2007) and MCS (Tomita et al., 2010) is greater than 115 days and 1.75 days respectively and for gen400_p0.9_75 it is greater than 115 days and 3.4 days respectively on a machine with Pentium4 3.6 GHz processor, compared to our HS_MCP algorithm obtaining global optimal in only few seconds for these two instances. The results of other three graph instances in Table III have not been reported in (Tomita et al., 2010).
Exact and heuristic methods must be considered as complementing rather than competing methodologies and each can be used in different situations to fulfill different objectives, these to methodologies can even be combined to produce superior procedures for problem solving, an excellent example in context of MCP is the recently proposed branch and bound based exact algorithm by Batsyn et al. (Batsyn et al., 2014) (we call it Improved MCS). In Improved MCS algorithm ILS heuristic (Andrade et al., 2012) is used to obtain an initial high quality solution which is then used to prune branching in branch and bound based MCS algorithm (Tomita et al., 2010). Improved MCS has been compared with MCS algorithm on DIMACS benchmark graphs and tremendous improvement has been reported particularly on big and dense graphs.

7.4 Conclusion

In this chapter a harmony search based algorithms namely HS_MCP has been proposed to address the computational limitation of classic methods for solving maximum clique problem particularly in large graphs. To increasing exploration capabilities of HS_MCP (by reducing exploitation potentiality) it has been modified to create another algorithm namely HSI_MCP. Since HS_MCP has significantly outperformed HSI_MCP on all DIMACS benchmark graphs leading to the conclusion that exploitation of promising search areas produces better results rather than exploration of the search space for MCP. Further it has been concluded that HS_MCP is very effective on problems of type Johnson, Hamming, Cfat, Gen, MANN and Keller. The results of HS_MCP has been compared with BHS algorithm proposed in (Afkhami et al., 2013) and it has been concluded that HS_MCP is the better performing algorithm particularly in terms of solution quality. On comparing HS_MCP with exact algorithm it has been concluded that heuristic and meta heuristic algorithms have a very significant role in solving MCP. Most of the algorithms for MCP including the algorithms proposed in (Afkhami et al., 2013; Batsyn et al., 2014; Brunato and Battiti, 2011; Katayama et al., 2005; Marchiori, 1998; Ordóñez-Guillén and Martínez-Pérez, 2015; Solnon and Fenet, 2006; Zhang et al., 2005) have reported results only on selected benchmark graphs however in this chapter we have been reported results on all the DIMACS benchmarks graphs so that not only the strength but also weakness of the proposed algorithms is highlighted, so that researchers in future can focus on weak spots and make the algorithms more efficient. In this chapter the focus was on Harmony Search algorithm and thus literature review on meta heuristic like genetic algorithms, tabu search, ACO, simulated annealing for MCP has not been given in detail.

Algorithm 12 Heuristic Repair Algorithm

```
Procedure Heuristic_Repair(New_Harmony)
1. Relax:(Enlarge the subgraph)
Add few new vertices randomly chosen from the graph to New_Harmony.
2. Repair:(Extract the clique)
Choose a random position Pos with $1 \leq pos \leq N$
for $i = Pos$ TO N do
  if New_Harmony[i]=1 then
    if rand$(0, 1)$ less than $\alpha$ then
      Set New_Harmony[i]=0
    else
      for $j = i + 1$ TO N do
        Set New_Harmony[j]=0 if $j^{th}$ vertex belongs to the New_Harmony and $j^{th}$ vertex is not
        connected to $i^{th}$ vertex.
      end for
      for $j = 1$ TO $i - 1$ do
        Set New_Harmony[j]=0 if $j^{th}$ vertex belongs to the New_Harmony and $j^{th}$ vertex is not
        connected to $i^{th}$ vertex.
      end for
    end if
  end if
end for
for $i = Pos - 1$ downto 1 do
  if New_Harmony[i]=1 then
    if rand$(0, 1)$ less than $\alpha$ then
      Set New_Harmony[i]=0
    else
      for $j = i - 1$ downto 1 do
        Set New_Harmony[j]=0 if $j^{th}$ vertex belongs to the New_Harmony and $j^{th}$ vertex is not
        connected to $i^{th}$ vertex.
      end for
      for $j = N$ TO $i + 1$ do
        Set New_Harmony[j]=0 if $j^{th}$ vertex belongs to the New_Harmony and $j^{th}$ vertex is not
        connected to $i^{th}$ vertex.
      end for
    end if
  end if
end for
3. Extend:(Enlarge the clique)
Choose a random position Pos with $1 \leq pos \leq N$
for $j = Pos$ TO N do
  add $j^{th}$ vertex if it is not in New_Harmony and is connected to all nodes of New_Harmony.
end for
for $j = 1$ TO $Pos - 1$ do
  add $j^{th}$ vertex if it is not in New_Harmony and is connected to all nodes of New_Harmony.
end for
return New_Harmony
End Procedure
```

Table 7.2: Performance of HS_MCP on DIMACS benchmark Graphs.

Instance	Order	Density	ω(G)	HS Best	HS Avg	HS SD	HS Time
brock200_1	200	0.745	21	**21**	20	0.775	0.575
brock200_2	200	0.496	12	**12**	10.7	0.9	0.441
brock200_3	200	0.605	15	**15**	11.3	0.64	0.58
brock200_4	200	0.658	17	**17**	16.10	0.300	0.628
brock400_1	400	0.748	27	21	20	0.632	1.208
brock400_2	400	0.749	29	25	24.1	0.3	1.517
brock400_3	400	0.748	31	22	20.9	0.7	1.247
brock400_4	400	0.749	33	25	24	0.447	1.51
brock800_1	800	0.649	23	11	10.4	0.49	1.925
brock800_2	800	0.651	24	20	19.3	0.458	2.722
brock800_3	800	0.649	25	10	9.8	0.4	1.904
brock800_4	800	0.65	26	20	18.9	0.7	2.753
c125.9	125	0.898	34	**34**	33.6	0.917	0.672
c250.9	250	0.899	44	**44**	42.7	0.9	1.005
c500.9	500	0.9	57	**57**	53.9	1.044	2.586
c1000.9	1000	0.901	68	66	62.70	1.418	5.444
c2000.5	2000	0.5	16	**16**	14.7	0.458	7.524
c2000.9	2000	0.9	77	71	68.4	2.01	12.567
c4000.5	4000	0.5	18	16	15.40	0.490	15.358
c-fat2001	200	0.077	12	**12**	12	0	0.001
c-fat2002	200	0.163	24	**24**	24	0	0.005
c-fat2005	200	0.426	58	**58**	58	0	0.016
c-fat5001	500	0.036	14	**14**	14	0	0.003
c-fat5002	500	0.073	26	**26**	26	0	0.01
c-fat5005	500	0.186	64	**64**	64	0	0.048
c-fat50010	500	0.374	126	**126**	126	0	0.177
dsjc500.5	500	0.5	13	**13**	12.1	0.3	1.529
dsjc1000.5	1000	0.5	15	**15**	13.70	0.64	3.464
gen200_p0.9_44	200	0.9	44	**44**	39	2.569	0.854
gen200_p0.9_55	200	0.9	55	**55**	45.9	7.476	0.63
gen400_p0.9_55	200	0.9	55	41	39.7	0.64	1.467
gen400_p0.9_65	400	0.9	65	**65**	50	5.099	1.793
gen400_p0.9_75	400	0.9	75	**75**	55.8	9.724	1.624
hamming6-2	64	0.905	32	**32**	32	0	0.002
hamming6-4	64	0.349	4	**4**	4	0	<0.001
hamming8-2	256	0.969	128	**128**	128	0	0.033
hamming8-4	256	0.639	16	**16**	16	0	0.003
hamming10-2	1024	0.99	512	**512**	512	0	1.325
hamming10-4	1024	0.829	40	**40**	38.3	2.002	2.571

Table 7.3: Performance of HS_MCP on DIMACS benchmark Graphs.

Instance	Order	Density	ω(G)	HS Best	HS Avg	HS SD	HS Time
johnson8-24	28	0.556	4	**4**	4	0	<0.001
johnson8-44	70	0.768	14	**14**	14	0	0.001
johnson16-24	120	0.765	8	**8**	8	0	<0.001
johnson32-24	496	0.879	16	**16**	16	0	0.004
keller4	171	0.649	11	**11**	11	0	0.002
keller5	776	0.751	27	**27**	26.4	0.8	1.644
keller6	3361	0.818	58	56	52.9	1.758	16.792
mann_a9	45	0.927	16	**16**	16	0	<0.001
mann_a27	378	0.99	126	**126**	125.4	0.49	3.087
mann_a45	1035	0.996	345	343	342.2	0.6	35.618
mann_a81	3321	0.999	1100	1096	1094.9	2.663	392.149
p_hat300-1	300	0.244	8	**8**	8	0	0.089
p_hat300-2	300	0.489	25	**25**	24.8	0.4	0.245
p_hat300-3	300	0.744	36	**36**	34.2	1.249	0.899
p_hat500-1	500	0.253	9	6	6	0	1.384
p_hat500-2	500	0.505	36	15	15	0	1.358
p_hat500-3	500	0.752	50	35	33.7	1.616	1.649
p_hat700-1	700	0.249	11	**11**	9.4	0.663	1.701
p_hat700-2	700	0.498	44	**44**	43	1	1.664
p_hat700-3	700	0.748	62	**62**	61	0.447	3.236
p_hat1000-1	1000	0.245	10	5	5	0	2.412
p_hat1000-2	1000	0.49	46	13	13	0	2.493
p_hat1000-3	1000	0.744	68	23	22.9	0.3	2.612
p_hat1500-1	1500	0.253	12	**12**	10.80	0.600	4.630
p_hat1500-2	1500	0.506	65	**65**	63.1	1.3	7.126
p_hat1500-3	1500	0.754	94	**94**	92.4	0.917	9.223
san200_0.7_1	200	0.7	30	18	18	0	0.678
san200_0.7_2	200	0.7	18	**18**	17.7	0.9	0.618
san200_0.9_1	200	0.9	70	**70**	65.6	8.8	0.269
san200_0.9_2	200	0.9	60	**60**	46.5	8.857	0.747
san200_0.9_3	200	0.9	44	**44**	38.3	2.865	0.903
san400_0.5_1	400	0.5	13	8	8	0	0.989
san400_0.7_1	400	0.7	40	18	18	0	1.11
san400_0.7_2	400	0.7	30	15	15	0	1.089
san400_0.7_3	400	0.7	22	15	14.8	0.4	1.117
san400_0.9_1	400	0.9	100	**100**	78.4	21.625	1.122
san1000	1000	0.502	15	9	9	0	2.487
sanr200_0.7	200	0.697	18	11	11	0	0.55
sanr200_0.9	200	0.898	42	**42**	41	0.447	0.965
sanr400_0.5	400	0.501	13	7	7	0	1.027
sanr400_0.7	400	0.7	21	18	16.3	0.781	1.133

Table 7.4: Performance of HSI_MCP on DIMACS benchmark Graphs.

Instance	Order	Density	ω(G)	HSI Best	HSI Avg	HSI SD	HSI Time
brock200_1	200	0.745	21	19	15	1.897	0.11
brock200_2	200	0.496	12	**12**	8.2	1.536	0.077
brock200_3	200	0.605	15	12	3.5	3.956	0.072
brock200_4	200	0.658	17	16	11.1	1.921	0.097
brock400_1	400	0.748	27	20	15.3	1.792	0.208
brock400_2	400	0.749	29	22	17.9	1.972	0.346
brock400_3	400	0.748	31	21	11.9	7.368	0.214
brock400_4	400	0.749	33	24	18	2.366	0.34
brock800_1	800	0.649	23	11	2.7	3.466	0.218
brock800_2	800	0.651	24	18	14.3	1.792	0.98
brock800_3	800	0.649	25	10	1.9	2.7	0.211
brock800_4	800	0.65	26	18	14.7	1.418	0.997
c125.9	125	0.898	34	**34**	26.2	2.993	0.089
c250.9	250	0.899	44	41	33.4	3.04	0.302
c500.9	500	0.9	57	53	39.7	4.562	1.39
c1000.9	1000	0.901	68	59	45.5	4.674	6.97
c2000.5	2000	0.5	16	14	10.4	1.356	4.267
c2000.9	2000	0.9	77	65	51.7	4.649	34.74
c4000.5	4000	0.5	18	15	12.5	1.36	17.349
c-fat2001	200	0.077	12	**12**	12	0	0.026
c-fat2002	200	0.163	24	**24**	24	0	0.068
c-fat2005	200	0.426	58	**58**	58	0	0.333
c-fat5001	500	0.036	14	**14**	14	0	0.133
c-fat5002	500	0.073	26	**26**	26	0	0.442
c-fat5005	500	0.186	64	**64**	64	0	2.391
c-fat50010	500	0.374	126	**126**	126	0	8.86
dsjc500.5	500	0.5	13	**13**	9.4	1.685	0.287
dsjc1000.5	1000	0.5	15	13	9.9	1.375	1.012
gen200_p0.9_44	200	0.9	44	**44**	33.4	6.053	0.194
gen200_p0.9_55	200	0.9	55	39	30.7	3.378	0.209
gen400_p0.9_55	200	0.9	55	41	16.1	15.514	0.351
gen400_p0.9_65	400	0.9	65	51	37.8	4.771	0.838
gen400_p0.9_75	400	0.9	75	52	39.7	4.818	0.885
hamming6-2	64	0.905	32	**32**	32	0	0.006
hamming6-4	64	0.349	4	**4**	4	0	<0.001
hamming8-2	256	0.969	128	**128**	128	0	0.75
hamming8-4	256	0.639	16	**16**	16	0	0.021
hamming10-2	1024	0.99	512	**512**	512	0	128.771
hamming10-4	1024	0.829	40	36	21.9	5.108	1.848

Table 7.5: Performance of HSI_MCP on DIMACS benchmark Graphs.

Instance	Order	Density	ω(G)	HSI Best	HSI Avg	HSI SD	HSI Time
johnson8-24	28	0.556	4	**4**	4	0	<0.001
johnson8-44	70	0.768	14	**14**	14	0	0.004
johnson16-24	120	0.765	8	**8**	8	0	0.005
johnson32-24	496	0.879	16	**16**	16	0	0.211
keller4	171	0.649	11	**11**	11	0	0.012
keller5	776	0.751	27	26	17.3	2.968	0.788
keller6	3361	0.818	58	52	35.1	6.139	35.485
mann_a9	45	0.927	16	**16**	16	0	0.003
mann_a27	378	0.99	126	**126**	121.7	3.348	4.792
mann_a45	1035	0.996	345	342	333.6	3.072	1188.198
mann_a81	3321	0.999	1100	1096	1088.00	8	94766.219
p_hat300-1	300	0.244	8	**8**	7.3	1.187	0.101
p_hat300-2	300	0.489	25	**25**	25	0	0.133
p_hat300-3	300	0.744	36	34	22.5	4.5	0.265
p_hat500-1	500	0.253	9	6	1.8	1.661	0.123
p_hat500-2	500	0.505	36	15	4.7	4.961	0.152
p_hat500-3	500	0.752	50	35	15.4	12.314	0.333
p_hat700-1	700	0.249	11	9	6.3	1.187	0.316
p_hat700-2	700	0.498	44	**44**	26.2	8.506	1.469
p_hat700-3	700	0.748	62	**62**	38.2	12.335	2.348
p_hat1000-1	1000	0.245	10	5	1.7	1.418	0.269
p_hat1000-2	1000	0.49	46	13	2.2	3.6	0.283
p_hat1000-3	1000	0.744	68	23	3.2	6.6	0.377
p_hat1500-1	1500	0.253	12	10	7.2	1.249	1.131
p_hat1500-2	1500	0.506	65	64	30.6	11.456	9.856
p_hat1500-3	1500	0.754	94	92	49.8	14.421	16.489
san200_0.7_1	200	0.7	30	18	12.6	4.543	0.098
san200_0.7_2	200	0.7	18	17	17	0	0.812
san200_0.9_1	200	0.9	70	**70**	55	15.925	0.296
san200_0.9_2	200	0.9	60	41	32.8	4.069	0.215
san200_0.9_3	200	0.9	44	36	28.2	2.993	0.188
san400_0.5_1	400	0.5	13	8	2.2	2.441	0.125
san400_0.7_1	400	0.7	40	18	4.2	6.416	0.119
san400_0.7_2	400	0.7	30	15	3.8	5.6	0.126
san400_0.7_3	400	0.7	22	15	5.7	5.832	0.143
san400_0.9_1	400	0.9	100	55	45.1	7.713	1.206
san1000	1000	0.502	15	9	2.4	2.835	0.299
sanr200_0.7	200	0.697	18	11	4	3.847	0.062
sanr200_0.9	200	0.898	42	41	31.2	4.308	0.207
sanr400_0.5	400	0.501	13	7	2.5	2.335	0.098
sanr400_0.7	400	0.7	21	15	9.1	5.394	0.15

Table 7.6: Comparison of HS_MCP and BHS algorithms on DIMACS Benchmark Graphs.

Instance	ω(G)	HS Best	HS Avg	HS Time	BHS Avg	BHS Best	BHS Time
brock200_2	12	**12**	10.7	0.441	11.23*	**12**	0.04
brock200_4	17	**17***	16.10*	0.628	14.1	15	0.06
brock400_2	29	25*	24.1*	1.517	20.3	21	0.14
brock400_4	33	25*	24*	1.51	21	21	0.31
c125.9	34	**34**	33.6	0.672	34*	**34**	1.01
c500.9	57	**57***	53.9*	2.586	45.5	46	0.47
gen200_p0.9_44	44*	**44***	39	0.854	39.4*	40	11.89
gen200_p0.9_55	55	**55**	45.9	0.63	55*	**55**	13.8
hamming8-4	16	**16**	16*	0.003	15.9	**16**	0.05
hamming10-4	40	**40***	38.3*	2.571	31.5	32	0.54
keller4	11	**11**	11	0.002	11	**11**	0.02
keller5	27	**27***	26.4*	1.644	23	24	0.7
mann_a27	126	**126***	125.4*	3.087	105.9	106	0.87
p_hat300-1	8	**8**	8	0.089	8	**8**	0.05
p_hat300-2	25	**25**	24.8	0.245	25*	**25**	8
p_hat300-3	36	**36**	34.2	0.899	36*	**36**	21.72
p_hat700-1	11	**11***	9.4*	1.701	9	9	0.22
p_hat700-2	44	**44***	43*	1.664	42	42	31.9

Table 7.7: Performance of Multi threading MCS algorithms on DIMACS Benchmark Graphs.

Instance	ω(G)	MTMCS Time
C4000.5	18	19 days using 32 threads on a 16-core hyper-threaded dual Xeon E5-2660 shared with other users
gen400_p0.9_65	65	4.99 days when run sequentially and 4.93 hours using 24 threads on a 12-core hyper-threaded dual Xeon E5645
gen400_p0.9_75	75	2.86 days when run sequentially on a computer with two 2.4GHz Intel Xeon E5645 processors
mann_a81	1100	31 days using 24 threads on a 12-core hyper-threaded dual Xeon E5645 shared with other users
p_hat1500-3	94	128 days using 32 threads on a 16-core hyper-threaded dual Xeon E5-2660 shared with other users

Chapter 8
Optimization of Lycopene extraction from tomato processing waste skin using Harmony Search Algorithm

Skin, rich in lycopene, is an important component of waste originating from tomato (lycopersicon esculentum) paste manufacturing plants. The lycopene content present in the skin fraction of tomato pomace is about 5 times higher than in the pulp . According to the World Processing Tomato Council 1,200,000 tons of tomato processing waste is produced worldwide annually. In this chapter a central composite design with five independent variables, namely solvent/meal , number of extractions , temperature, particle size, extraction time is used to study their effects on the extraction of lycopene from tomato skin. Prior art response surface analysis has been used to optimize the lycopene yield with respect to the above mentioned five independent variables and the maximum yield predicted was 1.99 mg/100 g. In this study music inspired Harmony Search (HS) metaheuristic algorithm is used to optimize the lycopene production and the maximum lycopene (4.8 mg/100 g) was predicted when the solvent/meal ratio, number of extractions, temperature, particle size and extraction time is 20:1 (v/w), 5, 60°C, 0.43 mm and 4 minutes, respectively.
The organization of this chapter is as follows. Section 8.1 provides the background art and Section 8.2 describes the methods and material used. The formulation of problem is described in Section 8.3. Section 8.4 describes the proposed procedure for solving the optimization problem at hand using HS algorithm and finally Section 8.5 provides conclusion & claim.

8.1 Literature Review

Lycopene belongs to the carotenoid family, is a bright red pigment that has received great interest due to its various biological activities. Lycopene is a potent antioxidant and has been found effective in reducing the risk of chronic diseases by protecting cells against oxidative damage (Rao and Agarwal, 1999). Various studies have shown that lycopene is associated with decreasing the risk of breast and prostate cancer (Chalabi et al., 2006; Holzapfel et al., 2013). Studies have also revealed its preventive effect on cardiovascular and coronary heart diseases. Lycopene exhibits anti-inflammatory activity by inhibiting the activation of inducible nitric oxide syntheses proteins (Rafi et al., 2007). Lycopene prevents low density lipoprotein oxidation and thus helps to reduce blood cholesterol levels (Rao and Agarwal, 1999). In food industry lycopene is used as a food additive to enhance nutritional properties and storage stability (Østerlie and Lerfall, 2005).

According to the World Processing Tomato Council 1,200,000 tons of tomato processing waste is produced worldwide annually (Zuorro et al., 2011). One of the main components of tomato processing waste is skin. The lycopene content present in the skin fraction of tomato pomace is about 5 times higher than in the pulp (Papaioannou and Karabelas, 2012). In common variants of tomato lycopene is found at a concentration of 4-14.3mg/100g (Kaur et al., 2006). At present, the tomato processing waste is either discarded or used as animal fodder, but its abundance in lycopene makes it a promising prospect as a sustainable, alternative and low-cost source of this nutraceutical compound.

Kaur et al. studied the effect of five independent values namely solvent/meal ratio, number of extractions, temperature, particle size and extraction time on lycopene extraction (Kaur et al., 2008). The optimal yield of lycopene predicted using Response Surface Analysis is 1.97mg/100g and the optimal yield experimentally obtained is 1.99mg/100g (Kaur et al., 2008). Poojary and Passamonti studied the effect of using acetone/n-hexane mixtures at different ratios (1:3, 2:2 and 3:1, v/v) and at different temperatures (30, 40 and 50°C) on lycopene extraction and the yield obtained was in the range 3.47-4.03mg/100g (Poojary and Passamonti, 2015) .

European Patent EP 1103579 discloses a method of extracting lycopene from tomato pomace, pink grapefruit, watermelon, guava and papaya. The process consists of removing the impurities and water from the product by subjecting the product to at least one washing with boiling ethanol having water content from 20% to 30%, then high content of hot ethanol is used to extract lycopene. Although this extraction process is relatively simple, it leads to low yield of lycopene due to very low solubility of lycopene in ethanol.

U.S. Pat. No. 5837311 discloses a process of obtaining oleoresin containing lycopene, including crushing tomato as raw material, pulping extracting with solvent and other processes. This process uses tomatoes as raw material without the treatment of dehydration, resulting in a low yield; a large amount of solvent is needed in this process, resulting in high cost and making large scale production unsuitable.

Chinese Patent Application Publication No. CN101449801A discloses processes for extracting lycopene as: Tomato skins are taken as raw material, followed by crushing the material with colloid grinder, then dehydrating with ethanol and extracting with organic solvent to produce lycopene.

Chinese Patent Application Publication Nos. CN1799674, CN1334328A and CN101298618A respectively disclose processes for extracting lycopene using tomato skins as raw material and applying supercritical carbon dioxide extraction. The use of supercritical extraction device increases the cost

and affects the industrial scale of producing lycopene.
Chinese Patent Application Publication No. CN101121631A discloses that tomatoes and tomato pomace are taken as raw materials, and extracted by using ultra-sound/microwave synergistic technology associated with solvent to obtain lycopene. Although the yield of lycopene may be relatively high using this process, it cannot be used in large scale industrial production due to use of ultra-sound/microwave technology, resulting in less practicability.

8.2 Material and methods

The method for extracting lycopene in this chapter is exactly the same as defined in (Kaur et al., 2008) and is briefly introduced in this section.

8.2.1 Material

Tomato pomace was obtained from a tomato past manufacturing unit located in Haridwar, Uttarakhand (India).

8.2.2 Sample Preparation

Skin was separated from pomace, obtained from tomato paste manufacturing unit by a continuous floatation-cum-sedimentation system. The separated skin was dried in a cabinet dryer according to the methods of (Kaur et al., 2006). The dried skin was ground in a mixer and then passed through different sieves of size 0.05, 0.15, 0.25, 0.35 and 0.43 mm.

8.2.3 Proximate Analysis

Moisture, ash, crude protein, crude fibre and crude fat content were determined according to (AoA, 1990). Carbohydrates are computed by subtracting percent content of all the above components from one hundred.

8.2.4 Pigment Extraction

Sample (1g) was extracted using solvent (hexane:acetone:alcohol 2:1:1) containing 0.05% (w/v) butylated hydroxytoluene (BHT). Cold distilled water (15 ml) was added and the suspension was agitated. The solution was then allowed to stand for 15 minutes for separation of polar and non-polar layers (Figure 8.1). The polar layer, containing lycopene was obtained and the absorbance was measured using a UV visible spectrophotometer (Figure 8.2) at 471 nm and expressed as mg/100 g using an extinction coefficient of 17.2×10^4 mol cm^{-1}.

8.3 Problem Formulation

The object of this chapter is to provide optimal setting of five input parameters for production of lycopene from tomato skin obtained as a waste originating from tomato paste manufacturing plants.

The effects of five independent variables X_1 (solvent/meal ratio), X_2 (number of extractions), X_3 (temperature), X_4 (particle size) and X_5 (time) on lycopene extraction is studied in (Kaur et al., 2008) and in this section the process is briefly introduced. Table 8.1 shows the independent variables and their levels used for central composite design in (Kaur et al., 2008). Thirty two combinations of the independent variables along with their experimental and predicted yield are shown in Table 8.2 (reproduced from (Kaur et al., 2008)). Data pertaining to five independent variables and one response variable were analyzed to get a quadratic regression equation of the form of Equation 8.1 and the regression equation is given as Equation 8.2 (Kaur et al., 2008).

$$Y = b_0 + \sum_{n=1}^{5} b_n X_n + \sum_{n=1}^{5} b_{mn} X_n^2 + \sum_{n<m}^{5} b_{mn} X_n X_m \tag{8.1}$$

where Y is the lycopene yield (mg/g), b_0 is the value for the fixed response at the central point of the experiment. b_n, b_m, b_{nm} are the linear, quadratic and cross product coefficients, respectively.

$$\begin{aligned} Y = 0.8615 - 0.0384X_1 + 0.1078X_2 + 0.0805X_3 + 0.0669X_4 + 0.0619X_5 + 0.11920011X_1^2 \\ +0.1155X_2^2 + 0.1452X_3^2 - 0.0127X_4^2 + 0.0468X_5^2 - 0.1133X_1X_2 - 0.0687X_1X_3 \\ -0.0019X_1X_4 + 0.0204X_1X_5 + 0.0985X_2X_3 + 0.0019X_2X_4 + 0.0167X_2X_5 \\ +0.0761X_3X_4 - 0.0798X_3X_5 - 0.0204X_4X_5 \end{aligned} \tag{8.2}$$

The predicted values of lycopene content were calculated using the regression model and compared with experimental values. The value for the coefficient of determination (R^2) was 0.99 which indicates the adequacy of the applied model. The statistical analysis of data revealed that linear, quadratic and interaction coefficients were significant. In (Kaur et al., 2008) the levels of independent variables for optimal extraction conditions of lycopene content were determined using response surface graphs plotted between two independent variables while remaining independent variables were kept at zero level and optimal predicted lycopene yield using response surface analysis is 1.97 mg/100g at parameter setting 30:1 v/w solvent/meal ratio, four extractions, 50°C temperature, 0.15 mm particle size and 8 minutes extraction time.

In this study we optimized the regression model given as Equation 8.2 developed in (Kaur et al., 2008) using Harmony search metaheuristic algorithm and the optimal value lycopene predicted is 4.8mg/100g (rather than 1.97 predicted using response surface analysis in (Kaur et al., 2008)). The predicted optimal parameter setting for the five independent variables is 20:1 for solvent/meal ratio (v/w), five extractions, 60 °C temperature, 0.43 mm particle size and 4 minutes extraction time. In order to verify the claim experiential yield of lycopene obtained at this parameter setting was 4.75mg/100g.

In the next section a detailed description of how HS has been utilized to solve the problem at hand is provided.

Table 8.1: Independent variables and their levels used for central composite design.

Independent variables	Symbol	Coded variable levels				
		-2	-1	0	1	2
Solvent/meal ratio (v/w)	X_1	20	30	40	50	60
Number of extractions	X_2	1	2	3	4	5
Temperature (°C)	X_3	20	30	40	50	60
Particle Size (mm)	X_4	0.05	0.15	0.25	0.35	0.43
Time (minutes)	X_5	4	8	12	16	20

8.4 Proposed methodology of Optimizing Lycopene extraction using Harmony Search algorithm

The value of HMCR and PAR adopted for HS (Algorithm 1) in this experimentation is respectively set as 0.9 and 0.3 (same as in Chapter 2), however it was observed small value of HMS produces inferior results occasionally and hence HMS was set as 100. The objective function to be maximized is given as Equation 8.2. Each harmony is a five dimensional vector such that the first dimension corresponds to independent variable X_1 (solvent/meal ratio, v/w), second dimension corresponds to X_2 (number of extractions), third dimension corresponds to X_3 (temperature, °C), fourth dimension corresponds to X_4 (particle size, mm) and fifth dimension corresponds to X_5 (time, minutes). The range of all the five independent variables is from -2 to 2 in coded form as given in Table 8.1.

The Harmony Memory is initialized such that each component of the harmony/solution is within the bounds from -2 to 2. Note that the third component of harmony corresponding to X_3 (No. of extractions) can take only integer values. Each component of the new harmony represented as H_i is generated either using HM or randomization and is slightly altered in step 12 (Algorithm 1) determined by parameter Bandwidth (BW). The value of BW for each component of harmony except X_3 is 0.04 (i.e. $(\frac{1}{100})^{th}$ of variable bounds) and for X_3 it is 1. Once the new harmony (H) is generated it is evaluated using objective function and replaces the worst harmony in HM incase it is better than the worst harmony. The procedure is repeated for 5000 generations.

The optimal yield of lycopene predicted using HS algorithm is 4.8mg/100g at parameter setting 20:1 (v/w solvent/meal ratio), five extractions, 60 °C temperature, 0.43 mm particle size and 4 minutes extraction time. In the coded form the optimal value of independent variables X_1, X_2, X_3, X_4 and X_5 obtained using HS is respectively -2, 2, 2, 2 and -2, hence at these value the objective function (Equation 8.2) evaluates to 4.8. The program was run for 50 independent runs and the yield predicted was exactly same (4.8mg/100g) in each run and thus the standard deviation is zero. The program was run on the machine whose specifications are given in Section 2.4 and the execution time was approximately 0.18 seconds. Figure 8.3 graphically shows lycopene yield predicted by HS with respect to function evaluations.

The optimal parameter setting obtained by HS is 20:1 (v/w solvent/meal ratio), five extractions, 60 °C temperature, 0.43 mm particle size and 4 minutes extraction time. In order to verify the claim experimentation was carried out at this parameter setting, the experiment was repeated three times and the lycopene yield obtained is shown in Table 8.3. The average yield of lycopene obtained at the optimal parameter setting is 4.75mg/100g experiential. Note that the experimental procedure adopted for lycopene extraction in this study is exactly same as given in (Kaur et al., 2008) and has been described in Section 8.3.

The variants of HS algorithm (HS-SA, TPHS, SMHS) proposed in previous chapters were also tested on the problem at hand and produced exactly same results as that of standard HS algorithm.

8.5 Conclusion & Claim

In this chapter the regression expression depicting the relation between five independent variables namely solvent/meal ratio, number of extractions, temperature, particle size and time was optimized using music inspired Harmony Search algorithm. The optimal yield of lycopene predicted using HS algorithm is 4.8mg/100g when the parameter setting adopted is 20:1 (v/w solvent/meal ratio), five extractions, 60 °C temperature, 0.43 mm particle size and 4 minutes extraction time. whereas the optimal parameter setting predicted by response surface analysis gives a yield of only 1.97 mg/100g (Kaur et al., 2008). In order to verify the claim experiment was carried out at the above mentioned optimal parameter setting obtained by HS, the lycopene yield obtained experimentally was 4.75mg/100g. Thus the optimal setting predicted by HS is quite in tune with the experimental results.

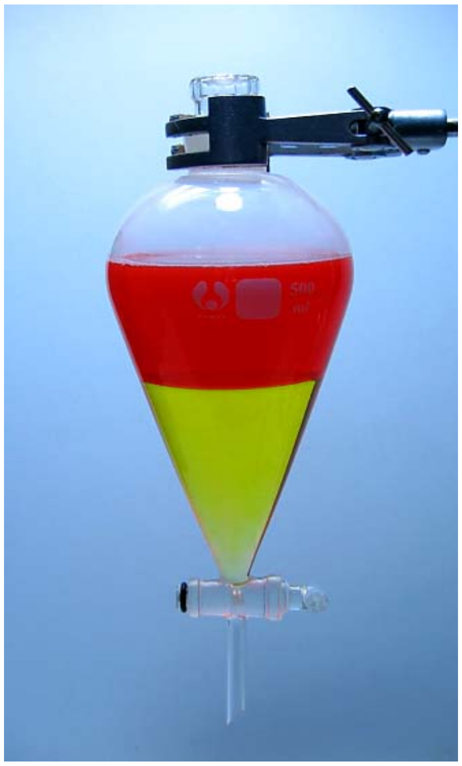
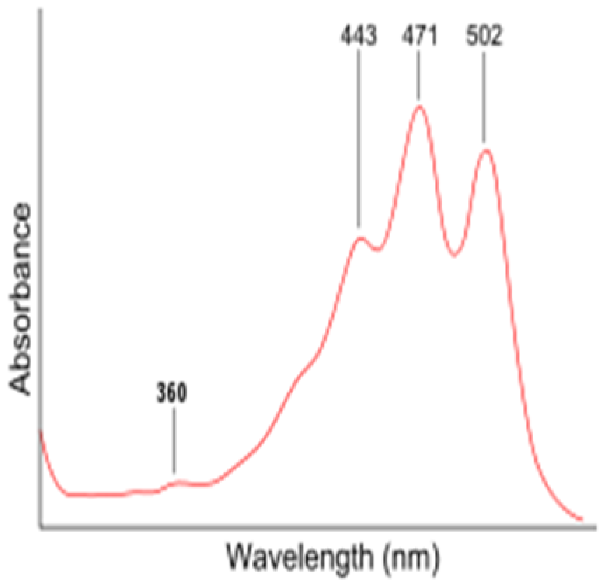


Figure 8.1: Polar and Non polar layer.

(a) HPLC-UV Visible Spectrometer



(b) Absorption spectrum of lycopene in hexane.

Figure 8.2: HPLC system and Absorption spectrum of Lycopene.

Table 8.2: Central composite arrangement for independent variables X_1, X_2, X_3, X_4, X_5 & their response (lycopene yield, mg/100 g).

Run	Variables levels (uncoded)					Lycopene yield (mg/100 g)	
	X_1	X_2	X_3	X_4	X_5	Experimental	Predicted
1	-1(30)	-1(2)	-1(30)	-1(0.35)	1(16)	1.19	1.18
2	1(50)	-1(2)	-1(30)	-1(0.35)	-1(8)	1.19	1.18
3	-1(30)	1(4)	-1(30)	-1(0.35)	-1(8)	1.13	1.13
4	1(50)	1(4)	-1(30)	-1(0.35)	1(16)	1.32	1.33
5	-1(30)	-1(2)	1(50)	-1(0.35)	-1(8)	1.04	1.04
6	1(50)	-1(2)	1(50)	-1(0.35)	1(16)	1.06	1.02
7	-1(30)	1(4)	1(50)	-1(0.35)	1(16)	1.65	1.64
8	1(50)	1(4)	1(50)	-1(0.35)	-1(8)	1.17	1.16
9	-1(30)	-1(2)	-1(30)	1(0.15)	-1(8)	0.951	0.949
10	1(50)	-1(2)	-1(30)	1(0.15)	1(16)	1.44	1.44
11	-1(30)	1(4)	-1(30)	1(0.15)	1(16)	1.38	1.4
12	1(50)	1(4)	-1(30)	1(0.15)	-1(8)	0.936	0.954
13	-1(30)	-1(2)	1(50)	1(0.15)	1(16)	1.23	1.21
14	1(50)	-1(2)	1(50)	1(0.15)	-1(8)	1.35	1.33
15	-1(30)	1(4)	1(50)	1(0.15)	-1(8)	1.98	1.97
16	1(50)	1(4)	1(50)	1(0.15)	1(16)	1.48	1.48
17	-2(20)	0(3)	0(40)	0(0.25)	0(12)	1.4	1.42
18	2(60)	0(3)	0(40)	0(0.25)	0(12)	1.25	1.26
19	0(40)	-2(1)	0(40)	0(0.25)	0(12)	1.06	1.11
20	0(40)	2(5)	0(40)	0(0.25)	0(12)	1.56	1.54
21	0(40)	0(3)	-2(20)	0(0.25)	0(12)	1.31	1.28
22	0(40)	0(3)	2(60)	0(0.25)	0(12)	1.55	1.6
23	0(40)	0(3)	0(40)	-2(0.25)	0(12)	0.639	0.677
24	0(40)	0(3)	0(40)	2(0.05)	0(12)	0.962	0.945
25	0(40)	0(3)	0(40)	0(0.43)	-2(4)	0.907	0.925
26	0(40)	0(3)	0(40)	0(0.25)	2(20)	1.16	1.17
27	0(40)	0(3)	0(40)	0(0.25)	0(12)	0.921	0.862
28	0(40)	0(3)	0(40)	0(0.25)	0(12)	0.892	0.862
29	0(40)	0(3)	0(40)	0(0.25)	0(12)	0.832	0.862
30	0(40)	0(3)	0(40)	0(0.25)	0(12)	0.862	0.862
31	0(40)	0(3)	0(40)	0(0.25)	0(12)	0.847	0.862
32	0(40)	0(3)	0(40)	0(0.25)	0(12)	0.847	0.862

Table 8.3: Lycopene yield (mg/100 g) obtained at optimal setting.

Run	Variables levels (uncoded)					Lycopene yield (mg/100 g)
	X_1	X_2	X_3	X_4	X_5	Experimental
1	-2(20)	2(5)	2(60)	2(0.43)	-2(4)	4.78
2	-2(20)	2(5)	2(60)	2(0.43)	-2(4)	4.72
3	-2(20)	2(5)	2(60)	2(0.43)	-2(4)	4.76
Average						**4.75**

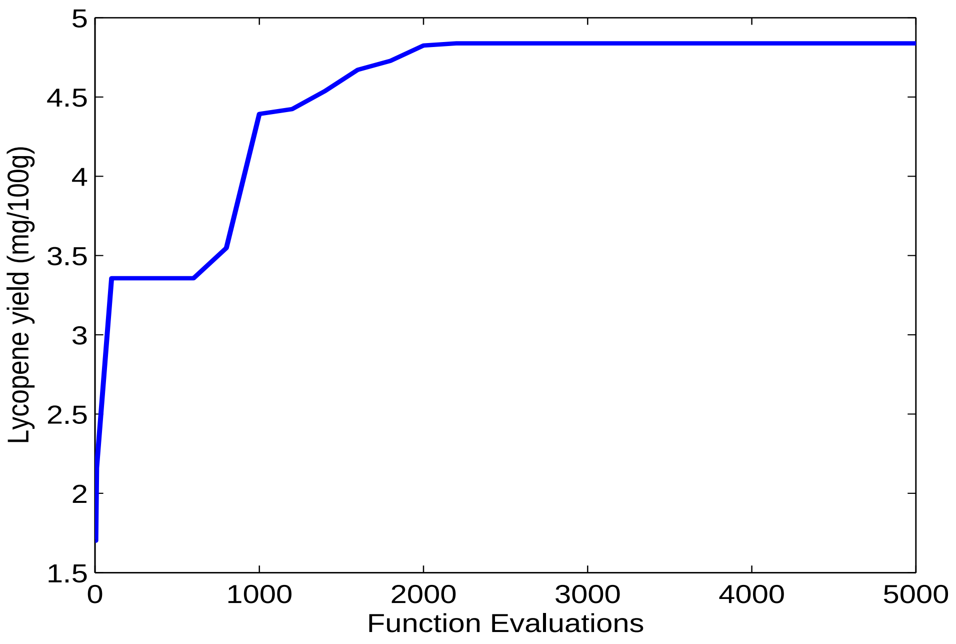


Figure 8.3: Predicted Lycopene yield w.r.t. no. of function evaluations.

Chapter 9
Concluding Observations

This is the concluding Chapter of this Thesis. It is organized as follows. In Section 9.1 the conclusions based on this study are outlined and in Section 9.2 some future research directions are proposed.

9.1 Conclusions

The aim of this Thesis is to enhance the balance between exploration and exploitation of the Harmony Search Algorithm, with a view to improve its efficiency and reliability to solve real life problems. To achieve this objective the HS algorithm is modified to create TPHS & SMHS algorithms, further HS is hybridized with a well known metaheuristic algorithm called simulated annealing to create HS-SA algorithm. The performance of HS and the three proposed variants is evaluated on IEEE 2014 benchmark problems and a real life problem from the field of computer vision called camera calibration problem- a highly non linear 12 dimensional unconstrained optimization problem from the field of computer vision. The HS algorithm enhanced by local search operator is used to solve sudoku and maximum clique problem, two well known combinatorial optimization problems. The chapter wise summary of this Thesis is given below:

Chapter 1 is introductory in nature. Besides relevant definitions, this Chapter discusses in details literature review on the related subject. Finally it outlines the Chapter-wise contents of the Thesis.

Chapter 2 introduces a novel algorithm based on hybridization of Harmony search and Simulated Annealing called HS-SA to inherit their advantages in a complementary way and overcome their limitations. HS suffers a limitation of premature convergence if one or more initially generated solutions/harmonies are in the vicinity of local optimal. In order to remove this limitation, taking the inspiration from Simulated Annealing the proposed HS-SA algorithm accepts even the inferior harmonies, compared to the harmonies already stored in Harmony Memory, with probability determined by parameter called Temperature. The Temperature parameter is initially kept high to favor exploration of search space and is linearly decreased to gradually shift focus to exploitation of promising search areas. The performance of HS-SA is analyzed and compared with Harmony Search algorithm and Simulated Annealing on IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed algorithm on multimodal benchmark functions, which are generally very challenging test beds for metaheuristic algorithms. Moreover, the local optima avoidance of an algorithm can be examined due to the massive number of local optima in such test functions.

Chapter 3 introduces Two phase Harmony search (TPHS) algorithm that attempts to strikes a balance between exploration and exploitation by concentrating on diversification in the first phase using catastrophic mutation and then switches to intensification using local search in the second phase, further the search area was gradually decreased by decreasing the BW and PAR with time.The performance of TPHS is analyzed and compared with 15 state-of-the-art metaheuristic algorithms on all the 30 IEEE CEC 2014 benchmark functions. Based on the numerical results it is established even though performance of TPHS is not very satisfactory on unimodal functions however it outperforms HS variants on multimodal functions. The performance of TPHS is particularly outstanding on composition functions when compared to state-of-the-art metaheurisctic algorithms for continuous optimization.

Chapter 4 introduces Shrinking Memory Harmony search (SMHS) algorithm, the SMHS attempts to establish a balance between the contradictory property of exploration and exploitation by concentrating on diversification in the beginning using extended memory and broad Bandwidth and then gradually switching to intensification by shrinking harmony memory and utilizing local search operator. The performance of SMHS is compared with nineteen state-of-the-art metaheuristic algorithms (four HS variants, five PSO variants, eight DE variants, and one variant each of GA and ES) on all the 30 IEEE CEC 2014 benchmark functions. The numerical results demonstrate the superiority of the proposed SMHS algorithm on multimodal function and its performance is outstanding on composition functions.

In Chapter 5 the performance of the three proposed algorithms namely HS-SA, TPHS & SMHS is compared on IEEE CEC 2014 benchmark suite and a real life problem called Camera Calibration (a highly non linear 12 dimensional optimization problem from the field of computer vision). It is established SMHS is the better performing algorithm on all the four categories of benchmark functions viz a viz unimodal, simple multimodal, hybrid multimodal and composition multimodal functions, further the performance of SMHS is excellent on Camera Calibration problem.

Chapter 6 explore the hybridization between Harmony Search (HS) and Hill Climbing (HC) algorithm by utilizing the exploration power of the former and exploitation power of the latter in the context of solving Sudoku which is a well-known hard Combinatorial Optimization problem. The hybrid algorithm is referred as Harmony Search Hill Climber (HSHC). In order to extend the exploration capabilities of HSHC it is further modified to create three different algorithms namely Retrievable Harmony Search Hill Climber (RHSHC), Global Best Retrievable Harmony Search Hill Climber (GB-RHSHC) and Random Best Retrievable Harmony Search Hill Climber (RB-RHSHC). Comparing the four proposed algorithms proposed in this chapter RHSHC outperforms its three variations in terms of effectiveness. Experimental results demonstrate that RHSHC perform significantly better than standard Harmony Search algorithm and standard Hill climber algorithm. On comparing RHSHC with the genetic algorithm it has been concluded that former outperforms latter both in terms of effectiveness and efficiency particularly for Hard and Expert level puzzles. Comparing RHSHC and hybrid AC3-tabu search algorithm it has been concluded that RHSHC is very competent to hybrid AC3-tabu search algorithm.

Chapter 7 investigates the capabilities of Harmony Search algorithm for solving maximum clique problem, a well known NP-Hard combinatorial optimization problem. Two different instantiations of a generic HS algorithm namely Harmony Search for MCP (HS_MCP) and Harmony Search with idiosyncratic harmonies for MCP (HSI_MCP) are proposed for this problem. HS_MCP has better exploitation and inferior exploration capabilities than HSI_MCP whereas HSI_MCP has better exploration and inferior exploitation capabilities than HSI_MCP, it has been concluded that former performs better than latter by testing them on all the instances of DIMACS benchmark graphs. HS_MCP has been compared with a recently proposed Harmony search based algorithm for MCP called Binary Harmony search (BHS) and the simulation results show that HS_MCP significantly outperforms BHS in terms of solution quality. The asymptotic time complexity of HS_MCP is $O(G \times N^3)$ where G is the number of generations and N is the number of nodes in the graph. A glimpse of effectiveness of some state-of-the-art exact algorithms on MCP has also been provided.

In Chapter 8 the regression expression depicting the relation between five independent variables namely solvent/meal ratio, number of extractions, temperature, particle size and time was optimized using Harmony Search algorithm. The optimal yield of lycopene predicted using HS algorithm is 4.8mg/100g when the parameter setting adopted is 20:1 (v/w solvent/meal ratio), five extractions, 60 $^{\circ}$C temperature, 0.43 mm particle size and 4 minutes extraction time. Prior art response surface analysis has been used to optimize the lycopene yield with respect to the above mentioned five independent variables and the maximum yield predicted was 1.99 mg/100 g. In order to verify the claim experiment was carried out at the above mentioned optimal parameter setting obtained by HS, the lycopene yield obtained experimentally was 4.75mg/100g. Thus the optimal setting predicted by HS is quite in tune with the experimental results.

The Thesis concludes with Chapter 9. It derives the overall conclusions of this Thesis and it outlines the limitations and scope of the proposed algorithms. Later it suggests future scope and new directions of research in this area.

9.2 Future Research Directions

Research is a never ending process. In order to carry out research in this direction, the following is suggested:

1. The proposed algorithms can be enhanced by using different constraint handling techniques available in literature so as to solve constraint optimization problems.

2. The efficiency of the proposed algorithms can be tested in solving multi objective optimization problems.

3. The proposed algorithms should be parallelized so that they can be used to solve optimization problems which have a costly objective function.

4. The proposed variants can be applied to more complex real life optimization problems.

Bibliography

Abdel-Aziz, Y. (1971). Direct linear transformation from comparator coordinates in close-range photogrammetry. In *ASP Symposium on Close-Range Photogrammetry in Illinois, 1971*.

Acampora, G., Pedrycz, W., and Vitiello, A. (2015). A competent memetic algorithm for learning fuzzy cognitive maps. *IEEE Transactions on Fuzzy Systems*, 23(6):2397–2411.

Afkhami, S., Ma, O. R., and Soleimani, A. (2013). A binary harmony search algorithm for solving the maximum clique problem. *International Journal of Computer Applications*, 69(12):38–43.

Ahmed, M. T., Hemayed, E. E., and Farag, A. A. (1999). Neurocalibration: a neural network that can tell camera calibration parameters. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 463–468. IEEE.

Al-Betar, M. A., Awadallah, M. A., Khader, A. T., and Bolaji, A. L. (2016). Tournament-based harmony search algorithm for non-convex economic load dispatch problem. *Applied Soft Computing*, 47:449–459.

Al-Betar, M. A., Khader, A. T., and Zaman, M. (2012). University course timetabling using a hybrid harmony search metaheuristic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(5):664–681.

Alfailakawi, M. G., Ahmad, I., and Hamdan, S. (2016). Harmony-search algorithm for 2d nearest neighbor quantum circuits realization. *Expert Systems with Applications*, 61:16–27.

Ali, M, M. and Kaelo, P. (2008). Improved particle swarm algorithms for global optimization. *Applied mathematics and computation*, 196(2):578–593.

Ali, M, M. and Zhu, W. (2013). A penalty function-based differential evolution algorithm for constrained global optimization. *Computational Optimization and Applications*, 54(3):707–739.

Andrade, D. V., Resende, M. G., and Werneck, R. F. (2012). Fast local search for the maximum independent set problem. *Journal of Heuristics*, 18(4):525–547.

AoA, C. (1990). Official methods of analysis. *Association of Official Analytical Chemists, Arlington, VA, USA*, 1:684.

Balasundaram, B. and Butenko, S. (2006). Graph domination, coloring and cliques in telecommunications. In *Handbook of Optimization in Telecommunications*, pages 865–890. Springer.

Balasundaram, B., Butenko, S., and Hicks, I. V. (2011). Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142.

Bansal, J. C., Sharma, H., Arya, K., and Nagar, A. K. (2013). Memetic search in artificial bee colony algorithm. *Soft Computing*, 17(10):1911–1928.

Bansal, J. C., Sharma, H., Jadon, S. S., and Clerc, M. (2014). Spider monkey optimization algorithm for numerical optimization. *Memetic Computing*, 6(1):31–47.

Batsyn, M., Goldengorin, B., Maslov, E., and Pardalos, P. M. (2014). Improvements to mcs algorithm for the maximum clique problem. *Journal of Combinatorial Optimization*, 27(2):397–416.

Battiti, R. and Protasi, M. (2001). Reactive local search for the maximum clique problem 1. *Algorithmica*, 29(4):610–637.

Bedi, P., Bansal, R., and Sehgal, P. (2011). Using pso in image hiding scheme based on lsb substitution. *Advances in Computing and Communications*, pages 259–268.

Bedi, P., Bansal, R., and Sehgal, P. (2013). Using pso in a spatial domain based image hiding scheme with distortion tolerance. *Computers & Electrical Engineering*, 39(2):640–654.

Bedi, P., Sharma, R., and Kaur, H. (2009). Recommender system based on collaborative behavior of ants. *Journal of Artificial Intelligence*, 2(2):40–55.

Benlic, U. and Hao, J.-K. (2013). Breakout local search for maximum clique problems. *Computers & Operations Research*, 40(1):192–206.

Bertsekas, D. P. (2014). *Constrained optimization and Lagrange multiplier methods*. Academic press.

Bertsekas, D. P. (2015). *Convex optimization algorithms*. Athena Scientific Belmont.

Bhardwaj, A., Tiwari, A., Krishna, R., and Varma, V. (2016). A novel genetic programming approach for epileptic seizure detection. *Computer Methods and programs in Biomedicine*, 124:2–18.

Bhardwaj, A., Tiwari, A., Varma, M. V., and Krishna, M. R. (2014). Classification of eeg signals using a novel genetic programming approach. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 1297–1304. ACM.

Bhattacharjee, P., Rakshit, P., Goswami, I., Konar, A., and Nagar, A. K. (2011). Multi-robot path-planning using artificial bee colony optimization algorithm. In *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, pages 219–224. IEEE.

Blum, C. and Roli, A. (2008). Hybrid metaheuristics: an introduction. In *Hybrid Metaheuristics*, pages 1–30. Springer.

Boginski, V., Butenko, S., and Pardalos, P. M. (2006). Mining market data: a network approach. *Computers & Operations Research*, 33(11):3171–3184.

Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. (1999). The maximum clique problem. In *Handbook of Combinatorial Optimization*, pages 1–74. Springer.

Boryczka, U. and Juszczuk, P. (2012). Solving the sudoku with the differential evolution. *Zeszyty Naukowe Politechniki Białostockiej. Informatyka*, pages 5–16.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University press.

Brest, J., Greiner, S., Boskovic, B., Mernik, M., and Zumer, V. (2006). Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657.

Brunato, M. and Battiti, R. (2011). R-evo: a reactive evolutionary algorithm for the maximum clique problem. *IEEE Transactions on Evolutionary Computation*, 15(6):770–782.

Cai, X., Cui, Z., Zeng, J., and Tan, Y. (2008). Dispersed particle swarm optimization. *Information Processing Letters*, 105(6):231–235.

Carraghan, R. and Pardalos, P. M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9(6):375–382.

Carter, M. W. and Johnson, D. (2001). Extended clique initialisation in examination timetabling. *Journal of the Operational Research Society*, pages 538–544.

Carter, M. W., Laporte, G., and Lee, S. Y. (1996). Examination time tabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, pages 373–383.

Carter, R. and Park, K. (1993). How good are genetic algorithms at finding large cliques: an experimental study. Technical report, Boston University Computer Science Department.

Chalabi, N., Delort, L., Le Corre, L., Satih, S., Bignon, Y.-J., and Bernard-Gallon, D. (2006). Gene signature of breast cancer cell lines treated with lycopene. *Future Medicine*, pages 663–672.

Chan, T.-M., Leung, K.-S., and Lee, K.-H. (2012). Memetic algorithms for de novo motif discovery. *IEEE Transactions on Evolutionary Computation*, 16(5):730–748.

Chandra, S., Jayadeva, and Mehra, A. (2009). *Numerical optimization with applications*. Alpha Science International.

Chen, F., Zhai, H., and Fang, Y. (2010). Available bandwidth in multirate and multihop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 28(3):299–307.

Cheng, M.-Y., Prayogo, D., Wu, Y.-W., and Lukito, M. M. (2016). A hybrid harmony search algorithm for discrete sizing optimization of truss structure. *Automation in Construction*, 69:21–33.

Cheng, R. and Jin, Y. (2015). A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, 291:43–60.

Cheng, Y., Li, L., Lansivaara, T., Chi, S., and Sun, Y. (2008). An improved harmony search minimization algorithm using different slip surface generation methods for slope stability analysis. *Engineering Optimization*, 40(2):95–115.

Chhabra, J. K. et al. (2017). Harmony search based remodularization for object-oriented software systems. *Computer Languages, Systems & Structures*, 47:153–169.

Cuevas, E., Cienfuegos, M., Zaldívar, D., and Pérez-Cisneros, M. (2013). A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems with Applications*, 40(16):6374–6384.

Das, K. N., Bhatia, S., Puri, S., and Deep, K. (2012). A retrievable ga for solving sudoku puzzles. *Technical report. (http://www.cse.psu.edu)*.

Das, S., Mukhopadhyay, A., Roy, A., Abraham, A., and Panigrahi, B. K. (2011). Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):89–106.

Dash, R. and Dash, P. (2016). Efficient stock price prediction using a self evolving recurrent neuro-fuzzy inference system optimized through a modified differential harmony search technique. *Expert Systems with Applications*, 52:75–90.

Dawkins, R. (2006). *The selfish gene (No. 199)*. Oxford university press.

De Castro, L. N. and Von Zuben, F. J. (2000). The clonal selection algorithm with engineering applications. In *Proceedings of GECCO*, volume 2000, pages 36–39.

Deng, L., Lu, G., Shao, Y., Fei, M., and Hu, H. (2016). A novel camera calibration technique based on differential evolution particle swarm optimization algorithm. *Neurocomputing*, 174:456–465.

Deng, X. Q. and Li, Y. D. (2013). A novel hybrid genetic algorithm for solving sudoku puzzles. *Optimization Letters*, pages 1–17.

Diao, R. and Shen, Q. (2012). Feature selection with harmony search. *IEEE Transactions onSystems, Man, and Cybernetics, Part B: Cybernetics*, 42(6):1509–1523.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39.

Dorndorf, U., Jaehn, F., and Pesch, E. (2008). Modelling robust flight-gate scheduling as a clique partitioning problem. *Transportation Science*, 42(3):292–301.

Dos Santos Coelho, L. and De Andrade Bernert, D. L. (2009). An improved harmony search algorithm for synchronization of discrete-time chaotic systems. *Chaos, Solitons & Fractals*, 41(5):2526–2532.

Duane, C. B. (1971). Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866.

Durstenfeld, R. (1964). Algorithm 235: random permutation. *Communications of the ACM*, 7(7):420.

El-Abd, M. (2013). An improved global-best harmony search algorithm. *Applied Mathematics and Computation*, 222:94–106.

Elola, A., Del Ser, J., Bilbao, M. N., Perfecto, C., Alexandre, E., and Salcedo-Sanz, S. (2016). Hybridizing cartesian genetic programming and harmony search for adaptive feature construction in supervised learning problems. *Applied Soft Computing*.

Elyasigomari, V., Lee, D., Screen, H. R., and Shaheed, M. H. (2017). Development of a two-stage gene selection method that incorporates a novel hybrid approach using the cuckoo optimization algorithm and harmony search for cancer classification. *Journal of Biomedical Informatics*, 67:11–20.

Engebretsen, L. and Holmerin, J. (2003). Towards optimal lower bounds for clique and chromatic number. *Theoretical Computer Science*, 299(1):537–584.

Esfahani, H. N., hossein Sobhiyah, M., and Yousefi, V. R. (2016). Project portfolio selection via harmony search algorithm and modern portfolio theory. *Procedia-Social and Behavioral Sciences*, 226:51–58.

Etzion, T. and Ostergard, P. R. (1998). Greedy and heuristic algorithms for codes and colorings. *IEEE Transactions on Information Theory*, 44(1):382–388.

Faig, W. (1975). Calibration of close-range photogrammetric systems: Mathematical formulation. *Photogrammetric engineering and remote sensing*, 41(12).

Fattahi, H., Gholami, A., Amiribakhtiar, M. S., and Moradi, S. (2015). Estimation of asphaltene precipitation from titration data: a hybrid support vector regression with harmony search. *Neural Computing and Applications*, 26(4):789–798.

Feige, U. (2004). Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics*, 18(2):219–225.

Feige, U., Goldwasser, S., Lovász, L., Safra, S., and Szegedy, M. (1991). Approximating clique is almost np-complete. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 2–12. IEEE.

Fesanghary, M., Mahdavi, M., Minary-Jolandan, M., and Alizadeh, Y. (2008). Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer Methods in Applied Mechanics and Engineering*, 197(33):3080–3091.

Formato, R. A. (2009). Central force optimization: A new deterministic gradient-like optimization metaheuristic. *Opsearch*, 46(1):25–51.

Ganjidoost, H., Mousavi, S. J., and Soroush, A. (2016). Adaptive network-based fuzzy inference systems coupled with genetic algorithms for predicting soil permeability coefficient. *Neural Processing Letters*, 44(1):53–79.

Gao, K., Zhang, Y., Sadollah, A., and Su, R. (2016). Optimizing urban traffic light scheduling problem using harmony search with ensemble of local search. *Applied Soft Computing*, 48:359–372.

Gao, X. Z., Wang, X., and Ovaska, S. J. (2008). Modified harmony search methods for uni-modal and multi-modal optimization. In *Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on*, pages 65–72. IEEE.

García-Martínez, C., Lozano, M., Herrera, F., Molina, D., and Sánchez, A. M. (2008). Global and local real-coded genetic algorithms based on parent-centric crossover operators. *European Journal of Operational Research*, 185(3):1088–1113.

García-Martínez, C., Lozano, M., and Rodríguez-Díaz, F. J. (2012). A simulated annealing method based on a specialised evolutionary algorithm. *Applied Soft Computing*, 12(2):573–588.

Garey, M. R. (1979). Computers and intractability: A guide to the theory of np-completeness. *Revista Da Escola De Enfermagem Da USP*, 44(2):340.

Garg, V. and Deep, K. (2016). Performance of laplacian biogeography-based optimization algorithm on cec 2014 continuous optimization benchmarks and camera calibration problem. *Swarm and Evolutionary Computation*, 27:132–144.

Geem, Z. W. (2005). Harmony search in water pump switching problem. *Advances in Natural Computation*, pages 445–445.

Geem, Z. W. (2007). Harmony search algorithm for solving sudoku. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 371–378. Springer.

Geem, Z. W. (2009). Particle-swarm harmony search for water network design. *Engineering Optimization*, 41(4):297–311.

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

Geem, Z. W. and Yoon, Y. (2017). Harmony search optimization of renewable energy charging with energy storage system. *International Journal of Electrical Power & Energy Systems*, 86:120–126.

Geng, X., Chen, Z., Yang, W., Shi, D., and Zhao, K. (2011). Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing*, 11(4):3680–3689.

Geng, X., Xu, J., Xiao, J., and Pan, L. (2007). A simple simulated annealing algorithm for the maximum clique problem. *Information Sciences*, 177(22):5064–5071.

Gennery, D. B. (1979). Stereo-camera calibration. In *Proceedings ARPA IUS Workshop*, pages 101–107.

Gomez Ravetti, M. and Moscato, P. (2008). Identification of a 5-protein biomarker molecular signature for predicting alzheimers disease. *PloS one*, 3(9):e3111.

Grosan, C. and Abraham, A. (2007). Hybrid evolutionary algorithms: methodologies, architectures, and reviews. In *Hybrid Evolutionary Algorithms*, pages 1–17. Springer.

Gunther, J. and Moon, T. (2012). Entropy minimization for solving sudoku. *IEEE Transactions on Signal Processing*, 60(1):508–513.

Guo, S.-M., Yang, C.-C., Hsu, P.-H., and Tsai, J. S.-H. (2015). Improving differential evolution with a successful-parent-selecting framework. *IEEE Transactions on Evolutionary Computation*, 19(5):717–730.

Guo, Z., Shi, L., Chen, L., and Liang, Y. (2017). A harmony search-based memetic optimization model for integrated production and transportation scheduling in mto manufacturing. *Omega*, 66:327–343.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Haralick, R. M., Joo, H., Lee, C.-N., Zhuang, X., Vaidya, V. G., and Kim, M. B. (1989). Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1426–1446.

Hart, W. E., Krasnogor, N., and Smith, J. E. (2004). *Recent advances in memetic algorithms*, volume 166. Springer Science & Business Media.

Hasançebi, O., Erdal, F., and Saka, M. P. (2009). Adaptive harmony search method for structural optimization. *Journal of Structural Engineering*, 136(4):419–431.

Håstad, J. (1996). Clique is hard to approximate within $n^{1-\epsilon}$. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 627–636. IEEE.

Hati, S. and Sengupta, S. (2001). Robust camera parameter estimation using genetic algorithm. *Pattern Recognition Letters*, 22(3):289–298.

Himmelblau, D. M. (1972). *Applied nonlinear programming*. McGraw-Hill Companies.

Hoang, D. C., Yadav, P., Kumar, R., and Panda, S. K. (2014). Real-time implementation of a harmony search algorithm-based clustering protocol for energy-efficient wireless sensor networks. *IEEE Transactions on Industrial Informatics*, 10(1):774–783.

Holland, J. H. (1975). Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*.

Holzapfel, N. P., Holzapfel, B. M., Champ, S., Feldthusen, J., Clements, J., and Hutmacher, D. W. (2013). The potential role of lycopene for the prevention and therapy of prostate cancer: from molecular mechanisms to clinical evidence. *International Journal of Molecular Sciences*, 14(7):14620–14646.

Hopper, E. and Turton, B. C. (2001). A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16(4):257–300.

Horowitz, E. and Sahni, S. (1978). *Fundamentals of computer algorithms*. Computer Science Press.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223.

Jaddi, N. S. and Abdullah, S. (2017). A cooperative-competitive master-slave global-best harmony search for ann optimization and water-quality prediction. *Applied Soft Computing*, 51:209–224.

Jadon, S. S., Bansal, J. C., Tiwari, R., and Sharma, H. (2015). Accelerating artificial bee colony algorithm with adaptive local search. *Memetic Computing*, 7(3):215–230.

Jain, K., Padhye, J., Padmanabhan, V. N., and Qiu, L. (2005). Impact of interference on multi-hop wireless network performance. *Wireless Networks*, 11(4):471–487.

Jang, W. S., Kang, H. I., and Lee, B. H. (2008). Hybrid simplex-harmony search method for optimization problems. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*.

Ji, Q. and Zhang, Y. (2001). Camera calibration with genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 31(2):120–130.

Jin, F., Song, S., and Wu, C. (2009). A simulated annealing algorithm for single machine scheduling problems with family setups. *Computers & Operations Research*, 36(7):2133–2138.

Jin, X. and Li, Z. (1997). Genetic-catastrophic algorithms and its application in nonlinear control system. *Journal of System Simulation*, 9(2):111–115.

Jin, Y. and Hao, J.-K. (2015). General swap-based multiple neighborhood tabu search for the maximum independent set problem. *Engineering Applications of Artificial Intelligence*, 37:20–33.

Jones, S. K., Roach, P. A., and Perkins, S. (2008). Construction of heuristics for a search-based approach to solving sudoku. In *Research and Development in Intelligent Systems XXIV*, pages 37–49. Springer.

Jung, D., Kang, D., and Kim, J. H. (2017). Development of a hybrid harmony search for water distribution system design. *KSCE Journal of Civil Engineering*, pages 1–9.

Kamboj, V. K., Bath, S., and Dhillon, J. (2016). Implementation of hybrid harmony search/random search algorithm for single area unit commitment problem. *International Journal of Electrical Power & Energy Systems*, 77:228–249.

Kapur, P., Pham, H., Gupta, A., and Jha, P. (2011). *Software reliability assessment with OR applications*. Springer.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karp, R. M. (1972). *Reducibility among combinatorial problems*. Springer.

Katayama, K., Hamamoto, A., and Narihisa, H. (2005). An effective local search for the maximum clique problem. *Information Processing Letters*, 95(5):503–511.

Kaur, D., Sharma, R., Abas Wani, A., Singh Gill, B., and Sogi, D. (2006). Physicochemical changes in seven tomato (lycopersicon esculentum) cultivars during ripening. *International Journal of Food Properties*, 9(4):747–757.

Kaur, D., Wani, A. A., Oberoi, D., and Sogi, D. (2008). Effect of extraction conditions on lycopene extractions from tomato processing waste skin using response surface methodology. *Food Chemistry*, 108(2):711–718.

Kennedy, J. (2011). Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer.

Kiran, R., Sircar, P., and Verma, N. K. (2016). Soft computing approaches for two-dimensional beamforming. In *Recent Developments and New Direction in Soft-Computing Foundations and Applications*, pages 301–314. Springer.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Krishnanand, K. and Ghose, D. (2006). Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. *Multiagent and Grid Systems*, 2(3):209–222.

Krishnanand, K. and Ghose, D. (2009). Glowworm swarm optimisation: a new method for optimising multi-modal functions. *International Journal of Computational Intelligence Studies*, 1(1):93–119.

Kumar, S., Thakur, M., Raman, B., and Sukavanam, N. (2008). Stereo camera calibration using real coded genetic algorithm. In *TENCON 2008-2008 IEEE Region 10 Conference*, pages 1–5. IEEE.

Lee, Y. C. and Zomaya, A. Y. (2009). Interweaving heterogeneous metaheuristics using harmony search. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE.

Lewis, R. (2007). Metaheuristics can solve sudoku puzzles. *Journal of Heuristics*, 13(4):387–401.

Li, C. M. and Quan, Z. (2010). An efficient branch and bound algorithm based on maxsat for the maximum clique problem. In *AAAI*, volume 10, pages 128–133.

Li, Y. and Deng, X. (2011). Solving sudoku puzzles based on improved genetic algorithm. *Jisuanji Yingyong yu Ruanjian*, 28(3):68–70.

Liang, J., Qu, B., and Suganthan, P. (2013). Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory*.

Liang, J. J., Qin, A. K., Suganthan, P. N., and Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281–295.

Lin, G., Zhu, W., and Ali, M. M. (2016). An effective hybrid memetic algorithm for the minimum weight dominating set problem. *IEEE Transactions on Evolutionary Computation*, 20(6):892–907.

Liu, L., Shao, L., Li, X., and Lu, K. (2016). Learning spatio-temporal representations for action recognition: A genetic programming approach. *IEEE Transactions on Tybernetics*, 46(1):158–170.

Lynce, I. and Ouaknine, J. (2006). Sudoku as a sat problem. In *ISAIM*.

Lynn, N. and Suganthan, P. N. (2015). Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. *Swarm and Evolutionary Computation*, 24:11–24.

Mahdavi, M., Fesanghary, M., and Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2):1567–1579.

Mallipeddi, R., Suganthan, P. N., Pan, Q.-K., and Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2):1679–1696.

Malod-Dognin, N., Andonov, R., and Yanev, N. (2010). Maximum cliques in protein structure comparison. In *Experimental Algorithms*, pages 106–117. Springer.

Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M. N., Salcedo-Sanz, S., and Geem, Z. W. (2013). A survey on applications of the harmony search algorithm. *Engineering Applications of Artificial Intelligence*, 26(8):1818–1831.

Mantere, T. and Koljonen, J. (2006). Solving and rating sudoku puzzles with genetic algorithms. In *New Developments in Artificial Intelligence and the Semantic Web, Proceedings of the 12th Finnish Artificial Intelligence Conference STeP*, pages 86–92.

Mantere, T. and Koljonen, J. (2007). Solving, rating and generating sudoku puzzles with ga. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1382–1389. IEEE.

Marchiori, E. (1998). A simple heuristic based genetic algorithm for the maximum clique problem. In *Symposium on Applied Computing: Proceedings of the 1998 ACM symposium on Applied Computing*, volume 27, pages 366–373. Citeseer.

Maslov, E., Batsyn, M., and Pardalos, P. M. (2014). Speeding up branch and bound algorithms for solving the maximum clique problem. *Journal of Global Optimization*, 59(1):1–21.

McCreesh, C. and Prosser, P. (2013). Multi-threading a state-of-the-art maximum clique algorithm. *Algorithms*, 6(4):618–635.

McCreesh, C. and Prosser, P. (2015). A parallel branch and bound algorithm for the maximum labelled clique problem. *Optimization Letters*, 9(5):949–960.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

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

Moayedikia, A., Ong, K.-L., Boo, Y. L., Yeoh, W. G., and Jensen, R. (2017). Feature selection for high dimensional imbalanced class data using harmony search. *Engineering Applications of Artificial Intelligence*, 57:38–49.

Mohan, C. and Deep, K. (2009). *Optimization Techniques*. New Age New Delhi.

Mohd Alia, O. (2017). Dynamic relocation of mobile base station in wireless sensor networks using a cluster-based harmony search algorithm. *Information Sciences*, 385:76–95.

Mohd Alia, O. and Mandava, R. (2011). The variants of the harmony search algorithm: an overview. *Artificial Intelligence Review*, 36(1):49–68.

Molina-Moreno, F., García-Segura, T., Martí, J. V., and Yepes, V. (2017). Optimization of buttressed earth-retaining walls using hybrid harmony search algorithms. *Engineering Structures*, 134:205–216.

Moon, T. K. and Gunther, J. H. (2006). Multiple constraint satisfaction by belief propagation: An example using sudoku. In *IEEE Mountain Workshop on Adaptive and Learning Systems, 2006*, pages 122–126. IEEE.

Moon, T. K., Gunther, J. H., and Kupin, J. J. (2009). Sinkhorn solves sudoku. *IEEE Transactions on Information Theory*, 55(4):1741–1746.

Moraglio, A., Togelius, J., and Lucas, S. (2006). Product geometric crossover for the sudoku puzzle. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 470–476. IEEE.

Mousavi, S., Nakhaei, P., Sadollah, A., and Kim, J. H. (2017). Optimization of hydropower storage projects using harmony search algorithm. In *International Conference on Harmony Search Algorithm*, pages 261–270. Springer.

Mullaney, D. (2006). Using ant systems to solve sudoku problems. *University College Dublin*.

Nasir, M., Das, S., Maity, D., Sengupta, S., Halder, U., and Suganthan, P. N. (2012). A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. *Information Sciences*, 209:16–36.

Nayak, S. K., Sahoo, N., and Panda, G. (2015). Demand side management of residential loads in a smart grid using 2d particle swarm optimization technique. In *Power, Communication and Information Technology Conference (PCITC), 2015 IEEE*, pages 201–206. IEEE.

Nicolau, M. and Ryan, C. (2006). Solving sudoku with the gauge system. In *European Conference on Genetic Programming*, pages 213–224. Springer.

Nwankwor, E., Nagar, A. K., and Reid, D. (2013). Hybrid differential evolution and particle swarm optimization for optimal well placement. *Computational Geosciences*, pages 1–20.

Okamoto, A. (1981). Orientation and construction of models. i- the orientation problem in close-range photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 47:1437–1454.

Omran, M. G. (2005). *Particle swarm optimization methods for pattern recognition and image processing*. PhD thesis, University of Pretoria.

Omran, M. G. and Mahdavi, M. (2008). Global-best harmony search. *Applied Mathematics and Computation*, 198(2):643–656.

Ong, Y.-S., Lim, M.-H., Zhu, N., and Wong, K.-W. (2006a). Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(1):141–152.

Ong, Y.-S., Nguyen, Q.-H., Lim, M.-H., and Jing, T. (2006b). A development platform for memetic algorithm design. In *SCIS & ISIS SCIS & ISIS 2006*, pages 1027–1032. Japan Society for Fuzzy Theory and Intelligent Informatics.

Ordóñez-Guillén, N. E. and Martínez-Pérez, I. M. (2015). Heuristic search space generation for maximum clique problem inspired in biomolecular filtering. *Journal of Signal Processing Systems*, pages 1–12.

Östergård, P. R. (2002). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1):197–207.

Østerlie, M. and Lerfall, J. (2005). Lycopene from tomato products added minced meat: Effect on storage quality and colour. *Food Research International*, 38(8):925–929.

Ouaddah, A. and Boughaci, D. (2016). Harmony search algorithm for image reconstruction from projections. *Applied Soft Computing*, 46:924–935.

P C Jha, M. N. H. (2011). *Mathematical Modelling, Optimization and Their Applications*. Narosa India.

Pan, Q.-K., Suganthan, P., Liang, J., and Tasgetiren, M. F. (2010a). A local-best harmony search algorithm with dynamic subpopulations. *Engineering Optimization*, 42(2):101–117.

Pan, Q.-K., Suganthan, P. N., Tasgetiren, M. F., and Liang, J. J. (2010b). A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, 216(3):830–848.

Pandiarajan, K. and Babulal, C. (2016). Fuzzy harmony search algorithm based optimal power flow for power system security enhancement. *International Journal of Electrical Power & Energy Systems*, 78:72–79.

Papa, J. P., Scheirer, W., and Cox, D. D. (2016). Fine-tuning deep belief networks using harmony search. *Applied Soft Computing*, 46:875–885.

Papaioannou, E. H. and Karabelas, A. J. (2012). Lycopene recovery from tomato peel under mild conditions assisted by enzymatic pre-treatment and non-ionic surfactants. *Acta Biochimica Polonica*, 59(1):71.

Paquette, L., Stampfler, R., Davis, W. A., and Caelli, T. M. (1990). A new camera calibration method for robotic vision. In *Close-Range Photogrammetry Meets Machine Vision*, volume 1395, pages 656–663.

Pardalos, P. M. and Xue, J. (1994). The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328.

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

Pattabiraman, B., Patwary, M. M. A., Gebremedhin, A. H., Liao, W.-k., and Choudhary, A. (2015). Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection. *Internet Mathematics*, 11(4-5):421–448.

Pattillo, J., Youssef, N., and Butenko, S. (2012). Clique relaxation models in social network analysis. In *Handbook of Optimization in Complex Networks*, pages 143–162. Springer.

Poli, R., Langdon, W. B., McPhee, N. F., and Koza, J. R. (2008). *A field guide to genetic programming*. Lulu. com.

Ponz-Tienda, J., Salcedo-Bernal, A., Pellicer, E., and Benlloch-Marco, J. (2017). Improved adaptive harmony search algorithm for the resource leveling problem with minimal lags. *Automation in Construction*, 77:82–92.

Poojary, M. M. and Passamonti, P. (2015). Extraction of lycopene from tomato processing waste: Kinetics and modelling. *Food Chemistry*, 173:943–950.

Pullan, W. (2006). Phased local search for the maximum clique problem. *Journal of Combinatorial Optimization*, 12(3):303–323.

Pullan, W., Mascia, F., and Brunato, M. (2011). Cooperating local search for the maximum clique problem. *Journal of Heuristics*, 17(2):181–199.

Purohit, A., Chaudhari, N. S., and Tiwari, A. (2010). Construction of classifier with feature selection based on genetic programming. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–5. IEEE.

Qin, A. K., Huang, V. L., and Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417.

Rafi, M. M., Yadav, P. N., and Reyes, M. (2007). Lycopene inhibits lps-induced proinflammatory mediator inducible nitric oxide synthase in mouse macrophage cells. *Journal of Food Science*, 72(1):S069–S074.

Rao, A. and Agarwal, S. (1999). Role of lycopene as antioxidant carotenoid in the prevention of chronic diseases: a review. *Nutrition Research*, 19(2):305–323.

Rao, S. S. (2009). *Engineering optimization: theory and practice*. John Wiley & Sons.

Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information Sciences*, 179(13):2232–2248.

Rossi, F., Van Beek, P., and Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

Roula, S., Gantayat, H., Panigrahi, T., and Panda, G. (2015). Maximum likelihood doa estimation in wireless sensor networks using comprehensive learning particle swarm optimization algorithm. In *Computational Intelligence in Data Mining-Volume 3*, pages 499–507. Springer.

Rout, N. K., Das, D. P., and Panda, G. (2016). Particle swarm optimization based nonlinear active noise control under saturation nonlinearity. *Applied Soft Computing*, 41:275–289.

Saka, M., Hasançebi, O., and Geem, Z. W. (2016). Metaheuristics in structural optimization and discussions on harmony search algorithm. *Swarm and Evolutionary Computation*, 28:88–97.

Saka, M, P. and Hasancebi, O. (2009). Adaptive harmony search algorithm for design code optimization of steel structures. In *Harmony Search Algorithms for Structural Design Optimization*, pages 79–120. Springer.

San Segundo, P. and Artieda, J. (2015). A novel clique formulation for the visual feature matching problem. *Applied Intelligence*, 43(2):325–342.

San Segundo, P., Lopez, A., and Pardalos, P. M. (2016). A new exact maximum clique algorithm for large and massive sparse graphs. *Computers & Operations Research*, 66:81–94.

Sato, Y., Hasegawa, N., and Sato, M. (2013). Acceleration of genetic algorithms for sudoku solution on many-core processors. In *Massively Parallel Evolutionary Computation on GPGPUs*, pages 421–444. Springer.

Sato, Y. and Inoue, H. (2010). Solving sudoku with genetic operations that preserve building blocks. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 23–29. IEEE.

Sawyerr, B. A., Adewumi, A. O., and Ali, M. M. (2014). Real-coded genetic algorithm with uniform random local search. *Applied Mathematics and Computation*, 228:589–597.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127.

Sharma, H., Bansal, J. C., and Arya, K. (2013). Power law-based local search in differential evolution. *International Journal of Computational Intelligence Studies*, 2(2):90–112.

Sharma, H., Bansal, J. C., Arya, K. V., and Yang, X.-S. (2016). Lévy flight artificial bee colony algorithm. *International Journal of Systems Science*, 47(11):2652–2670.

Shourian, M., Mousavi, S. J., Menhaj, M., and Jabbari, E. (2008a). Neural-network-based simulation-optimization model for water allocation planning at basin scale. *Journal of Hydroinformatics*, 10(4):331–343.

Shourian, M., Mousavi, S. J., and Tahershamsi, A. (2008b). Basin-wide water resources planning by integrating pso algorithm and modsim. *Water Resources Management*, 22(10):1347–1366.

Simonis, H. (2005). Sudoku as a constraint problem. In *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, volume 12, pages 13–27. Citeseer.

Singh, A. and Bhukya, W. N. (2011). A hybrid genetic algorithm for the minimum energy broadcast problem in wireless ad hoc networks. *Applied Soft Computing*, 11(1):667–674.

Singh, A. and Gupta, A. K. (2006). A hybrid heuristic for the maximum clique problem. *Journal of Heuristics*, 12(1):5–22.

Sobel, I. (1974). On calibrating computer controlled cameras for perceiving 3-d scenes. *Artificial Intelligence*, 5(2):185–198.

Solnon, C. and Fenet, S. (2006). A study of aco capabilities for solving the maximum clique problem. *Journal of Heuristics*, 12(3):155–180.

Soto, R., Crawford, B., Galleguillos, C., Monfroy, E., and Paredes, F. (2013). A hybrid ac3-tabu search algorithm for solving sudoku puzzles. *Expert Systems with Applications*, 40(15):5817–5821.

Soto, R., Crawford, B., Galleguillos, C., Monfroy, E., and Paredes, F. (2014). A prefiltered cuckoo search algorithm with geometric operators for solving sudoku problems. *The Scientific World Journal*, 2014.

Soto, R., Crawford, B., Galleguillos, C., Paredes, F., and Norero, E. (2015). A hybrid alldifferent-tabu search algorithm for solving sudoku puzzles. *Computational Intelligence and Neuroscience*, 2015:40.

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

Subramanian, P., Ramkumar, N., Narendran, T., and Ganesh, K. (2013). Prism: Priority based simulated annealing for a closed loop supply chain network design problem. *Applied Soft Computing*, 13(2):1121–1135.

Taherinejad, N. (2009). Highly reliable harmony search algorithm. In *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pages 818–822. IEEE.

Takayuki, Y. and Takahiro, S. (2003). Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer sciences*, 86(5):1052–1060.

Tam, V. and Ma, K. T. (2004). Combining meta-heuristics to effectively solve the vehicle routing problems with time windows. *Artificial Intelligence Review*, 21(2):87–112.

Tanweer, M., Suresh, S., and Sundararajan, N. (2015). Self regulating particle swarm optimization algorithm. *Information Sciences*, 294:182–202.

Tomita, E. and Kameda, T. (2007). An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *Journal of Global Optimization*, 37(1):95–111.

Tomita, E. and Seki, T. (2003). An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete Mathematics and Theoretical Computer Science*, pages 278–289. Springer.

Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., and Wakatsuki, M. (2010). A simple and faster branch-and-bound algorithm for finding a maximum clique. In *WALCOM: Algorithms and Computation*, pages 191–203. Springer.

Valaei, M. and Behnamian, J. (2017). Allocation and sequencing in 1-out-of-n heterogeneous cold-standby systems: Multi-objective harmony search with dynamic parameters tuning. *Reliability Engineering & System Safety*, 157:78–86.

Valente, J. M., Moreira, M. R., Singh, A., and Alves, R. A. (2011). Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs. *The International Journal of Advanced Manufacturing Technology*, 54(1):251–265.

Verma, N. K. (2012). Future image frame generation using artificial neural network with selected features. In *Applied Imagery Pattern Recognition Workshop (AIPR), 2012 IEEE*, pages 1–8. IEEE.

Verma, N. K., Tamrakar, P., and Agrawal, S. (2010). Generating future satellite image frame using artificial neural network. In *Image and Video Processing and Computer vision (IVPCV-2010), International Conference*, pages 158–164.

Wan, X. and Xu, G. (1996). Camera parameters estimation and evaluation in active vision system. *Pattern Recognition*, 29(3):439–447.

Wang, C.-M. and Huang, Y.-F. (2010). Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, 37(4):2826–2837.

Wang, D., Tu, Y., and Zhang, T. (2008a). Research on the application of pso algorithm in non-linear camera calibration. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 4495–4500. IEEE.

Wang, L. (1998). Learning and retrieving spatio-temporal sequences with any static associative neural network. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 45(6):729–738.

Wang, L. and Li, L.-p. (2013). An effective differential harmony search algorithm for the solving non-convex economic load dispatch problems. *International Journal of Electrical Power & Energy Systems*, 44(1):832–843.

Wang, L., Liu, W., and Shi, H. (2008b). Noisy chaotic neural networks with variable thresholds for the frequency assignment problem in satellite communications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):209–217.

Wang, L. and Smith, K. (1998). On chaotic simulated annealing. *IEEE Transactions on Neural Networks*, 9(4):716–718.

Wang, X., Gao, X.-Z., and Ovaska, S. J. (2009). Fusion of clonal selection algorithm and harmony search method in optimisation of fuzzy classification systems. *International Journal of Bio-Inspired Computation*, 1(1-2):80–88.

Wang, Y., Cai, Z., and Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15(1):55–66.

Wang, Z., Yasuda, T., and Ohkura, K. (2015). An evolutionary approach to sudoku puzzles with filtered mutations. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 1732–1737. IEEE.

Weng, J., Cohen, P., and Herniou, M. (1992). Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):965–980.

Weyland, D. (2012). A rigorous analysis of the harmony search algorithm: how the research community can be. *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends: Advancements and Trends*, page 72.

Weyland, D. (2015). A critical analysis of the harmony search algorithmhow not to solve sudoku. *Operations Research Perspectives*, 2:97–105.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Wolpert, D. H., Macready, W. G., et al. (1995). No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute.

Wong, K. W. (1975). Mathematical formulation and digital analysis in close-range photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 44(11).

Wu, G., Mallipeddi, R., Suganthan, P., Wang, R., and Chen, H. (2016). Differential evolution with multi-population based ensemble of mutation strategies. *Information Sciences*, 329:329–345.

Wu, Q. and Hao, J.-K. (2015a). A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3):693–709.

Wu, Q. and Hao, J.-K. (2015b). Solving the winner determination problem via a weighted maximum clique heuristic. *Expert Systems with Applications*, 42(1):355–365.

Xinchao, Z. (2011). Simulated annealing algorithm with adaptive neighborhood. *Applied Soft Computing*, 11(2):1827–1836.

Xing, Y., Sun, J., and Chen, Z. (2007). Analyzing and improving of neural networks used in stereo calibration. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 1, pages 745–749. IEEE.

Yadav, P., Kumar, R., Panda, S. K., and Chang, C. (2012). An intelligent tuned harmony search algorithm for optimisation. *Information Sciences*, 196:47–72.

Yang, G., Yi, J., Zhang, Z., and Tang, Z. (2009). A tcnn filter algorithm to maximum clique problem. *Neurocomputing*, 72(4):1312–1318.

Yoo, D. G., Kim, J. H., and Geem, Z. W. (2014). Overview of harmony search algorithm and its applications in civil engineering. *Evolutionary Intelligence*, 7(1):3–16.

Zainuddin, Z., Lai, K. H., and Ong, P. (2016). An enhanced harmony search based algorithm for feature selection: Applications in epileptic seizure detection and prediction. *Computers & Electrical Engineering*, 53:143–162.

Zarro, R. D. and Anwer, M. A. (2016). Recognition-based online kurdish character recognition using hidden markov model and harmony search. *Engineering Science and Technology, an International Journal*.

Ze-Tao, J., Wenhuan, W., and Min, W. (2008). Camera autocalibration from kruppa's equations using particle swarm optimization. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 1, pages 1032–1034. IEEE.

Zeng, B. and Dong, Y. (2016). An improved harmony search based energy-efficient routing algorithm for wireless sensor networks. *Applied Soft Computing*, 41:135–147.

Zhan, Z.-H., Zhang, J., Li, Y., and Chung, H. S.-H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1362–1381.

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

Zhang, J., Wu, Y., Guo, Y., Wang, B., Wang, H., and Liu, H. (2016). A hybrid harmony search algorithm with differential evolution for day-ahead scheduling problem of a microgrid with consideration of power flow constraints. *Applied Energy*, 183:791–804.

Zhang, J.-d., Lu, J.-g., Li, H.-l., and Xie, M. (2012). Particle swarm optimisation algorithm for non-linear camera calibration. *International Journal of Innovative Computing and Applications*, 4(2):92–99.

Zhang, Q., Sun, J., and Tsang, E. (2005). An evolutionary algorithm with guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation*, 9(2):192–200.

Zhao, S.-Z., Suganthan, P. N., Pan, Q.-K., and Tasgetiren, M. F. (2011). Dynamic multi-swarm particle swarm optimizer with harmony search. *Expert Systems with Applications*, 38(4):3735–3742.

Zuckerman, D. (2006). Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690. ACM.

Zuorro, A., Fidaleo, M., and Lavecchia, R. (2011). Enzyme-assisted extraction of lycopene from tomato processing waste. *Enzyme and Microbial Technology*, 49(6):567–573.

Publications from this research work

Research Papers in International Journals

1. Assad, A., & Deep, K. (2017). Harmony search based memetic algorithms for solving sudoku. International Journal of System Assurance Engineering and Management, 1-14.Springer.(Accepted)

2. Assad, A., & Deep, K. (2017). A Two Phase Harmony Search Algorithm for Continuous Optimization. Computational Intelligence. Wiley. (Accepted)

3. Assad, A., & Deep, K. (March 2017). A Heuristic Based Harmony Search Algorithm for Maximum Clique Problem. Opsearch. Springer. (Accepted)

4. Assad, A., & Deep, K. (April 2017). A Hybrid Harmony Search and Simulated Annealing Algorithm for Continuous Optimization. Information Sciences. Elsevier. (Under Review)

5. Assad, A., & Deep, K. (July 2017). Shrinking Memory Harmony Search Algorithm for Continuous Optimization. IEEE Transactions on Cybernetics.IEEE. (Under Review)

Research Papers in International Conferences

1. Assad, A., & Deep, K. (2016). Applications of Harmony Search Algorithm in Data Mining: A Survey. In Proceedings of Fifth International Conference on Soft Computing for Problem Solving (pp. 863-874)., Advances in Intelligent Systems and Computing, Springer Singapore.

Appendix A
Sample Sudoku puzzles used for experimentation

Figures A.1, A.2, A.3, A.4 and A.5 respectively shows the sodoku puzzles of level Beginner, Easy, Medium, Hard and Expert used for numerical experimentation in this Thesis.

9		7	3	5		1		4
3			6	4	1			
	6	1			7	5		2
	1					4		7
2	3						5	1
7		5					2	
5		4	8			3	7	
			7	3	5			8
8		3		6	4	9		5

2	3		4	8	5	6	9	1
							3	4
8	4			1				5
9					8	5		7
			3	7	9			
6		4	2					9
3				9			2	8
4	1							
7	8	9	5	6	2		4	3

5			1			7		
6	1			9				5
2		3	4				8	
7	6	5		1	2	3		
3	4		6		9		5	8
		9	5	4		6	1	7
	5				4	9		6
9				6			4	1
		6			1			3

	4					1	7	9
		2			8		5	4
		6			5			8
	8			7		9	1	
	5			9			3	
	1	9		6			4	
3			4			7		
5	7		1			2		
9	2	8					6	

8		2		5		7		1
		7		8	2	4	6	
	1		9					
6					1	8	3	2
5								9
1	8	4	3					6
					4		2	
	9	5	6	1		3		
3		8		9		6		7

Figure A.1: Beginner level sudoku puzzles.

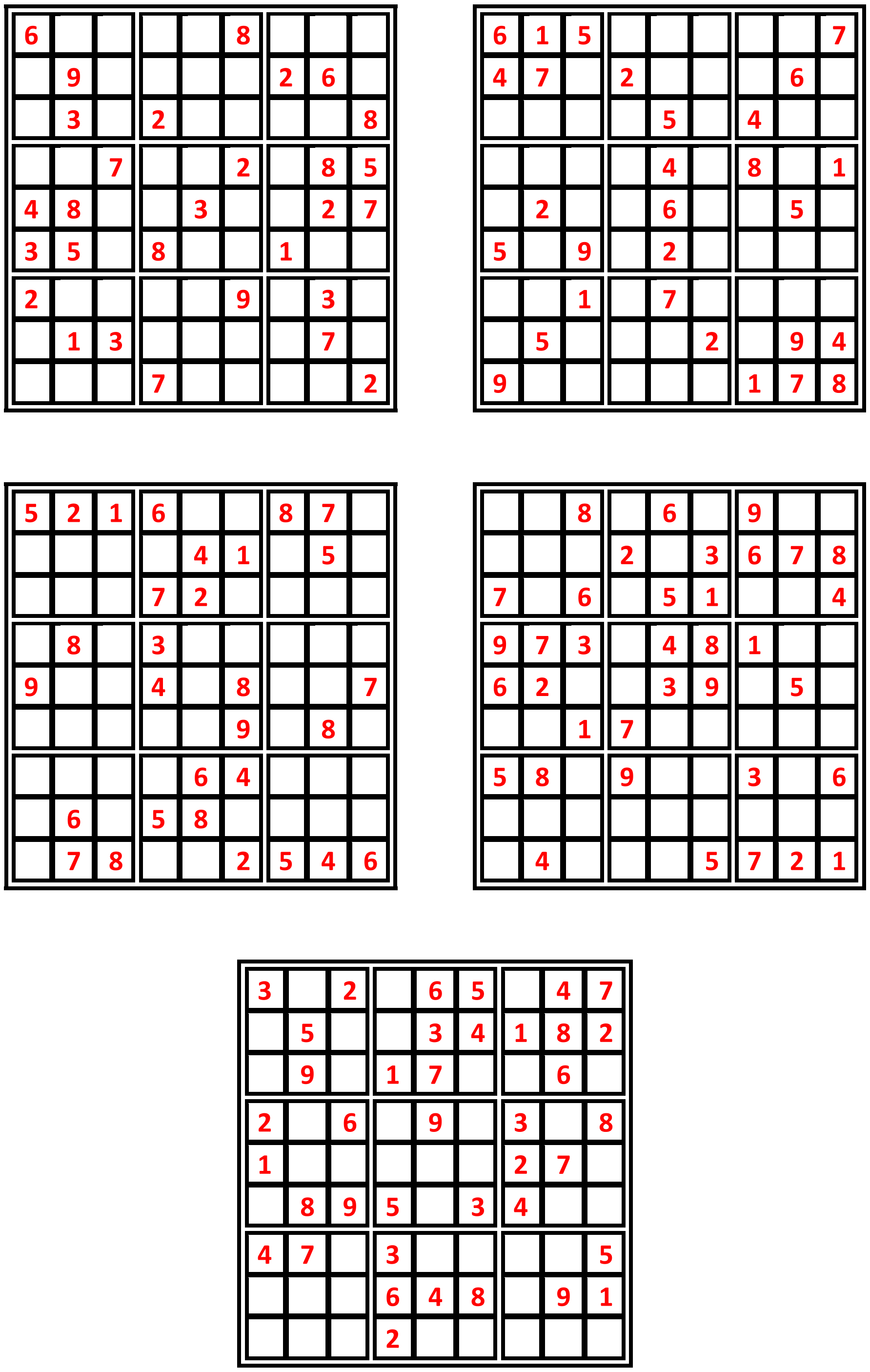
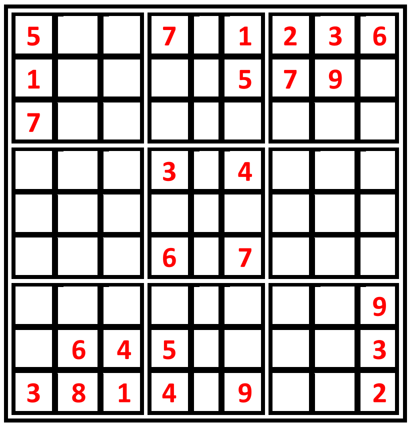
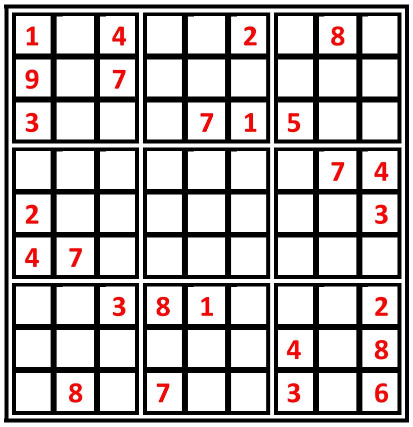


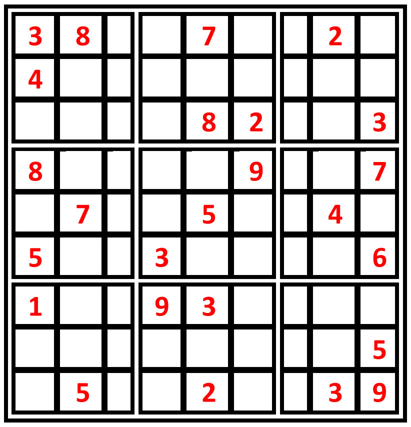
Figure A.2: Easy level sudoku puzzles.



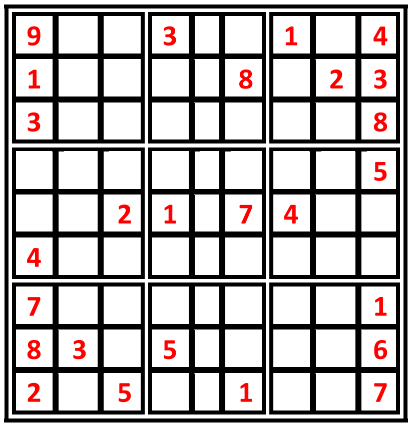
5			7		1	2	3	6
1					5	7	9	
7								
			3		4			
			6		7			
								9
	6	4	5					3
3	8	1	4		9			2



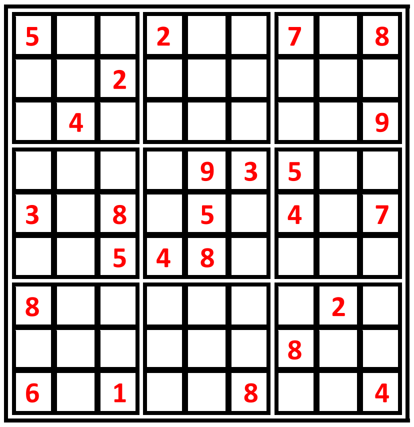
1		4			2		8	
9		7						
3				7	1	5		
							7	4
2								3
4	7							
		3	8	1				2
						4		8
	8		7			3		6



3	8			7			2	
4								
				8	2			3
8					9			7
	7			5			4	
5			3					6
1			9	3				
								5
	5			2			3	9



9			3			1		4
1					8		2	3
3								8
								5
		2	1		7	4		
4								
7								1
8	3		5					6
2		5			1			7



5			2			7		8
		2						
	4							9
				9	3	5		
3		8		5		4		7
		5	4	8				
8							2	
						8		
6		1			8			4

Figure A.3: Medium level sudoku puzzles.

					6	4		
	8		4	5	3	1	2	
	6							
				6				
							4	
	4	6	5	2	9		3	
		2	7					

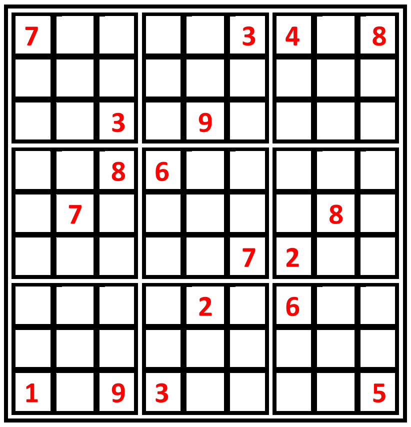
5				2				
		4			9			
9			4				2	1
		6		9		4		
4	2				1			8
			9			2		
				1				6

1			6	8				
	9		4				5	
3								
8				5				
			1		8			
				9				2
								3
	6				1		4	
				4	3			5

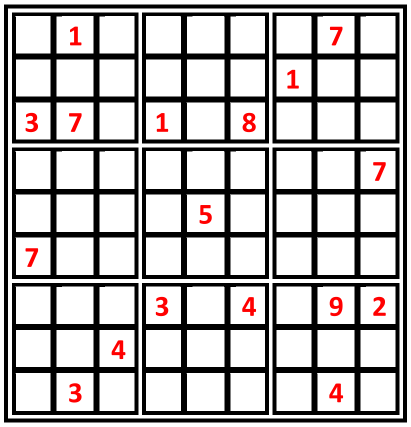
5			2					
8						2	5	
		3	7					
	6		1	8	2		9	
					9	7		
	5	2						3
					1			2

		3						
2						5	4	
9			8					
5			6					
		9		7		6		
					3			4
					2			5
	7	5						1
						3		

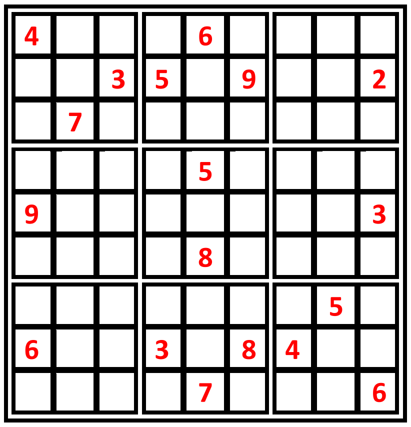
Figure A.4: Hard level sudoku puzzles.



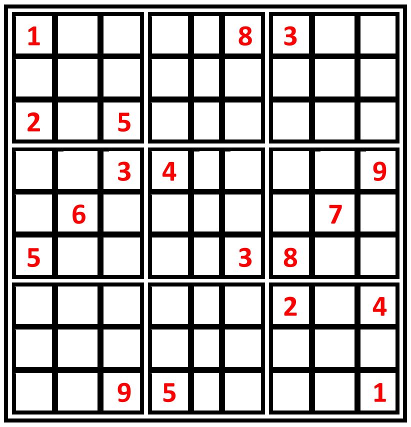
7					3	4		8
		3		9				
		8	6					
	7						8	
					7	2		
				2		6		
1		9	3					5



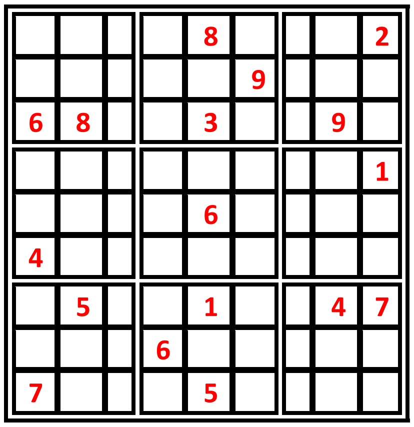
	1						7	
						1		
3	7		1		8			
								7
				5				
7								
			3		4		9	2
		4						
	3						4	



4				6				
		3	5		9			2
	7							
				5				
9								3
				8				
							5	
6			3		8	4		
				7				6



1					8	3		
2		5						
		3	4					9
	6						7	
5					3	8		
						2		4
		9	5					1



				8				2
					9			
6	8			3			9	
								1
				6				
4								
	5			1			4	7
			6					
7				5				

Figure A.5: Expert level sudoku puzzles.