This is to certify that Dissertation entitled

Prototype Virtual Reality Enabled Particle Swarm Optimizer

Presented by

Arpit M. Patel

has been accepted toward fulfillment of the requirement
for the degree of
Master of technology in Computer Science & Engineering

Prof. S. N. Pradhan	Prof. D. J. Patel
Professor In Charge	Head of The Department

Prof. A. B. Patel
Director, Institute of Technology

CERTIFICATE

This is to certify that the Major Project entitled "Embedded System Modeling and Simulation for In Bus System" submitted by Mr. Arpitk M. Patel (05MCE008), towards the partial fulfillment of the requirements for the degree of Master of Technology in Compute Science & Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any Master degree.

Project Guide

Dr. (Prof.) S. N. Pradhan
P. G. Coordinator,
Department of Computer Engineering,
Institute of Technology,
Nirma University,
Ahmedabad

Date :-

ACKNOWLEDGEMENT

This indeed is the moment of proud pleasure to present "Prototype Virtual Reality Enabled Particle Swarm Optimizer" - a work of comparatively shorter duration that would go a long way in making the systems.

Getting an opportunity to work in one of the country's most scientifically organized institution – the prestigious IPR, is like a dream come true. My limited association with IPR has provided me an unparallel exposure to the challenging task of developing the system, which is the first ever of its type, in a very short period of time. Exposure to the work culture, discipline, high professionalism and dedication, I was so fortunate to get while working at IPR, has immensely benefited my perspective and has left deep imprints on my mind to last throughout my career.

I do not know how to express my gratitude to the Institute, the authorities of IPR and more particularly to my project guide **Dr. Shashank Chaturvedi** and **Mr. Ritesh Sugandhi,** sufficiently, for allowing me this opportunity.

I wish to thank **Dr S.N.Pradhan**, whose keen interest and substantial encouragement motivated me to take up this challenging project.

I am thankful to all faculty members of Department of Computer Science & Engineering, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

The blessings of God and my family members makes the way for completion of major project. I am very much grateful to them. Finally I thank one and all those who helped and encouraged me in every manner.

Arpit M. Patel
05MCE008

ABSTRACT

Prototype Virtual Reality Enabled Particle Swarm Optimizer

By Arpit M. Patel

Plasma Physics is a complex nonlinear system. The problems of the field involve the optimization of a set of a parameters and their constraint checking on a set of boundary conditions. The convergence of algorithm for local and global minima is of importance. Due to unpredictable nature of solution, use of virtual reality based modeling will be of due importance for better understanding and sharing it for discussion. Numerous optimization algorithms have been proposed to solve these problems with varying degree of success. The particle swarm optimization is relatively new techniques that have been empirically shown to perform well on many of this optimization problem. It is inspired by the behavior of the swarm found in nature. The basic optimization algorithm of PSO has been enhanced for guaranteed convergence with various configurations and topologies. This project will be focus on study of Particle Swarm Optimization algorithms and modeling of physical problem of 3rd degree nonlinear equations using it and comparing it with empirical results. Empirical results are presented to support theoretical properties predicted by the various models, using synthetic benchmark functions to investigate specific properties. In the later phase more complex problems related to plasma physics will be tried out and results will be achieved.

CONTENTS

LIST of FIGURES

LIST of TABLES

1. INTRODUCTION

> *"You awaken to the sound of your alarm clock. A clock that was manufactured by a company that tried to maximize its profit by looking for the optimal allocation of the resources under its control. You turn on the kettle to make some coffee, without thinking about the great lengths that the power company went to in order to optimize the delivery of your electricity. Thousands of variables in the power network were configured to minimize the losses in the network in an attempt to maximize the profit of your electricity provider. You climb into your car and start the engine without appreciating the complexity of this small miracle of engineering. Thousands if parameters were fine-tuned by the manufacturer to deliver a vehicle that would live up your expectations, ranging from the aesthetic appeal of the bodywork to the specially shaped side-mirror cowls, designed to minimize drag. As you hit the gridlock traffic, you think "Couldn't the city planners have optimized the road layout so that I could get to work in under an hour?"*

Optimization forms an important part of our day-to-day life. Many scientific, social, economic and engineering problems have parameters that can be adjusted to produce a more desirable outcome. Over the years numerous techniques have been developed to solve such optimization problems. This thesis investigates the behavior of a relatively new technique known as Particle Swarm Optimization, a technique that solves problems by simulating swarm behavior.

1.1 About Organization: Institute for Plasma Research

Institute for Plasma Research was established in 1986, as an autonomous institution, funded by the Department of Atomic Energy. Institute for Plasma Research is a Premier Institute; the major objectives of institute carrying out fundamental theoretical and experimental research on magnetically confined hot Plasmas and nonlinear plasma phenomena. It has its branch, Facilitation Center for

Industrial Plasma Technology (FCIPT), at GIDC, Gandhinagar, which basically builds industrial plasma appliances and makes some funding to the Institute. The Institute has various working groups, viz., Pulsed Power Group, RF Group, Cryogenics Group, Vacuum Group, Modeling Group, Neutral Beam Injection Group, etc.

1.2 Motivation

Plasma physics is a complex non-linear system. These systems have various parameters whose values define various system properties, system behavior, system architecture, system design, system requirement, etc…. The problem of the field involves the optimization of these set of parameters and their constraint checking on a set of boundary conditions. Thus the important task is the convergence of the algorithm for the local and global minima in order to optimize complex non-linear problems.

The nature of the solution to these complex nonlinear problems is unpredictable, so use of virtual reality based modeling will be of importance for better understanding and analysis. By using virtual reality based modeling the problems and their solutions can be studied in microscopic way. To solve these kinds of problems numerous techniques have been proposed which has given varying degree of success.

The Particle Swarm Optimization (PSO), which is inspired by the behavior of the swarm found in nature, is a relatively new technique that has been empirically shown to perform well on many of the optimization problems other than Plasma Physics related optimization problems like neural network learning algorithm, Human tremor analysis, Reactive power and voltage control, Distribution state estimation, state of charge of battery pack, rule extraction in fuzzy neural network, etc...[2].

The main motivation is that, the PSO has been never applied in the area of Plasma Physics, so here it is to be applied first time in the area of plasma physics.

1.3 Objectives

The primary objectives of this thesis can be summarized as follows:

To develop Particle Swarm Optimization engine with different variations of PSO algorithm and various topologies to solve various kinds of optimization problems

To simulate the behavior of the convergence of 3rd degree nonlinear equations using Java3d API

To select optimum values for parameters of the PSO for better performance of PSO

To analyze the performance of various PSO using various benchmark functions

To apply PSO for optimization of various Plasma Physics related problems

1.4 Scope of Project

The PSO has been applied to a vast number of problems, though not all of these applications have been described in published material yet. This section will briefly describe scope of the PSO in some areas.

Neural Network training was one of the first applications of the PSO, Kennedy and Eberhart reported that the PSO was successful in training a network to correctly classify the XOR system, a process involving the minimization of function in a 13-dimensional search space [1]. They also reported that the PSO could train a neural network to classify Fisher's Iris Data set.

In fact, most PSO applications reported in the literature involve neural network training. Eberhart and Hu used the PSO to train a network to correctly classify a patient as exhibiting *essential tremor,* or suffering from Parkinson's disease. Their PSO implementation used an inertia weight that decreased linearly from 0.9 to 0.4 over 2000 iterations.

The PSO has also been used to evolve the architecture of a neural network in tandem with the training of the weights of the network.

Eberhart describes several other applications of PSO in [2], including some more neural network training applications. Tandom used the PSO to train a neural

network used to simulate and control an end milling process. End milling involves the removal of metal in a manufacturing process, using computer numerically controlled (CNC) machine tools. Another neural network training application described by Eberhart is that of training a network to estimate the state-of-charge of a battery pack. Another neural network training application, Fuzzy Neural Network was studied by He[2].

Another scope of PSO unrelated to neural network was used to optimize to ingredient mix of chemicals used to facilitate the growth of strains of micro-organisms. Another scope of PSO unrelated to neural network was published by Fukuyama and Yoshiba. They have shown that the PSO is very effective at optimizing continuous and discrete variables simultaneously.

PSO is also useful for Reactive Power and Voltage Control and Power System Stabilizer Design[2][3].

In the area of Plasma Physics, PSO is used for optimum parameters selections of Nuclear Fusion Reactors for minimization of cost per unit energy. It is also useful for atomic cluster configuration for lowest energy configurations.

The virtual reality enable Particle Swarm Optimizer is useful for the modeling the behavior of various 3^{rd} degree nonlinear equations for better understanding and analysis. Virtual reality enable PSO can be useful for analysis of various problems in microscopic way.

1.5 Thesis Outline

Chapter 2 starts with an introduction to the theory of optimization. This is followed by a description of the Particle Swarm Optimizer. In this, the focus will be on background of PSO, optimization process of PSO, neighborhood topologies used in PSO and variants of PSO.

Chapter 3 presents requirement specification of system, which is to be made. Here system requirements are identified and their feasibility is checked for implementation.

Chapter 4 describes the design of the system, which is to be implemented. Here system's class diagram, sequence diagram, etc… are discussed. In this section, overall structure and style of system are discussed.

Chapter 5 discussed about performance analysis of the system. In this section, optimum parameter selection of the PSO is discussed. After that different variants of the PSO's results are analyzed.

Chapter 6 explains applications of PSO. These applications are of the field of Plasma Physics. Here various PSO are used to solve the problems of Plasma Physics and the results are analyzed.

Chapter 7 presents a summary of the findings of this thesis. Some topics for future research are also discussed.

The appendices present, in order, a list of useful websites, a list of published paper.

2. Background

This chapter reviews some of the basic definitions related to optimization. Then the origin of Particle Swarm Optimizer is discussed, followed theory of Particle Swarm Optimizer and variants of PSO are described. At the end, Virtual Reality for PSO is discussed.

2.1 Optimization

The task of optimization is that of determining the values of a set of parameters so that some measure of optimality is satisfied, subject to certain constraints. This task is of great importance to many professions, for example chemists, scientists, engineers etc. The term optimization refers to both minimizations and maximization tasks. A task involving the maximization of the function is equivalent to the task of minimizing –f, therefore the terms minimization, maximization and optimization will be used interchangeably.

Techniques used to solve the minimization problems defined above can be placed into two categories: Local and Global Optimization algorithms

1. *Local optimization*

   A local minimiser, x_B^* of the region B is defined so that $f(x_B^*) \quad f(x), \quad x \quad B$, where $B \quad S \quad R^n$, and S denoted the search space. B is a proper subset of S. Here, the whole search space (S) is divided into number of blocks with smaller search space (B). The optimization or finding minima or maxima is performed on these smaller blocks of search spaces.

2. *Global optimization*

   A global minimiser, x^* of the region S is defined so that $f(x^*) \quad f(x), \quad x \quad S$, Where S is the search space. The optimization or finding minima or maxima is performed on the whole search space (S).

2.2 Particle Swarm Optimizers

Particle Swarm Optimization is a population based search technique, was first introduced by Dr. Russell C. Eberhart and Dr. James Kennedy [1] in 1995. As

described by Eberhart and Kennedy, "Particle Swarm algorithm imitates human (or insects) social behavior. Individuals interact with one another while learning from their own experience, and gradually the population members move into better regions of the problem space". Here they called each individual as "Particle" because they felt that velocities and accelerations are more appropriately applied to particles.

2.2.1 Background of PSO

Natural creatures sometimes behave as a swarm. One of the main streams of artificial life researches is to examine how natural creatures behave as a swarm and reconfigure the swarm models inside a computer. Swarm behavior can be modeled with a few simple rules. School of fishes and swarm of birds can be modeled with such simple models. Namely, even if the behavior rules of each individual (particle) are simple, the behavior of the swarm can be complicated. Reynolds called this kind of particle as *boid* and generated complicated swarm behavior by CG animation [3]. He utilized the following three vectors as simple rules.

(1) To step away from the nearest particle
(2) To go toward the destination
(3) To go to the center of the swarm

Namely, behavior of each particle inside the swarm can be modeled with simple vectors. This characteristic is one of the basic concepts of PSO. Boyd and Richerson examine the decision process of human being and developed the concept of individual learning and cultural transmission [4]. According to their examination, people utilize two important kinds of information in decision process. The first one is their own experience; that is, they have tried the choices and know which state has been better so far, and they know how good it was. The second one is other people's experiences; that is, they have knowledge of how the other peoples around them have performed. Namely, they know which choices their neighbors have found are most positive so far and how positive the best pattern of choices was. Namely each person decides his decision using his own experiences and other peoples' experiences. This characteristic is another basic concept of PSO.

2.2.2 Optimizer Engine

Particle Swarm has two primary operators: Velocity update and Position update. During each generation each particle is accelerated toward the particles previous best position and the global best position. At all iteration new velocity value for each particle is calculated based on its current velocity, the distance from its previous best position, and the distance from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a set number of times or until a minimum error is achieved. Namely, PSO is basically developed through simulation of bird flocking in two-dimension space. The position of each particle is represented by XY axis position and also the velocity is expressed by V_X (the velocity of X axis) and V_Y (the velocity of Y axis). Modification of the particle's position is realized by the position and velocity information. Bird flocking optimizes a certain objective function. Each particle knows its best value so far (*pBest*) and its XY position. This information is analogy of personal experiences of each particle. Moreover, each particle knows the best value so far in the group (*gBest*) among pBests. This information is analogy of knowledge of how the other particles around them have performed. Namely, each particle tries to modify its position using the following information: current positions (x, y, z), velocities (V_X, V_Y, V_Z), distance between the current position and pBest and distance between the current position and gBest.

This modification can be represented by the concept of velocity. Velocity of each particle can be modified by the following equation:

$$V_i^{k+1} = V_i^k + C_1 * rand_1 * (pBest_i - S_i^k) + C_2 * rand_2 * (gBest - S_i^k) \quad \ldots\ldots\ldots (1)$$

Where V_i^{k+1} = Velocity of particle i at iteration k,

C_1 = Self confidence parameter

C_2 = Group confidence parameter

$rand_{1,2}$ = Random number between 0 and 1

S_i^{k+1} = position of particle i at iteration k

$pBest_i$ = pBest of particle i

$gBest_i$ = gBest of particle i

Using the above equation, a certain velocity, which gradually gets close to pBest and gBest, can be calculated. The current position (searching point in the solution space) can be modified by the following equation:

$$S_i^{k\ 1} \quad V_i^{k\ 1} \quad S_i^{k} \qquad \text{..... (2)}$$

Figure 2.1 shows a concept of modification of a searching point by PSO and Figure 2.2 shows a searching concept with particles in a solution space. Each particle changes its current position using the integration of vectors as shown in Figure 2.1.

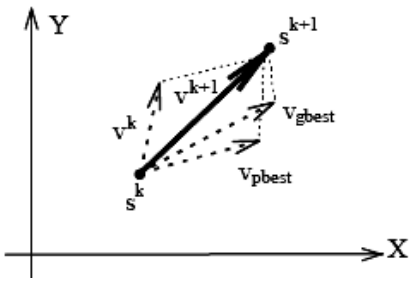

Fig.2.1 Concept of modification of a searching point by PSO

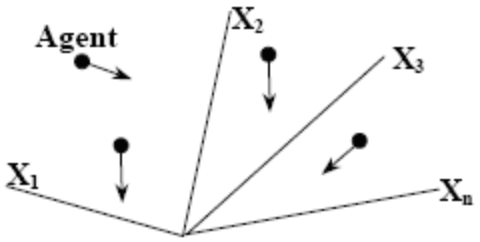

Fig 2.2 Searching concept with particles in a solution space by PSO

The general flow chart for the PSO can be described as follows:

Step: 1 Generation of initial condition of each particle

Initial searching points (S_i^0) and velocities (V_i^0) of each particle are usually generated randomly within the allowable range. The current searching point is set to pBest for each particle. The best-evaluated value of pBest is set to gBest and the particle number with the best value is stored

Step: 2 Evaluation of searching point of each particle

The objective function value is calculated for each particle. If the value is better than the current pBest of the particle, the pBest value is replaced by the current value. If the best value of pBest is better than the current gBest, gBest is replaced by the best value and the particle number with the best value is stored.

Step: 3 Modification of each searching point

The current searching point of each particle is changed using (1) and (2).

Step: 4 Checking the exit condition

The current iteration number reaches the predetermined maximum iteration,

number, then exit. Otherwise, go to step 2.

Figure 2.3 shows the general flowchart of PSO. The features of the searching procedure of PSO can be summarized as follows:

(a) As shown in (1) and (2), PSO can essentially handle continuous optimization problem.

(b) PSO utilizes several searching points like genetic algorithm (GA) and the searching points gradually get close to the optimal point using their pBests and the gBest.

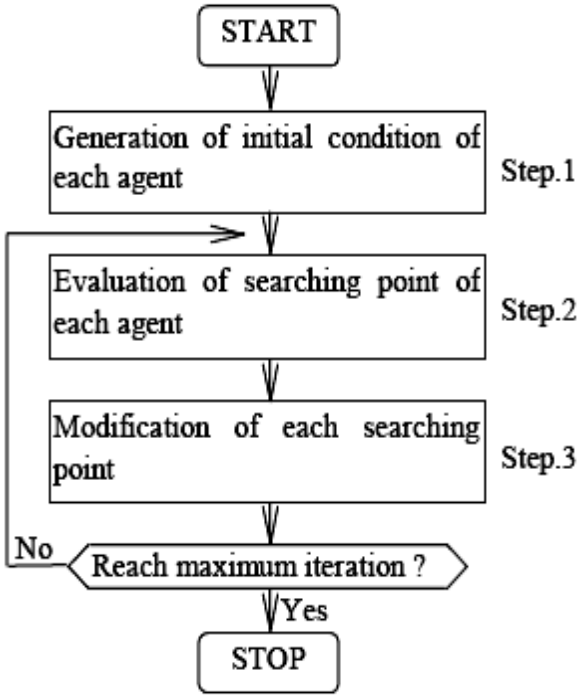

Figure2.3: General Flow Chart of PSO

(c) The first term of right-hand side (RHS) of (1) is corresponding to diversification in the search procedure. The second and third terms of that are corresponding to intensification in the search procedure. Namely, the method has a well-balanced mechanism to utilize diversification and intensification in the search procedure efficiently.

(d) The above concept is explained using only XY-axis (two- dimension space). However, the method can be easily applied to n-dimension problem. Namely, PSO

can handle continuous optimization problems with continuous state variables in a n-dimension solution space.

The above feature (c) can be explained as follows [3]. The RHS of (1) consists of three terms. The first term is the previous velocity of the particle. The second and third terms are utilized to change the velocity of the particle. Without the second and third terms, the particle will keep on "flying" in the same direction until it hits the boundary. Namely, it tries to explore new areas and, therefore, the first term is corresponding to diversification in the search procedure. On the other hand, without the first term, the velocity of the "flying" particle is only determined by using its current position and its best positions in history. Namely, the particles will try to converge to their pBests and/or gBest and, therefore, the terms are corresponding to intensification in the search procedure.

The basic algorithm for the particle swarm optimizer is shown as follows:

```
Procedure PSO
  Repeat
    For i = 1 to number of individuals do
      If G( S_i ) > pBest_i then                    • G() evaluates goodness
        For d = 1 to dimensions do
          pBest_i,d = S_i,d                         • pBest_i,d is the best state found
                                                      so far
        End for
      End if
      g = i                                         • arbitrary
      For j = indexes of neighbors do
        If G( S_j ) > gBest_j then
          g = j                                     • g is the index of the best
                                                      performer in the neighborhood
        End if
      End for
```

For d = 1 to number of dimensions do

$V_{id}(t)$ $f(S_{id(t-1)}, V_{id}(t-1), p_{id}, p_{gd})$ • Update velocity

$V_{id}(t)$ $(-V_{max}, V_{max})$

$S_{id}(t)$ $f(V_{id}(t), S_{id}(t-1))$ • Update position

End for

End for

Until stopping criteria

End procedure

2.2.3 Neighborhood Topologies

There are three main neighborhood topologies used in PSO: Circular topology, wheel topology and star topology [6]. The choice for neighborhood topology determines which individual to use for gBest. The topology affects the rate of convergence and the parallelism of the search.

In circle topology, each individual in society is connected to its k- nearest topological neighbors, from which each particle obtains its gBest value, which is the best individual result among its k-nearest neighbors, where k is typically 2. Figure 2.4 shows five individual connected with circular topology.

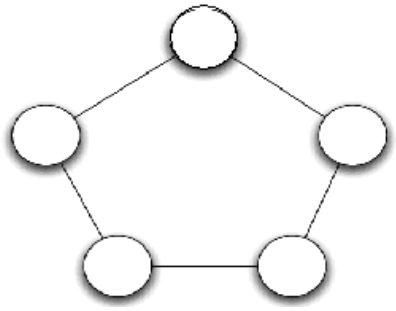

Figure 2.4 Circular Topology

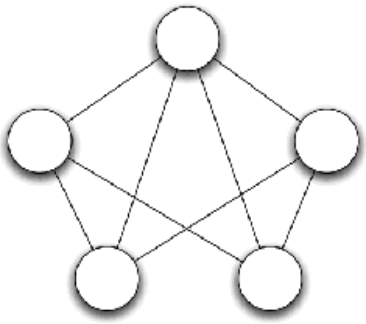

Figure 2.5 Circular Topology

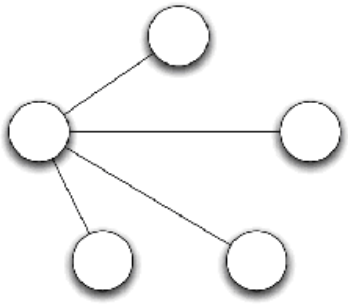

Figure 2.6 Wheel topology

The star topology (see Fig. 2.5) is better known as the global best topology. Here every individual is connected to every other individual in the swarm. Here gBest becomes the best individual's result in the whole population. The Wheel

topology (see Fig. 2.6) effectively isolates individuals from one another, as information has to be communicated through a focal individual, whose best value is known as fBest (focal best). From that, the gBest value can be found as gBest = best {fBest, pBest}.

2.2.4 Variations of Particle Swarm Optimization

Researchers have uncovered many variants determined to be better with respect to convergence speed and robustness. Numerous improvements to the PSO have been proposed. These improvements usually involve changes to the PSO update equations, without changing the structure of the algorithm.
Here three variants of PSO are discussed.

1. Inertia Weight PSO

Some of the earliest modifications to the original PSO were aimed at further improving the rate of convergence of the algorithm. One of the most widely used improvements is the introduction of the *inertia weight,* which is developed by Shi and Eberhart in 1998 [5]. The inertia weight is a scaling factor associated with the velocity during the previous time step, resulting in a new velocity update equation, so that

$$V_i^{k\ 1} \quad W * V_i^k \quad C_1 * rand_1 * (pBest_i \quad S_i^k) \quad C_2 * rand_2 * (gBest \quad S_i^k)$$

Where W = Inertia Weight

Based on the values of inertia weight there are two variants of Inertia Weight PSO:

1) Constant Inertia Weight PSO

   In Constant Inertia Weight PSO, the w value is kept constant for iterations in the execution of optimization algorithm. From the empirical results, it can be shown that the inertia weight w is selected in the range of [0.8, 1.2] [Chapter 4] for better convergence speed and robustness.

2) Variable Inertia Weight PSO

   In Variable Inertia Weight PSO, the w value is decreased linearly over time from 1.0 to 0.5. At the start of simulation runs, the inertia weight value is

1.0 and as the time goes the value of inertia weight is decreased linearly up to 0.5. This approach allows the PSO to explore a large area at the start of simulation run (when the inertia weight is large), and to refine the search later by using a smaller inertia weight. This approach gives the better performance compare to constant inertia weight PSO (Chapter 4). So, decreasing the inertia weight over time introduces a shift from the exploratory (global search) to exploitive (local search) mode.

The inertia weight governs how much of the previous velocity should be retained from the previous time step. To briefly illustrate the effect of w, let C_1 = C_2 =0. Now, a w value greater than 1.0 will cause the particle to accelerate up to maximum velocity V_{max} (or $-V_{max}$) where it will remain as it is. A w value less than 1.0 will cause particle to decelerate until its velocity reaches zero. When C_1 = C_2 0, the behavior of the algorithm is harder to predict, but based on the results of Shi and Eberhart [5] it would appear that w values close to 1.0 are preferable.

2. Constriction PSO

Another PSO variation named Constriction Coefficient PSO was developed by Clerc [6] in 2000. In this variation of PSO, one new *constriction factor* is introduced for ensuring convergence. The constriction factor model describes a way of choosing the values of w, C_1 and C_2 so that the convergence is ensured. By choosing these values correctly, the need for clamping the values of velocity to the range Vmax can be omitted.

A modified velocity update equation, corresponding to constriction model is presented in below equation.

$V_i^{k\ 1}$ $*(V_i^k$ $C_1 * rand_1 * (pbest_i$ $S_i^k)$ $C_2 * rand_2 * (gbest$ $S_i^k))$ Whe

re Constriction factor, $\frac{2}{\left|2 \quad \sqrt{\ ^2 \quad 4}\right|}$

and C_1 C_2, 4

Clert, et al., found that by modifying , the convergence characteristics of the system could be controlled.

Let $C_1 = C_2 = 2.05$ and substitute $C_1 \quad C_2 \quad 4.1$ into above equation of constriction factor, will give value of 0.7298.

Substituting value of constriction factor in the above velocity update equation, will result in

$$V_i^{k\ 1} \quad 0.7298 * (V_i^k \quad 2.05 * rand_1 * (pbest_i \quad S_i^k) \quad 2.05 * rand_2 * (gbest \quad S_i^k))$$

This is equivalent to choosing $C_1 = C_2 = 2.05*0.7298 = 1.4962$ and $w = 0.7298$. So the modified equation can be represented with w, C_1 and C_2 as follows:

$$V_i^{k\ 1} \quad 0.7298 * V_i^k \quad 1.4962 * rand_1 * (pbest_i \quad S_i^k) \quad 1.4962 * rand_2 * (gbest \quad S_i^k))$$

By using constriction factor, the amplitude of the particle's oscillation decreases. But for some functions, however, the PSO with the constriction factor failed to reach the specified error threshold for the specified number of iterations. To mitigate this effect, use of velocity constriction is applied, which gives improved performance for almost all the functions- both in terms of the rate of convergence and the ability of the algorithm to reach the error threshold. The particle will oscillate around the weighted mean of *pBest* and *gBest*, if the previous best position and the neighborhood best position are near each other the particle will perform a local search. If the previous best position and the neighborhood best position are far apart from each other the particle will perform a more exploratory search. During the search the neighborhood best position and previous best position will change and the particle will shift from local search back to global search.

1. Fully Informed PSO

This variation of PSO [7] is made up on the basis of constriction PSO. A particle searches through its neighbors in order to identify the one with the best result so far, and used information from that one source to bias its search in promising

direction. In the constriction PSO the two terms $(pbest_i \quad S_i^k)$ and $(gbest \quad S_i^k)$ are of the same kind. Hence it can be condensed to the following.

Velocity updates equation: $V_i^{k\ 1} \quad *(V_i^k \quad {}_i*(p_i \quad S_i^k))$

Position updates equation: $S_i^{k\ 1} \quad S_i^k \quad V_i^k$

Where ${}_i \quad 1 \quad 2$

and $p_i \quad (\ 1* pbest \quad 2* gbest)/(\ 1 \quad 2)$

This shows that the particle tends to converge towards a point determined by p_i, which a weighted average of its previous best pBest and the neighborhood's best gBest.

Here variation is introduced in several ways: First, the term is weighted by a random number. This in itself would not prevent the velocity from approaching a zero limit. For instance, if the pm-Si difference equals zero, the velocity will still converge to zero. Another important source of variation is the difference between P_i and S_i. As long as the position of the particles differs from the previous best position, then there will be movement. Here pm does not remain fixed, and a key source of variation is the updating of Pm over time as new points are found in the search space, which are better than those previous ones. For convergence it is necessary for Pm to remain fixed.

The velocity update equation can be further generalized to any number of terms:

$$V_i^{k\ 1} \quad *(V_i^k \quad {}_{k\ N} Rand\ 0, \frac{}{|N|} \ *(p_k \quad S_i^k))$$

Here N denotes the neighborhood and p_k denotes the best previous position by the k-th particle in N. if |N| equals 2 then above is a generalization of the Standard PSO. The results (Chapter 4) of the FIPS shows that the rate of convergence of this algorithm is very fast compare to all above approaches. The number of faults is also very less compare to others. In FIPS, all the informants take part into the

calculation of new velocity of particle. In FIPS, increasing the size of the neighborhood seems to deteriorate the performance of the swarm.

2.3 Virtual Reality for PSO

Virtual reality (VR) is a technology, which allows a user to interact with a computer-simulated environment, be it a real or imagined one. Most current virtual reality environments are primarily visual experiences, displayed either on a computer screen or through special stereoscopic displays. In nature, real world problems can be expressed as numerical equations. For better understanding and analysis of these equations i.e. real problems, they should be studied in a microscopic way. These problems can be solved and analyzed using mathematical models or optimization theories. This can be used for function understanding or algorithm analysis method. So, as the microscopic analysis requirement, there is a need of Virtual reality modeling. Here the behavior of the PSO is modeled and analyzed using virtual reality. Users can interact with a virtual environment for analyzing PSO behavior; he can use keyboard or mouse. The simulated environment can be similar to the real world

3. Requirement Analysis

Requirement is a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose. Since Particle Swarm Optimizer has been widely studied and has been implemented in many applications, but as this thesis deals with the use of PSO in Plasma Field, the requirement analysis is very important. There are various kinds of requirements, which are discussed below for the PSO.

3.1 Particle Swarm Optimizer Input Requirements:

Before starting the project, there is lot of research and study work required. This input requirement not only includes the study of Particle Swarm Optimization but also the study of all the basic requirement of the project.

The sub requirements are as follows:

Input Requirement 1: Understanding of Optimization

1.1 Understanding of Local Optimization

1.2 Understanding of Global Optimization

Input Requirement 2: Study of Swarm Behavior

Input Requirement 3: Study of PSO Algorithm

Input Requirement 4: Study of Social Behavior

3.2 Particle Swarm Optimizer Topology Requirements:

To implement any system or algorithm in various environment or field, the various topologies should be required. There are mainly three-topology requirement in PSO.

Topology Requirement 1: Study of Circular topology (section 2.2.3)

Topology Requirement 2: Study of Star topology (section 2.2.3)

Topology Requirement 1: Study of Wheel topology (section 2.2.3)

3.3 PSO Implementation Requirements:

The PSO can be implemented in two types. The types are differentiated based on the type of solution to be produced. Based of local and global optimization there are two model of PSO: Lbest – local best model and Gbest – global best model.

Lbest Model: The Lbest model tries to prevent premature convergence by maintaining multiple attractors. A subset of particles is defined for each particle from which the local best particle is then selected.

Gbest Model: The Gbest model offers a faster rate of convergence at the expense of robustness. This model maintains only a single "best solution", called the global best particle, across all the particles in the swarm. This particle acts as an attractor, pulling all the particles towards it. Eventually all particles will converge to this position, so if it is not updated regularly, the swarm may converge prematurely.

Different variants of PSO's implementation are also required for analysis of all the variants to find the best one from all. Here there are mainly four variants of PSO's implementation is required.

Implementation Requirement 1: Inertia Weight PSO (constant)

Implementation Requirement 2: Inertia Weight PSO (variable)

Implementation Requirement 3: Constriction PSO

Implementation Requirement 4: Fully Informed PSO

3.4 Function Evaluation

Here the problems, which are to be optimized, are defined by the mathematical functions. These functions have relevance with the real world problems. These functions are evaluated on a single computer or at remote computer. Based on location of evaluator there are two requirements for the function evaluation:

Stand Alone Function Evaluation:

The function should be evaluated and optimized on a single computer. Mainly, when the function, which is to be optimized, is not very complex and doesn't

require much computation time than the function can be implemented and optimized on a same computer.

Remote Function Evaluation

Here the function, which is to be optimized, is evaluated at remote system. Mainly, very complex functions, which require large amount of computation time, are evaluated on large computers. To optimize these kinds of problems, there is an interface requirement to communicate with them and to transfer data to each other. In Plasma Physics, functions, which are to be optimized, are very complex and nonlinear in nature. As most of scientific and complex mathematical libraries are implemented in FORTRAN, it is desirable to use FORTRAN for the calculation of the fitness function which is to be optimized. Here FORTRAN is used as an implementation platform for the evaluation of the function. These evaluation functions are called by the C server, which communicates with optimization algorithm (in Java). The FORTRAN evaluator gets each particle's position from the C server and returns back the fitness value (see Fig. 3.1). The fitness value specifies the goodness of the particle's position in the search space or how it fits in the search space.

Fig. 3.1 C and FORTRAN interaction

3.5 Behavior Modeling in VR – Active Animation

As the project is virtual reality enabled, there is requirement of the modeling of the PSO in the virtual environment. The Virtual Reality environment can be simulated using JAVA 3D technology, which provides rich set of libraries for the implementation of various 3D graphics rendering, modeling and visualizing. Here, the PSO is used to model the behavior the molecular dynamics, so for the better analysis and understanding of the behavior of the convergence and performance of

the algorithm, it is necessary to use JAVA3D or any other Virtual Reality Modeling Language.

The behavior of the molecular dynamics is like an animation in which number of particles moving and colliding with each other. So the active animation of the modeling is required. The active animation of the molecular behavior can be used to plot convergence graph of the particles.

3.6 Virtual Reality Requirement Specification

There are two types of requirement to create virtual worlds. These are VR modeling requirement and interaction requirement.

VR Modeling Requirement:

In this requirement, there is a basic need of Virtual Space representation. The virtual space is also known as virtual universe, in which different visual objects (particle) are displayed and rendered.

If the problem, which is solved by PSO is 3-dimentional then the simulation of the PSO algorithm behavior should be displayed using the VR modeling.

Interaction Requirement:

These requirements are related to interaction associated with the virtual worlds. These requirements are of Instantaneous type. For each virtual world, there is a need of interaction by which virtual worlds can be navigated. In this, there is need of rotational view in which the virtual worlds are rotated as user requirement for better visualization and understanding of the system. Navigation system is also required in which the virtual worlds objects can be positioned, scaled etc… by VR devices. The viewing system is also required in which from different views the virtual worlds can be analyzed and displayed.

4. System Design

System design is the first stage in which the basic approach to solving the problem is selected. During system designing the overall structure and style are decided.

4.1 Use Case Diagram

A use case is a set of scenarios that describing an interaction between user and a system. A use case diagram displays the relationship among actors and use cases. Following shows the use case diagram of the system which is to be made(see Figure 4.1).

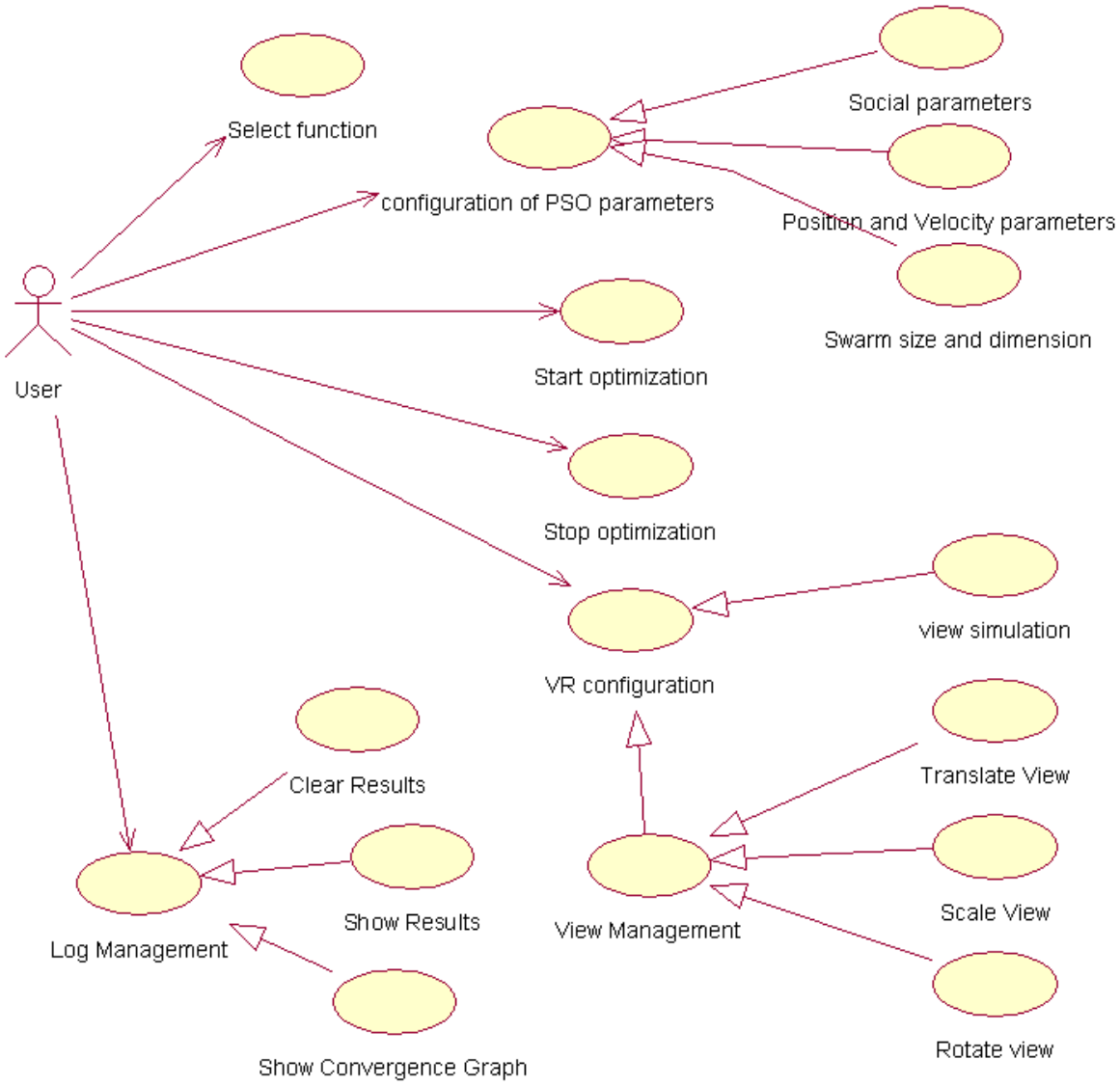

Figure 4.1 Use Case Diagram

Here user interacts with the system. At start, user can specify the function, which is to be optimized. After that, user configures the parameters of PSO. The user is allowed to configure following parameters: (1) Social parameters and (2) particles position and velocity ranges within which particle can move, and (3) Swarm size (or number of particles) and dimension.

After validation of input parameters, user can start the optimization process with the selection of various kinds of PSO algorithms. User can stop the optimization whenever he desire. If the problem is of 3^{rd} degree equation (dimension = 3) user can see the simulation of the PSO behavior in virtual world created using Java3d. When the simulation is shown, user can interact with it using mouse to navigate the virtual world, in which virtual worlds can be rotated, scaled and translated for better visualization and understanding purpose.

User can see the solutions and other results after execution of the optimization process using Log management system. Here, user can compare the results with other results also for identifying better solution, configuration parameters and algorithm. In this Log management, user can see the convergence plot of the optimization process, which displays the convergence graph of PSO, means how the PSO finds the best solution step by step.

4.2 Class Diagram

Class diagrams are used to describe the types of objects in a system and their relationships. To implement complete Particle Swarm Optimizer, six classes are used here. In PSO each class communicates with some class to achieve the desired result. The class diagram is shown in Figure 4.2.

The Particle class contains the basic properties of particles. These properties includes dimensions of the search space, position of particle, velocity of particle, particle's current fitness and initial fitness, number of neighbors and their id, best position in the history, and group best position.

The main heart of the whole system is defined in the ParticleSwarmOptimizer class. This class defines various configuration parameters of the PSO. This includes self confidence parameter C_1 and group confidence parameter C_2, randomness

parameters Si_1 and Si_2. It also defines maximum number of times the PSO is iterated, target fitness which is to be achieved, achieved fitness which is obtained by PSO, best solution to the problem and function which is to be optimized. When optimization starts, createSwarm() initializes the particles in the search space with random values to their position and velocity using getRandomPosition() and getRandomVelocity() function. After that, the topology is set with specified number of neighbors using setTopology().

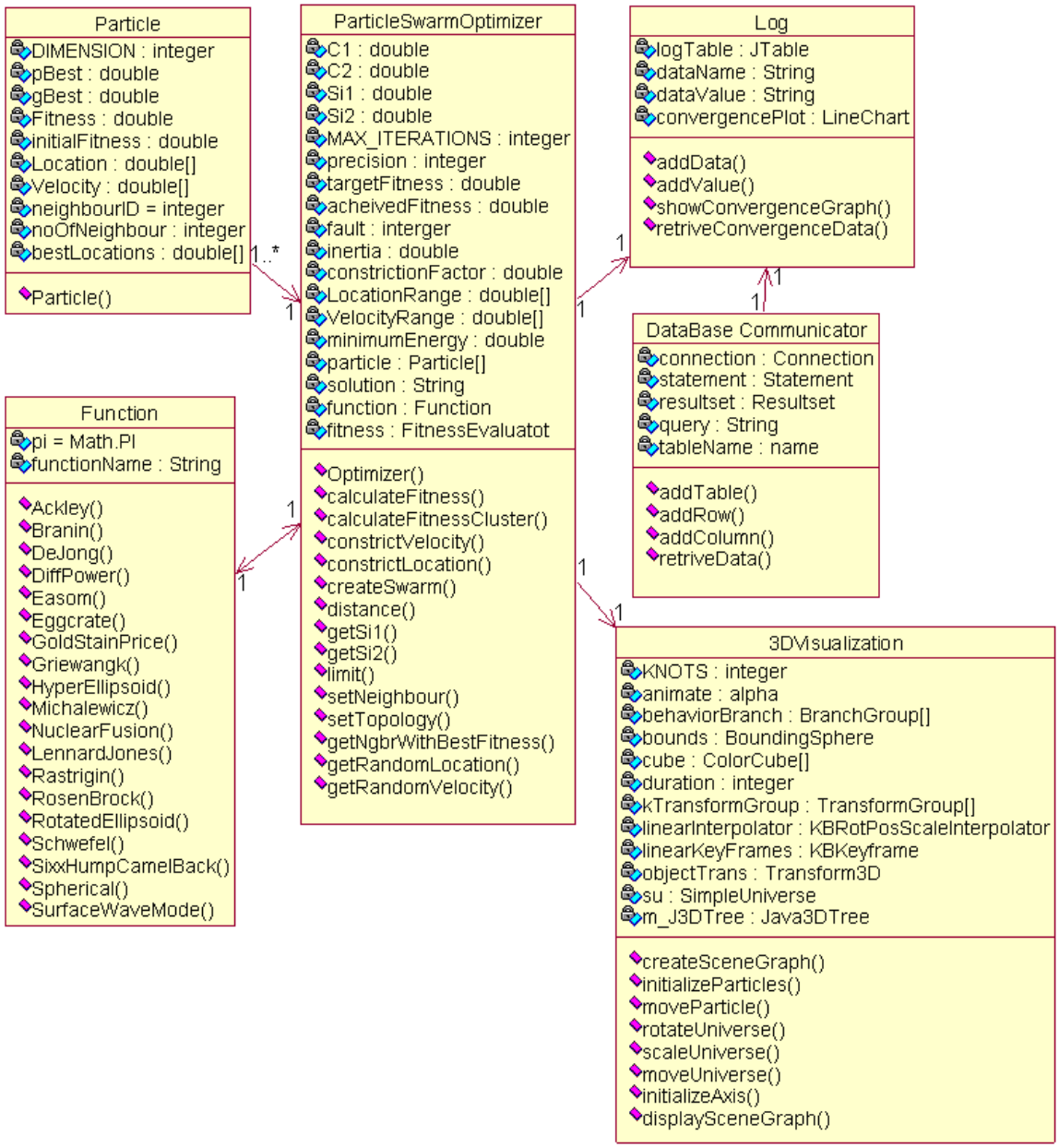

Figure 4.2 Class Diagram

The calculateFitness() function is used to calculate fitness of each particle. To identify the best neighbor in the group getNgbrWithBestFitness() is used. This class interacts with Particle to get and set particles position, velocity, fitness, neighborhood topology, its best value and group best value.

The Function class is used as a function library in which different synthetic benchmark functions are implemented. These class also contain some real world problems like SurfaceWaveMode() is used to define problem of surface wave mode differentiation, NuclearFusion() is used to define problem related to parameter selection problem of fusion reactor, LennardJones() defines the problem of atomic cluster optimization.

The Log class is used to store the results of various functions after their optimization. It also includes the facility of showing Convergence plot which is provided by showConvergenceGraph(), in which it interacts with DataBase communicator class to retrieve the convergence values of the algorithm for each function, which stores different algorithms results.

The 3DVisualization class is used to display the simulation of the PSO behavior. This class uses various objects of Java3D to create virtual world. The class initializes the particles in the virtual world. It than simulate the particles using movePartricle(). The position parameters of each particles' are obtained by interacting with ParticleSwarmOptimizer class. The virtual universe can be navigated using rotateuniverse() for rotation, scaleUniverse() for scaling and moveUniverse() for translation of universe. The createSceneGraph() is used to create the scene graph structure of the virtual world. The displaySceneGraph() function shows the tree like structure of the virtual world, which shows the structure of visual and non visual objects used in creating virtual world.

4.3 Sequence Diagram

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. A sequence

diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects (see Figure 4.3 (sequence diagram of PSO system)).

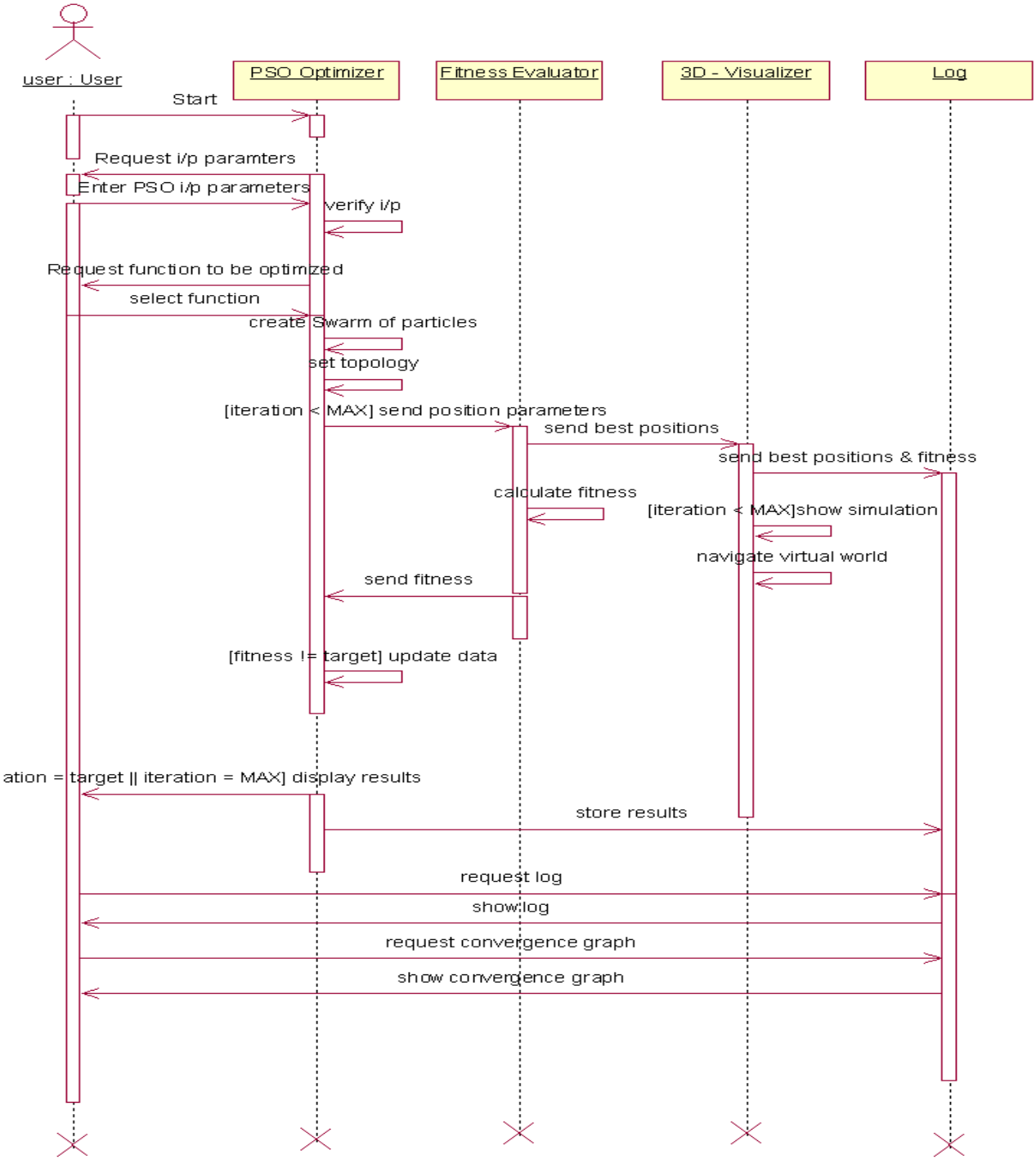

Figure 4.3 Sequence Diagram

The user first starts the PSO system. Than PSO system requests the input parameters and function to be optimized from user. The user responses with configuring PSO parameters and giving function. After that, user starts the optimization process, which is followed by the optimization process of the PSO optimizer.

The PSO optimizer sends the parameters to the fitness evaluator to calculate the fitness. While optimization runs, PSO optimizer sends particles best position to the 3D visualizer when the function is of 3rd degree. The 3D visualizer simulates the behavior of the PSO in 3D world. The 3D world can be navigated by user using mouse. PSO also sends best value calculated in each iteration to the Log, where the results are stored. After execution of the optimization process, the results are displayed to the user. The user can request to plot the convergence graph of the PSO to Log. The Log responds by displaying the convergence plot of the desired function with desired PSO variant.

4.4 Collaboration Diagram

UML collaboration diagrams illustrate the relationship and interaction between system objects. The collaboration diagram illustrates messages being send between classes and objects (instances). For each system operation the collaboration is defined. Here as the main operation is of optimization, only one collaboration diagram exists.

There are four main objects defined for designing Particle Swarm Optimizer system (see Figure 4.4). These objects are as follows: PSO Optimizer, Fitness Evaluator, 3D Visualizer and Log. Initially user starts the optimization process by interacting with PSO Optimizer, which requests input parameters and problem to be optimized from user. After that, PSO optimizer starts the optimization process. While optimization, it interact with Fitness Evaluator and Log. When the function is of 3rd degree, it also interacts with 3D visualizer.

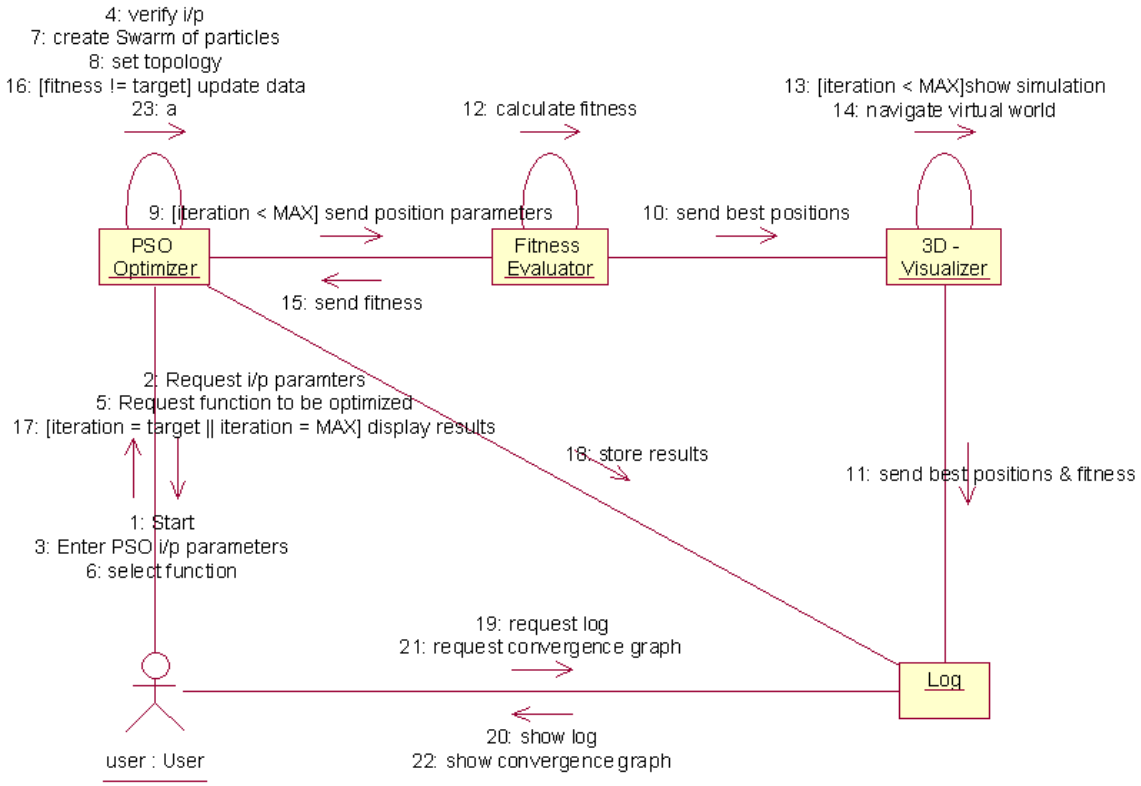

Figure 4.4 Collaboration Diagram

PSO optimizer updates the information of particles depending on the fitness values obtained from the fitness evaluator. It than calculates best value at each iteration and sends it to Log and 3D visualizer (in case of 3rd degree problem). The 3D visualizer simulates the behavior of PSO algorithm for certain function. The Log stores the best values over time and displays the results to the user. User can show the convergence graph of the PSO by interacting with Log, which store the results and PSO's convergence parameters values.

4.5 State chart Diagram

State chart diagrams provide a way to model the workflow of a system process. We can also use state chart diagrams to model code-specific information such as a class operation. State chart diagrams are very similar to a flowchart because we can model a workflow from state to state. The state chart diagrams are state centric. The state chart is best suited for the discrete stages of an object's life time.

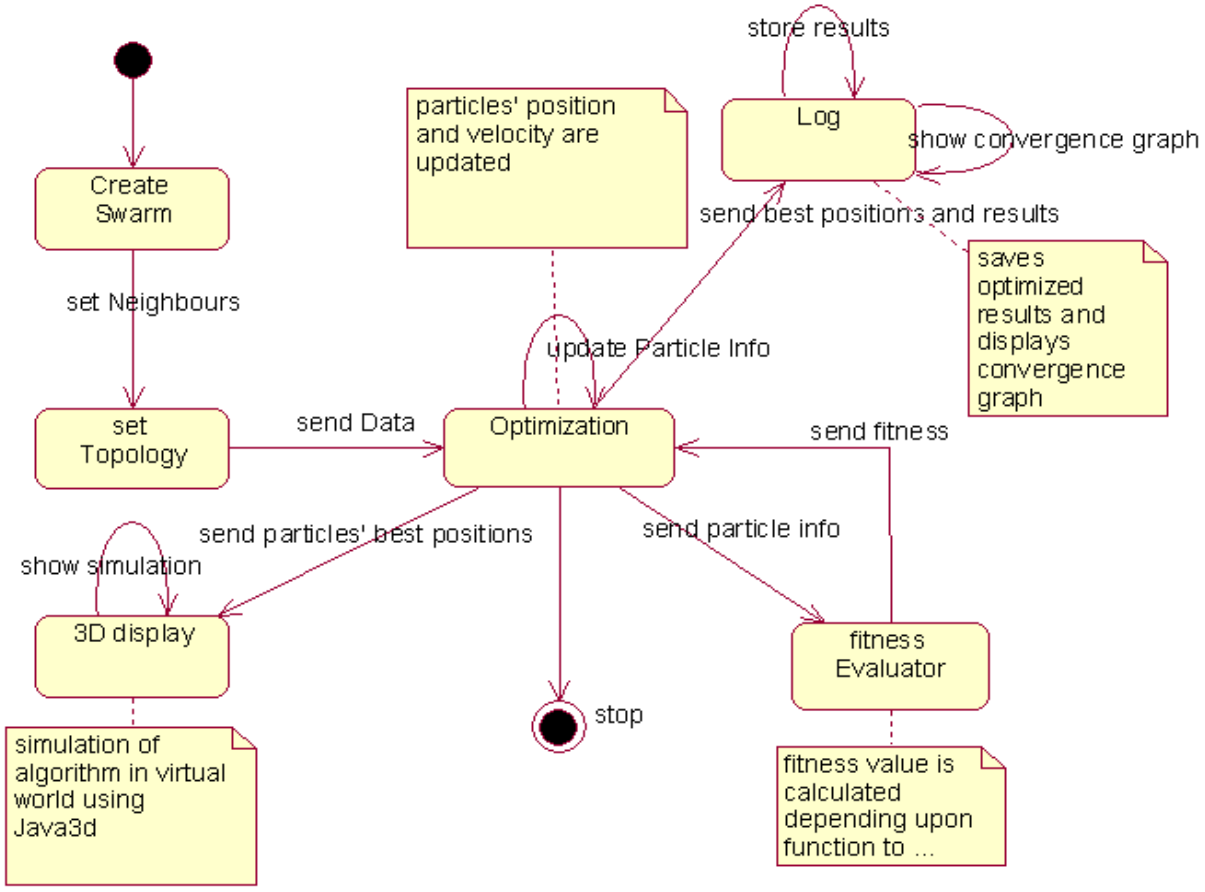

Figure 4.5 State chart diagram

When PSO starts, it first enters into "createSwarm" state, where the swarm is initialized in which numbers of particles are initialized with random position and velocities. After that it enters into "set Topology" state, where groups of particles are created. Than the algorithms optimization process starts in optimization state. During optimization it interacts with 3D display, fitness evaluator and log. In "Fitness Evaluator" state, the fitness is calculated according to the function which is to be optimized by getting input values from optimizer. After obtaining the fitness values from fitness evaluator, optimizer updates particles information and sends particles best information to the 3D display, where particles are displayed and Log where they are stored. In "3D display" state, the simulation of PSO is displayed and can be navigated using mouse. In "Log" state, the results of various functions are shown with respect to the configuration of parameters and the type of PSO. The convergence plot of the algorithm for optimized function can be shown in this state.

5. Performance Analysis

This chapter includes analysis of parameter selection of PSO. Optimum values of parameters of PSO are selected by analyzing the performance of PSO for range of values of all parameters. It also investigates analysis of different variants PSO, which are explained in Chapter 2.

5.1 Methodology

This investigates the performance of PSO using several benchmark functions.
These benchmark functions are synthetic functions, which are used to test and compare various algorithms. Although these functions may not necessarily give an accurate indication of the performance of an algorithm on real world problems, they can be used to investigate certain aspects of the algorithm under construction.

Below certain benchmark functions are described briefly. Note that all the functions described here are synthetic benchmark functions.

Ackley:

This function is originally proposed by Ackley [8], and generalized by Back, has an exponential term that covers its surface with numerous local minima (see Appendix D). In order to obtain good results for this function, the search strategy must combine the exploratory and exploitative components efficiently. The function is defined as below:

$$f_{Ack}(\mathbf{x}) = 20 + e - 20exp\left(-0.2\sqrt{\frac{1}{p}\sum_{i=1}^{p} x_i^2}\right) - exp\left(\frac{1}{p}\sum_{i=1}^{p} cos(2\pi x_i)\right)$$

Where, $x_i \in [-30, 30]$.

The function has global minima at $\mathbf{x}^* = (0, 0, \ldots, 0)$ at $f_{Ack}(\mathbf{x}^*) = 0$.

Here various types of PSO are used to optimize the Ackley function. The results for this are shown in Table 2.1.

Spherical:

This [8] is a very simple function (see Appendix D). It is used in the development of theory of evolutionary strategies. The function is defined as:

$$f_{Sph}(\mathbf{x}) = \sum_{i=1}^{p} x_i^2 \quad \text{, where } x_i \in [-5.12, 5.12]$$

The function has minima at $\mathbf{x}^* = (0, 0, \ldots, 0)$, where $f_{Sph}(\mathbf{x}^*) = 0$

Rastrigin:

This [8] function is constructed from sphere adding a modular term $\alpha \cdot cos(2\pi x_i)$. Its contour is made up of a large number of local minima whose value increases with the distance to the global minimum (see Appendix D). It is defined as:

$$f_{Ras}(\mathbf{x}) = 10p + \sum_{i=1}^{p} (x_i^2 - 10\cos(2\pi x_i))$$

$$\text{where } x_i \in [-5.12, 5.12]$$

The function has global minimum at $\mathbf{x}^* = (0, 0, \ldots, 0)$ where $f_{Ras}(\mathbf{x}^*) = 0$

Griewangk:

This [8] function has a product term that introduces interdependence among the variable. The aim is the failure of the technique that optimizes each variable independently. The optima of Griewangk function are regularly distributed (see Appendix D).

The function is defined as:

$$f_{Gri}(\mathbf{x}) = 1 + \sum_{i=1}^{p} \frac{x_i^2}{4000} - \prod_{i=1}^{p} cos\left(\frac{x_i}{\sqrt{i}}\right)$$

$$\text{where } x_i \in [-600, 600] .$$

The function has global minimum at $\mathbf{x}^*(0, 0, \ldots, 0)$ where $f_{Gri}(\mathbf{x}^*) = 0$.

Table 5.1 lists the lists the parameter settings for the functions in the benchmark suite. Note that all functions were tested using 25 particles (swarm size). All functions are solved in 5-dimensional search space.

Table 5.1 Parameter settings for benchmark functions

Function	**No. of Particles**	**Dimension**	**Domain**
Ackley	25	5	30
Spherical	25	5	5.12
Rastrigin	25	5	5.12
Griewangk	25	5	600

The PSO algorithm was initialized so that initial particles in the swarm were distributed thought the search space using a random number generator. The value of each dimension j of particle i was thus sampled from a distribution of the form

$$S_{i,j} \sim U(-d, d)$$

Where d is the appropriate domain value for the function under consideration. The domain value specifies the range of the particle's search area (the area within which particle can move).

The position and velocity of each particle are selected randomly using the random function of the Math library of JAVA. The function is Math.nextDouble() which generates random numbers from 0.0 to 1.0 in double format. The random numbers generated from this function are then scaled for obtaining desired random values in the desired domain.

For all functions specified above are optimized using various PSO. Here the total number of "iterations" required and number of "faults" occurred while optimizing problems are used to compare performance i.e. convergence speed and robustness of PSO. "Iterations" is the total number of iterations required to find the optimum value of the function. When particle crosses boundary of search space, then it is known as fault.

5.2 Optimum Configuration of Parameters of PSO

The standard PSO (Constant Inertia Weight PSO with inertia weight = 1.0) is used to find the optimal values for parameters of PSO. For that, PSO is used to optimize 3-dimensional Rastrigin function which is unconstrained benchmark function. Following are the parameters of PSO, which are defined in algorithm: Number of Particles (Swarm size), Velocity Limits, Self Confidence Parameter (C_1), Group Confidence Parameter (C_2), and Inertia Weight (w).

To select optimum parameters of PSO, The PSO is used to optimize the Rastrigin function in which different parameters values are varying over range of values. For each configuration of parameters, number of iterations and faults required are compared with each other.

Following discusses the steps of fixing the values for the parameters of PSO:

Step 1: Fixing Number of Particles (Swarm Size)

To fix the swarm size, PSO is executed with the initial values of $C_1=C_2=2.0$ and $Si_1=Si_2=w=1.0$, with $V_{max} = S_{max} = 5.12$. And the swarm size is varied from 5 to 35 at a step of 5. Figure 5.1 shows the graph of iterations, faults vs. swarm size. From the graph, when swarm size is 25, PSO converges with less number of iterations than other values of swarm size (or it converges faster than others).

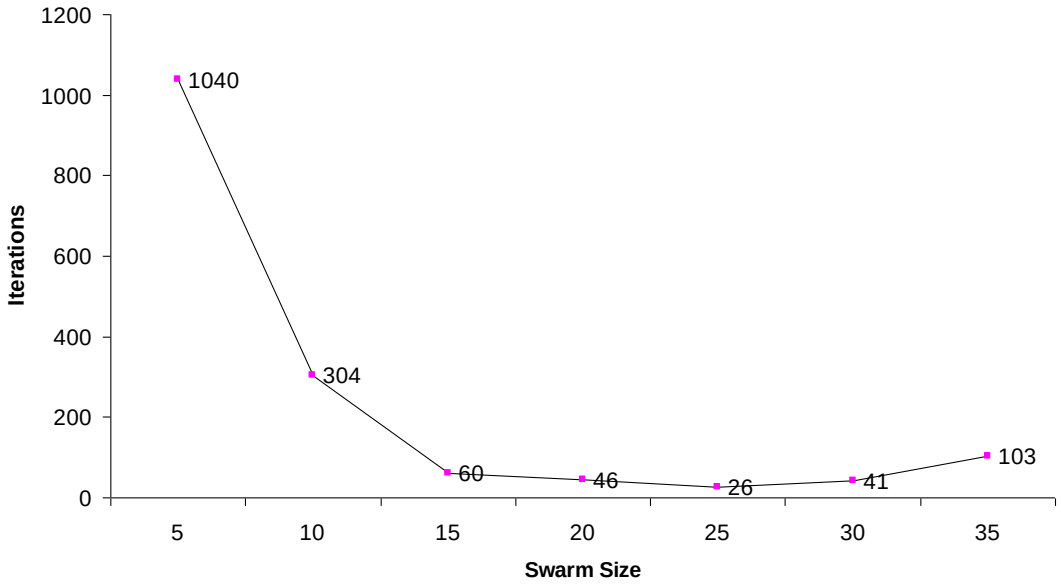

Figure 5.1 iterations vs. Swarm size

Step 2: Fixing Velocity Range (V_{max})

To fix the velocity range, PSO is executed with the initial values of $C_1=C_2=2.0$ and $Si_1=Si_2=w=1.0$, with swarm size = 25. And the velocity range is varied from S_{max} to $4*S_{max}$ at a step of S_{max}. Figure 5.2 shows the graph of iterations, faults vs. velocity range. From the graph, when velocity range is same as position range, PSO converges faster than others.

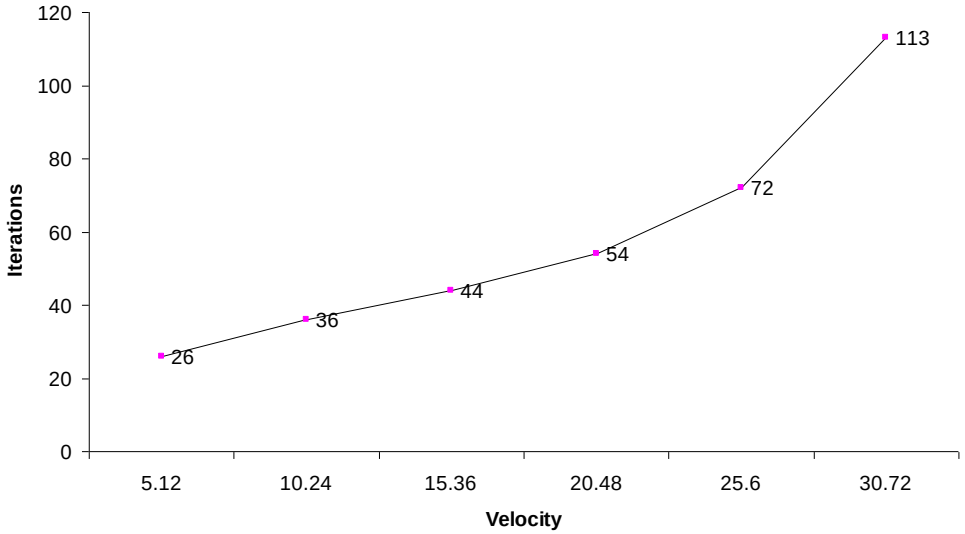

Figure 5.2 iterations vs. velocity

Step 3: Fixing Self Confidence Parameter (C_1)

To fix the self confidence parameter C_1, PSO is executed with the initial values of C_2=2.0 and Si_1=Si_2=w=1.0, with swarm size = 25 and V_{max} = S_{max}. And the C_1 is varied from 0.5 to 2.5 at a step of 0.5. Figure 5.3 shows the graph of iterations, faults vs. C_1. From the graph, when C_1 is 2.0, PSO converges faster than others.

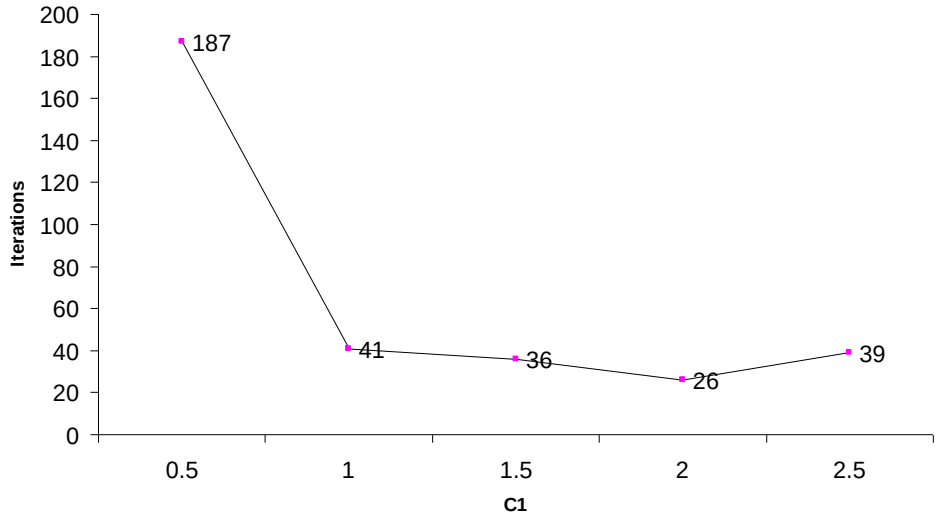

Figure 5.3 iterations vs. C_1

Step 4: Fixing Group Confidence Parameter (C_2)

To fix the group confidence parameter C_2, PSO is executed with the initial values of C_1=2.0 and Si_1=Si_2=w=1.0, with swarm size = 25 and V_{max} = S_{max}. And the C_2 is varied from 0.5 to 2.5 at a step of 0.5. Figure 5.4 shows the graph of iterations, faults vs. C_2. From the graph, when C_2 is 2.0, PSO converges faster than others.

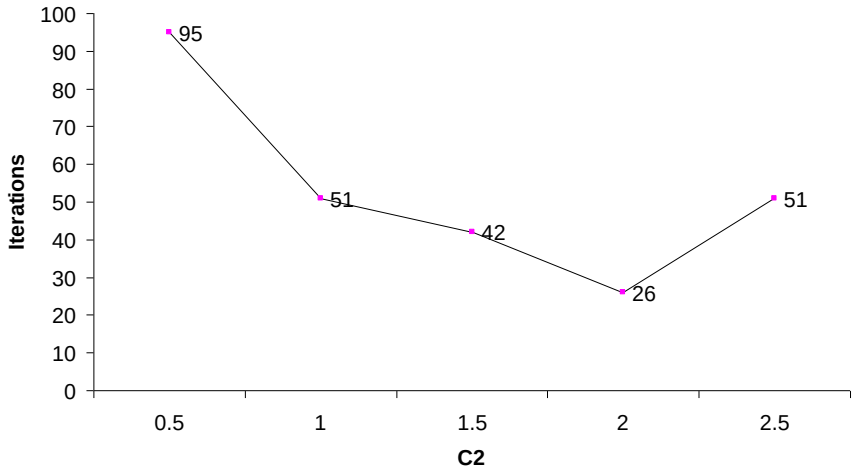

Figure 5.4 iterations vs. C_2

Step 5: Fixing Inertia weight (w)

To fix the group confidence parameter C_2, PSO is executed with the initial values of C_1=C_2=2.0 and Si_1=Si_2=1.0, with swarm size = 25 and V_{max} = S_{max}. And the inertia weight w is varied from 0.5 to 1.5 at a step of 0.1. When w is 0.8, 0.9 or 1.0, PSO converges faster than others (Figure 5.5).

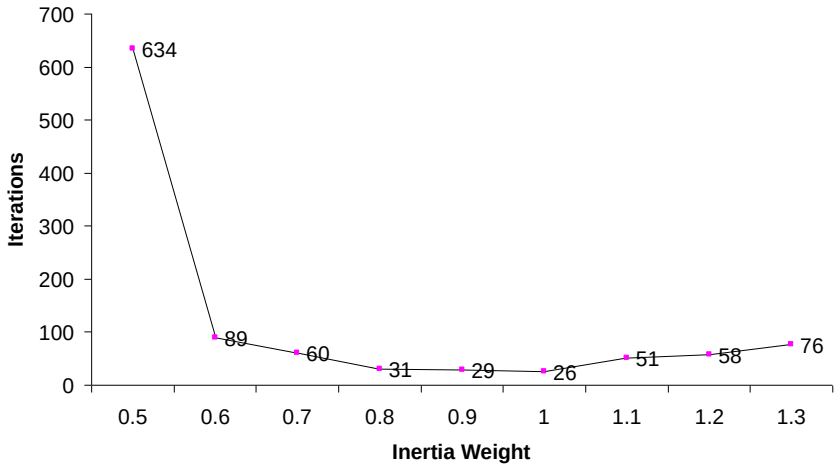

Figure 5.5 Iterations vs Inertia Weight

Table 5.2 shows the optimum values for parameters of PSO, which are derive from above results for faster convergence of algorithm.

Table 5.2 Optimum values of parameters of PSO

Parameter	Optimum Value
Number of Particles (swarm size)	20 to 30
Velocity Range (V_{max})	S_{max} (position range)
Self confidence (C_1)	2.0
Group confidence (C_2)	2.0
Inertia Weight (w)	0.8,0.9,1.0

5.3 Performance analysis of variants of PSO

Synthetic benchmark functions, which are described in section 5.1, are used to analyze the performance of variants of PSO. Here all defined functions are optimized using variants of PSO. After optimizing all the functions, number of iterations and number of faults are used for the comparison. When any algorithm optimizes function with less number of iterations and less number of faults compare to others, it can be say that the algorithm performs better than the others. When faults are higher, it shows that the particles are crossing the boundary many times and it may possible that the solution may not be found within defined number of iterations. From the update equations of each variants of PSO, each update equation requires number of additions and multiplications, which affect the convergence speed of the algorithm. Table 5.3 shows the number of additions and multiplication requires for each variant of PSO. N is Group size (or neighborhood size), the number of individuals in group.

Table 5.3 Optimum values of parameters of PSO

PSO variant	No. of Additions	No. of Multiplications
Standard PSO	4	4
Inertia Weight PSO	4	5
Constriction PSO	4	5
Fully Informed PSO	3*N + 1	6*N + 1

Following Table5.4 shows the results of various PSO when they are used to solve various benchmark functions. Here the results are in terms of number of faults occurred and number of iterations required for solving the function. Here four types of PSO is used which are explained in chapter 2.

All functions are of 5 dimensional with swarm size is 25 and neighborhood topology is circular with 2 nearest neighbors, means there are 2 left and 2 right neighbors for each individual. The social parameters C1 and C2 = 2.05 and the domain size for all functions is defined in Table5.1.

Table 5.4 Results of variants of PSO for different benchmark functions (No. of Iterations)

PSO Type	Standard PSO		Inertia weight PSO		Constriction PSO		FIPSO	
Function	Iter.	Faults	Iter.	Faults	Iter.	Faults	Iter.	Faults
Spherical	277	19465	126	4027	116	2948	12	30
Rastrigin	-	-	180	6757	170	3490	19	35
Griewangk	38	2689	12	864	128	5688	108	76
Ackley	-	-	491	13001	664	7006	30	13

Table 5.5 shows the results of variants of PSO for different benchmark functions in terms of number of additions and multiplications requires for the optimization of the defined functions.

Table 5.5 Results of variants of PSO for different benchmark functions (No. of Evaluations)

PSO Type	Standard PSO		Inertia weight PSO		Constriction PSO		FIPSO	
Function	Add.	Mult.	Add.	Mult.	Add.	Mult.	Add.	Mult.
Spherical	1108	1108	504	630	464	580	192	324
Rastrigin	-	-	720	900	680	850	301	513
Griewangk	152	152	48	60	512	640	1728	2808
Ackley	-	-	1964	2455	2656	3220	480	810

From the above results, when the sphere function is optimized using different variants of PSO, the FIPSO gives the best results among them. It optimizes the function in 12 iterations means it takes 192 additions and 324 multiplications to find the optimum value. The inertia weight PSO and Constriction PSO gives almost identical performance but less compare to FIPSO. The Standard PSO gives very poor performance than others. For Rastrigin and Ackley function also above is true.

In the case of Griewangk, the results are totally different than the above one. The inertial weight PSO optimizes the function within 12 iterations, which require 48 additions and 60 multiplications and Standard PSO optimized with 152 additions and multiplications. Constriction PSO requires 512 additions and 648 multiplications, while FIPSO requires 1728 additions and 2808 multiplications. So here, the Inertia weight PSO gives better performance than the others.

Form the result, it can be said that the FIPSO is better to use when the function complexity is not so high. In the case of higher complexity of function, there is always a trade off between the selection of optimization algorithms.

6. Application of PSO

This section investigates the application of Particle Swarm Optimization in the area of plasma physics. Here PSO is used for the optimum parameter selection for Nuclear Fusion Reactors and Atomic cluster optimization [13] problem. Virtual Reality enabled PSO is useful for modeling the behavior of the convergence of PSO. It shows how the optimization is achieved by PSO graphically.

6.1 Atomic cluster optimization problem

The problem of minimizing the potential energy function of cluster of atoms is generally known as molecular conformation problem. Cluster sizes can range from a few atoms to several atoms. The determination of global minima of these energy functions is of particular interest to researchers in chemistry, biology, physics and optimization methods. One part of these, the Lennard-Jones potential function is concerned with determining the lowest-energy configuration of a cluster of neutral atoms interacting via the Lennard Jones potential. This turns out to be very difficult problem to solve since the number of local minima has been estimated to increase exponentially with the number of atoms. Studies have shown that at N = 13 there are 988 local minima and for N = 98 the number grows to the order of 10^{40}. The function to be minimized is the total energy (in reduced units) of the Lennard-Jones cluster as computer from

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(r_{ij}^{-12} - r_{ij}^{-6} \right)$$

Where N = number of atoms in cluster

$r_{i,j}$ = distance separating atom i and atom j

It serves as a reasonably accurate mathematical model of a real physical system, namely that of low-temperature micro clusters of heavy rare-gas atoms such as argon, krypton and xenon.

The problem is optimized by various types of PSO, which are discussed earlier. PSO is executed for 10000 iterations and the minimum cluster energy is calculated for

different sizes of number of atoms. Following parameters are selected in the execution of PSO.

Here N is varied from 5 to 13, for which the minimum cluster energy is defined as in Table 6.1. Swarm size is 25. Atoms velocity values can range from -0.2 to 0.2 and the C1 and C2 are selected as 2.0. The neighborhood size is 5.Table 6.1 shows the results of various PSO for the cluster energy minimization problem.

Table 6.1 Results of Atomic cluster optimization

	Cluster Energy	Inertia Weight PSO				Fully Informed PSO			
N		Fault	Iter.	Add	Mult	Fault	Iter	Add	Mult
5	-9.90	996	47	188	235	28	2	144	256
6	-9.90	1330	52	212	265	81	4	64	104
7	-9.90	980	69	276	345	58	4	64	104
8	-9.90	1289	17	68	85	45	19	304	496
9	-9.90	1082	54	216	270	91	10	160	260
10	-9.90	2075	65	260	325	51	15	240	390
11	-9.90	2501	69	276	345	62	27	432	702
12	-9.90	3213	56	224	280	71	15	240	390
13	-9.90	2697	32	128	160	64	21	336	546

Here, for larger sizes of cluster (>9) the inertia weight PSO gives better performance than the FIPSO (less number of additions and multiplications). When cluster size is smaller (<10), the FIPSO gives better performance than the Inertia weight PSO (less number of additions and multiplications). So, it can be say that various PSO can be used effectively for this kind of atomic cluster optimizations problems.

The calculations show that the PSO is indeed an effective optimization method. The potential energy surfaces for these problems are known to contain large numbers of local minima, often very close to the global minima for certain values of N. Hence it is not unexpected to find that many of the runs converge to one of these local minima.

6.2 Optimum Parameter selection for Nuclear Fusion Reactors for minimum cost per unit energy

Nuclear Fusion Reactors are used generate power. Here donut shaped reactor is shown in Fig. 6.1 which is known as tokomak. Here the cavity of tokomak contains plasma, which is used to generate energy.

The cost of energy generated using this kind of nuclear reactors is depends on the three parameters of the reactor: Reactor Radius, Reactor width and Reactor temperature. These parameters are shown in the cross section of tokomak (Figure 6.2)

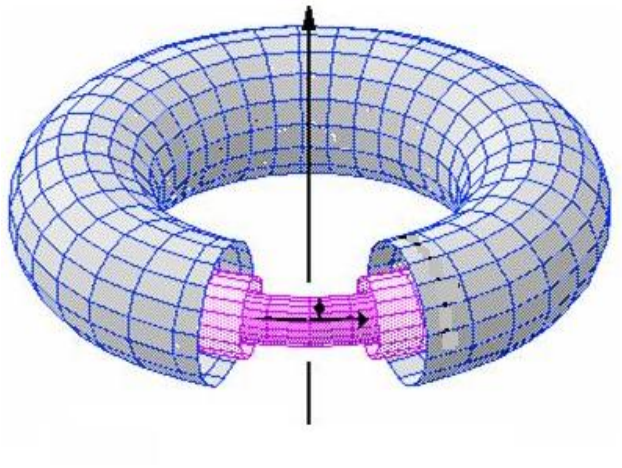

Fig 6.1 Tokomak

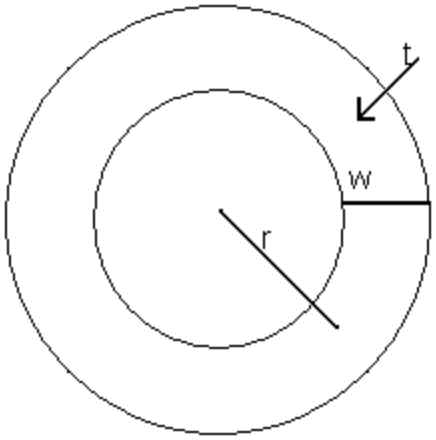

Fig 6.2 cross-section of reactor

The cost per unit energy is calculated using following function.

$$f(r,w,t) \quad 10^6 * t^2 * (C1 * (r^2 \quad \frac{2rw}{3} \quad \frac{w^2}{6}) \quad C_2 * (C_4 \quad \frac{C_4^2}{r^2 \quad C_4^2})) \quad C_3 * (\frac{1}{2} \quad \frac{r}{w}) * (10^3 t)^{1.46}$$

where r = reactor radius

w = reactor width

t = reactor temperature

C1 = C2 = C3 = C4 = 1, generally between 0 and 2.

Here the aim is to achieve the minimum cost per unit energy for values of r, w and t.

Following table shows the results of r, w, t and the cost per unit energy obtained by using variants of PSO.

Table 6.2 Results of Nuclear Fusion Reactor Parameters optimum values for minimum cost/kw/hr

	Radius(meter)	Width(meter)	Temperature(M 'C)	Cost/KW/hr
Inertia Weight	1.581	0.494	480.1	$9.98
Constriction	1.632	0.22	543.6	$9.974
FIPSO	1.78	0.32	513.8	$9.998

From the above table, it can be say that, Constriction PSO gives minimum cost of 9.974 Rupees per kilo watt energy generated by nuclear fusion reactor, with values of radius, width and temperature as 1.632meter, 0.22meter, and 543.6 million 'C.

6.3 Differentiation and Distribution of Surface wave modes on a Dielectric Coated Cylinder

Plasmas that are excited by propagation of electromagnetic surface waves are called surface-wave-sustained. The surface wave mode allows to generate uniform high-frequency-excited plasmas in volumes whose lateral dimensions extends over several wavelengths of the electromagnetic wave, e.g. for microwaves of 2.45 GHz in vacuum the wavelength amounts to 12.2 cm. When electromagnetic waves are excited on a dielectric coated cylinder it produces surface waves (see Figure 6.6). These surface waves' modes are of interest in the design of conformal antenna and in the control of scattering properties of metallic cylindrical structures that have been coated with dielectric materials. The waves propagating along a curved surface with respect to features of their modal propagations constants are very important for the analysis. Here the analysis is performed on the data, which is generated using 3-d FDTD (Finite Difference Time Domain) method [12].

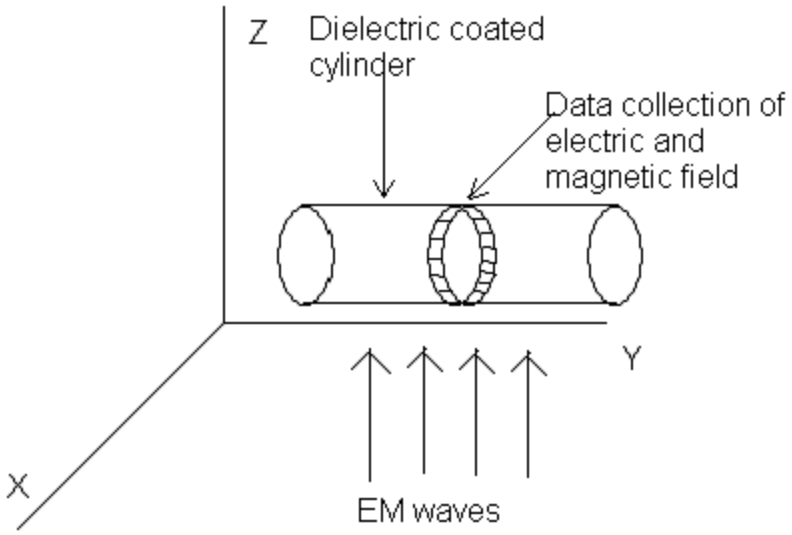

Figure 6.3 Dielectric coated cylinder excited by EM waves

FDTD is a popular computational electrodynamics modeling technique. As it is time domain method, solutions cover a wide frequency range. The FDTD method belongs in the general class of grid based differential time- domain numerical modeling methods. The equations are solved in leapfrog manner: the electric field is solved at given instant time, than the magnetic field is solved at the next instant time, and the process is repeated over and over time.

Using 3-d FDTD code, researchers have generated data, electric and magnetic field [19/Mr. Bhaskar], on the surface of a dielectric coated cylinder, which is being lit by a plane EM wave.

Researchers [19, Bhaskar] have generated a time series of electric and magnetic fields at different locations on the surface of the cylinder. The surface wave modes are characterized by amplitude, phase angle, damping coefficient, mode number and phase angle.

These characteristic parameters are calculated by optimizing following equation:

$$F(t,\theta) = \sum_{t}\sum_{\phi} \left(Ai * \cos(wt) * e^{-\alpha 1\theta} * \cos(\alpha 2*\theta + \Phi) - E_{(t,\theta)} \right)^2 \quad \ldots (6.1)$$

where Ai = Amplitude,
α_1 = Damping Coefficient,
α_2 = Mode number,
Φ = Phase Angle,
W = Omega = 18GHz

Here t is a time step, which specifies the data (electric field E_X and magnetic field E_y) collected using FDTD at t. Researchers had collected data for 1000 time steps. θ specifies the angle at which the data is to be collected from the surface wave. Researches had collected data from 342 different angles from the surface wave. Each θ value is used to extract components E_X and E_y. $E(t, \theta)$ is a matrix of data collected in terms of Ex and Ey corresponding to t and θ. Here as there are 1000 time steps and 342 angles, $E(t, \theta)$ consists of 1000 rows corresponding to time step and 685 columns (one is of time step, 342 of E_x and 342 of E_y). Here the aim is to find the values of Ai, α_1, α_2 and Φ such that the $F(t, \theta)$ gives the value 0.0. These values can be calculated by fitting the first term of the equation of the right hand side to the second term $E(t, \theta)$ of the equation of the right hand side. In other words, the analytical data is to be fitted with empirical data, for that values of Ai, α_1, α_2 and Φ are calculated.

The PSO is used to find these values of Ai, α_1, α_2 and Φ, such that the analytical data fits with empirical data. Table 6.4 shows the values of Ai, α_1, α_2 and Φ achieved by optimizing the equation (6.1) using PSO.

Table 6.3 Results of equation 6.1

Amplitude (Ai)	Damping Co-eff (α_1)	Mode Number(α_2)	Phase angle (Φ)	Achieved Fitness
894.8	6.2	20.1	1.6	-8403492.54
1033.7	2.8	22.5	1.2	-8204038.41
234.2	1.2	1.6	1.2	1940692.77
404.7	**1.4**	**9.8**	**1.5**	**-135660.77**
83.5	0.0	0.9	0.0	-744892.27

From these values of Ai, α_1, α_2 and Φ, the best value is achieved when amplitude I 404.7, damping coefficient is 1.4, mode number is 9.8 and phase angle is 1.5 with fitness value as -135660.77. These values are useful to differentiate the surface wave modes and in calculation of each distribution mode, which are useful in the design of conformal antennas and in the control of scattering properties of metallic cylindrical structures that have been coated with dielectric materials.

6.4 Modeling the behavior of PSO convergence using Virtual Reality

The Virtual Reality is used to create the virtual worlds, in which different visual objects are rendered and navigated through navigation systems like mouse, keyboard, hapitcs, etc… Here, virtual reality is used to model the behavior of the PSO. When PSO is used to optimize the 3rd degree nonlinear equation, the behavior of the PSO is modeled in the virtual world. The modeling of the PSO in virtual reality is useful for understanding the convergence of PSO and to analyze and understand behavior of the particles.

Here Java3D API is used to create virtual worlds. The Java3D is a freely available toolkit from SUN, which supports mouse and keyboard for interaction with virtual worlds. It has 255 C and Java functions which are used to implement virtual reality enable programs. Fig.6.4 shows the initial positions of the particles in a virtual world, which is created using Java3d. In virtual world, the particles are displayed using spheres. The Rastrigin function is used to model the behavior of the Inertia weight PSO. Here 3rd degree Rastrigin function, with $\pm V_{max} = 5.12$, with 30 particles is optimized.

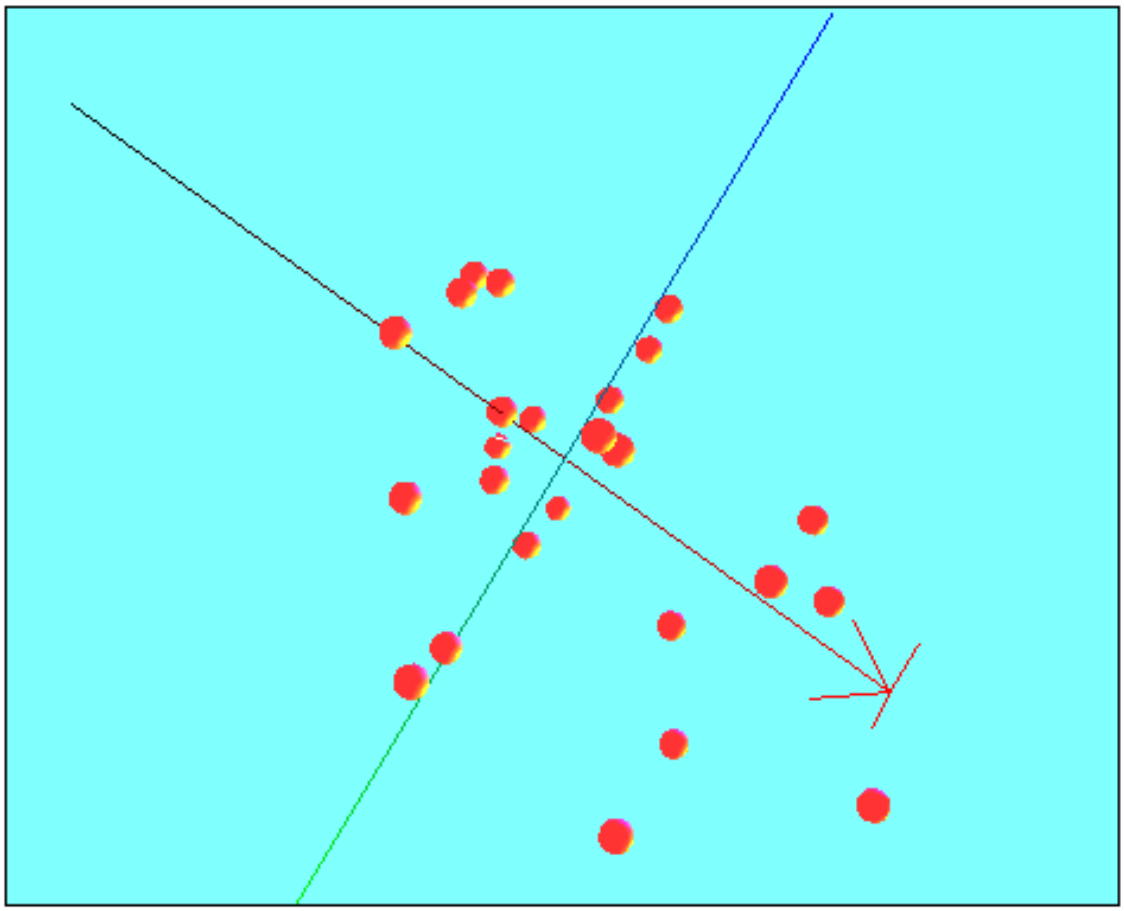

Figure 6.4 Initial Positions of Particles (Rastrigin Function)

The V_{max} is set to 5.12, so any particle will never cross the boundary of the ±5.12. The particles can freely move in the range of [-5.12, 5.12].

When PSO is running, the particles try to converge to the target value. This convergence of particles is simulated in the virtual world. For Rastrigin function, as algorithms runs, the particles try to come to origin. From the movement of the particles, which depends on the best value found by algorithm, PSO behavior can be analyzed. Figure 6.5 shows the positions of the particles at 607^{th} iteration. Note that the algorithms had founded the target value for some particles and some of them are converged to target value (0.0, 0.0, 0.0). Here some particles have not yet found the target, which are away from the origin and are displayed in the virtual world.

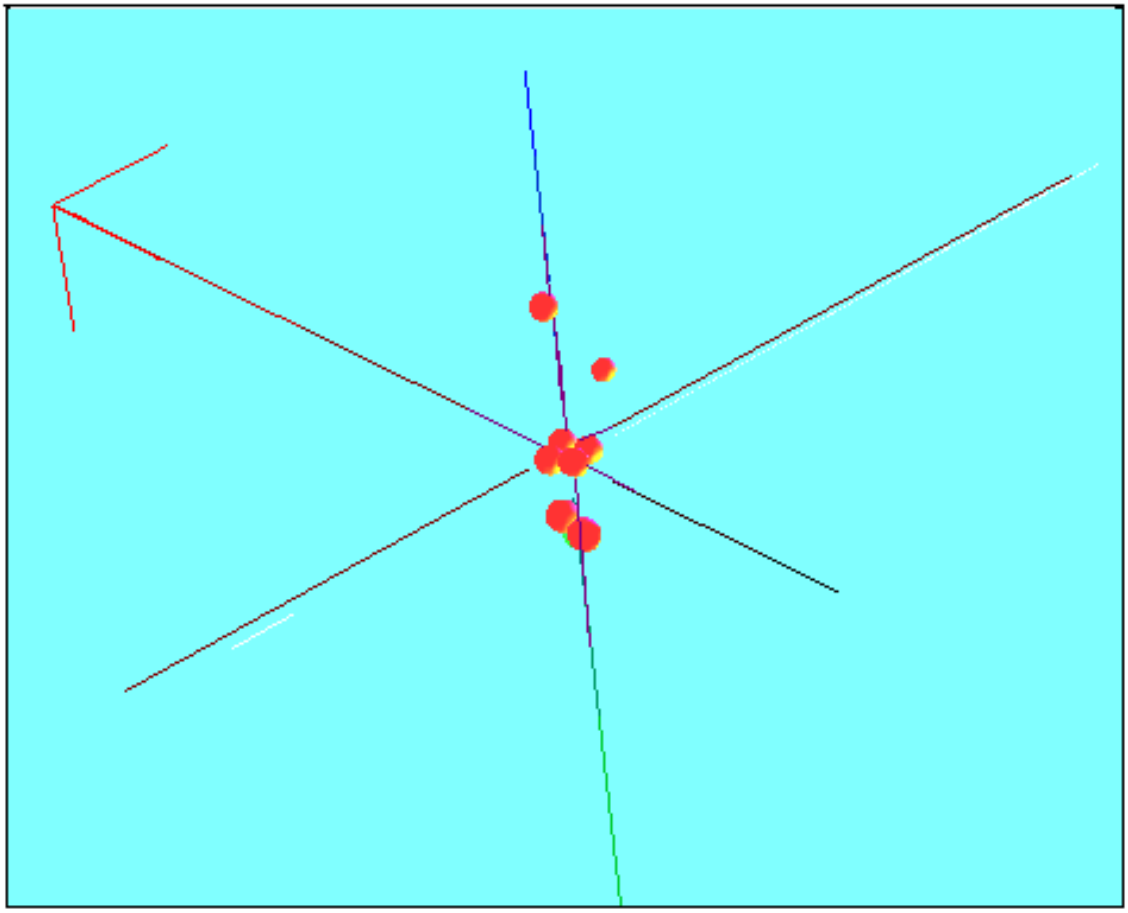

Figure 6.5 Positions of Particles (607th iteration)

At the end of the simulation, all the particles converge to the target value. Here, as Rastrigin function is used, all particles will converge to (0.0, 0.0, 0.0) (see Figure 6.6). Here three axis are shown, to understand the positions of the particles.
Here, the Virtual world can be rotated, scaled and translated for better view and understanding of PSO.

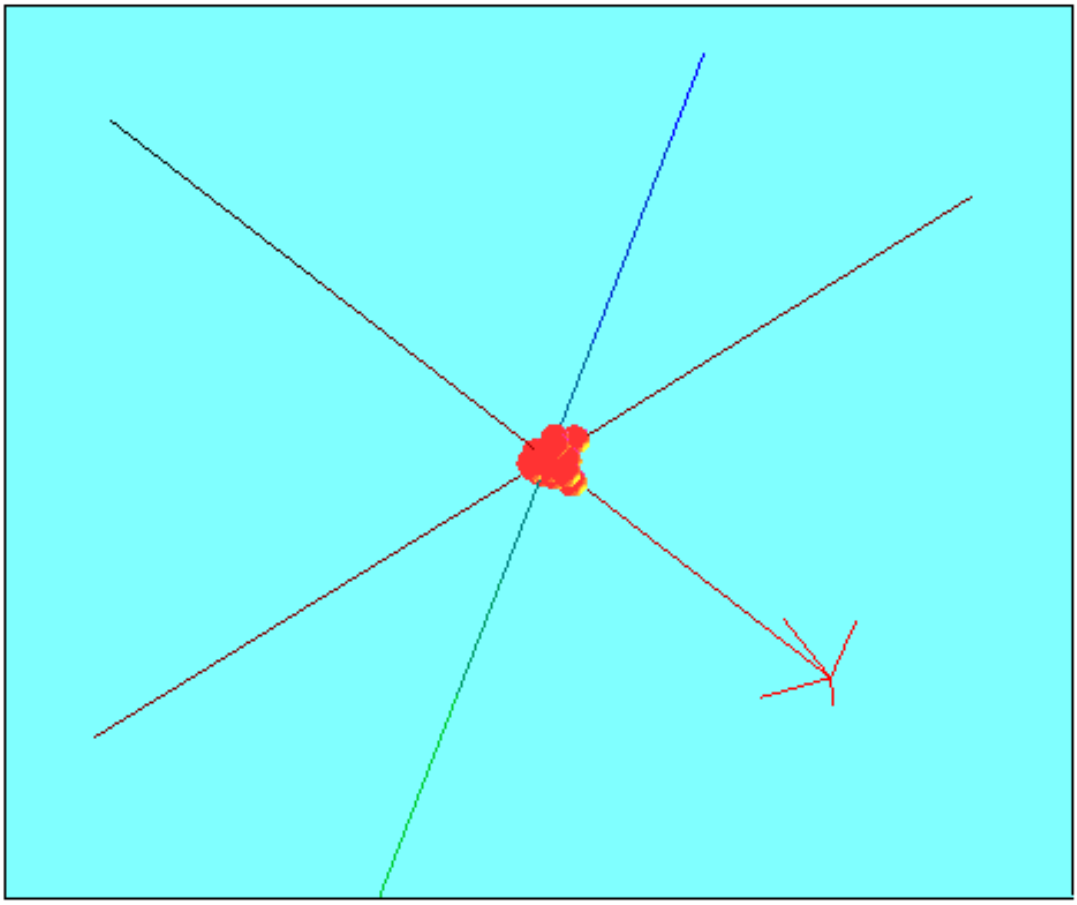

Figure 6.6 Positions of Particles (712th iteration)

Here, the virtual reality enabled PSO is used for the analysis of PSO behavior and its understanding.

7. Summary and Conclusion

This chapter briefly summarizes the findings and contributions of this thesis, followed by a conclusion and discussion of numerous directions for future research.

7.1 Summary

The thesis investigated the detailed study of Particle Swarm Optimizer algorithms, its performance analysis and its uses in the area of plasma physics. Virtual Reality enabled PSO investigated the modeling of PSO behavior in the 3D world.

Chapter 2 discusses about the basic optimization concepts of local and global minima. It also describes about the road map of Particle Swarm Optimization including its background and its basic optimization algorithm, followed by the various neighborhood topologies used in the PSO. At the end, variants of PSO, Constant Inertia Weight PSO, Variable Inertia Weight PSO, Constriction PSO and Fully Informed PSO are discussed, which are created by involving improvement in the PSO update equation. It also discusses about the Virtual reality, which is used to simulate the behavior of the PSO.

In Chapter 3 and 4, the requirement and the design of the PSO engine is discussed. These are the basic building blocks of any system which is to be implemented.

Chapter 5 discusses about the performance of the PSO. In this chapter, various synthetic benchmark functions are discussed which are used to test the algorithms. It also discusses about the optimum parameter selection of PSO. To select the optimum values of parameters of PSO, the PSO is used to optimize certain benchmark function using different ranges of the parameters of PSO. From these ranges of values, optimum values are selected. At the end of the chapter, various benchmark functions are optimized using defined variants of PSO, with optimum values of parameters of PSO. These variants require distinct number of evaluations.

Chapter 6 applied the variants of PSO to optimize various real world problems of plasma physics like atomic cluster optimization, optimizing parameters

of Nuclear Fusion Reactors for optimum cost and surface wave mode differentiation and contribution. In Atomic cluster optimization, the Lennard Jones problem, which is used to configure the lowest energy clusters, is optimized using PSO. The results are than compared with Genetic Algorithm, whose results are empirically shown. In Nuclear Fusion, the values of parameters of Reactors are optimized for lowest cost per energy (KiloWatt/hour). Another physics problem, differentiation and contribution of surface wave modes is also optimized using PSO, which is used in the design of conformal antenna and control of scattering properties of the metallic dielectric cylinder. At the end of the chapter, Virtual Reality enabled PSO is discussed, which is used to analyze the convergence behavior of the PSO visually.

7.2 Conclusion

Plasma Physics is a complex nonlinear system. The problems defined in the plasma physics involve the optimization of a set of parameters and their constraint checking on a set of boundary conditions. In optimization, the convergence of the algorithm for local and global minima is of interest. Numerous optimization techniques have been proposed. Particle Swarm Optimization is one of them, which is relatively new technique inspired by the behavior of swarm found in nature. The PSO has certain parameters. The convergence speed of the algorithm is depends on these parameters. To improve the convergence speed, the optimum value of parameters of PSO should be extracted. For that, PSO is used to solve certain benchmark functions using different ranges of values of parameters of PSO. From the result (Section 6.2), for certain values of parameters, the PSO converges faster, which can be considered as optimum values. So, it is concluded that, by using optimum values of parameters of PSO, the convergence speed is increased.

By using these optimum values of parameters, different variants of PSO are required to analyze for their comparison with each other. For comparison, different synthetic benchmark functions, which have different characteristics, are optimized using different variants of PSO. The results shows that, for simple functions like Spherical, Fully Informed PSO gives the best results compare to other ones and Inertia weight PSO and Constriction PSO gives almost identical performance, but less compare to FIPSO. The Standard PSO gives the lowest performance among all

the PSO. For complex functions, there is always a trade off between the selection of Inertia weight PSO and Fully Informed PSO, because for some functions like Griewangk Inertia weight PSO and Standard PSO gives the better results while for some functions like Ackley, FIPSO gives best results. So, it is required to study the functions characteristics for the purpose of classifying functions in which Inertia weight PSO or FIPSO can be used. Variable Inertia Weight PSO converges with lowest number of additions and multiplications. Constriction PSO and Constant Inertia weight PSO performances are less compare to other two PSO. So, it can be concluded that, the convergence speed of the algorithm depends on the parameters of PSO and the characteristics of function which is to be optimized.

To date, Particle Swarm Optimization was never used in the field of PSO. Empirical results shows that, PSO had performed well in different area of problem space like neural network, microorganisms, power and voltage control, stabilizer design, optimizing continuous and discrete variables simultaneously, etc… By taking inspiration from these results, PSO is tried to optimize plasma physics problems. Here the PSO is used to solve atomic cluster optimization problem in which from results (Figure 6.1), the PSO gives better results than Genetic Algorithm. So, it can be concluded that, for minimizing atomic cluster energy PSO performs better than GA. PSO is also used to calculate the optimum parameters of Nuclear Fusion Reactor for optimum cost per unit energy generated. When PSO is used to solve these problem, it gives different optimum values for parameters of Nuclear Fusion Reactor, from which, depending on the requirement the values of parameters can be selected to design the Nuclear Fusion Reactor to generate energy with optimum cost.

When EM waves are excited on a dielectric cylinder, it produces surface wave modes. These surface wave modes are used in the design of conformal antenna design and also to control the scattering properties of metallic dielectric cylinder. The calculation of these surface wave modes is very complex. Here. PSO is used to find the characteristics of surface wave modes from which surface wave modes and its distribution can be calculated. Even though, the results obtained are not giving the desirable ones, but it can be used as the base for further improvement.

Today, Virtual Reality is very important, as its uses are increasing day by day in the area of scientific calculation, medical, chemistry, education and gaming. Here to understand the behavior and to analyze the convergence of PSO, the Virtual Reality is used. For that, the PSO made virtual reality enabled using Java3d. This virtual reality enable PSO can be used to simulate the behavior of PSO.

REFERENCES

[1] J. Kennedy and R.C.Eberhart. Particle Swarm Optimization. *In Proceeding of IEEE International Conference on Neural Networks,* volume IV, pages 1942-1948, Perth, Australia, 1995, IEEE Service Centerr, Piscataway

[2] An Analysis of Particle Swarm Optimization
By Frans van den Bergh
University of Pretoria, November 2001

[3] *Modern* Heuristic Optimization Techniques with Application to Power Systems, IEEE PPS tutorial,
Chapter 5 : Fundamentals of Particle Swarm Optimization

[4] R. Boyd and P. Recharson, *Culture and the Evolutionary Process*, University of Chicago Press, 1985.

[5] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer", *Proceedings of IEEE International Conference on Evolutionary Computation (ICEC'98)*, pp.69-73, Anchorage, May 1998.

[6] "An Introduction to Particle Swarm Optimization"
Matthew Settles,Department of Computer Science, University of Idaho, Moscow, Idaho U.S.A 83844,November 7, 2005

[7] The Fully Informed Particle Swarm: Simpler,Maybe Better
Rui Mendes, *Member, IEEE,* James Kennedy and José Neves
IEEE TRANSACTIONS OF EVOLUTIONARY COMPUTATION, VOL. 1, NO. 1, JANUARY 2025 1

[8] GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab – com//index.html**}** 1994-2005 Hartmut Pohlheim

[9] http:// www.particleswarm.

[10] http://www.swarmintelligence.org

[11] "Virtual Reality Technology", wiley publications

[12] Finite Difference Time Domain Technique,
http://www..pas.rochester.edu/~icpark/Vinos/whatisfdtd.html

[13] Particle Swarm Optimization Applied to the Atomic Cluster Optimization Problem, R.J.W. Hodgson, Physics Department
University of Ottawa, Ottawa, Ontario, Canada K1N 6N5

[14] "Swarm Intelligence Systems", Nadia Nedjah and Luiza de Macedo Mourelle. An Introduction to Particle Swarm Opt,
Springer

[15] An Introduction to Particle Swarm Optimization
March, 2003. By Paul Pomeroy.
http://www.adaptiveview.com/articles/ipsop1.html

[16] THE INTERACTIVE DATA VISUALIZATION WITH Java 3D
M. Emoto, M. Shoji, S. Suzuki, S. Yamaguchi, NIFS, Toki, 509-5292 Japan
R. Muratsu, J. Narlo, M. Tamura, Y. Tanahashi, Nihon Sun Microsystems,
Y. Teramachi, University of Industrial Technology, Sagamihara, 229-1196
J. Kariya, Yamaguchi University, Ube, 755-8611
H. Okumura, Matsusaka University, Matsusaka, 515-8511, Japan

[17] Parameter Selection in Particle Swarm Optimization
Yuhui Shi and Russell C. Eberhart, Department of Electrical Engineering
Indiana University Purdue University Indianapolis, 723 W. Michigan St., SL160, Indianapolis, IN 46202

[18] Automobile Conformal Antenna Design and Optimization using Genetic Algorithm, Yongjin Kim' and Eric K. Walton, ElectroScience Laboratory, Department of Electrical Engineering, The Ohio State University, Columbus, OH 43212, Tel: 614-292-798 1, Kim.743(4osu.edu, Walton. I t3osu.edu

[19] "Java3D Programming", Danial Selman, Hamming publications

[20] How Nuclear Fusion Reactors Work
by Craig Freudenrich, Ph.D.
http://science.howstuffworks.com/fusion-reactor.htm

Here for all the functions are plotted according to their ranges specified in the chapter 5.1. Except for Griewangk function, it is plotted over [-30,30].

Spherical:

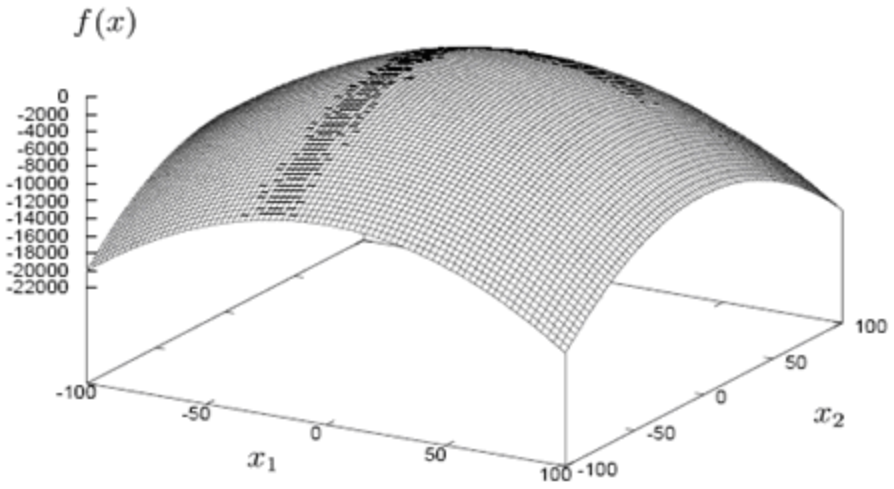

Figure A.1 Spherical function

Rastrigin:

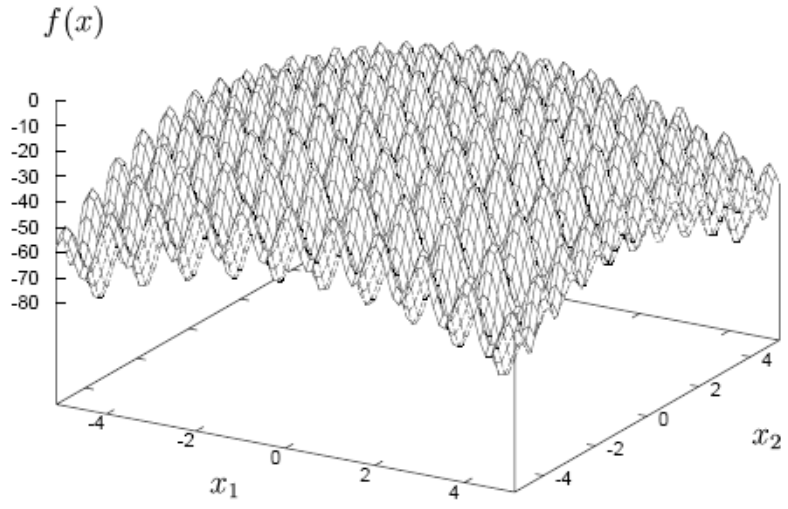

Figure A.2 Rastrigin function

Ackley:

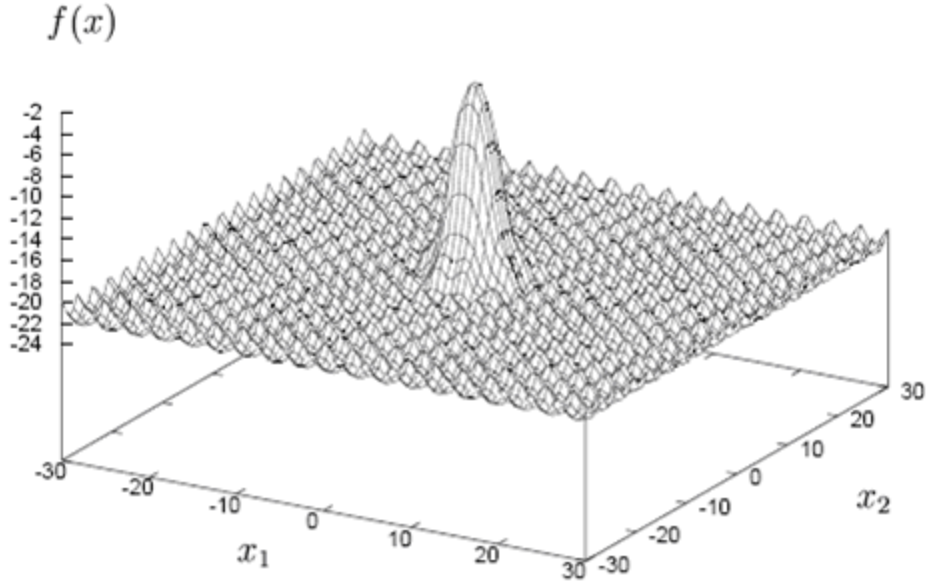

Figure A.3 Ackley function

Griewangk:

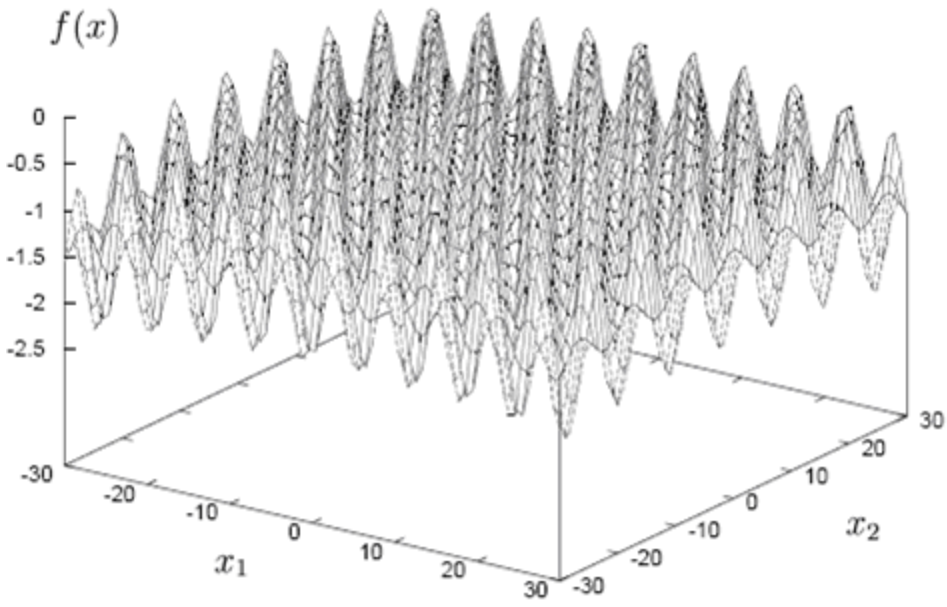

Figure A.4 Griewangk function

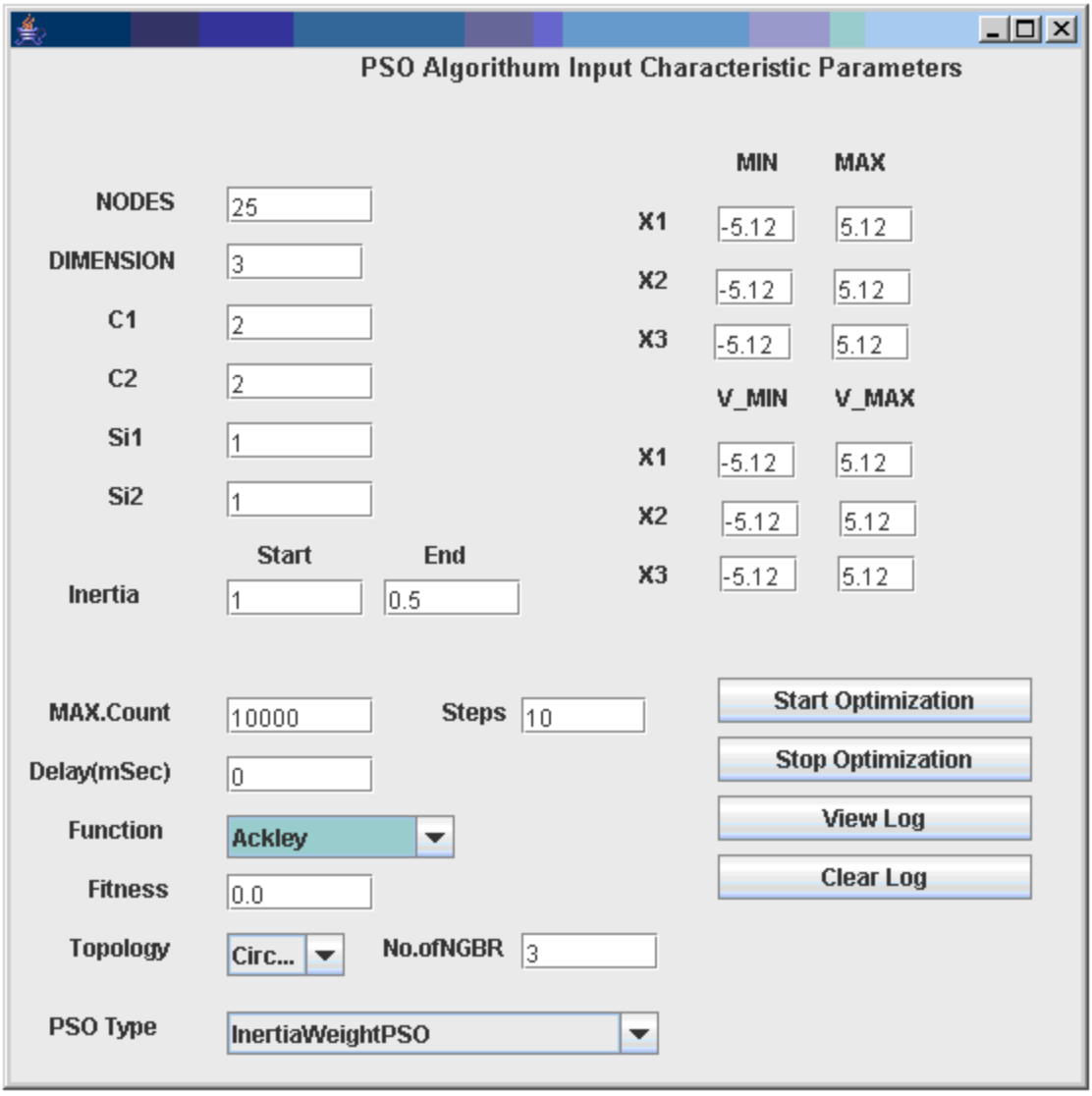

Figure B.1 PSO Algorithm Input Characteristics Frame

The above figure shows the PSO Algorithm input frame. This is used to setup the configuration parameters of PSO, to select the function to be optimized and the selection of PSO variant type.

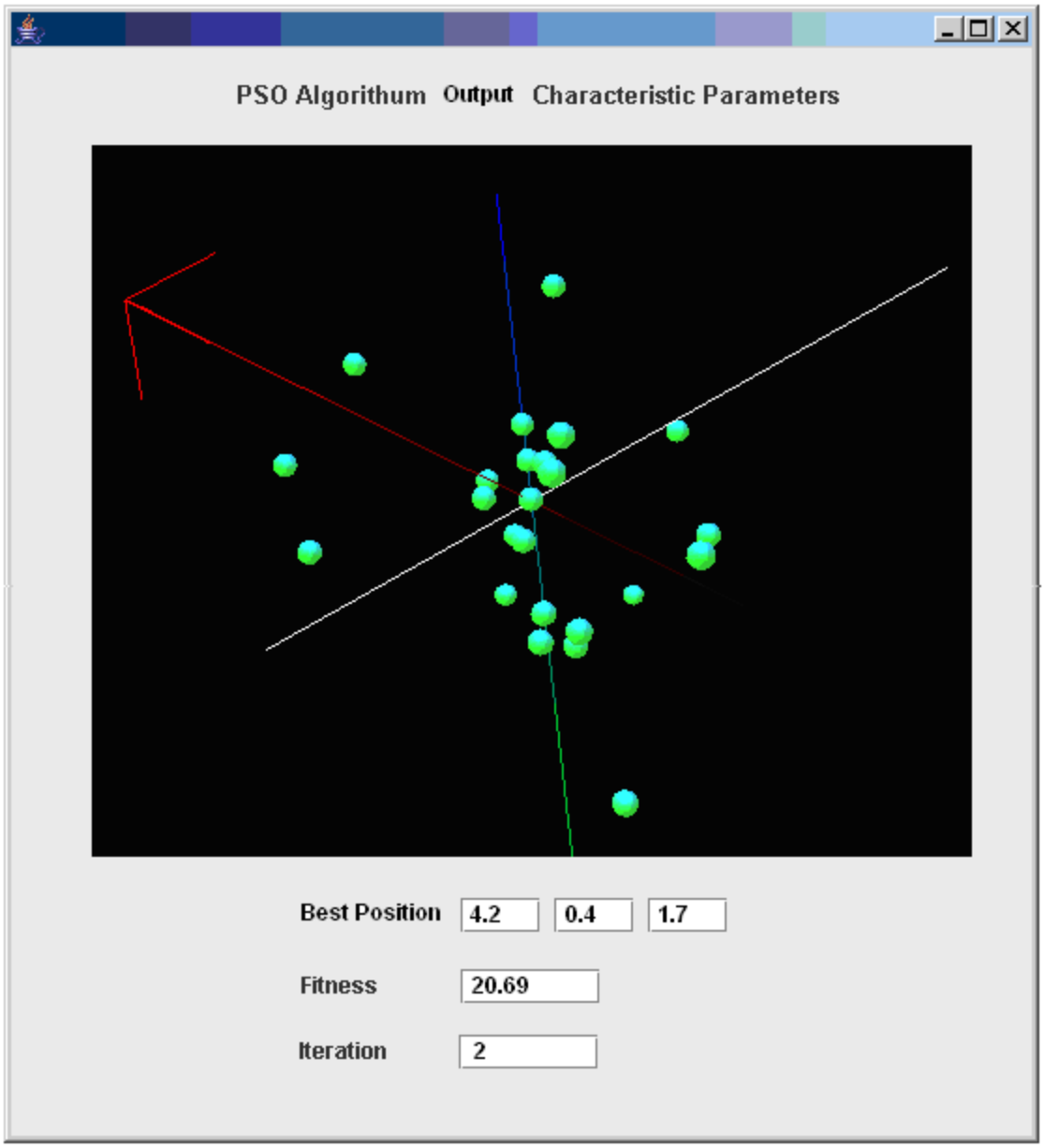

Figure B.2 PSO Algorithm Output Characteristics Frame (with 3-D visualization)

The above figure shows the output frame of PSO system. When the problem is of 3rd degree, the simulation of the PSO is displayed. The green spheres identify the particles and axes are used to understand the position of particles. It also shows the best position till now and the current best fitness and the iteration number.

Funtion	Type	NO..	DI...	C1	C2	Si1	Si2	ITER...	FAULTS	Target	Achieced	SOLUTION	X_Ran...	V_Ran...	...	INERTIA	Graph
Spherical1	InertiaWe...	25	5	2.0	2.0	1.0	1.0	126	4027	0.0	0.0	(0.0,0.0,0.0,0.0,0.0,)	5.12	5.12	5	0.5	ShowGraph
Spherical2	Constricti...	25	5	2.0	2.0	1.0	1.0	116	2948	0.0	0.0	(0.0,0.0,0.0,-0.0,0.0,)	5.12	5.12	5	0.5	ShowGraph
Spherical3	FIPSO	25	5	2.0	2.0	1.0	1.0	12	30	0.0	0.0	(0.0,0.0,0.0,0.0,0.0,)	5.12	5.12	5	0.5	ShowGraph
Spherical4	InertiaW...	25	5	2.0	2.0	1.0	1.0	277	19486	0.0	0.0	(0.0,0.0,0.0,0.0,0.0,)	5.12	5.12	5	1.0	ShowGraph
Rastrigin1	InertiaWe...	25	5	2.0	2.0	1.0	1.0	180	6757	0.0	-0.1	(0.1,-0.0,0.0,0.0,0.0,)	5.12	5.12	5	0.5	ShowGraph
Rastrigin2	InertiaWe...	25	5	2.0	2.0	1.0	1.0			0.0			5.12	5.12	5	0.5	ShowGraph
Rastrigin3	FIPSO	25	5	2.0	2.0	1.0	1.0	19	35	0.0	0.0	(0.0,0.0,0.0,0.2,0.1)	5.12	5.12	5	0.5	ShowGraph
Rastrigin4	Constricti...	25	5	2.0	2.0	1.0	1.0	170	5688	0.0	0.0	(0.0,0.2,0.0,0.0,0.1)	5.12	5.12	5	0.5	ShowGraph
Griewangk	InertiaWe...	25	5	2.0	2.0	1.0	1.0	12	864	0.0	0.0	(0.0,0.0,0.2,0.3,0.0,)	100	100	5	1.0	ShowGraph
Ackley	FIPSO	25	5	2.0	2.0	1.0	1.0	30	13	0.0	0.0	(0.3,0.1,0.0,0.0,0.0,)	30	30	3	0.5	ShowGraph

Figure B.3 Log Table

The above figure shows the Log table of the PSO system. This Log is displayed by pressing "View Log" button of the PSO input frame. It shows the results of the PSO and associated parameter's value.

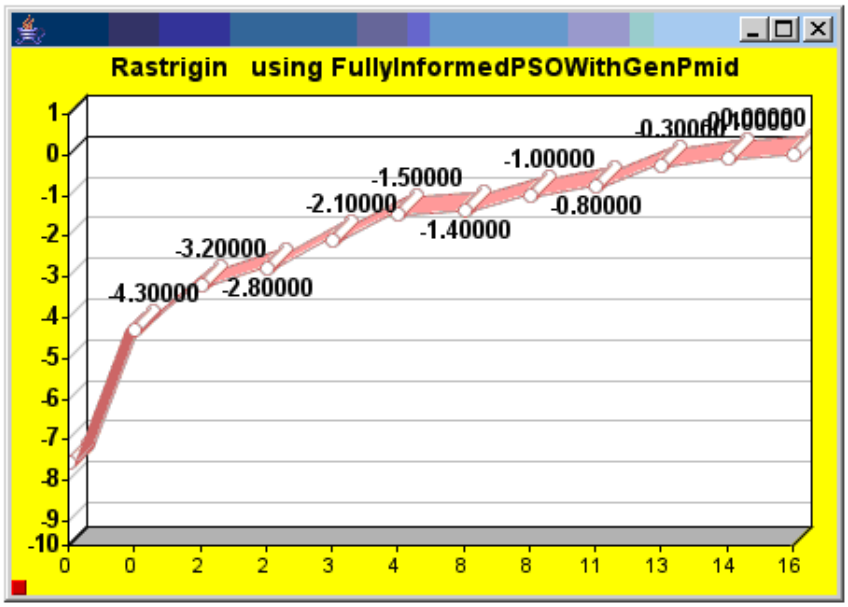

Figure B.4 Convergence Graph

The above graph shows the convergence graph of the Rastrigin function. This graph can be displayed by pressing the "Show Graph" button of Log table. The Y-axis presents the fitness and X-axis presents the iteration number.