Convergence Analysis and Numerical Simulation of Particle Swarm Optimization

by

Ganesha Weerasinghe

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 7, 2017

Keywords: Particle Swarm Optimization, Stochastic optimization simulation, quasi-Monte Carlo Simulation, Randomized Low-discrepancy sequences, optimal Halton sequence, Almost Sure Convergence

Approved by

Yanzhao Cao, Chair, Professor of Mathematics
Xiaoying Han, Professor of Mathematics
Junshan Lin, Assistant Professor of Mathematics
Hans-Werner van Wyk, Assistant Professor of Mathematics
Kaijun Liu, Assistant Professor of Physics

Abstract

Nature, including mankind, loves to optimize and the world is far from being deterministic. Thus, stochastic optimization is an important and active area of research in applied mathematics. Inspired by the social behavior of bird flocking or fish schooling, particle swarm optimization (PSO) is a population based stochastic optimization method developed by Eberhart and Kennedy in 1995. It has been used across a wide range of applications. Uniform random numbers are used to initialize the particles in the PSO algorithm. In literature, Faure, Halton and Van der Corput sequences have been used for initializing the swarm in PSO. Quasi-random (or low-discrepancy) sequences such as Faure, Halton, Van der Corput, etc, are deterministic and suffer from correlations between radical inverse functions with different bases used for different dimensions. These correlations results in poorly distributed $2D$ projections. A standard solution to these is to use randomized versions of the quasi-random sequences. In this dissertation, we investigate the effect of initializing the swarm with the optimal Halton sequence, which is a randomized quasi-random sequence. This ensures that we still have the uniformity properties of quasi-random sequences while preserving the stochastic behavior for particles in the swarm.

The central part of this dissertation is the convergence analysis of PSO. Though there have been several studies on the convergence of PSO, the convergence proofs in these studies are either imprecise or based on unrealistic assumptions and, to the best of our knowledge, incomplete. Here, we present almost surely convergence analysis of PSO, based on the properties of the objective function.

The last part of this dissertation is about numerical experiments based on quasi Monte Carlo sampling algorithms. Numerical experiments are conducted with benchmark objective functions in high dimensions (starting from two dimensions and then in 10, 20, 30, 50, 70

and 100 dimensions) to verify the convergence and effectiveness of the proposed initialization of PSO.

Acknowledgments

First and foremost I would like to express my profound gratitude to my advisor, Professor Yanzhao Cao, for his continuous support and encouragement of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me through the years of research and writing that it took to produce this thesis. I could not imagine having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Xiaoying Han, Dr. Hans-Werner van Wyk, and Dr. Junshan Lin for their insightful comments. In particular, I would like to thank Dr. Kaijun Liu from the Department of Physics for taking on the responsibility of reading and evaluating my thesis and for his genuine interest in my work.

My sincere thanks also goes to Dr. Ming Liao for checking the accuracy of content in some parts of Chapter 4.

In addition, I would like to express my sincerest appreciation to Dr. Hongmei Chi at Florida A&M University, for offering me the opportunity to work with her during the summer of 2015. I wish to take this opportunity to thank her for all of her continuous support and guidance.

Furthermore, it is with great pleasure that I express my thankfulness to Prof. Claude Bélisle from Department of Mathematics and Statistics at Université Laval (Quebec, Canada) for the highly valuable discussion through emails. I appreciate his patience and the time he spent on writing the detailed e-mails.

I also want to thank my fellow colleagues in the Department of Mathematics at Auburn University, especially Tahir Issa, Rachidi Salako and Xiangqian Meng for their valuable discussions and time that they have spent with me.

Last but not the least, I would like to thank my parents and my sisters for their innumerable sacrifices unchanging support and confidence in me. I would also express my gratitude to my parents in-law for their great support. My appreciation goes to my husband, Asanka for providing me with unfailing support and continuous encouragement throughout my graduate studies and writing this thesis. This accomplishment would not have been possible without you. Thank you. I dedicate this thesis to my son, Sawain for having patience with me when I am not around.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

a.s	almost sure convergence
ACO	Ant Colony Optimizaton
GA	Genetic Algorithm
PSO	Particle Swarm Optimization

Chapter 1

Introduction

Optimization plays an important role in our everyday life. Managers at a firm are faced with making short and long-term investment decisions in order to increase profit and/or minimize cost, researchers design laboratory experiments to extract the maximum information about the efficiency of a new drug, aerospace engineers run computer simulations to refine the design of a missile or aircraft, traffic engineers set the timing strategy for the signals in a traffic network to minimize the delay incurred by vehicles in the network. Optimization offers a technique for solving problem of this type. Other applications of optimization include engineering design in mechanical, civil, and chemical engineering, structural optimization, applications in electronics and electromagnetic, chip design, antenna design, power generation and power system, design and optimization in communication networks, biological, medical and pharmaceutical applications, scheduling, combinatorial optimization problems, neural networks, robotics, security and military applications, finance and economics. The goal of a global optimization algorithm is to find the absolute maximum or minimum of the objective function involving unknown variables in which the variables may or may not be restricted by constraints. A global optimization problem may contain several local optimum or stationery points.

Global optimization can be classified as either deterministic or stochastic. Many phenomena in the real world involve uncertainty. Due to the difficulties in many real world problems and the randomness in information, stochastic optimization methods are often applied to real-life problems.

According to James Spall [48] the stochastic optimization problems exhibit at least one of the following two features.

1. There is a random noise in the measurements of objective function.

   and/or

2. There is a random choice made in search direction as the algorithm iterates toward a solution.

In classical deterministic search and optimization, it is assumed that one has perfect information about the objective function and its derivatives. This information is then used to determine the search direction in a deterministic manner. In many practical problems, such information is not available, indicating that deterministic algorithms are inappropriate. In such cases, it is beneficial to deliberately introduce randomness into the search process. This injected randomness causes the problem to fall under the category 2 above. These random search methods have been shown to have a potential to solve problems efficiently in a way that is not possible for deterministic algorithms. Another advantage is that they are relatively easy to implement and more efficient to compute complex problems. These random search methods are widely applied to continuous and discrete global optimization problems. Because the methods rely only on function evaluations, rather than gradient and hessian information, they can be coded quickly and applied to broad class of ill structured problems.

Stochastic approximation (Robbins-Monro algorithm 1951) [42], simulated annealing (1979) [24, 25], tabu search (1986) [14, 15, 16], genetic algorithms (GAs) , evolutionary programming (1966) [12], ant colony optimization (ACO) (1992) [32, 8], particle swarm optimization (1995) [9], nested partitioning method (2000) [45] are some of the random

search methods and these methods are applied widely to global optimization problems [57], [58]. Stochastic optimization can be further classified as population-based optimization algorithms. They differ from other optimization methods, in the fact that they maintain a population of potential solutions to a problem, and not just one solution. In population-based algorithms first a population of individuals is randomly initialized. Here each individual represents a potential solution to the problem. The current population is used to generate candidate points for a new population, by using some mechanisms such as mutation, crossover or reproduction. The current and new points are then evaluated and compared to update the population. The quality of each solution is evaluated using the objective function. This procedure is repeated until a certain criteria is met. The motivation behind population-based algorithms is to parallel the process of biological evolution. Examples for population-based algorithms are genetic algorithms (GAs), evolutionary programming, ant colony optimization (ACO), particle swarm optimization (PSO), differential evolution [49] and fire fly algorithm (2008) [55].

Among these methods we focus on particle swarm optimization due to several reasons. Firstly, researchers conclude that PSO was generally found to perform better than other algorithms in terms of success rate, solution quality, convergence speed, computational efficiency, and quality of the results while being second best (after ACO) in terms of processing time [10] , [19]. Secondly, it is one of the simplest algorithms in terms of complexity of equations and thirdly, it is relatively new emerging computational methodology compared to other techniques and thus less research is focused on it when compared to other methods.

Inspired by the social behavior of bird flocking or fish schooling, PSO [9] was developed by Eberhart and Kennedy in 1995. Since then, PSO has been used across a wide range of applications. Areas where PSO have shown particular promise include black-box functions, multimodal problems and problems for which there is no specialized method available or all specialized methods give unsatisfactory results [40], [28], [29], [23].

Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate position with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, also called as particles, and moving these particles around in the search-space according to simple mathematical formulae that specify the particle's position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions it has achieved so far. More about PSO method and it's algorithm is discussed in chapter 2.

Original version of PSO was unable to solve discontinuous problems. Thus, it was hybridized in literature to have a better performance. Over the past two decades researches of PSO have been attentive on several aspects of PSO algorithm such as initialization of particles [38], [37], parameter selection [46], [50], [22], convergence analysis [36], [7], [56], modification or hybridized versions [1], [30], [54], [43], [21], [20], [41] and applications of PSO. [39] contains a list of applications within each different application area.

Initialization of particles plays an important role in population based optimization techniques. If the initial population does not cover the search area efficiently, it may not be able to locate the global optimum and may converge to a local optimum point. In population based algorithms, pseudo-random numbers (monte carlo sequences) are used to produce the initial population. Because pseudo-random numbers generate uniformly distributed samples and thus covers promising regions of the search space.

Since the population size is limited, the chance for a population to cover the promising regions of the search space decreases as the dimension increases. Quasi-random sequences which is also called as quasi-monte carlo sequences or low-discrepancy sequences are more evenly distributed over the D dimensional unit cube, thus it improves the accuracy of the estimation and provides better estimation in population search algorithms rather than monte

carlo sequences. Researchers in population based algorithms has used quasi-random sequences like Van der Corput, Halton, Faure and Sobel sequences to initialize the population [3], [26], [38], [37], [34] and using numerical experiments they showed that these initializations improves the performance when compared to monte carlo sequences. Kimura and Matsumara [26], performed GA using Halton sequence and discovered that performance of GA using Halton sequence is superior to the same GA using psedo-ramdom number generator. More about quasi-random sequences are discussed in chapter 3.

Although quasi-random sequences have advantages over monte carlo sequences, they have some limitations. The drawback here is, that these sequences suffers from correlation effect between radical inverse functions with different bases used for different dimensions [33], [4]. Also quasi-random sequences provides deterministic behavior as opposed to the stochastic behavior. To address these drawbacks of quasi-Monte Carlo methods, researchers introduced randomized versions of quasi-Monte Carlo sequences either by random shifting, random permutation of digits or linear scrambling [17]. This guarantees the uniformity properties of the quasi-random sequences and random shifting, permutation or scrambling provides stochastic behavior. It is important to have stochastic initialization, since PSO is a stochastic optimization method. In this dissertation, we initialize the particles using optimal Halton sequence [4], which belongs to the family of randomized quasi-random sequences. More about these sequences are discussed in chapter 3. Numerical experiments are performed to check the convergence and it can be seen that particles converge to the optimum value when particles are initialized using randomized quasi-random sequence. These numerical results are performed in section 5.2 under chapter 5.

Further, the theoretical analysis of the PSO is lacking in the literature. According to Poli and et al. [40], this is due to several reasons.

1. The PSO is made up of a large number of interacting elements (the particles).
   Although the nature of the elements and of the interactions is simple, understanding the dynamics of the whole is nontrivial.

2. The particles are provided with memory and the ability to decide when to update the memory.

3. Forces are stochastic. This prevents the use of standard mathematical tools used in the analysis of dynamical systems.

4. The behavior of the PSO depends crucially on the structure of the fitness function. However, there are infinitely many fitness functions, and so it is extremely difficult to derive useful results that apply to all, even when complete information about the fitness function is available.

Because of these difficulties researchers have been used simplifying assumptions in order to obtain models that could be studied mathematically. These include: assuming that there is only one particle in the swarm (isolated single individuals), assuming that particle's personal best value and global best values are constant at each iteration, assuming that no improved solutions are found (search stagnation), and assuming that the model is deterministic (absence of randomness) [11].

Although these assumptions allow a researcher to observe a particular aspect of the particle's dynamics, their effect on the system's behavior may be severe. For example, the problem in the assumption of one particle is, when there is one particle in the swarm the particle's personal best value and global best values coincide. This is not true in general. The assumption that particle's personal best value and global best values are constant is not practical as well because these values keep updating at each iteration. So, the conclusions drawn from the resulting mathematical models tend to be approximate and need to be verified empirically.

Ozcan and Mohan (1998) [36] theoretically analyzed the model assuming one particle, in isolation, in one dimension, during stagnation, in the absence of stochasticity and in 1999 the work was extended to multiple multi dimensional particles. Clerc and Kennedy (2002) [7] assumed one particle, one dimension, deterministic behaviour and stagnation. They predict

that the particles will converge to the equilibrium if the magnitude of the eigen values of the position and velocity transition matrix is less than 1. They derived relationships between the parameters that would guarantee convergence. This now has become almost the standard PSO algorithm. van den Bergh (2001) [51] assumed one particle, with no stochasticity, during stagnation and showed that the particles are attracted towards a fixed point corresponding to a weighted sum of particle's global best value and personal best value. The work also suggested the possibility that particles may converge to a point which is not an global optimum. Chuan and Quanyuan (2007) [5] studied convergence analysis of PSO assuming the model is deterministic. There has been conspicuously lacking in stochastic convergence analysis of PSO. Jiang and et al [22] analysed convergence stochastically assuming one particle and particle's global best value is constant.

There is no analysis based on the characteristics of objective function. Properties of objective function plays a main role in any optimization problem and the behavior of the PSO depends crucially on the structure of objective function [40]. Thus properties of the objective function needs to be considered. Further, the convergence analysis proofs in the literature do not guarantee the convergence to the optimizer. i.e those studies do not imply that the equilibrium is the optimizer. Thus, we argue that the theoretical studies in the literature and the convergence proofs in these studies are either imprecise or based on unrealistic assumptions. In this dissertation, we present stochastic convergence analysis i.e almost sure convergence analysis of PSO algorithm to the optimizer based on following assumptions. These assumptions are on the objective function and there is another one on the algorithm of PSO. The assumptions on objective function are,

1. The objective function is bounded.

2. The objective function is continuous.

3. There is a unique optimum value.

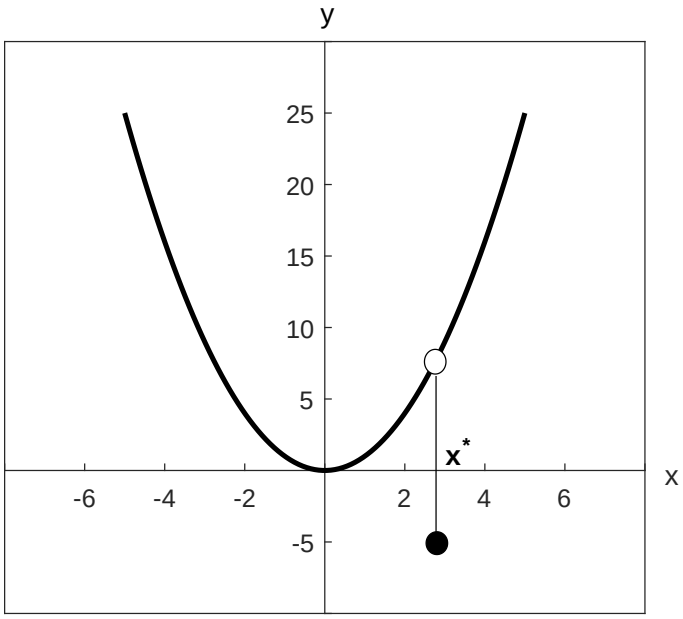

Figure 1.1: Example for a discontinuous function

These assumptions are reasonable. Because if the function is not bounded, the optimum point will be either ∞ or $-\infty$ and in real world problems we don't get these values. We have the second assumption as we avoid including situations where the minimum of the function occurs at which the function is singularly discontinuous. For example consider figure 1.1, for the function on figure 1.1 there is no guarantee of obtaining the global optimum without prior knowledge of existence of such a point. According to [48], no typical algorithm would be able to locate this point and in that case true minimum will never be discovered. The third assumption is due to the fact that PSO algorithm cannot detect the number of global optimal points and it will provide one of the optimal points. Thus here we restrict our proof to the case where there is only one global optimum value.

We have one assumption on algorithm of PSO. We assume that the two random numbers in the velocity equation are not 0 simultaneously. If both random numbers are zero we assign two other random numbers.

Using these assumptions first we prove that particle's personal and global best values converge to the optimum value almost surely. This proof is inspired by the convergence proof for sequential random search algorithms by Solis and Wets [47] and proof of convergence analysis of simulated annealing by Bélisle [2]. Then using the fact that particle's personal and global best values converge to the optimum value almost surely, we prove the almost sure convergence of the swarm particles to the optimizer of the objective function. This is one of the main results of this dissertation.

The outline of this work is as follows. In Chapter 2, we give a brief overview of the particle swarm optimization algorithm that will be used throughout the rest of the chapters. In Chapter 3, we present the importance of initialization of particles and describe the Halton sequence and optimal Halton sequence (scrambled Halton sequence). Also, we present our proposed initialization of PSO. In Chapter 4, almost sure convergence of PSO is proved. In Chapter 5, numerical experiments are carried out to demonstrate the performance and efficiency of the proposed initialization. Finally conclusion and future work are presented in Chapter 6.

Chapter 2

Particle Swarm Optimization

2.1 Introduction to Optimization

In this chapter we first introduce some basic concepts relating to optimization, followed by an introduction of particle swarm optimization.

In mathematical terms, optimization is the minimization or maximization of a function (called objective function or fitness function) subject to constraints on its variables. Without loss of generality, consider a minimization problem in the D dimensional space, where $f : \Re^D \to \Re$.

$$\begin{aligned} &Minimize : f(\vec{a}) \\ &Subject\ to : \\ &g_k(\vec{a}) \leq 0 \qquad k = 1, 2, ...p \\ &h_m(\vec{a}) = 0 \qquad m = 1, 2, ...q \\ &a^j_{min} \leq a^j \leq a^j_{max} \qquad j = 1, 2, ...D \end{aligned} \tag{2.1}$$

where $\vec{a} = (a^1, a^2, ..., a^D)$ is the vector of variables, also known as unknowns or parameters; f is the objective function with variables $\vec{a}$ to be optimized and p, q are number of inequality and equality constraints respectively.

Hereafter all the optimization problems are assumed to find minima; a maximal problem could be transformed to minimal form by conveniently multiplying the objective function by -1. The solution to the above system may be a unique point, a countable collection of points, or a set containing an uncountable number of points. To solve the optimization problem, there are optimization algorithms inspired by different techniques. But they all share some common characteristics. They are iterative, i.e they all begin with an initial

guess of the optimal values of the variables and generate a sequence of improved estimates until they converge to a solution. The strategy used to move from one potential solution to the next is what distinguishes one algorithm from another. For instance, some of them use gradient based information and other similar methods (e.g. the first and second derivatives of these functions) to lead the search toward more promising areas of the design domain, whereas others use problem specific heuristics (e.g. accumulated information gathered at previous iterations).

PSO is an iterative algorithm that engages a number of particles flying iteratively over the search space of the given objective function. Each particle determines its movement through the search space by combining information about its current fitness, its best fitness from previous locations (individual perspective) and best fitness locations with regards to one or more members of the swarm (social perspective), with some random perturbations.

2.2 Classical Particle Swarm Optimization

The Classical PSO algorithm maintains a population of M particles, and places them in the search space of the objective function. PSO defines each particle's position as a potential solution to the function to be optimized, and then searches for the optima by updating its position in every iterative step. Each particle is associated with a velocity which directs the flying of the particle toward a new, presumably better, position/solution. The particles fly through the problem space by following the current optimum particles.

In every iteration, each particle's velocity is updated by following two best values. The first one is current best solution it has achieved so far. This value is called *pbest*. Another best value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called *gbest*.

After finding the two best values, the particle updates its velocity and positions with following equations.

$$\vec{v}_{t+1} = w\vec{v_t} + c_1 r_{1,t+1}(\vec{pbest}_t - \vec{a_t}) + c_2 r_{2,t+1}(\vec{gbest}_t - \vec{a_t}). \tag{2.2}$$

$$\vec{a}_{t+1} = \vec{a_t} + \vec{v}_{t+1}. \tag{2.3}$$

Let D be the dimension of the search space, and let $a_t^{k,j}$ denote the k^{th} particle's j^{th} component at time t. Then $\vec{a^k}_t = (a_t^{k,1}, a_t^{k,2}, ..., a_t^{k,D})$ is the k^{th} particle's position at time t, $\vec{v^k}_t = (v_t^{k,1}, v_t^{k,2}, ..., v_t^{k,D})$ is the k^{th} particle's velocity at time t. $\vec{pbest^k}_t = (pbest_t^{k,1}, pbest_t^{k,2}, ..., pbest_t^{k,D})$ and $\vec{gbest}_t = (gbest_t^1, gbest_t^2, ..., gbest_t^D)$ are the vectors of current best values and global best values, respectively. $r_{1,t+1}$ and $r_{2,t+1}$ are uniformly distributed random numbers between 0 and 1. There are three factors w, c_1 and c_2 in equation 2.2. w factor is called the inertia weight and is simply a positive constant less than 1. c_1 and c_2 are the cognitive(or personal or local) weight and social or global weight respectively which are positive constants.

2.3 Algorithm of PSO Simulation

Algorithm of Standard PSO

1. Initialize a population array of M particles with random positions and velocities on D dimensions, in the search space.

   Loop

2. For each particle evaluate the objective function in D variables.

3. Compare each particle's objective function value with its $\vec{pbest^k}_{t-1}$ value. If current value is better than $\vec{pbest^k}_{t-1}$, then set $\vec{pbest^k}_t$ equal to the current value. i.e

$$\vec{pbest^k}_t = \begin{cases} \vec{a^k}_t, & \text{if } f(\vec{a^k}_t) < f(\vec{pbest^k}_{t-1}). \\ \vec{pbest^k}_{t-1}, & \text{if } f(\vec{a^k}_t) \geq f(\vec{pbest^k}_{t-1}). \end{cases} \tag{2.4}$$
$$for\ k = 1, 2, ..., M.$$

4. Identify the particle in the swarm with the best success so far and assign its position to the variable $\vec{gbest}_t$.

   Choose $\vec{a^p}_t$ s.t.$f(\vec{a^p}_t) \leq f(\vec{a^k}_t)$ for all $k = 1, 2, ..., M$

$$\vec{gbest}_t = \begin{cases} \vec{a^p}_t, & \text{if } f(\vec{a^p}_t) < f(\vec{gbest}_{t-1}). \\ \vec{gbest}_{t-1}, & \text{if } f(\vec{a^p}_t) \geq f(\vec{gbest}_{t-1}). \end{cases} \tag{2.5}$$

5. Update each particle's j^{th} dimension of velocity according to the following equation, for $k = 1, 2, ..., M$ and $j = 1, 2, ..., D$

$$v^{k,j}_{t+1} = wv^{k,j}_t + c_1 r_{1,t+1}(pbest^{k,j}_t - a_t) + c_2 r_{2,t+1}(gbest^j_t - a_t). \tag{2.6}$$

   To ensure that each component of $\vec{v^k}_{t+1}$ is kept within the search space, make the following modification

$$v^{k,j}_{t+1} = \begin{cases} v^j_{min}, & \text{if } v^{k,j}_{t+1} < v^j_{min}. \\ v^{k,j}_{t+1}, & \text{if } v^j_{min} \leq v^{k,j}_{t+1} \leq v^j_{max} \\ v^j_{max} & \text{if } v^j_{max} < v^{k,j}_{t+1}. \end{cases} \tag{2.7}$$

   where v^j_{min} and v^j_{max} are determined from constraints of the objective function or by setting $v^j_{min} = a^j_{min}$ and $v^j_{max} = a^j_{max}$ [46].

6. Update each particle's j^{th} dimension of position according to the following equation, for $k = 1, 2, ..., M$ and $j = 1, 2, ..., D$

$$a_{t+1}^{k,j} = a_t^{k,j} + v_{t+1}^{k,j}. \tag{2.8}$$

To ensure that each component of $\vec{a^k}_{t+1}$ is kept within the search space, make the following modification

$$a_{t+1}^{k,j} = \begin{cases} a_{min}^j, & \text{if } a_{t+1}^{k,j} < a_{min}^j. \\ a_{t+1}^{k,j}, & \text{if } a_{min}^j \leq a_{t+1}^{k,j} \leq a_{max}^j. \\ a_{max}^j & \text{if } a_{max}^j < a_{t+1}^{k,j}. \end{cases} \tag{2.9}$$

End of the Loop

7. If a criterion is met (after a certain number of iterations or until particle position converge to a certain value) exit loop.

Note: Here we define our stopping criteria as maximum number of iterations. If we set the stopping criteria with error bounds, for example if $\mid \vec{a^k}_{t+1} - \vec{a^k}_t \mid < \epsilon$ for some $\epsilon > 0$, then the particles trap in local optimum points. Thus we always set the stopping criteria as maximum number of iterations.

Figure 2.1 shows flow chart of the PSO algorithm.

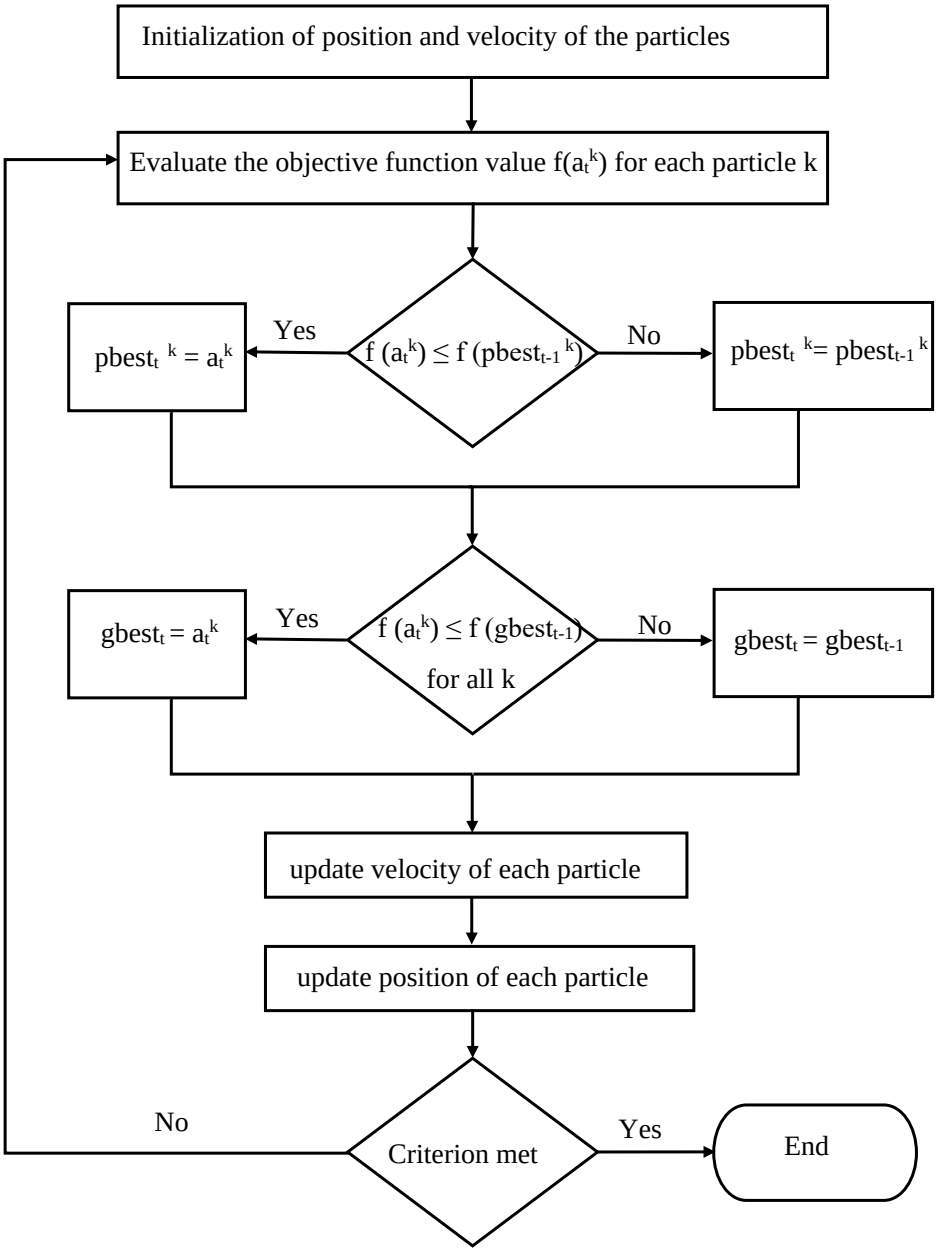

Figure 2.1: Flow chart for PSO algorithm

Chapter 3

Initialization of Particles

Initialization of particles plays an important role in population based optimization techniques. Initialization consists of choosing swarm size, choosing particle positions randomly in the search space and choosing random velocities. If the swarm population does not cover the search area efficiently, it may not be able to locate the global optima and/or may converge to a local optimum point.

In the classical PSO method particles are randomly initialized using pseudo-random numbers (monte carlo sequences). A problem with pseudo-random sequences is that values may clutter in one region, and other regions would be left out with a lower density of points. Hence, pseudo-random sequences cannot achieve optimal performance. Because of this reason, quasi-random sequences (also called as quasi-Monte carlo or low discrepancy sequences) have been proposed in the literature and later have been used in population-based methods to initialize particles. More details about quasirandom sequences are discussed in Section 3.1

3.1 Quasi-Monte Carlo / Quasi-random sequences

Quasi-Monte Carlo methods are often described as the deterministic version of Monte Carlo method. Quasi-Monte carlo methods, use sequences of numbers, called low discrepancy sequences, which are designed to be more evenly distributed over the D-dimensional unit cube. Thus they improve the accuracy of the estimation. Moreover, quasi-random initialization often leads to a faster convergence rate than normal Monte Carlo initialization.

Examples for quasi-random sequences are Van der Corput sequence, Halton sequence, Faure sequence, Sobol sequence. These sequences have been used to initialize population

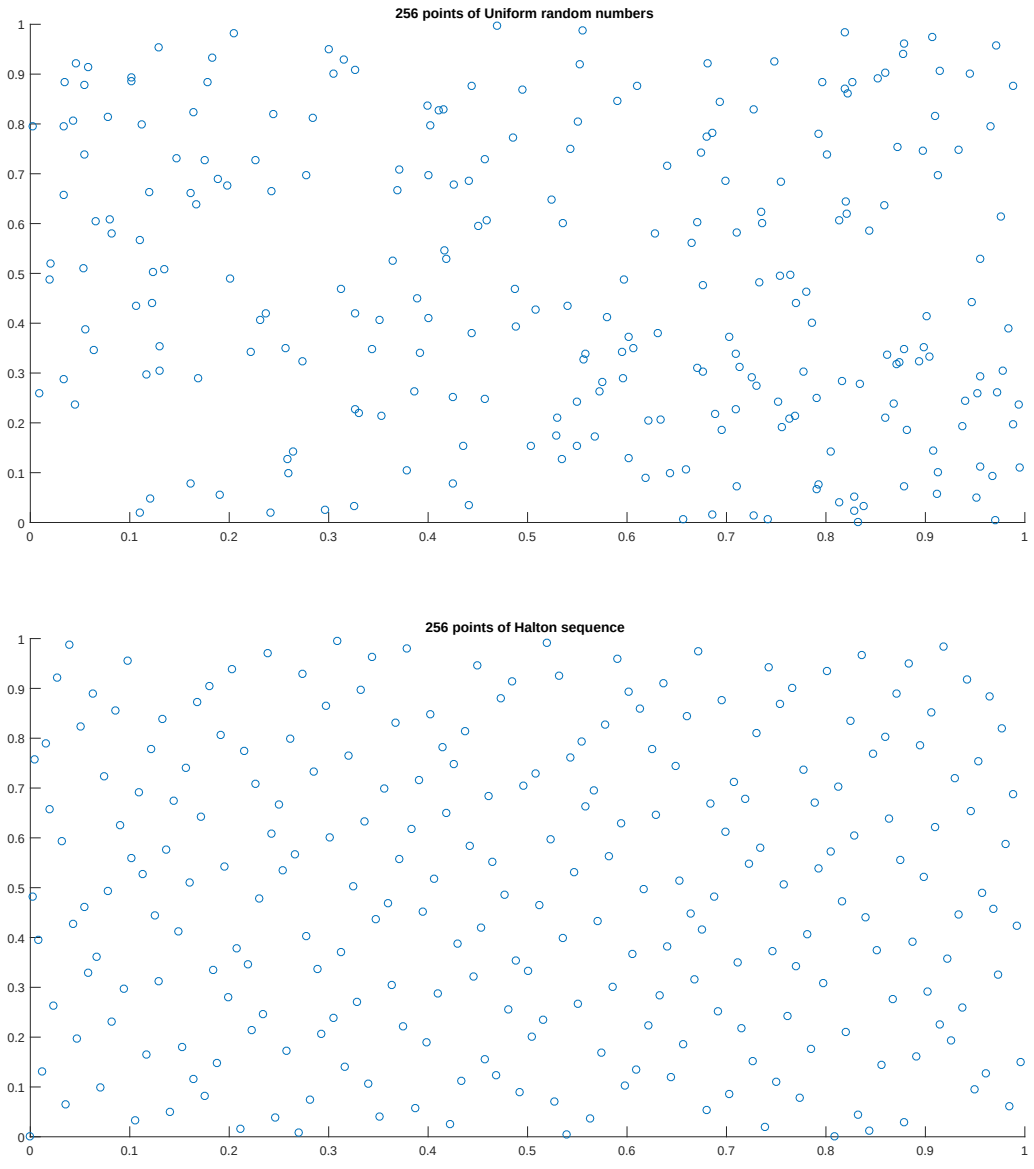

Figure 3.1: Comparison of uniform random sequence (top) and Halton sequence (bottom)

[3], [26], [38], [37], [34] and numerical results show that these initialization provides better estimation in population search algorithms rather than random (Monte Carlo) sequences. Omran et al (2013) [35] has used Halton and Sobol sequences to initialize the particles in two population based algorithms (PSO and Differential evolution) and compare the results. They claim that the results has no evidence of significant difference.

Figure 3.1 shows 256 points from a uniform pseudo-random number sequence (top) compared with the first 256 points from the 2,3 Halton sequence (bottom). We can see that Halton sequence covers the space more evenly than the uniform pseudo-random number sequence.

The Van der Corput sequence illustrates the main idea of construction of low discrepancy sequences.

3.1.1 Van der Corput Sequence

A Van der Corput sequence is an example of the simplest one-dimensional low-discrepancy sequence over the unit interval; it was first described in 1935 by the Dutch mathematician J. G. Van der Corput [52]. It is constructed by reversing the base-n representation of the sequence of natural numbers.

For any natural number n, let $n = b_0 + b_1 p + ... + b_m p^m$ be the $p-$adic expansion with $b_j \in \{0, 1, ..., p-1\}$ and $T = \lfloor \log_p n \rfloor$, then n^{th} term in the Van der Corput sequence is defined as,

$$\phi_p(n) \equiv \frac{b_0}{p} + \frac{b_1}{p^2} + ... + \frac{b_m}{p^{m+1}} \tag{3.1}$$

For example, Van der Corput sequence with base 2 is,

$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \frac{5}{16}, \frac{13}{16}, \frac{3}{16}, \frac{11}{16}, \frac{7}{16}, \frac{15}{16}, ...$

Note that Van der Corput sequence in any base is dense in $[0, 1)$. That is, for any real number in $[0, 1)$ there exists a subsequence of the Van der Corput sequence that converges to that number.

In [37], [38] M. Pant et al used a Van der Corput sequence to initialize the particles in the swarm of PSO.

3.1.2 Halton Sequence

A classical family of low-discrepancy sequences are the Halton sequences [18], which are bases on the radical inverse function defined as follows:

$$\phi_p(n) \equiv \frac{b_0}{p} + \frac{b_1}{p^2} + ... + \frac{b_m}{p^{m+1}} \tag{3.2}$$

where p is a prime number and the expansion of n in base p is given as $n = b_0 + b_1 p + ... + b_m p^m$, with integers $0 \leq b_j < p$.

The Halton sequence X_n in $[0, 1)^s$ is defined as

$$X_n = (\phi_{p_1}(n), \phi_{p_2}(n), \ldots, \phi_{p_s}(n)) \tag{3.3}$$

where $p_1, p_2, ..., p_s$ are pairwise co-primes. In practice, we always use the first s primes as the bases.

Simply, the Halton sequence generalizes the one dimensional Van der Corput sequence and uses prime numbers as its base.

For example, Halton Sequence with base 2 is,

$\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{4}$, $\frac{1}{8}$, $\frac{5}{8}$, $\frac{3}{8}$, $\frac{7}{8}$, $\frac{1}{16}$, $\frac{9}{16}$, $\frac{5}{16}$, $\frac{13}{16}$, $\frac{3}{16}$, $\frac{11}{16}$, $\frac{7}{16}$, $\frac{15}{16}$, ...

and Halton Sequence with base 3 is,

$\frac{1}{3}$, $\frac{2}{3}$, $\frac{1}{9}$, $\frac{4}{9}$, $\frac{7}{9}$, $\frac{2}{9}$, $\frac{5}{9}$, $\frac{8}{9}$, $\frac{1}{27}$, $\frac{10}{27}$,$\frac{19}{27}$, $\frac{4}{27}$, $\frac{13}{27}$, $\frac{22}{27}$, $\frac{7}{27}$,...

In dimension n, we create n-tuples of such sequences. For example, a two dimensional Halton sequence with base 2 and 3 is,

$(\frac{1}{2}, \frac{1}{3})$, $(\frac{1}{4}, \frac{2}{3})$, $(\frac{3}{4}, \frac{1}{9})$, $(\frac{1}{8}, \frac{4}{9})$, $(\frac{5}{8}, \frac{7}{9})$, $(\frac{3}{8}, \frac{2}{9})$, $(\frac{7}{8}, \frac{5}{9})$, $(\frac{1}{16}, \frac{8}{9})$, $(\frac{9}{16}, \frac{1}{27})$,...

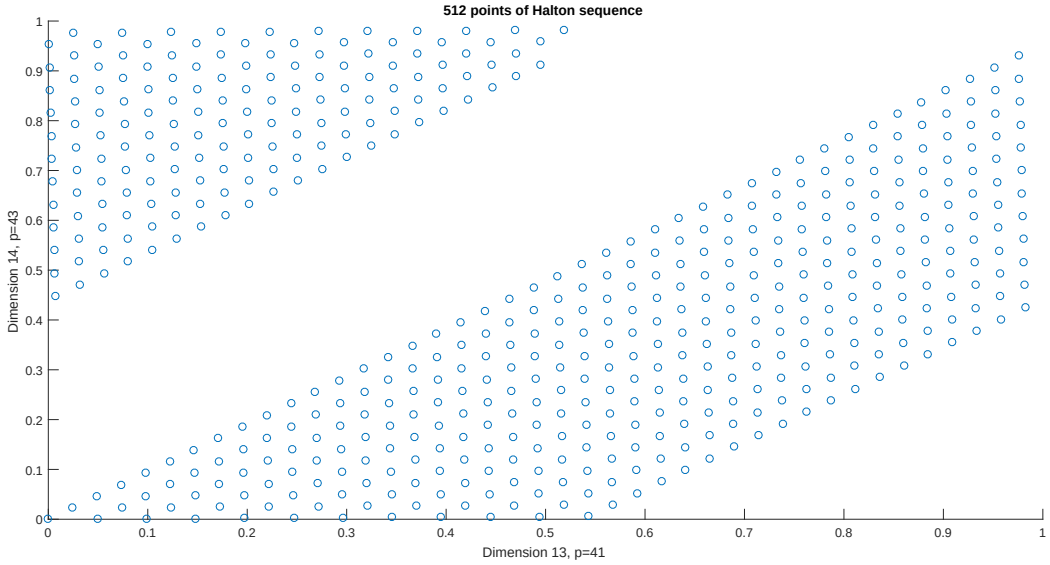

Figure 3.2: 512 points of Halton sequence for dimension 13^{th} and 14^{th}

Compared to other low-discrepancy sequences, Halton sequences are easier to implement due to the ease of implementation of the radical inverse function. However, a problem with Halton sequence comes from the correlations between the radical inverse functions for different dimensions [4], [33]. The correlations cause the Halton sequence to have poor $2-D$ projection for some pairs of dimensions. For example, the two dimensional projections of the 13^{th} and 14^{th} (Figure 3.2), 28^{th} and 29^{th} (Figure 3.3) and 39^{th} and 40^{th} (Figure 3.4) dimensions are not evenly distributed.

Another drawback of quasi-random sequences is that they are deterministic, hence we get a single estimate of the results as having many estimates of the unknown quantity and we can find mean and variance. In other words, it does not allow multiple starts of the algorithm. To address these drawbacks of quasi-Monte Carlo methods, researchers introduced randomized versions of quasi-Monte Carlo methods, that still have the good uniformity properties of low discrepancy sequences.

Therefore, in this dissertation we use optimal Halton sequence, which belongs to the family of randomized quasi-Monte Carlo methods to initialize the particles

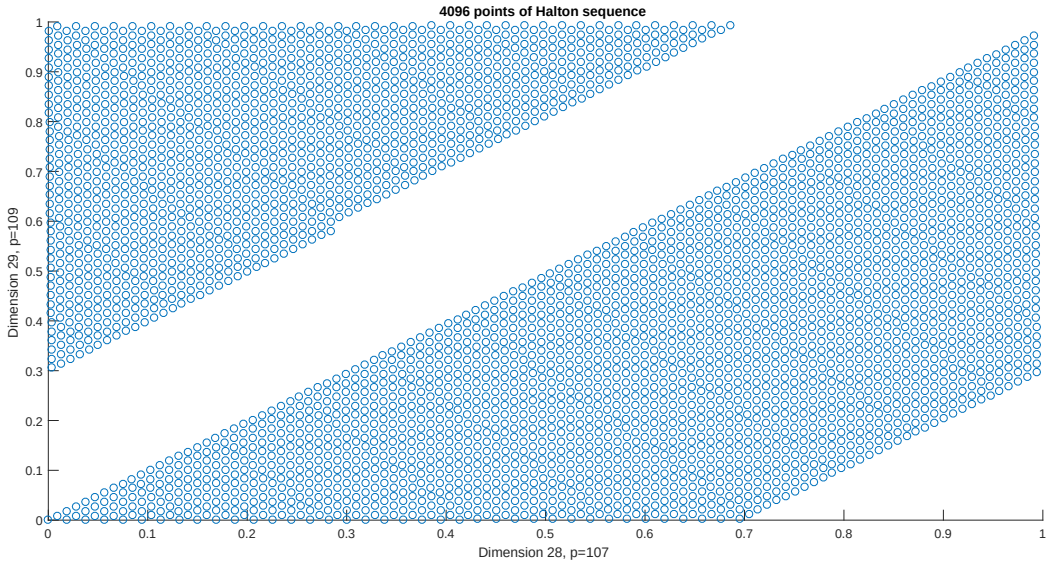

Figure 3.3: 4096 points of Halton sequence for dimension 28^{th} and 29^{th}

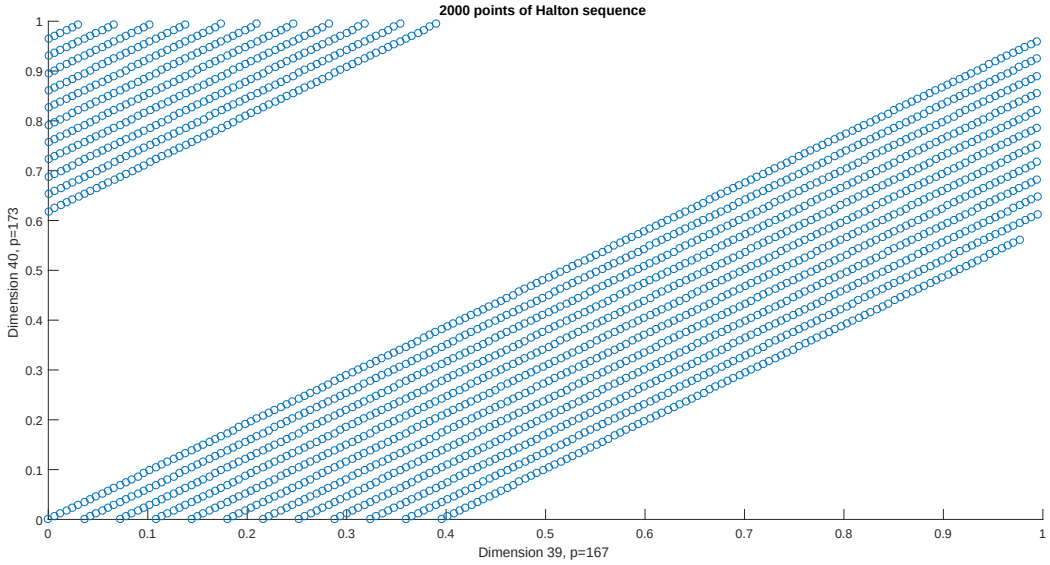

Figure 3.4: 2000 points of Halton sequence for dimension 39^{th} and 40^{th}

3.2 Randomized Quasi-Random Sequences

To address the drawbacks of quasi-random sequences, researchers have introduced Randomized quasi-random sequences. They are obtained either by, random shifting, permutation of the digits or by linear scrambling. Unlike pseudo-random numbers, there are only a few common choices for quasi-random number generation. However, by scrambling a quasi-random sequence, one can produce a family of related quasi-random sequences.

Finding one or a group of optimal quasi-random sequences within this family is an interesting problem, as such optimal quasi-random sequences can be quite useful for enhancing the performance of ordinary quasi-Monte Carlo sequences. The process of finding such optimal quasi-random sequences is called the derandomization of a randomized (scrambled) family of quasi-random sequences. In addition to providing more quasi-random sequences for quasi-Monte Carlo applications, derandomization can help us improve the accuracy of error estimation provided by randomized quasi-Monte Carlo. This is due to the fact that one can find a set of optimal sequences within a family of scrambled sequences, and use sequences within this set for error estimation. In next section, we give a detailed description for derivation of optimal Halton sequences.

3.2.1 Optimal Halton Sequence

In order to improve the quality of Halton sequence, the optimal Halton sequence is introduced by Chi and et al [4]. It can break the correlation among dimensions. Optimal Halton sequences can help us obtain good quality of Halton sequence. The linear scrambling method is defined as follows.

$$\phi_{\pi_p}(n) \equiv \frac{\pi_p(b_0)}{p} + \frac{\pi_p(b_1)}{p^2} + ... + \frac{\pi_p(b_m)}{p^{m+1}} \tag{3.4}$$

where linear congruential operator, $\phi_{\pi_p}(b_j) = wb_j(mod\ p)$ with $1 \leq w \leq p-1$ and $0 \leq j \leq m$. p refers to the base of the Halton sequence and w refers to the best choice for

the linear scrambling. According to Chi et al [4] optimal w values for linear scrambling are shown in table 3.1.

s	p	w	s	p	w	s	p	w	s	p	w
1	2	1	11	31	12	21	73	47	31	127	56
2	3	2	12	37	13	22	79	30	32	131	50
3	5	3	13	41	17	23	83	47	33	137	52
4	7	3	14	43	18	24	89	65	34	139	61
5	11	8	15	47	29	25	97	71	35	149	108
6	13	11	16	53	14	26	101	28	36	151	56
7	17	12	17	59	18	27	103	40	37	157	66
8	19	14	18	61	43	28	107	60	38	163	63
9	23	7	19	67	41	29	109	79	39	167	60
10	29	18	20	71	44	30	113	89	40	173	66

Table 3.1: Optimal values for w for the first 40 dimensions of the optimal Halton sequence

By analyzing the inner property of points in each coordinate, they find that correlations are related to the most significant bit. They permute the most significant bits of each Halton point according to the coordinate. The period of points in each coordinate is the base. Permutation of the most significant bit of each point is the same as a permutation in $\{\phi_p(1), \phi_p(2), \ldots, \phi_p(b)\}, \{\phi_p(b+1), \phi_p(b+2), \ldots, \phi_p(2b)\}, \ldots$. The advantage of this procedure is that we have the same code as the original Halton sequence. The only thing we need to do is to permute the points according to each coordinate, and output the scrambled Halton sequence. This permutation of the Halton sequence does not change its uniformity in one dimension. It just changes the position of one point [4].

Figure 3.5 shows 256 points from optimal Halton sequence in 13^{th} and 14^{th} dimension. Figure 3.6 shows 4096 points from optimal Halton sequence in 28^{th} and 29^{th} dimension followed by figure 3.7 which shows the 2000 points of optimal Halton sequence for dimension 39^{th} and 40^{th}. Comparison of figure 3.2 with 3.5, figure 3.3 with 3.6 and figure 3.4 with 3.7 , we can see that the performance of optimal Halton sequence is exceptional.

In this practice, we followed the recommendation from an early implementation[13], to skip a certain number of n instead $n = 1$, we dropped first 5000 points and start with

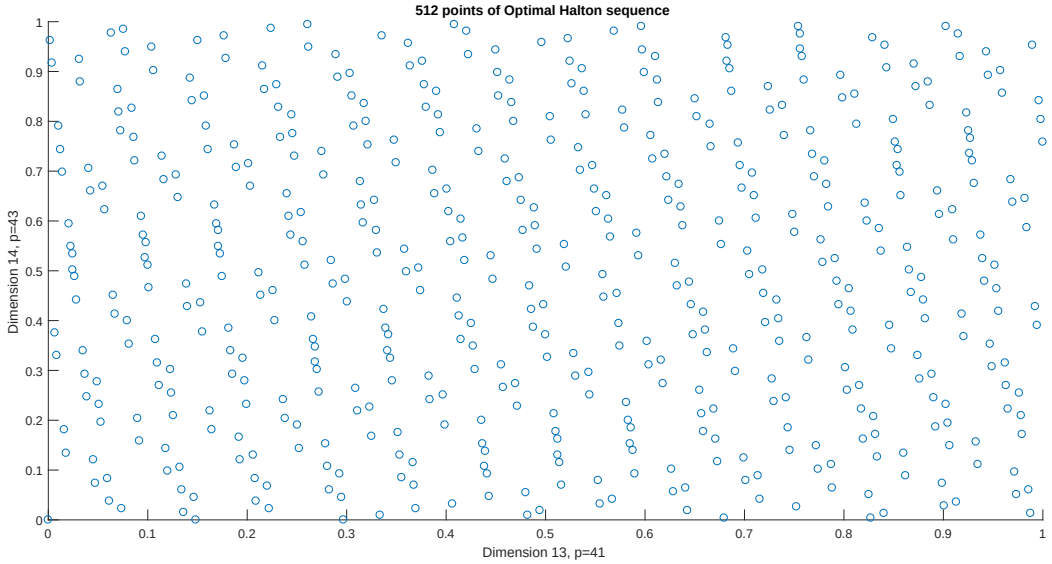

Figure 3.5: 512 points of optimal Halton sequence for dimension 13^{th} and 14^{th}

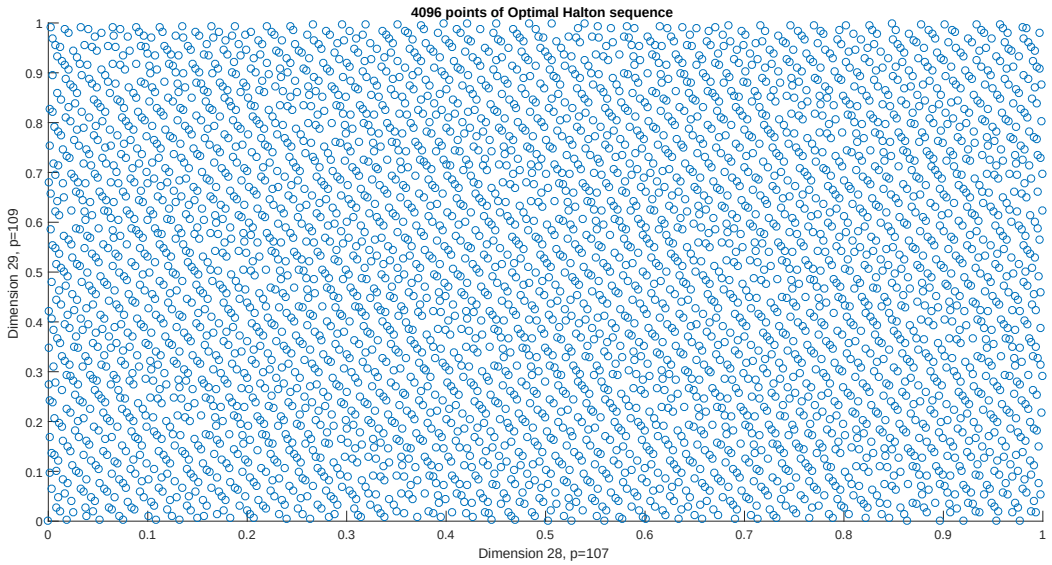

Figure 3.6: 4096 points of optimal Halton sequence for dimension 28^{th} and 29^{th}

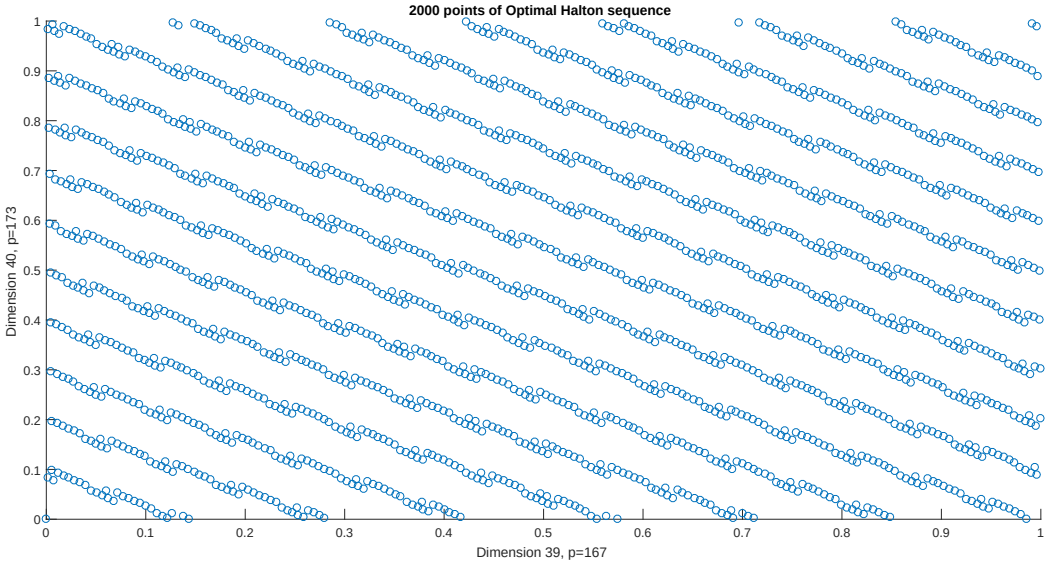

Figure 3.7: 2000 points of optimal Halton sequence for dimension 39^{th} and 40^{th}

$n = 5001$. This has no effect on asymptotic performance, but it dramatically improves practical performance when the dimension is high. It is always safe to skip a couple thousand points when we are using quasi-random sequences.

Chapter 4

Convergence Analysis

In this chapter we first give some definitions in measure theory and mathematical formulas of general probability theory. Then we introduce our theorem followed by the proof of the theorem with two lemmas.

4.1 Mathematical Preliminaries

Definition 1 *($\sigma-$ algebra) Given a set Ω , a collection $\mathcal{A}$ of subsets of Ω is a $\sigma-$ algebra iff,*

1. $\emptyset \in \mathcal{A}$
2. $A \in \mathcal{A} \Rightarrow A^c \in \mathcal{A}$
3. $A_1, A_2, ... \in \mathcal{A} \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$

Definition 2 *(measurable space) A space Ω and a $\sigma-$ algebra $\mathcal{A}$ on Ω , $(\Omega, \mathcal{A})$ is a measurable space.*

Definition 3 *(measure) Let $(\Omega, \mathcal{A})$ be a measurable space. A map $\mu : \mathcal{A} \to [0, \infty]$ is a measure iff*

1. $\mu(\emptyset) = 0$
2. *If $A_i \bigcap A_j = \emptyset$ for $\{A_n : n = 1, 2, ...\} \subset \mathcal{A}$ then $\mu(\bigcup_{i=1} A_i) = \sum_{i=1} \mu(A_i)$*

The triple $(\Omega, \mathcal{A}, \mu)$ is called a measure space.

Example: Lebesgue measure Let

- $\Omega = \Re^D$,

- $\mathcal{A} = \mathcal{B}(\Re^D)$
- $\upsilon(A) = \prod_{k=1}^{D}(a_k^R - a_k^L)$ for rectangles $A = \{x \in \Re^D : a_k^L < x_k < a_k^R, k = 1, 2, ...D\}$

Note: Borel sets $\mathcal{B}$ is the smallest $\sigma-$ algebra that contains all open sets.

Definition 4 *(Probability space) A measure space* $(\Omega, \mathcal{A}, \mathrm{P})$ *is a probability space if* $\mathrm{P}(\Omega) = 1$. *In this case* P *is called a probability measure.* $\mathrm{P}(A) = \mathrm{P}(X \in A)$ for $A \in \mathcal{A}$

4.2 Convergence Analysis of PSO

In this section we introduce our theorem and prove the convergence of PSO algorithm to the optimizer under reasonable conditions.

Theorem 4.1 *(Weerasinghe, Cao, 2017)*
Let $f(\vec{a})$ *be a continuous and bounded function from* $\Theta \subset \Re^D$ *to* $\Re$. *Assume the following two conditions,*
(A1). $\vec{a}^*$ *is the minimizer of* f *on the domain* Θ *such that* $f(\vec{a}^*) = \inf_{\vec{a} \in \Theta} f(\vec{a})$
(A2). $\vec{a}^*$ *is the unique minimizer, that means* $\inf_{\vec{a_t} \in \Theta, \|\vec{a_t} - \vec{a}^*\| \geq \eta} f(\vec{a_t}) > f(\vec{a}^*)$.
Suppose further that $r_{1,t+1}$ *and* $r_{2,t+1}$ *in* (2.2) *are chosen s.t.* $r_{1,t+1} \neq r_{1,t+1} \neq 0$ *simultaneously. Then, the particle swarm optimization algorithm converges a.s to the optimizer as* $t \to \infty$

To prove the above theorem we suggest and prove the following two lemmas.

Lemma 1 :

Under above conditions, $\{\vec{pbest}_t^{\,k}\}$ *and* $\{\vec{gbest}_t\}$ *converge to the optimizer a.s.*

Proof :

According to our notation defined in chapter 2, $\{\vec{pbest}_t^{\,k}\}$ denotes k^{th} particle's personal best position at time t. First we consider convergence of k^{th} particle's personal best value. From

here onwards for notational convenience let $\{\vec{pbest}_t^{\,k}\}$ be denoted by $\{\vec{pbest}_t\}$.
According to equation (2.4), $\{f(\vec{pbest}_t)\}$ is a decreasing sequences and $f(\vec{a}^*) > -\infty$. Thus by Monotone Convergence Theorem in real analysis (for example, Principles of mathematical analysis by Walter Rudin [44], page 55), $\lim_{t\to\infty}\{f(\vec{pbest}_t)\}$ exists and

$$\lim_{t\to\infty}\{f(\vec{pbest}_t)\} = \inf_{\vec{pbest}_t\in\Theta}\{f(\vec{pbest}_t)\} \geq \inf_{\vec{a}\in\Theta}\{f(\vec{a})\} = f(\vec{a}^*) \tag{4.1}$$

Let $\mathrm{S}_\eta = \{\vec{a} : f(\vec{a}) < f(\vec{a}^*) + \eta\}$ for any $\eta > 0$.

From update equation (2.4) it is clear that,

$$\begin{gathered} \vec{pbest}_k \notin \mathrm{S}_\eta \subseteq \bigcap_{t=0}^{k} \vec{pbest}_t \notin \mathrm{S}_\eta \\ \text{Thus by probability theory,} \quad \mathrm{P}[\vec{pbest}_k \notin \mathrm{S}_\eta] \leq \mathrm{P}[\bigcap_{t=0}^{k} \vec{pbest}_t \notin \mathrm{S}_\eta] \end{gathered} \tag{4.2}$$

and also from update equation (2.4) we have that,

$$\begin{gathered} \vec{pbest}_0 \in \mathrm{S}_\eta \subseteq \vec{pbest}_t \in \mathrm{S}_\eta \ \text{ for all } t \geq 1. \\ \text{Thus,} \quad \mathrm{P}[\vec{pbest}_0 \in \mathrm{S}_\eta] \leq \mathrm{P}[\vec{pbest}_t \in \mathrm{S}_\eta] \ \text{ for all } t \geq 1. \end{gathered} \tag{4.3}$$

From probability theory ,

$$\mathrm{P}[\vec{pbest}_k \in \mathrm{S}_\eta] = 1 - \mathrm{P}[\vec{pbest}_k \notin \mathrm{S}_\eta] \tag{4.4}$$

From equation (4.2), we have

$$
\begin{aligned}
&\mathrm{P}(\vec{pbest}_k \in \mathrm{S}_\eta) \\
&\geq 1 - \mathrm{P}(\bigcap_{t=0}^{k} \vec{pbest}_t \notin \mathrm{S}_\eta) \\
&= 1 - \mathrm{P}(\vec{pbest}_0 \notin \mathrm{S}_\eta) \cdot \prod_{j=1}^{k} \mathrm{P}[\vec{pbest}_j \notin \mathrm{S}_\eta | \bigcap_{t=0}^{j-1} \vec{pbest}_t \notin \mathrm{S}_\eta] \qquad (4.5) \\
&= 1 - (1 - \mathrm{P}[\vec{pbest}_0 \in \mathrm{S}_\eta]) \cdot \prod_{j=1}^{k} (1 - \mathrm{P}[\vec{pbest}_j \in \mathrm{S}_\eta | \bigcap_{t=0}^{j-1} \vec{pbest}_t \notin \mathrm{S}_\eta]) \qquad (4.6)
\end{aligned}
$$

Equation (4.5) follows from conditional probability formula.

From probability theory, equation for computing a probability by conditioning on a random variable is given by, $P[A] = \int P[A|W = w]d\mu(w)$ where μ denotes the distribution of W, i.e., the probability measure defined by $\mu(H) = P[W \in H]$

Using the above equation we have,

$$P[A|B] = \int P[A|B \cap (W = w)]d\mu(w) \qquad (4.7)$$

where μ denotes the conditional distribution of W given B, i.e., the probability measure defined by $\mu(H) = P[W \in H|B]$

Let $W_m = w_m$ denote

$[(\vec{a_0}, \vec{a_1}, ..., \vec{a_m}), (\vec{v_0}, \vec{v_1}, ..., \vec{v_m}), (\vec{pbest}_0, \vec{pbest}_1, ..., \vec{pbest}_m)]$

$= [(\vec{\vec{a_0}}, \vec{\vec{a_1}}, ..., \vec{\vec{a_m}}), (\vec{\vec{v_0}}, \vec{\vec{v_1}}, \cdots, \vec{\vec{v_m}}), (\vec{\widetilde{pbest}}_0, \vec{\widetilde{pbest}}_1, ..., \vec{\widetilde{pbest}}_m)]$

Using above equation (4.7), for any $0 < m \leq k$, we have

$$\begin{aligned} &\mathrm{P}[\vec{pbest}_m \in \mathrm{S}_\eta | \bigcap_{t=0}^{m-1} \vec{pbest}_t \notin \mathrm{S}_\eta] \\ &= \int \mathrm{P}[\vec{pbest}_m \in \mathrm{S}_\eta | \bigcap_{t=0}^{m-1} \vec{pbest}_t \notin \mathrm{S}_\eta \bigcap (W_{m-1} = w_{m-1})] \cdot d\mu \end{aligned} \tag{4.8}$$

Here μ denotes the conditional distribution of

$[(\vec{a_0}, \vec{a_1}, ..., \vec{a_{m-1}}), (\vec{v_0}, \vec{v_1}, ..., \vec{v_{m-1}}), (\vec{pbest}_0, \vec{pbest}_1, ..., \vec{pbest}_{m-1})]$ given $[(\vec{pbest}_0 \notin \mathrm{S}_\eta) \cap (\vec{pbest}_1 \notin \mathrm{S}_\eta) \cdots (\vec{pbest}_{m-1} \notin \mathrm{S}_\eta)]$

i.e., the probability measure defined by

$\mu(H) = P[((\vec{a_0}, \vec{a_1}, ..., \vec{a_{m-1}}), (\vec{v_0}, \vec{v_1}, ..., \vec{v_{m-1}}), (\vec{pbest}_0, \vec{pbest}_1, ..., \vec{pbest}_{m-1})) \in H | ((\vec{pbest}_0 \notin \mathrm{S}_\eta) \cap (\vec{pbest}_1 \notin \mathrm{S}_\eta) \cdots (\vec{pbest}_{m-1} \notin \mathrm{S}_\eta))]$

Let,

$$[\bigcap_{t=0}^{m-1} \vec{pbest}_t \notin \mathrm{S}_\eta \cap (W_{m-1} = w_{m-1})] = (W_{m-1} = \tilde{w}_{m-1}) \tag{4.9}$$

Here $\tilde{w}_{m-1}$ are the values for which $\vec{pbest}_0 \notin \mathrm{S}_\eta$, $\vec{pbest}_1 \notin \mathrm{S}_\eta, \cdots \vec{pbest}_{m-1} \notin \mathrm{S}_\eta$

Equation (4.8) can be simplified to,

$$\begin{aligned}
&\mathrm{P}[\vec{pbest}_m \in \mathrm{S}_\eta | \bigcap_{t=0}^{m-1} \vec{pbest}_t \notin \mathrm{S}_\eta] \\
&= \int \mathrm{P}[\vec{pbest}_m \in \mathrm{S}_\eta | \bigcap_{t=0}^{m-1} \vec{pbest}_t \notin \mathrm{S}_\eta \bigcap (W_{m-1} = w_{m-1})] \cdot d\mu \\
&= \int \mathrm{P}[\vec{pbest}_m \in \mathrm{S}_\eta | W_{m-1} = \tilde{w}_{m-1}] \cdot d\mu \qquad (4.10) \\
&= \int \mathrm{P}[\vec{pbest}_m \in \mathrm{S}_\eta | (\vec{a}_{m-1} = \widetilde{\vec{a_{m-1}}}, \vec{v}_{m-1} = \widetilde{\vec{v_{m-1}}}, \vec{pbest}_{m-1} = \widetilde{\vec{pbest}_{m-1}})] \cdot d\mu \qquad (4.11) \\
&= \int \mathrm{P}[\vec{a}_m \in \mathrm{S}_\eta | (\vec{a}_{m-1} = \widetilde{\vec{a_{m-1}}}, \vec{v}_{m-1} = \widetilde{\vec{v_{m-1}}}, \vec{pbest}_{m-1} = \widetilde{\vec{pbest}_{m-1}})] \cdot d\mu \qquad (4.12)
\end{aligned}$$

Equation (4.11) is obtained because according equation (2.4) selection of $\vec{pbest}_m$ depends only on $\vec{pbest}_{m-1}$ among the given values in W_{m-1}. The reason for last equality (equation (4.12)) follows again from equation (2.4). It is given that $\vec{pbest}_{m-1} \notin \mathrm{S}_\eta$. Thus the event that $\vec{pbest}_m \in \mathrm{S}_\eta$ is same as the event $\vec{a}_m \in \mathrm{S}_\eta$.

$$\mathrm{P}[\vec{a}_m \in \mathrm{S}_\eta | (\vec{a}_{m-1} = \widetilde{\vec{a_{m-1}}}, \vec{v}_{m-1} = \widetilde{\vec{v_{m-1}}}, \vec{pbest}_{m-1} = \widetilde{\vec{pbest}_{m-1}})] = \int_{\mathrm{S}_\eta} p(x) \upsilon(dx)$$

in the above equation, $p(x)$ is the probability density function of $(\vec{a}_m \in \mathrm{S}_\eta | (\vec{a}_{m-1}, \vec{v}_{m-1}, \vec{pbest}_{m-1}))$ and $\upsilon(\cdot)$ denotes Lebesgue measure.

$(\vec{a}_m \in \mathrm{S}_\eta | (\vec{a}_{m-1}, \vec{v}_{m-1}, \vec{pbest}_{m-1}))$ follows uniform distribution according to equations (2.2) and (2.3). Thus, $p(x) = \frac{1}{\prod_{j=1}^{D}(a^j_{max} - a^j_{min})}$. Here a^j_{max} and a^j_{min} are maximum and minimum values in the search space in j^{th} direction. Thus above equation can be simplified

to,

$$\begin{aligned}
&\mathrm{P}[\vec{a}_m \in \mathrm{S}_\eta | (\vec{a}_{m-1} = \widetilde{\vec{a_{m-1}}}, \vec{v}_{m-1} = \widetilde{\vec{v_{m-1}}}, \vec{pbest}_{m-1} = \widetilde{\vec{pbest}_{m-1}})] \\
&= p(x) \int_{\mathrm{S}_\eta} \upsilon(dx) \\
&= \frac{\upsilon(\mathrm{S}_\eta)}{\upsilon(\Theta)} = \beta_\eta
\end{aligned} \tag{4.13}$$

Recall that Θ denotes search space.

According to the definition of S_η , $a^* \in \mathrm{S}_\eta$ and since f is continuous S_η is open. S_η is non-empty open set. Hence by basic property in Lebesgue measure (For example [31], page 43, Property, 2.1.3) that every non- empty open set has strictly positive lebesgue measure, we have that $\upsilon(\mathrm{S}_\eta) > 0$.

Then $0 < \beta_\eta \leq 1$ for each $\eta \geq 0$.

Equation (4.12) can be written as,

$$\mathrm{P}[\vec{pbest}_m \in \mathrm{S}_\eta | \bigcap_{t=0}^{m-1} \vec{pbest}_t \notin \mathrm{S}_\eta] = \int \beta_\eta d\mu = \beta_\eta \int d\mu = \beta_\eta \int f_X dm = \beta_\eta \cdot 1 \tag{4.14}$$

where f_X denotes the probability density function. The last equality in equation (4.14) is due to the fact that, $\int_{-\infty}^{\infty} f_X(x)dx = 1$ for a continuous random variable x.

$$1 - \mathrm{P}[\vec{pbest}_m \in \mathrm{S}_\eta | \bigcap_{t=0}^{m-1} \vec{pbest}_t \notin \mathrm{S}_\eta] = 1 - \beta_\eta \tag{4.15}$$

At initial stage, $\vec{pbest}_0 = \vec{a}_0$. Thus, $\mathrm{P}(\vec{pbest}_0 \in \mathrm{S}_\eta) = \mathrm{P}(\vec{a}_0 \in \mathrm{S}_\eta)$. Initialization of $\vec{a}_0$ follows uniform distribution. Thus $\mathrm{P}(\vec{pbest}_0 \in \mathrm{S}_\eta) = \int_{\mathrm{S}_\eta} p(x)\upsilon(dx) = \frac{\upsilon(\mathrm{S}_\eta)}{\upsilon(\Theta)} = \beta_\eta$.

Let $\beta = min\{\beta_\eta : \eta > 0\}$.

Now equation (4.6) simplifies to,

$$\mathrm{P}[p\vec{best}_k \in \mathrm{S}_\eta] \geq 1 - \prod_{t=0}^{k}(1-\beta)$$

$$\mathrm{P}[p\vec{best}_k \in \mathrm{S}_\eta] \geq 1 - (1-\beta)^{k+1}$$

$$\lim_{k\to\infty} \mathrm{P}[p\vec{best}_k \in \mathrm{S}_\eta] \geq 1 - \lim_{k\to\infty}(1-\beta)^{k+1} \tag{4.16}$$

Since $0 < \beta \leq 1$, we have, $0 \leq 1-\beta < 1$. Thus, $\lim_{k\to\infty}(1-\beta)^{k+1} = 0$.

$$\lim_{k\to\infty} \mathrm{P}(p\vec{best}_k \in \mathrm{S}_\eta) \geq 1 \tag{4.17}$$

$$\text{also} \quad \mathrm{P}(p\vec{best}_k \in \mathrm{S}_\eta) \leq 1 \tag{4.18}$$

$$\text{Thus,} \quad \lim_{k\to\infty} \mathrm{P}(p\vec{best}_k \in \mathrm{S}_\eta) = 1 \tag{4.19}$$

Then there exists $\hat{t}$ such that for all $t > \hat{t}$,

$$f(p\vec{best}_t) < f(\vec{a}^*) + \eta \quad \text{with probability 1.}$$

$$\lim_{t\to\infty} f(p\vec{best}_t) \leq f(\vec{a}^*) + \eta$$

$$\text{From equation (4.1),} \quad f(\vec{a}^*) \leq \lim_{t\to\infty} f(p\vec{best}_t) \leq f(\vec{a}^*) + \eta$$

$$0 \leq \lim_{t\to\infty} f(p\vec{best}_t) - f(\vec{a}^*) \leq \eta$$

$$\text{Thus,} \quad |\lim_{t\to\infty} f(p\vec{best}_t) - f(\vec{a}^*)| \leq \eta$$

$$\begin{gathered}\text{Thus,} \quad \lim_{t\to\infty} f(p\vec{best}_t) = f(\vec{a}^*) \\ f(p\vec{best}_t) \ \text{ converges to } \ f(\vec{a}^*) \ \text{ in probability.}\end{gathered} \tag{4.20}$$

Then by standard result in probability theory (e.g Probability Theory by Laha and Rohatgi, proposition 1.3.4 [27]) there exists a subsequence of $\{f(p\vec{best}_t)\}$ which converges almost surely to $f(\vec{a}^*)$. We proved that the convergence of $\{f(p\vec{best}_t)\}$ to some limit and the fact that the subsequence has to converge to the same limit as the full sequence , we have $\{f(p\vec{best}_t)\}$ converges to $f(\vec{a}^*)$ almost surely.

We want to show that $\{p\vec{best}_t\}$ converge to $\vec{a}^*$ with probability 1.

$\{p\vec{best}_t\}$ is bounded. Thus there exists t_{n_k} such that $\{p\vec{best}_{t_{n_k}}\}$ converges to $\vec{b}$ for some $\vec{b}$. Consider

$$f(\vec{b}) = f(\lim_{k\to\infty} p\vec{best}_{t_{n_k}}) = \lim_{k\to\infty} f(p\vec{best}_{t_{n_k}}) = f(\vec{a}^*) \tag{4.21}$$

By uniqueness $\vec{b} = \vec{a}^*$

This is true for any sub-sub sequence. Thus, $\{p\vec{best}_t\}$ converge to $\vec{a}^*$ almost surely.

By the similar argument, we can show $\{g\vec{best}_t\}$ converge to $\vec{a}^*$ almost surely.

This completes the proof of lemma 1.

Lemma 2

If $A_{t+1} = \begin{bmatrix} (1 - c_1 r_{1,t+1} - c_2 r_{2,t+1}) & \omega \\ -c_1 r_{1,t+1} - c_2 r_{2,t+1} & \omega \end{bmatrix}$, *Then* $\rho(A_{t+1}) < 1$.

here $r_{1,t+1}$ *and* $r_{2,t+1}$ *are uniformly distributed random numbers between* 0 *and* 1, *and* $r_{1,t+1} \neq r_{2,t+1} \neq 0$ *simultaneously,* w, c_1 *and* c_2 *are positive constants and* $w < 1$,

$\rho(.)$ *is the spectral radius of the matrix.*

Proof:

The eigen values of the matrix can found by setting, $det|A - \lambda I| = 0$. Thus,

$$det|A - \lambda I| = \begin{vmatrix} 1 - c_1 r_{1,t+1} - c_2 r_{2,t+1} - \lambda & \omega \\ -c_1 r_{1,t+1} - c_2 r_{2,t+1} & \omega - \lambda \end{vmatrix}$$

$$\lambda^2 - (1 + \omega - c_1 r_{1,t+1} - c_2 r_{2,t+1})\lambda + \omega = 0$$

$$\lambda = \frac{1+\omega-c_1 r_{1,t+1}-c_2 r_{2,t+1} \pm \sqrt{(c_1 r_{1,t+1}+c_2 r_{2,t+1}-1-\omega)^2 - 4\omega}}{2}$$

Case I : $(c_1 r_{1,t+1} + c_2 r_{2,t+1} - 1 - \omega)^2 - 4\omega = 0$

$$\lambda = \frac{1 + \omega - c_1 r_{1,t+1} - c_2 r_{2,t+1}}{2}$$
$$= \pm\sqrt{\omega}$$
$$|\lambda| < 1$$

Case II : $(c_1 r_{1,t+1} + c_2 r_{2,t+1} - 1 - \omega)^2 - 4\omega > 0$

$$\lambda = \tfrac{1+\omega-c_1 r_{1,t+1}-c_2 r_{2,t+1}}{2} \pm \tfrac{\sqrt{(c_1 r_{1,t+1}+c_2 r_{2,t+1}-1-\omega)^2-4\omega}}{2}$$

assume $|\lambda| \geq 1$

then,

$$\frac{1 + \omega - c_1 r_{1,t+1} - c_2 r_{2,t+1} \pm \sqrt{(c_1 r_{1,t+1} + c_2 r_{2,t+1} - 1 - \omega)^2 - 4\omega}}{2} \geq 1$$
$$1 + \omega - c_1 r_{1,t+1} - c_2 r_{2,t+1} \pm \sqrt{(c_1 r_{1,t+1} + c_2 r_{2,t+1} - 1 - \omega)^2 - 4\omega} \geq 2$$
$$\pm \sqrt{(c_1 r_{1,t+1} + c_2 r_{2,t+1} - 1 - \omega)^2 - 4\omega} \geq 1 - \omega + c_1 r_{1,t+1} + c_2 r_{2,t+1} > 0$$
$$(c_1 r_{1,t+1} + c_2 r_{2,t+1} - 1 - \omega)^2 - 4\omega \geq (1 - \omega + c_1 r_{1,t+1} + c_2 r_{2,t+1})^2$$
$$- 4(c_1 r_{1,t+1} + c_2 r_{2,t+1}) \geq 0$$

since $r_{1,t+1} \neq r_{2,t+1} \neq 0$ simultaneously, $(c_1 r_{1,t+1} + c_2 r_{2,t+1}) > 0$. This gives a contradiction.

Thus $|\lambda| < 1$

Case III : $(c_1 r_{1,t+1} + c_2 r_{2,t+1} - 1 - \omega)^2 - 4\omega < 0$

$$\lambda = \frac{1+\omega-c_1r_{1,t+1}-c_2r_{2,t+1}}{2} \pm \frac{\sqrt{4\omega-(c_1r_{1,t+1}+c_2r_{2,t+1}-1-\omega)^2}}{2}i$$

$$|\lambda|^2 = \frac{(1+\omega-c_1r_{1,t+1}-c_2r_{2,t+1})^2+4\omega-(c_1r_{1,t+1}+c_2r_{2,t+1}-1-\omega)^2}{4}$$

$$= \omega$$

$$|\lambda| = \sqrt{\omega} < 1$$

In all three cases we have $|\lambda| < 1$. thus $\rho(A_{t+1}) < 1$.

This completes the proof of lemma 2.

Proof of Theorem:

Let $\vec{a^*}$ be the minimizer of the objective function and let $\vec{e_t} = \vec{a_t} - \vec{a^*}$. From lemma 1, $\lim_{t\to\infty} p\vec{best}_t^k = \vec{a^*}$ for each $k = 1, 2, ...M$ a.s and
$\lim_{t\to\infty} g\vec{best}_t = \vec{a^*}$ a.s

Consider (2.2) ,

$$\vec{v}_{t+1} = w\vec{v_t} + c_1r_{1,t+1}(\vec{a^*}-\vec{a_t}) + c_2r_{2,t+1}(\vec{a^*}-\vec{a_t}) + c_1r_{1,t+1}(p\vec{best}_t-\vec{a^*}) + c_2r_{2,t+1}(g\vec{best}_t-\vec{a^*})$$

$$\vec{v}_{t+1} = w\vec{v_t} - (c_1r_{1,t+1}+c_2r_{2,t+1})\vec{e_t} + c_1r_{1,t+1}(p\vec{best}_t-\vec{a^*}) + c_2r_{2,t+1}(g\vec{best}_t-\vec{a^*}). \tag{4.22}$$

From (2.3), we have

$$\vec{e}_{t+1} = \vec{e_t} + \vec{v}_{t+1}. \tag{4.23}$$

Putting (4.22) and (4.23) together we have the system,

$$\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{e}_{t+1} \\ \vec{v}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -c_1 r_{1,t+1} - c_2 r_{2,t+1} & \omega \end{bmatrix} \begin{bmatrix} \vec{e}_t \\ \vec{v}_t \end{bmatrix} + \begin{bmatrix} 0 \\ c_1 r_{1,t+1}(p\vec{best}_t - \vec{a^*}) + c_2 r_{2,t+1}(g\vec{best}_t - \vec{a^*}) \end{bmatrix}$$

$$\begin{bmatrix} \vec{e}_{t+1} \\ \vec{v}_{t+1} \end{bmatrix} = \begin{bmatrix} (1 - c_1 r_{1,t+1} - c_2 r_{2,t+1}) & \omega \\ -c_1 r_{1,t+1} - c_2 r_{2,t+1} & \omega \end{bmatrix} \begin{bmatrix} \vec{e}_t \\ \vec{v}_t \end{bmatrix} + \begin{bmatrix} c_1 r_{1,t+1}(p\vec{best}_t - \vec{a^*}) + c_2 r_{2,t+1}(g\vec{best}_t - \vec{a^*}) \\ c_1 r_{1,t+1}(p\vec{best}_t - \vec{a^*}) + c_2 r_{2,t+1}(g\vec{best}_t - \vec{a^*}) \end{bmatrix}$$

we have, $X_{t+1} = A_{t+1} X_t + b_t$.

$$\| \vec{X}_{t+1} \| \leq \| A_{t+1} \| \| \vec{X}_t \| + \| \vec{b}_t \| \tag{4.24}$$

Let $\| A_{t+1} \| = \rho(A_{t+1})$. Denote $\rho = max\{\rho(A_{t+1})\}$. By lemma 2, $\rho < 1$.

$$\begin{aligned} \| \vec{X}_{t+1} \| &\leq \rho \| \vec{X}_t \| + \| \vec{b}_t \| \\ &\leq \rho(\rho \| \vec{X}_{t-1} \| + \| \vec{b}_{t-1} \|) + \| \vec{b}_t \| \\ &= \rho^2 \| \vec{X}_{t-1} \| + \rho \| \vec{b}_{t-1} \| + \| \vec{b}_t \| \\ &\leq \rho^3 \| \vec{X}_{t-2} \| + \rho^2 \| \vec{b}_{t-2} \| + \rho \| \vec{b}_{t-1} \| + \| \vec{b}_t \| \\ &\vdots \\ &\leq \rho^{t+1} \| \vec{X}_0 \| + \sum_{i=0}^{t} \rho^{t-i} \| \vec{b}_i \| \end{aligned}$$

Let $\varepsilon > 0$ be given. By lemma 1, $\lim_{t\to\infty} p\vec{best}_t = \vec{a^*}$ and $\lim_{t\to\infty} g\vec{best}_t = \vec{a^*}$

Then there exists N_0 s.t. $\| \vec{b}_t \| \leq \varepsilon$ for all $t \geq N_0$

$$\| \vec{X}_{t+1} \| \leq \rho^{t+1} \| \vec{X}_0 \| + \sum_{i=0}^{N_0} \rho^{t-i} \| \vec{b}_i \| + \sum_{i=N_0+1}^{t} \rho^{t-i} \| \vec{b}_i \|$$
$$\leq \rho^{t+1} \| \vec{X}_0 \| + \sum_{i=0}^{N_0} \rho^{t-i} \| \vec{b}_i \| + \varepsilon \sum_{i=N_0+1}^{t} \rho^{t-i}$$

Let $B = sup \| \vec{b}_t \|$. Since $\{\vec{b}_t\}$ is convergent, $B < \infty$. Thus we have,

$$\begin{aligned}
\| \vec{X}_{t+1} \| &\leq \rho^{t+1} \| \vec{X}_0 \| + B \sum_{i=0}^{N_0} \rho^{t-i} + \varepsilon \sum_{i=N_0+1}^{t} \rho^{t-i} \\
&\leq \rho^{t+1} \| \vec{X}_0 \| + B \frac{\rho^{t-N_0} - \rho^{t+1}}{1-\rho} + \varepsilon \frac{1-\rho^{t-N_0}}{1-\rho} \\
&\leq \rho^{t+1} \| \vec{X}_0 \| + B \frac{\rho^{t-N_0} - \rho^{t+1}}{1-\rho} + \varepsilon \frac{1}{1-\rho} \\
&= \rho^{t+1} \| \vec{X}_0 \| + B \frac{\rho^t}{(1-\rho)} \frac{(1-\rho^{N_0+1})}{\rho^{N_0}} + \varepsilon \frac{1}{1-\rho}
\end{aligned}$$

There exists $N_1 > N_0$ s.t. $\rho^t \frac{(1-\rho^{N_0+1})}{\rho^{N_0}} < \frac{\varepsilon}{B}$ for all $t > N_1$.

There exists N_2 s.t. $\rho^{t+1} < \frac{\varepsilon}{(1-\rho)\|\vec{X}_0\|}$ for all $t > N_2$.

Let $N_3 = max\{N_1, N_2\}$

Then for $t > N_3$,

$$\| \vec{X}_{t+1} \| < \| \vec{X}_0 \| \frac{\varepsilon}{(1-\rho) \| \vec{X}_0 \|} + \frac{B}{(1-\rho)} \frac{\varepsilon}{B} + \frac{\varepsilon}{1-\rho}$$
$$\| \vec{X}_{t+1} \| < \frac{3}{1-\rho} \varepsilon$$

Thus, $\lim_{t\to\infty} \| \vec{X}_{t+1} \| = 0$. This implies $\lim_{t\to\infty} \| \vec{e}_{t+1} \| = 0$ and $\| \vec{v}_{t+1} \| = 0$

Hence, $\lim_{t\to\infty} \vec{e}_{t+1} = 0$ and $\lim_{t\to\infty} \vec{v}_{t+1} = 0$

Thus,

$\lim_{t\to\infty} \vec{a}_t = \vec{a}^*, a.s.$

This completes the proof.

Chapter 5

Numerical Results

5.1 Numerical Results for Two Dimensional Problems

For illustrative purposes, we have chosen two dimensional functions to show how particles evolve in the search space through time. Among common functions which are used for testing optimization algorithms, we have chosen six functions which differ in their significant physical properties and shapes. The first function is the Bohachevsky function (f_1), which is a bowl shaped function. The second is the Ackley function (f_2), which is a multimodal function and widely used for testing optimization algorithms. In its 2D form as shown in Figure 5.4, it has a large hole at the center and has many local minima, thus many algorithms tend to become trapped at local minima. The Matyas function (f_3), has a flat surface. The flatness of the function does not give the algorithm any information to direct the search process towards the minima. The fourth is the Easom function (f_4). In f_4 , the area that contains the global minima are very small, when compared to the whole search space. The function value rapidly approaches zero away from (π, π), which is the minimizer. Fifth is the Schaffer function (f_5), whose global minimum is located very close to the local minima. Finally, tripod function (f_6), which is a semi-continuous function [6].

Particles are initialized using the optimal Halton sequence with base 2,3 which is a randomized quasi-random sequence. Thus it guarantees the stochastic properties of the PSO algorithm [53]. To implement the PSO algorithm, the values of the parameters $\{\omega, c_1, c_2\}$ are needed. According to Ming Jiang et al [22], the suggested parameter tuple in the literature is $\omega = 0.729, c_1 = 2.8\omega, c_2 = 1.38\omega$, or $\omega = 0.729, c_1 = c_2 = 1.49$, or $\omega = 0.6, c_1 = c_2 = 1.7$. Among these three, $\omega = 0.729, c_1 = c_2 = 1.49$ is the most widely used parameter tuple, so we choose that to perform our numerical results. We evaluate values for different numbers

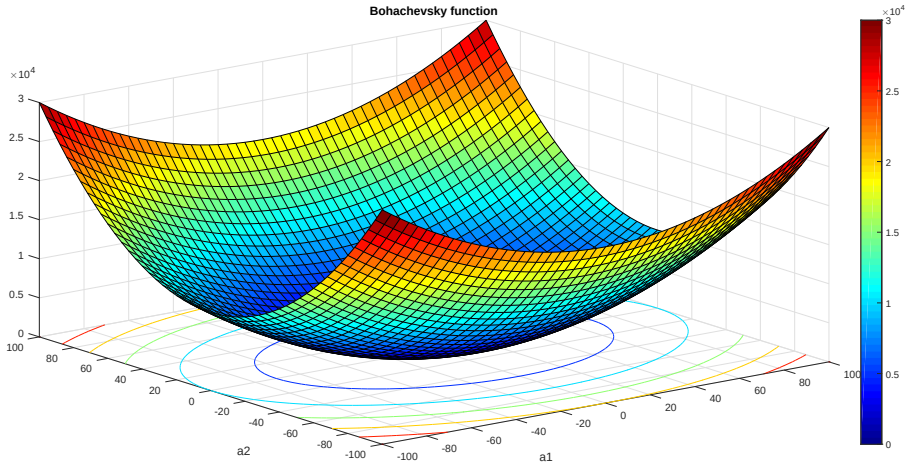

Figure 5.1: Bohachevsky Function

of particles (M) and iteration numbers (N) at random evaluations. Stopping criteria for the algorithm is taken as the maximum number of iterations.

5.1.1 Bohachevsky Function

The Bohachevsky function for dimension 2 is denoted by,

$$f_1 = a_1^2 + 2a_2^2 - 0.3cos(3\pi a_1) - 0.4cos(4\pi a_2) + 0.7 \tag{5.1}$$

It is a bowl shaped function with a unique global minimum $f(a^*) = 0$ at $a^* = (0, 0)$. The graph of the two dimensional function is shown in Figure 5.1. The function is evaluated in $[-100, 100]^2$ cube and the numerical results are shown in Table 5.1. Figure 5.2 shows the evolution of particles for different number of iterations and figure 5.3 shows a closer look at particle's position at final iteration.

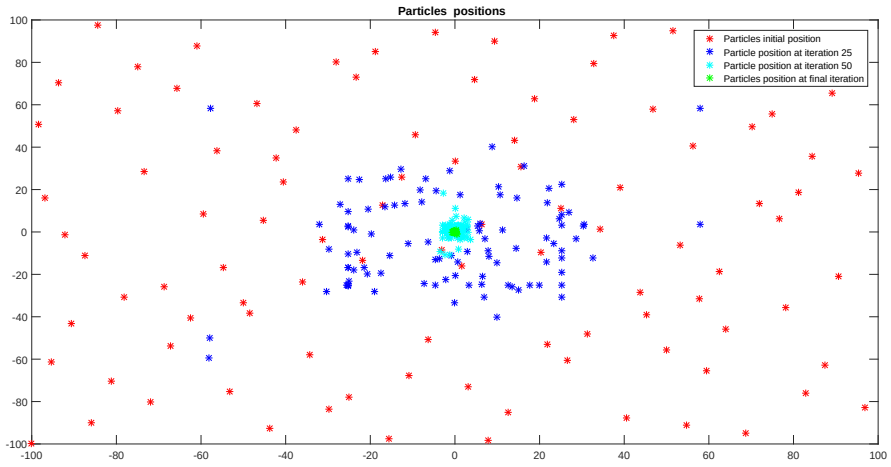

Figure 5.2: Evolution of Particles for Bohachevsky function when there are 100 particles for 75 iterations

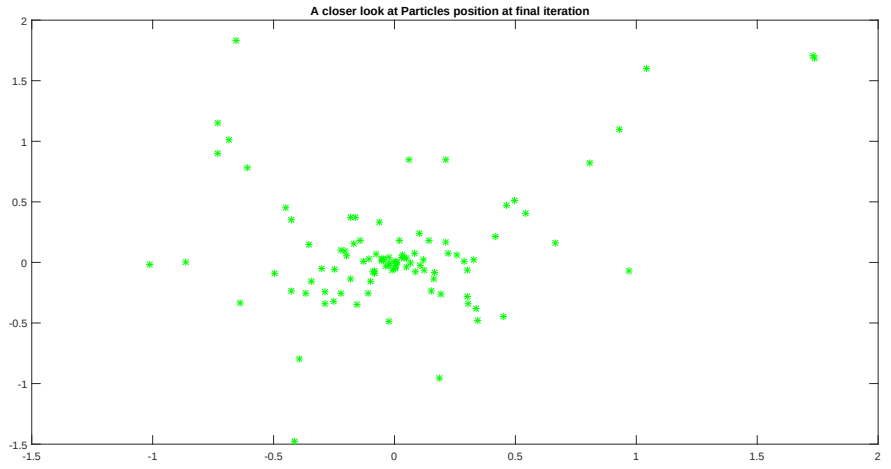

Figure 5.3: A closer look of particles position at final iteration for Bohachevsky function for 75 particles

M	N	Optimum position	Optimum value	error
50	50	(0.0019,-0.0009)	0.0001	0.0001
	75	$(0.8486e^{-5}, -0.1157e^{-5})$	$0.0001e^{-5}$	$0.0001e^{-5}$
75	50	$(0.2495e^{-4}, -0.3592e^{-4})$	$0.0005e^{-4}$	$0.0005e^{-4}$
	75	$(-0.2062e^{-6}, -0.4150e^{-6})$	$0.0000e^{-6}$	$0.0000e^{-6}$
100	50	$(0.5351e^{-3}, 0.0182e^{-3})$	$0.0041e^{-3}$	$0.0041e^{-3}$
	75	$(0, 0)$	0	0

Table 5.1: Results for Bohachevsky Function

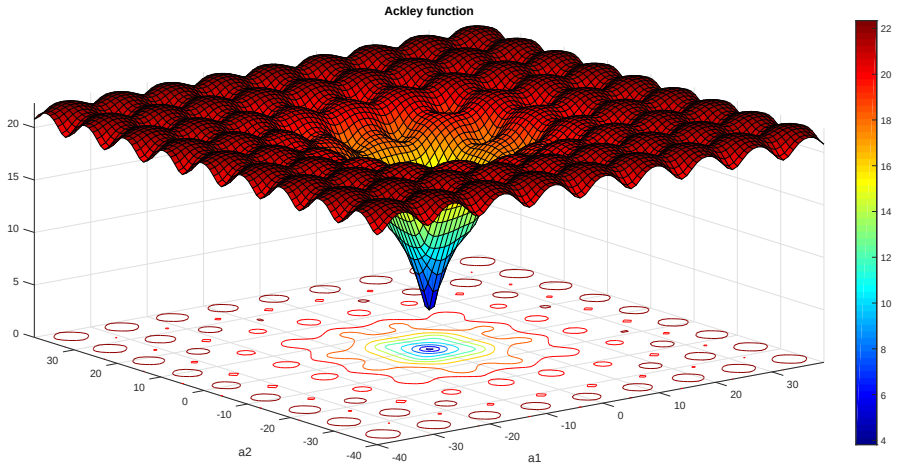

Figure 5.4: Ackley Function

5.1.2 Ackley Function

The Ackley function for dimension 2 is denoted by,

$$f_2 = 20 + e - 20\exp\left(-0.2\sqrt{\frac{a_1^2 + a_2^2}{2}}\right) - \exp\left(\frac{\cos(2\pi a_1) + \cos(2\pi a_2)}{2}\right) \tag{5.2}$$

It has a unique global minimum $f(a^*) = 0$ at $a^* = (0, 0)$ and several local minima. The graph of the two dimensional function is shown in Figure 5.4. The function is evaluated on a $[-35, 35]^2$ cube and the numerical results are shown in Table 5.2 .

M	N	Optimum position	Optimum value	error
50	50	$(-0.0755e^{-3}, 0.2394e^{-3})$	$0.7116e^{-3}$	$0.7116e^{-3}$
	75	$(0.0524e^{-3}, -0.0612e^{-3})$	$0.2281e^{-3}$	$0.2281e^{-3}$
75	50	(-0.0007,-0.0017)	0.0055	0.0055
	75	$(-0.0416e^{-3}, 0.0108e^{-3})$	$0.1218e^{-3}$	$0.1218e^{-3}$
100	50	$(0.0412e^{-3}, -0.2520e^{-3})$	$0.7239e^{-3}$	$0.7239e^{-3}$
	75	$(0.0435e^{-3}, 0.0270e^{-3})$	$0.1448e^{-3}$	$0.1448e^{-3}$

Table 5.2: Results for Ackley Function

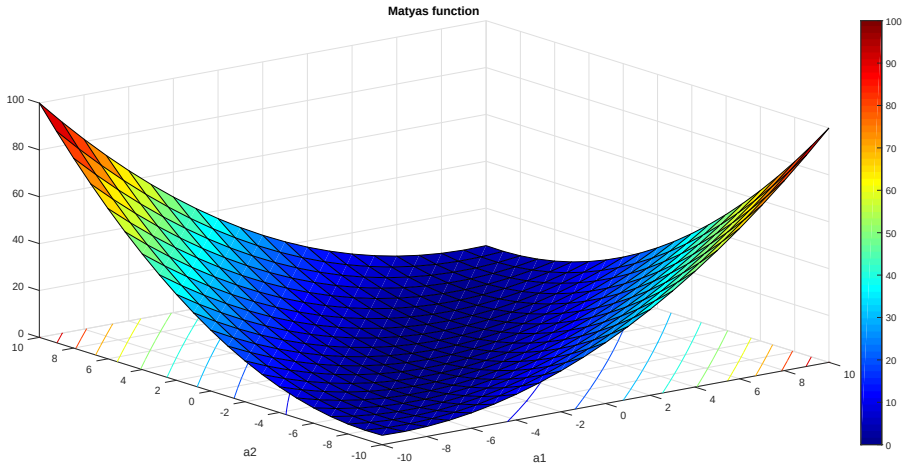

Figure 5.5: Matyas Function

5.1.3 Matyas Function

The Matyas function is denoted by,

$$f_3 = 0.26(a_1^2 + a_2^2) - 0.48a_1a_2. \tag{5.3}$$

It is a plate shaped function. It has no local minima other than the global one $f(a^*) = 0$ at $a^* = (0, 0)$. The graph of the function is shown in Figure 5.5. The function is evaluated in $[-10, 10]^2$ cube and the numerical results are shown in Table 5.3 .

M	N	Optimum position	Optimum value	error
50	50	$(-0.0020, -0.0028)$	0.0000	$3.7663e^{-7}$
	75	$(-0.3105e^{-4}, -0.1024e^{-4})$	0.0000	$1.2536e^{-10}$
75	50	$(-0.3331e^{-3}, -0.1974e^{-3})$	$0.0000e^{-3}$	$7.4187e^{-9}$
	75	$(-0.2613e^{-4}, -0.2422e^{-4})$	$0.0000e^{-4}$	$2.6269e^{-11}$
100	50	(0.0022,0.0025)	0.0000	$2.5717e^{-7}$
	75	$(0.7064e^{-5}, 0.7474e^{-5})$	$0.0000e^{-5}$	$2.1554e^{-12}$

Table 5.3: Results for Matyas Function

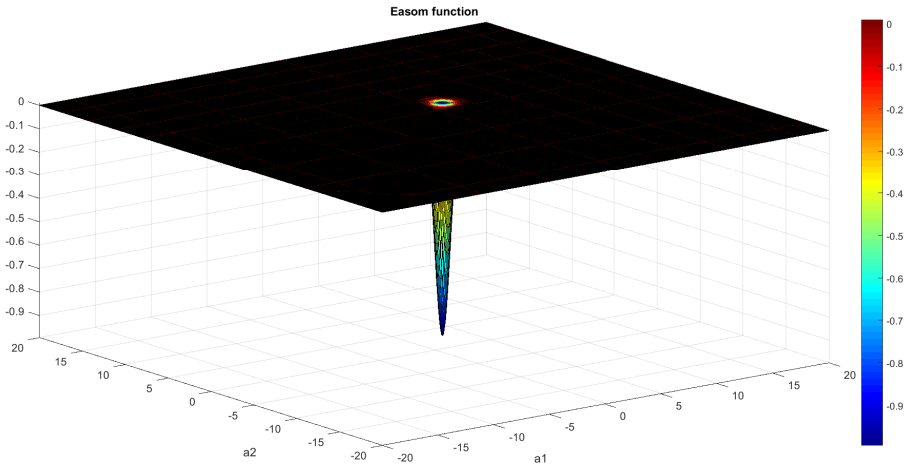

Figure 5.6: Easom Function

5.1.4 Easom Function

The Easom function is denoted by,

$$f_4 = -\cos(a_1)\cos(a_2)\exp[-(a_1-\pi)^2-(a_2-\pi)^2] \tag{5.4}$$

It has several local minima with a unique global minimum $f(a^*) = -1$ at $a^* = (\pi, \pi)$. It is unimodal, and the global minimum has a small area relative to the search space. The graph of the function is shown in Figure 5.6. The function is evaluated in $[-100, 100]^2$ cube and the numerical results are shown in Table 5.4 .

M	N	Optimum position	Optimum value	error
50	50	(3.1459,3.1404)	-1.0000	$2.9603e^{-5}$
	75	(3.1416,3.1416)	-1.0000	$7.4437e^{-10}$
75	50	(3.1445 ,3.1395)	-1.0000	$1.9510e^{-5}$
	75	(3.1416,3.1418)	-1.0000	$7.5945e^{-8}$
100	50	$(3.1410, 3.1421)$	-1.0000	$9.9443e^{-7}$
	75	$(3.1413, 3.1417)$	-1.0000	$1.7289e^{-7}$

Table 5.4: Results for Easom Function

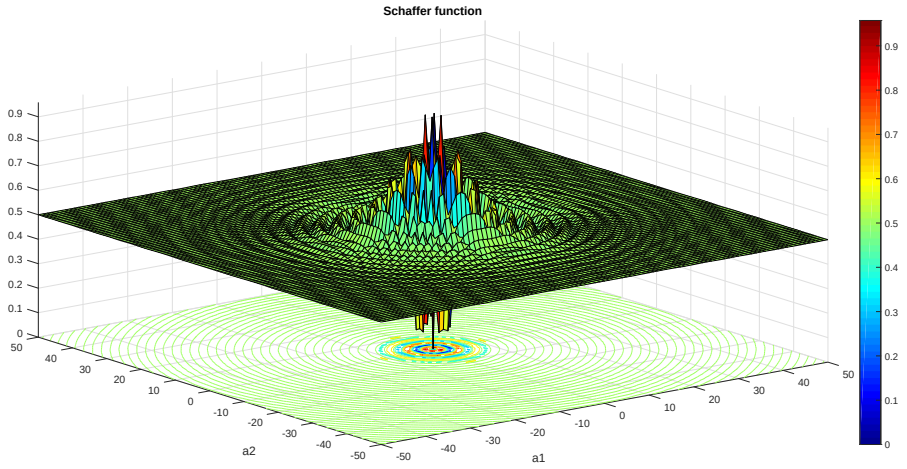

Figure 5.7: Schaffer Function

5.1.5 Schaffer Function

The Schaffer function for dimension 2 is denoted by,

$$f_5 = 0.5 + \frac{\sin^2(\sqrt{a_1^2 + a_2^2}) - 0.5}{(1 + 0.001(a_1^2 + a_2^2))^2} \tag{5.5}$$

It has a unique global minimum at $f(a^*) = 0$ at $a^* = (0, 0)$. The graph of the two dimensional function is shown in Figure 5.7. The function is evaluated in $[-100, 100]^2$ cube and the numerical results are shown in Table 5.5 .

M	N	Optimum position	Optimum value	error
50	50	$(0.0125, -0.0191)$	0.0005	$5.2211e^{-4}$
	75	$(0.1965, -3.1324)$	0.0097	0.0097
75	50	$(0.0047, -0.0104)$	0.0001	$1.2998e^{-4}$
	75	$(-0.3847e^{-3}, -0.0906e^{-3})$	$0.0002e^{-3}$	$1.5638e^{-7}$
100	50	$(0.0008, 0.0013)$	0.0000	$2.1825e^{-6}$
	75	$(-0.3570e^{-3}, 0.7082e^{-3})$	$0.0006e^{-3}$	$6.2965e^{-7}$

Table 5.5: Results for Schaffer Function

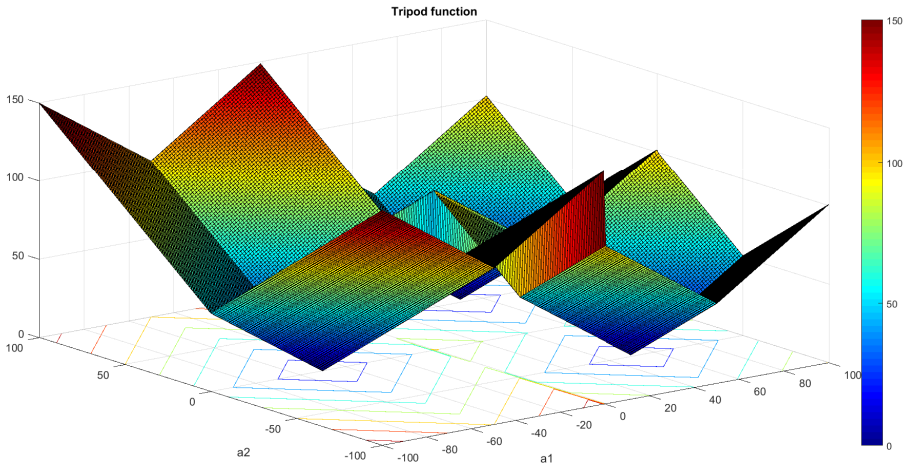

Figure 5.8: Tripod Function

5.1.6 Tripod Function

The Tripod function is denoted by,

$$f_6 = p(a_2)(1 + p(a_1)) + \mid a_1 + 50p(a_2)(1 - 2p(a_1)) \mid + \mid a_2 + 50(1 - 2p(a_2)) \mid$$
$$\text{where } p(s) = \begin{cases} 1; s \geq 0 \\ 0; s < 0 \end{cases} \tag{5.6}$$

It is a semicontinuous function with a global minimum $f(a^*) = 0$ at $a^* = (0, -50)$. The graph of the function is shown in Figure 5.6. The function is evaluated in $[-100, 100]^2$ cube and the numerical results are shown in Table 5.6 .

M	N	Optimum position	Optimum value	error
50	50	$(0.0000, -50.0017)$	0.0017	0.0017
	75	$(-0.0000, -49.9999)$	0.0002	$1.5971e^{-4}$
75	50	$(-0.0066, -50.0003)$	0.0069	0.0069
	75	$(0.0002, -49.9992)$	0.0009	$9.2871e^{-4}$
100	50	$(0.0040, -50.0010)$	0.0050	0.0050
	75	$(0.0001, -50.0000)$	0.0001	$1.3751e^{-4}$

Table 5.6: Results for Tripod Function

5.2 Comparison of Convergence by Using Different Sequences for Initialization

We analyze the performance of PSO by initializing the particles with different sequences. Particles are initialized using 1) the Halton sequence with base 3 for each direction , 2) the optimal/ scrambled Halton sequence with base 3 for each direction and 3) dropping the first 5000 points of optimal Halton sequence. Millie Pant et. al. [11] have used benchmark problems to analyze the performance of PSO, if particles are initialized with a Van der Corput sequence. We have chosen the same functions to compare the optimality of the solution, which are summarized in Table 5.7. The optimum point of each function is at 0. In order to have a fair comparison , we have set number of particles to 20 and 40. Also we have our comparison in the same ranges. "Halton" denotes the result for initializing using the Halton sequence with base 3, "Scrambled H." denotes the result for initializing using the optimal Halton sequence with base 3 and "Drop 5000" denotes result for initializing after dropping the first 5000 points of the optimal Halton sequence with base 3. Since the initialization of the particles follows a random behavior each test is executed 30 times and the mean of the values is taken as the optimum value. The results are summarized in Tables 5.8-5.11. Here M denotes the number of particles and D denotes the dimension of the function. Millie Pant et. al. [11], have performed the algorithm for 1000, 1500 and 2000 iterations. But since our value converges to the optimum point at 50 iterations, we performed the results only for 50 iterations. Also we checked the convergence for high dimensions i.e 70 and 100.

Objective function	Range 1	Range 2
$f_1 = \sum_{i=1}^{D}(x_i^2 - 10cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^D$	$[-100, 100]^D$
$f_2 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \sum_{i=1}^{D} cos(\frac{x_i}{\sqrt{i+1}}) + 1$	$[-600, 600]^D$	$[-1000, 1000]^D$
$f_3 = \sum_{i=1}^{D-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-30, 30]^D$	$[-100, 100]^D$
$f_4 = 20 + e - 20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{D} cos(2\pi x_i))$	$[-32, 32]^D$	$[-100, 100]^D$

Table 5.7: Optimization test functions and its Ranges

M	D	R1:[-5.12, 5.12]			R2:[-100, 100]		
		Halton	Scrambled H.	Drop 5000	Halton	Scrambled H.	Drop 5000
20	10	4.5031e-05	9.4292e-05	1.4199e-05	0.0183	0.0047	0.0093
	20	6.0832e-05	1.7520e-04	9.1973e-06	0.0201	0.0047	0.0070
	30	6.9443e-06	2.6703e-04	1.3455e-04	0.0199	0.0226	0.0153
	50	5.7875e-04	5.5034e-05	1.5749e-05	0.6320	0.1052	0.0104
	70	8.0016e-05	8.1385e-05	7.6740e-05	0.0084	0.0428	0.0163
	100	3.7532e-04	5.7959e-05	1.8847e-04	0.1467	0.0216	0.0068
40	10	3.6110e-06	1.6048e-06	3.6206e-06	0.0040	0.0073	0.0051
	20	6.9260e-05	4.3515e-05	8.5975e-06	0.0059	0.0028	0.0079
	30	1.8150e-05	4.9339e-05	1.4129e-05	0.0058	0.0010	0.0240
	50	6.1877e-06	1.6712e-05	2.3661e-05	0.0015	0.0564	0.0022
	70	1.0746e-04	9.1970e-06	8.7913e-05	0.0032	0.0072	0.0044
	100	1.3386e-04	3.1211e-04	2.5485e-05	0.0043	0.0150	0.0181

Table 5.8: Results of test function f_1

Figures 5.7 - 5.10 show the convergence of global best value in the swarm for each function at 100 dimensions evaluated with 40 particles, in the range 1, and the one on the right is an enlarged view of the same graph showing the convergence for iteration number 25 to 50. The Numerical results shows that for f_1, f_3 and f_4 all three sequences provide similar results. For f_2 , optimal Halton and drop 5000 sequences provides better estimation than Halton sequence when the range is expanded and the number of particles is less.

M	D	R1:[-600, 600]			R2:[-1000, 1000]		
		Halton	Scrambled H.	Drop 5000	Halton	Scrambled H.	Drop 5000
20	10	4.9462e-05	2.2319e-04	1.5571e-04	0.0014	7.6142e-04	1.6756e-04
	20	2.9423e-04	3.2445e-04	1.6124e-04	0.0040	6.5461e-04	5.4820e-04
	30	2.0535e-04	2.0806e-05	5.2344e-05	0.0016	8.8793e-04	2.1003e-04
	50	1.5963e-04	1.7287e-04	2.0185e-04	0.0032	6.7706e-04	7.7382e-04
	70	1.1883e-04	9.1621e-04	2.6643e-04	0.0018	9.9320e-05	0.0029
	100	1.4924e-04	3.6559e-04	8.6489e-04	0.0010	3.5493e-04	0.0022
40	10	1.1053e-05	2.0025e-05	8.0946e-06	3.4932e-04	2.4241e-04	4.9436e-05
	20	2.8379e-05	1.3041e-05	4.0735e-05	1.3493e-04	7.8787e-05	1.0553e-04
	30	2.9824e-04	5.4652e-04	7.1948e-04	6.9671e-05	5.0725e-05	8.3411e-04
	50	1.6720e-04	5.3928e-05	8.3615e-05	7.7936e-04	1.1867e-04	9.9799e-05
	70	3.1517e-04	3.2714e-05	3.6104e-05	5.2766e-04	4.2098e-04	1.4845e-04
	100	1.5062e-04	5.5221e-05	9.3912e-05	1.4093e-04	6.6336e-04	3.1576e-04

Table 5.9: Results of test function f_2

M	D	R1:[-30, 30]			R2:[-100, 100]		
		Halton	Scrambled H.	Drop 5000	Halton	Scrambled H.	Drop 5000
20	10	0.0342	0.0038	0.0023	0.0951	0.0048	0.1557
	20	0.0148	0.0011	0.0047	0.0090	0.1181	0.0222
	30	0.0051	0.0036	0.0021	0.0157	0.0383	0.0076
	50	0.0346	0.0013	4.2693e-04	0.0418	0.1161	0.0031
	70	0.0034	0.0427	0.0025	0.0527	0.0941	0.0272
	100	0.0018	0.0312	0.0087	0.1361	0.0305	0.2750
40	10	3.1437e-04	6.1063e-04	1.9113e-04	0.0134	0.0038	9.7550e-04
	20	7.4913e-04	1.8756e-04	1.7657e-04	0.0263	0.0070	0.0057
	30	7.2197e-04	1.2019e-04	3.9049e-04	0.0584	0.0077	0.0225
	50	0.0049	7.9631e-04	2.1287e-04	0.0257	0.0030	0.0017
	70	0.0031	6.0691e-04	0.0056	0.1823	0.0567	0.0495
	100	0.0026	7.2791e-04	0.0032	0.0412	0.0539	0.0209

Table 5.10: Results of test function f_3

M	D	R1:[-32,32]			R2:[-100, 100]		
		Halton	Scrambled H.	Drop 5000	Halton	Scrambled H.	Drop 5000
20	10	0.0014	0.0072	8.7490e-04	0.0035	0.0060	0.0194
	20	0.0025	0.0026	8.0518e-04	0.0067	0.0028	0.0057
	30	0.0021	9.5752e-04	0.0072	0.0044	0.0018	0.0039
	50	0.0032	0.0022	0.00241	0.0186	0.0062	0.0041
	70	0.0022	0.0017	0.0011	0.0028	0.0026	0.0062
	100	0.0025	5.9831e-04	4.4743e-04	0.0033	0.0033	0.0019
40	10	0.0010	7.9967e-04	5.4900e-04	0.0018	8.6368e-04	
	20	7.4003e-04	5.0134e-04	3.8245e-04	0.0041	0.0025	0.0022
	30	8.6885e-04	6.2990e-04	0.0015	6.2206e-04	0.0043	0.0018
	50	8.8984e-04	0.0010	0.0011	0.0038	0.0016	0.0029
	70	0.0016	5.4819e-04	4.3998e-04	0.0025	0.0025	0.0016
	100	0.0015	7.1370e-04	6.2701e-04	0.0044	0.0012	7.1727e-04

Table 5.11: Results of test function f_4

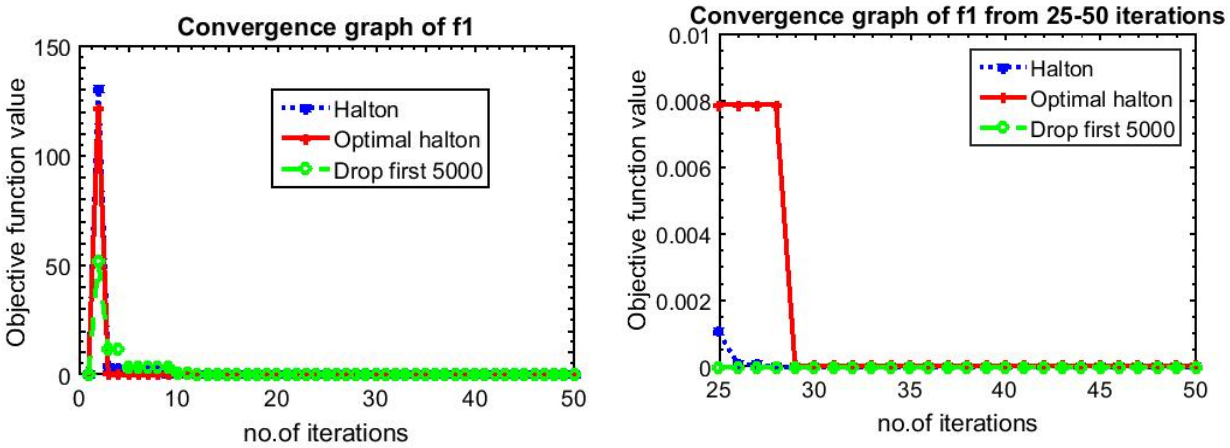

Figure 5.9: Convergence graph for function f_1

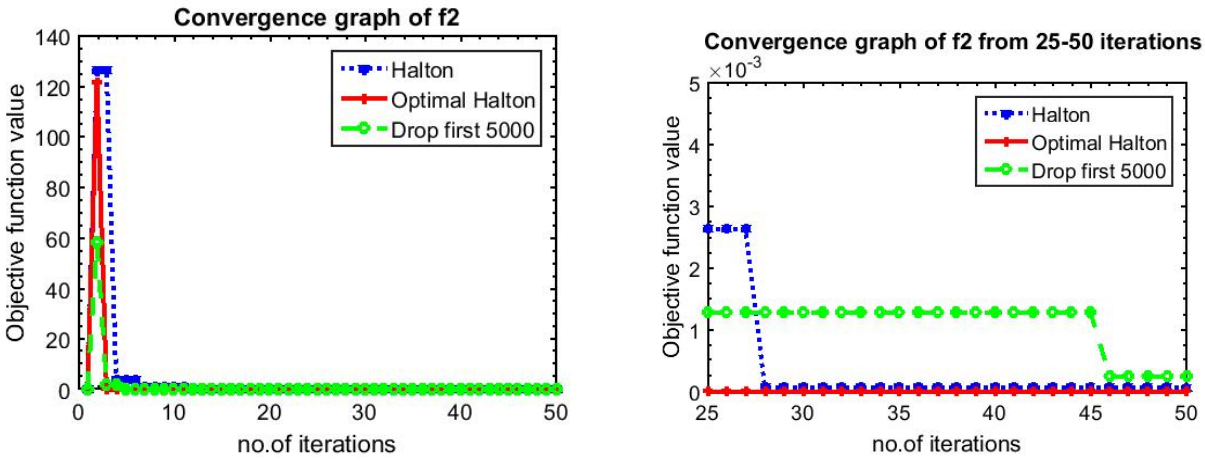

Figure 5.10: Convergence graph for function f_2

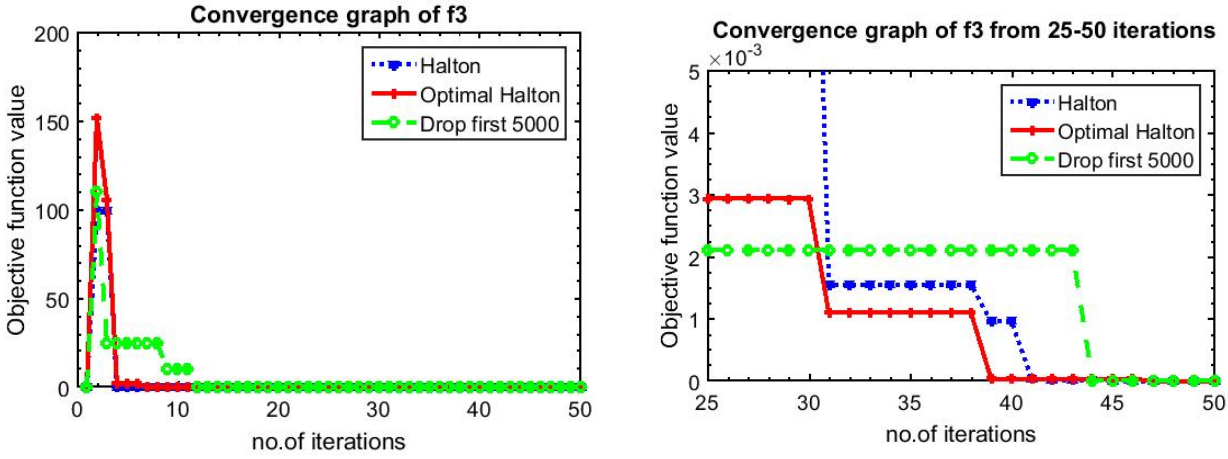

Figure 5.11: Convergence graph for function f_3

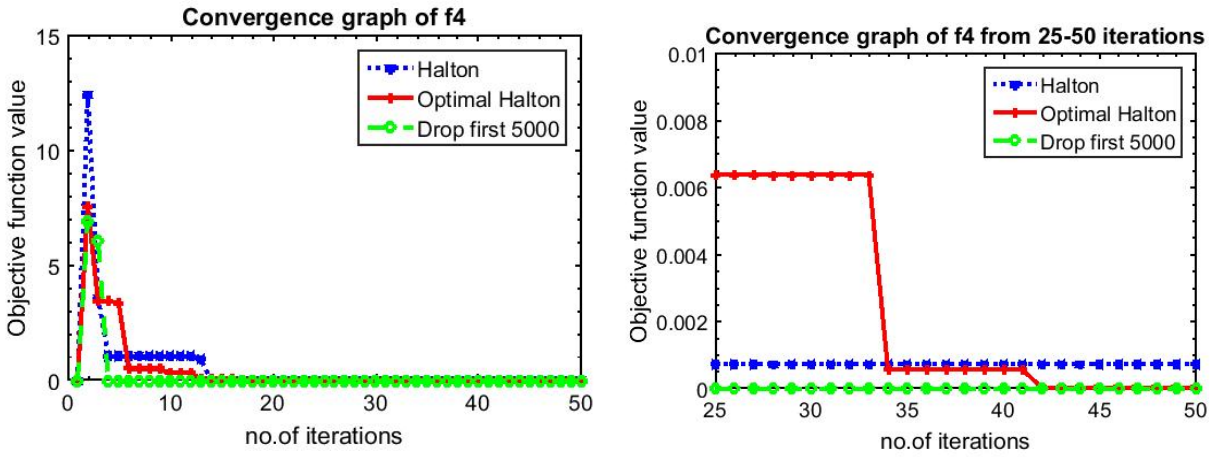

Figure 5.12: Convergence graph for function f_4

Chapter 6

Concluding Remarks

In PSO, particles velocities are updated according to a random manner. Hence the particle's positions are random vectors. So it is important to initialize particles using random sequences. In this dissertation we showed the importance of using randomized quasi-random sequences in initializing the particles. Optimal Halton sequence is one such sequence and we used that sequence for numerical experiments.

From this analysis we conclude that, even though all three sequences provides similar results, optimal Halton sequence and Drop 5000 sequences are better in initializing the swarm due to their random behavior. Also it is clear that accuracy can be improved by increasing the number of particles in the swarm, since it covers the search space more efficiently.

In PSO, Particle's velocity are updated according to a random manner. Hence the particle's positions are random vectors. Thus convergence of particle's position should be analyzed using stochastic process theory. In this dissertation we proved almost sure convergence of PSO, under reasonable assumptions. Given a continuous and bounded objective function with a unique global minimum, we can guarantee the convergence to the optimum value almost surely.

Numerical results from section 5.1 show that particles converge to the optimum value. Functions f_1, f_2, f_3, f_4 and f_5 are continuous functions and f_6 is a semi continuous function. But it also converges to the optimum value. Thus by examining numerical results we can argue that even if the function is semi-continuous near the optimum value it converges to the optimum point.

We proved that convergence of PSO to optimum value as number of iterations goes to infinity. This is inefficient as in the real world our number of iterations is finite. Thus,

future work for this can be focus on to finding expected number of iterations needed so that approximated optimizer falls within a $D-$dimensional ball of radius ϵ about the global optimizer. Further, in future one can compare the rate of convergence when the particles are initialize using, randomized quasi-random sequence, quasi-random sequence and random sequence.

Bibliography

[1] Peter J Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In *International Conference on Evolutionary Programming*, pages 601–610. Springer, 1998.

[2] Claude JP Bélisle. Convergence theorems for a class of simulated annealing algorithms on d. *Journal of Applied Probability*, 29(04):885–895, 1992.

[3] Riaan Brits, Andries P Engelbrecht, and F Van den Bergh. A niching particle swarm optimizer. In *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, volume 2, pages 692–696. Singapore: Orchid Country Club, 2002.

[4] Hongmei Chi, Michael Mascagni, and T Warnock. On the optimal halton sequence. *Mathematics and computers in simulation*, 70(1):9–21, 2005.

[5] Lin Chuan and Feng Quanyuan. The standard particle swarm optimization algorithm convergence analysis and parameter selection. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 3, pages 823–826. IEEE, 2007.

[6] Maurice Clerc. Semi-continuous challenge. *Retrieved from clerc. maurice. free. fr/pso/semi-continuous_challenge/semi-continuous_ challenge. htm*, 2004.

[7] Maurice Clerc and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[8] Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[9] Russ C Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[10] Emad Elbeltagi, Tarek Hegazy, and Donald Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced engineering informatics*, 19(1):43–53, 2005.

[11] Andries P Engelbrecht. *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006.

[12] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[13] Bennett L Fox. Algorithm 647: Implementation and relative efficiency of quasirandom sequence generators. *ACM Transactions on Mathematical Software (TOMS)*, 12(4):362–376, 1986.

[14] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533 – 549, 1986.

[15] Fred Glover. Tabu searchpart i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[16] Fred Glover. Tabu searchpart ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[17] Ahmet Goncu. *Monte Carlo and quasi-Monte Carlo methods in financial derivative pricing*. PhD thesis, THE FLORIDA STATE UNIVERSITY, 2009.

[18] John H Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964.

[19] Rania Hassan, Babak Cohanim, Olivier De Weck, and Gerhard Venter. A comparison of particle swarm optimization and the genetic algorithm. In *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2005.

[20] Tim Hendtlass. A combined swarm differential evolution algorithm for optimization problems. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 11–18. Springer, 2001.

[21] Nicholas Holden and Alex Alves Freitas. A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 100–107. IEEE, 2005.

[22] Ming Jiang, YP Luo, and SY Yang. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, 102(1):8–16, 2007.

[23] A Rezaee Jordehi. Enhanced leader pso (elpso): a new pso variant for solving global optimisation problems. *Applied Soft Computing*, 26:401–417, 2015.

[24] A. Khachaturyan, S. Semenovsovskaya, and B. Vainshtein. Statistical-Thermodynamic Approach to Determination of Structure Amplitude Phases. *Sov.Phys. Crystallography*, 24(5):519524, 1979.

[25] A. Khachaturyan, S. Semenovsovskaya, and B. Vainshtein. The thermodynamic approach to the structure analysis of crystals. *Acta Crystallographica Section A*, 37(5):742–754, Sep 1981.

[26] Shuhei Kimura and Koki Matsumura. Genetic algorithms using low-discrepancy sequences. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1341–1346. ACM, 2005.

[27] R.G. Laha and V.K. Rohatgi. *Probability theory*. Wiley series in probability and mathematical statistics. Wiley, 1979.

[28] Jing J Liang, A Kai Qin, Ponnuthurai Nagaratnam Suganthan, and S Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on*, 10(3):281–295, 2006.

[29] Yanmin Liu, Zhuanzhou Zhang, Yuanfeng Luo, and Xiangbiao Wu. An improved pso for multimodal complex problem. In *Intelligent Computing in Bioinformatics*, pages 371–378. Springer, 2014.

[30] Morten Løvbjerg, Thomas Kiel Rasmussen, and Thiemo Krink. Hybrid particle swarm optimiser with breeding and subpopulations. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 469–476. Morgan Kaufmann Publishers Inc., 2001.

[31] B. Makarov and A. Podkorytov. *Real Analysis: Measures, Integrals and Applications*. Universitext. Springer London, 2013.

[32] Alberto Colorni Marco Dorigo Vittorio Maniezzo. Distributed optimization by ant colonies. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, page 134. Mit Press, 1992.

[33] Michael Mascagni and Hongmei Chi. On the scrambled halton sequence. *Monte Carlo Methods and Applications mcma*, 10(3-4):435–442, 2004.

[34] Tentu Monica, Anguluri Rajasekhar, Millie Pant, and Ajith Abraham. Enhancing the local exploration capabilities of artificial bee colony using low discrepancy sobol sequence. In *International Conference on Contemporary Computing*, pages 158–168. Springer, 2011.

[35] Mahamed GH Omran, Salah Al-Sharhan, Ayed Salman, and Maurice Clerc. Studying the effect of using low-discrepancy sequences to initialize population-based optimization algorithms. *Computational Optimization and Applications*, 56(2):457–480, 2013.

[36] Ender Ozcan and Chilukuri K Mohan. Particle swarm optimization: surfing the waves. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 1939–1944. IEEE, 1999.

[37] Millie Pant, Radha Thangaraj, and Ajith Abraham. Low discrepancy initialized particle swarm optimization for solving constrained optimization problems. *Fundamenta Informaticae*, 95(4):511, 2009.

[38] Millie Pant, Radha Thangaraj, Crina Grosan, and Ajith Abraham. Improved particle swarm optimization with low-discrepancy sequences. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3011–3018. IEEE, 2008.

[39] Riccardo Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008, 2008.

[40] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[41] Riccardo Poli, William B Langdon, and Owen Holland. Extending particle swarm optimisation via genetic programming. In *European Conference on Genetic Programming*, pages 291–300. Springer, 2005.

[42] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[43] Jacob Robinson, Seelig Sinton, and Yahya Rahmat-Samii. Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In *Antennas and Propagation Society International Symposium, 2002. IEEE*, volume 1, pages 314–317. IEEE, 2002.

[44] W. Rudin. *Principles of Mathematical Analysis*. International series in pure and applied mathematics. McGraw-Hill, 1976.

[45] Leyuan Shi and Sigurdur Ólafsson. Nested partitions method for global optimization. *Operations research*, 48(3):390–407, 2000.

[46] Yuhui Shi and Russell C. Eberhart. *Parameter selection in particle swarm optimization*, pages 591–600. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[47] Francisco J Solis and Roger J-B Wets. Minimization by random search techniques. *Mathematics of operations research*, 6(1):19–30, 1981.

[48] James C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2003.

[49] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[50] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317 – 325, 2003.

[51] Frans Van Den Bergh. An analysis of particle swarm optimizers (pso). 2001.

[52] J.G. van der Corput. Verteilungsfunktionen. I. Mitt. *Proc. Akad. Wet. Amsterdam*, 38:813–821, 1935.

[53] Ganesha Weerasinghe, Hongmei Chi, and Yanzhao Cao. Particle swarm optimization simulation via optimal halton sequences. *Procedia Computer Science*, 80:772–781, 2016.

[54] Chengjian Wei, Zhenya He, Yifeng Zhang, and Wenjiang Pei. Swarm directions embedded in fast evolutionary programming. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1278–1283. IEEE, 2002.

[55] Xin-She Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[56] Keiichiro Yasuda, Azuma Ide, and Nobuhiro Iwasaki. Adaptive particle swarm optimization. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 2, pages 1554–1559. IEEE, 2003.

[57] Z.B. Zabinsky. *Stochastic Adaptive Search for Global Optimization*. Nonconvex Optimization and Its Applications. Springer US, 2013.

[58] Zelda B. Zabinsky. *Random Search Algorithms*. John Wiley & Sons, Inc., 2010.

Appendices