Guiding Particle Dynamics in PSO Algorithm Using Lyapunov Functions

THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology in Intelligent Automation and Robotics

by

Desilva Roy

Roll No. : M6IAR19012

Registration No. : 137302 OF 2016-2017

Under the Guidance of

Prof. Amit Konar

DEPARTMENT OF ELECTRONICS TELECOMMUNICATION ENGINEERING

JADAVPUR UNIVERSITY

KOLKATA-7000032 MAY, 2019

<u>**CERTIFICATE**</u>

This is to certify that that the dissertation entitled "**Guiding Particle Dynamics in PSO Algorithm Using Lyapunov Functions**" has been carried out Desilva Roy (University Registration No : 137302 OF 2016-2017) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Electronics & Telecommunication Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree to any other University or Institute.

____

Prof. Amit Konar(Supervisor)

Dept. of Electronics & Telecommunication

Engineering

Jadavpur University

____

Prof. Sheli Sinha Chaudhuri
Head of the Department
Electronics & Telecommunication
Engineering
Jadavpur University

____

Prof. Chiranjib Bhattacharjee

Dean, Faculty Council of Engineering and Technology

Jadavpur University

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL*

The forgoing thesis is hereby approved as a creditable study of an engineering subject and presented in a manner satisfactory to warrant acceptance as prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for which it is submitted.

Committee on final examination
For the evaluation of the thesis

____

Signature of the Examiner

*Only in the case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC THESIS

I hereby declare that the thesis entitled "**Guiding Particle Dynamics in PSO Algorithm Using Lyapunov Functions**

" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Electronics & Telecommunication Engineering.

All information have been obtained and presented with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: DESILVA ROY

Roll No. : M6IAR19012

Registration No. : 137302 OF 2016-2017

Thesis Title: **Guiding Particle Dynamics in PSO Algorithm Using Lyapunov Functions**

Signature of the candidate

ACKNOWLEDGEMENT

First and foremost, I would like to express my earnest gratitude and heartfelt indebtedness to my supervisor, Prof. Amit Konar, Department of Electronics and Tele - communication Engineering, for the privilege and the pleasure, of allowing me to work under him towards my Degree of Master of Electronics & Telecommunication Engineering. This work would not have been materialized, but for his whole-hearted help and support. Working under him has been a great experience. I sincerely thank my supervisor, particularly for all the faith he had in me.

I am thankful to Prof. Sheli Sinha Chowdhury who has acted as Head of the Department of Electronics and Telecommunication Engineering during the tenure of my studentship. I would also like to show my special gratitude to Miss. Lidia Ghosh and Mrs Mousumi Laha, PhD Research Scholar of the Department of Electronics and Telecommunication Engineering for her constant guidance and valuable advices.

I am indebted to my classmates and friends for their constant support and good wishes. Lastly, I would like to thank my parents and sister for their love, support and guidance through the course work.

Date : Desilva Roy

Place : Jadavpur University Roll No. : M6IAR19012

PREFACE

The objective of the thesis is to determine a suitable Particle Swarm Optimization and then attempt to empirically determine the optimal parameters setting. The formal basis of our study originates from the well known Lyapunov's theorem of classical control theory . In this thesis we indirectly used lyapunov's stability theorem to determine the dynamics that necessarily converges to an optima of the lyapunov – like search landscape . We have realized different variants of the classical PSO Dynamics . Such as

a)Replacement of the inertial term by a negative partial derivative of the lyapunov – like search landscape.

b)Inclusion of a negative particle position in the velocity adaptation rule.

c)Replacement of the inertial term by the negative position term in the

dynamics.

Computer simulations undertaken on a set of 8 benchmark function reveals that the modification in the PSO dynamics results in a significant improvement in the PSO algorithm with respect to both speed-up and accuracy . Comparisons are also done with respect to the image segmentation problem ,which too ,ensures the superiority of the modified dynamics over the Classical PSO dynamics.

Contents

Chapter 1

Introduction

1.1 General Introduction and Motivation

Optimization is the action of finding the best-suited solution of some problem within the given constraints and flexibilities. While optimizing a system performance we aim at finding out such a set of values of the system parameters for which the overall performance of the system will be the best. From the very beginning of the journey of development of optimization algorithms , different kind of optimization algorithms have been invented. This is because in real life we have to optimize many different objective functions those may have very rough landscape with multiple local minima or even discontinuous at a number of points .For example ,consider Fig 1.1

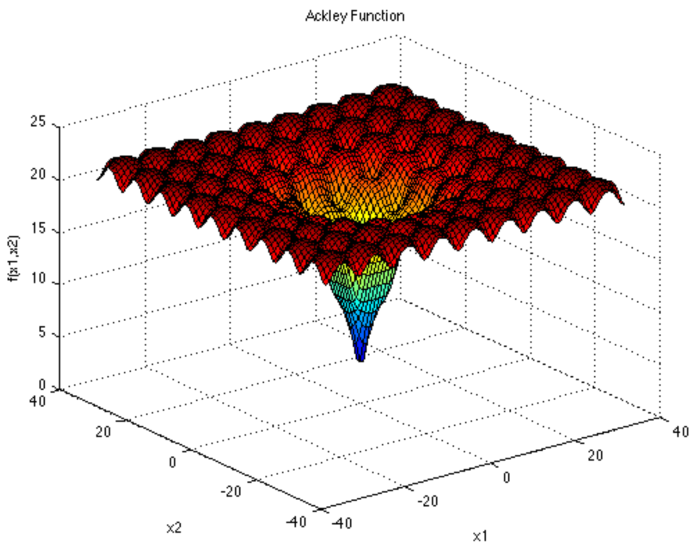


Fig 1.1 Ackley function with a huge number of local minima and maxima

Now a days ,there is a growing interest to the biological-inspired optimization algorithm ,which take inspiration from the natural selection and survival of the fittest principle of the biological world. These algorithms named as Evolutionary Algorithm [1],can perform very complex search and optimization . Evolutionary Algorithm differ from more traditional optimization techniques in that they involve a search from a "population" of solution ,not from a single point .The main step of a simple evolutionary algorithm are presented below in Procedure Evolutionary Computation [2].Here **p**(t) denotes a population of solution at time **t** .The procedure initializes **p**(t) at time t=0 .The function : Evaluate **p**(t) computes the fitness of the solution by employing a specially constructed fitness function .Each iteration of an evolutionary algorithm involves a competitive selection based on the fitness function evaluation that weeds out poor solutions. The solution with high "fitness" then go the next iteration and the process is continued ,until the terminating condition is reached.

Procedure Evolutionary-Computation :

```
Begin
    t ← 0;
    Initialize p(t);
    Evaluate p(t);
    While(Terminating condition not reached)do
        Begin
        t ← t+1;
        select p(t) from p(t-1);
        Alter p(t);
        Evaluate p(t);
    End While;
End.
```

Depending on the choice of the function : Alter **p**(t),Evolutionary Algorithm can have different variants . Rechenberg and Schwefel [3],[4] first used evolution for finding the approximate solutions of the shape designing problem of aeroplane wings . The optimization method was the Evolutionary Strategy ,where one investigates a new candidate solution found by applying a normally distributed mutation .If the new point has the better fitness ,then the solution is selected and the search continue from there .The technique is simple and has a low computational requirement .A series of other evolutionary algorithm were invented later .These includes Genetic Algorithm (GA) [5] , Genetic Programming(GP) , Ant Colony Optimization (ACO) , Differential Evolution (DE) , Particle Swarm Optimization (PSO) and others .

Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation. In simple words, they simulate “survival of the fittest” among individual of consecutive generation for solving a problem. **Each generation consist of a population of individuals** and each individual represents a point in search space and possible solution .Once the initial generation is created, the algorithm evolve the generation using following operators **–Selection Operator, Crossover Operator, Mutation Operator.**

The whole algorithm can be summarized as –

```
1) Randomly initialize populations p
2) Determine fitness of population
3) Until convergence repeat:
   a) Select parents from population
   b) Crossover and generate new population
   c) Perform mutation on new population
   d) Calculate fitness for new population
```

genetic programming (**GP**) is a technique of evolving programs, starting from a population of unfit (usually random) programs, fit for a particular task by applying operations analogous to natural genetic processes to the population of programs. It is essentially a heuristic search technique often described as 'hill climbing', i.e. searching for an optimal or at least suitable program among the space of all programs.

GP evolves computer programs, traditionally represented in memory as tree structures.[30] Trees can be easily evaluated in a recursive manner. Every tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate. Thus traditionally GP favors the use of programming languages that naturally embody tree structures (for example, Lisp; other functional programming languages are also suitable).

Differential Evolution is a Stochastic Direct Search and Global Optimization algorithm, and is an instance of an **Evolutionary** Algorithm from the field of E**volutionary** Computation . In same year , Kennedy and Eberhart [9] introduced the Particle Swarm Optimization (PSO) technique , inspired by the social behavior of bird flocking or fish schooling.

PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, met heuristics such as PSO do not guarantee an optimal solution is ever found. Also, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods.

1.2. Objective of the Thesis:

The objective of the thesis is to determine a suitable Particle Swarm Optimization and then attempt to empirically determine the optimal parameters setting. The formal basis of our study originates from the well known Lyapunov's theorem of classical control theory . In this thesis we indirectly used lyapunov's stability theorem to determine the dynamics that necessarily converges to an optima of the lyapunov – like search landscape . We have realized different variants of the classical PSO Dynamics . Such as

a)Replacement of the inertial term by a negative partial derivative of the lyapunov – like search landscape.

b)Inclusion of a negative particle position in the velocity adaptation rule.

c)Replacement of the inertial term by the negative position term in the dynamics.

Computer simulations undertaken on a set of 8 benchmark function reveals that the modification in the PSO dynamics results in a significant improvement in the PSO algorithm with respect to both speed-up and accuracy . Comparisons are also done with respect to the image segmentation problem ,which too ,ensures the superiority of the modified dynamics over the Classical PSO dynamics.

1.3. Organization of the thesis:

Chapter 2 provides an exhaustive review of a classical PSO , its parameters and its variants . This chapter acts as the basic of chapter 3 where the classical PSO dynamics has been modified in 3 ways .In chapter 4 image segmentation problem is solved by means of classical PSO and two different modified dynamics of PSO ,which were introduced in chapter 3.Finally the thesis is summarized and concluded in chapter 5.The description of the statistical methods to compare performance of the modified PSO dynamics with the classical one are presented in appendix A . Computer programs in the accompanying CD-ROM are given in appendix B . Bibliography is given at the end of the each chapter.

References :

[1] L. J. Fogel, A. J. Owens, M. J. Walsh: Artificial Intelligence through Simulated Evolution, John Wiley & Sons, New York, 1966.

[2] A. Konar, Computational Intelligence: Principles, Techniques and Applications, Springer-Verlag, 2005.

[3] I. Rechenberg, Evolution strategy: Optimization of technical systems by means of biological evolution, Fromman-Holzboog, Stuttgart, 1973.

[4] H. P. Schwefel, Numerical optimization of Computer models, John Wiley & Sons Ltd., Chichester, 1981.

[5] M. Mitchell, An introduction to genetic algorithms, MIT Press, 1996.

[6] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.

[7] J. R. Koza, Genetic Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, (1992).

[8] R. Storn, K. Price: "Differential Evolution: A Simple and Efficient Heuristics for Global Optimization over Continuous Spaces", Journal of Global Optimization, 11(4) (1997) 341-359.

[9] J. Kennedy, R. Eberhart: "Particle Swarm Optimization", proceedings of IEEE International conference on Neural Networks. (1995) 1942-1948.

Chapter 2

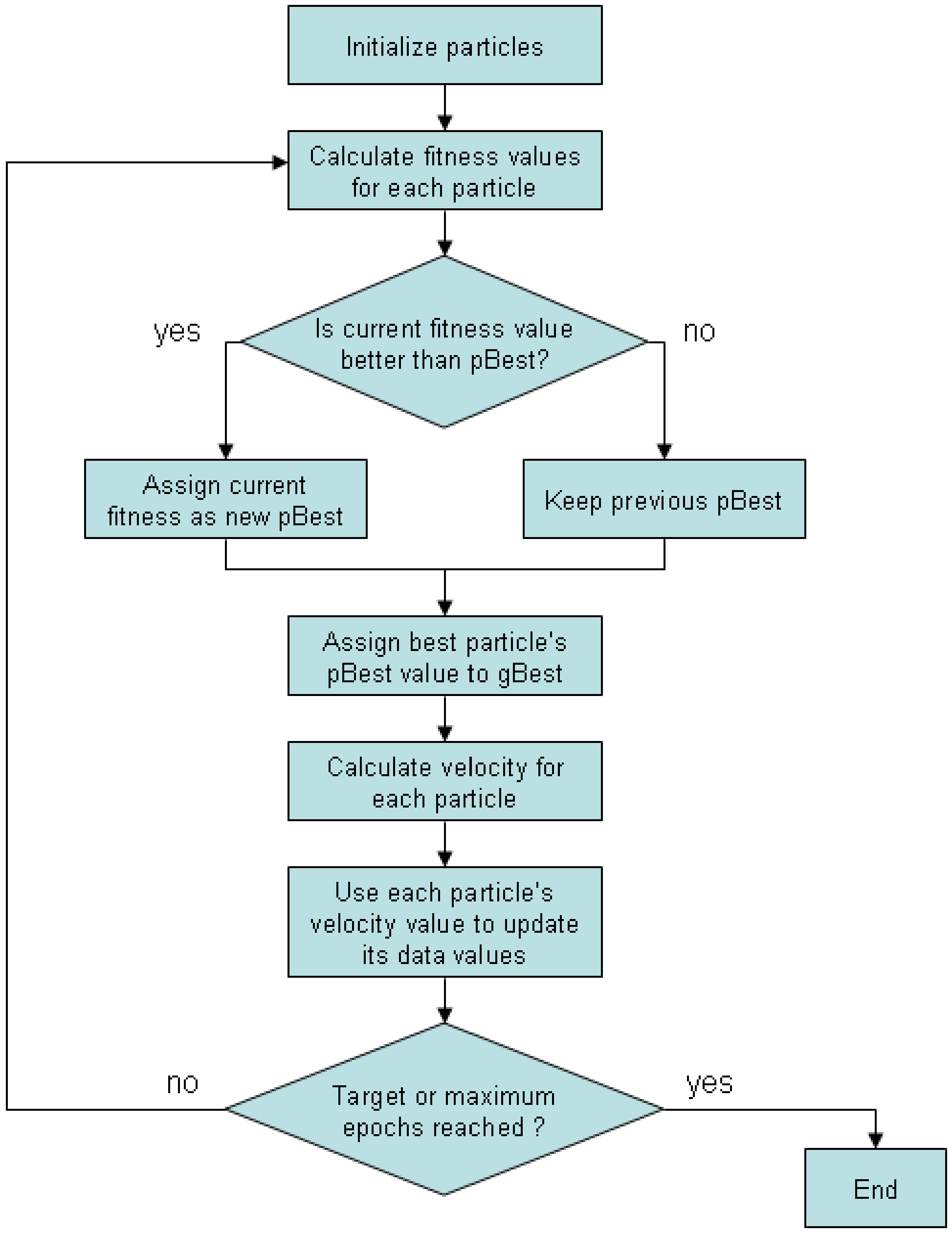
Overview of Particle swarm optimization

The concept of particle swarm, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient search and Optimization techniques. The particle Swarm Optimization(PSO),as it is called now ,does not require any gradient information of the function to be optimized, uses only primitive mechanical operation and is conceptually very simple . since its advent in 1995,PSO has attracted the attention of a lot of researchers all over the world resulting into a huge number of variants of the basic algorithm as well as many parameters automation strategies. This chapter starts with the conceptual outline of classical particle Swarm Optimization as a function optimization process . Then it goes on exploring the variants and parameters of the algorithm in great details .It also focuses on the application of PSO.

2.1.Introduction

The concept of particle swarm, although initially introduced for simulating human social behavior, has become very popular these days as an efficient means for intelligent search and optimization. The Particle Swarm Optimization (PSO) [1], [2], as it is called now, does not require any gradient information of the function to be optimized. PSO uses only primitive mathematical operators and is conceptually very simple. PSO emulates the swarming behavior of insects, animals herding, birds flocking and fish schooling, where these swarms forage for food in a collaborative manner. PSO also draws inspiration from the boids method of Craig Reynolds and Socio-Cognition [2].

A swarm is commonly used to describe a group of living creatures such as flies, bees and wasps. We have e experienced that the flies individually or in a group attempt to identify fruit juice or similar items . We have also seen that if fruit juice is kept in a largest jar usually attracts most of the Flies and only a few are attracted by the relatively small jar . The Flies hopefully have an objective to fulfill their maximum thrust ,naturally prevailing them to find the cup with maximum content .If we consider that the fulfillment of the maximum thrust of a fly to be an objective function, than identifying the largest cup perhaps is the solution of the problem. Naturally a question arises: why should we consider a swarm of flies to handle the search Problem? In fact the problem could have been defined using a single fly, where the problem refers to identifying the location of the largest cup. But if we employ a swarm of flies then the global optima which in present case is the location of the largest cup ,can be attained by identifying the cup with maximum number of flies.


Initialize particles
Calculate fitness values for each particle
yes
Is current fitness value better than pBest?
no
Assign current fitness as new pBest
Keep previous pBest
Assign best particle's pBest value to gBest
Calculate velocity for each particle
Use each particle's velocity value to update its data values
no
Target or maximum epochs reached ?
yes
End

There exist quite a large number of similar observations related to many lower class creatures. These observations motivated Kennedy and Eberhart [3] to describe a search problem by a specialized dynamics called stochastic differential dynamics. A stochastic differential dynamics is similar with typical differential/difference equations with stochastic parameters as the coefficient of the differential terms. Kennedy and Eberhart considered motion of the particles in a multi-of search space. Their objective was to ultimately use the global minima of the search space as an attractor to a ball rolling on the space by the action of a forcing function [4]. The forcing function in the present context represents the dynamics itself. Naturally, construction of a suitable forcing function that is always attracted towards the global (and also local) minimum in the process of exploring the search space is an interesting problem. The following example illustrates the concept outlined above.

Example2.1:

This example illustrates the attraction of a dynamics towards the stable point of the Well-known sphere function. The sphere function is given by,

$$f(x_1, x_2) = x_1^2 + x_2^2 \tag{2.1}$$

Is shown in the following figure.

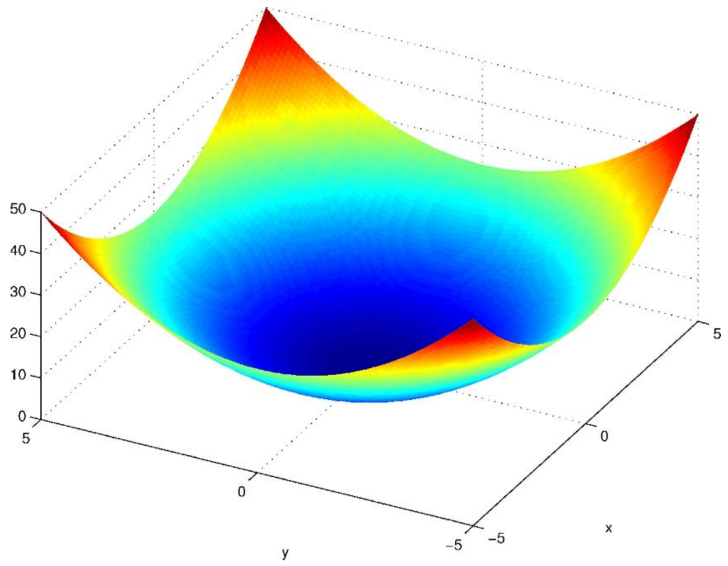


Fig. 2.1: The sphere function in two dimension

Let us consider a simple dynamics of the following form which essentially goes to the origin (0, 0) at steady state:

$$\frac{dx_1}{dt} = -ax_1(t)$$

$$\frac{dx_1}{dt} = -ax_2(t)$$

(2.2)

Now suppose a particle of zero mass is placed on the surface and a forcing function described by equation 2.2 is applied onto it to cause its motion on the surface. Here we are ignoring the effect of gravity.

The position of the particle in each iteration on the surface is represented by the iterative solution of the above set of differential equations. It is clear from the iterative response of the dynamics that it is gradually moving towards the origin (0, 0).

2.2 Construction of a dynamics causing motion towards Minima

Let us consider a search space in two dimensions x_1 andx_2.Let f be the objective function we need to minimize. Also assume that $[x_1^*, x_2^*]$ be one minima (local or global)in the search space . Suppose we need to direct the motion of the particles towards this minimum . Then we can define the velocity of a particle as a vector.

$$\text{V(t+1)} = [v_1, v_2]^T$$

where T denotes the transpose of the vector

$$\text{V(t+1)} = [\begin{matrix} v_1^{(t+1)} \\ v_2^{(t+1)} \end{matrix}] = \alpha[\begin{matrix} x_1(t) - x_1^* \\ x_2(t) - x_2^* \end{matrix}]$$

Where α is the rate constant of the dimension 1/t where t may be measured in seconds . The position of a particle in the next instance can be computed by constructing a time difference equation of the following form :

$$\text{X(t+1)} = [\begin{matrix} x_1^{(t+1)} \\ x_2^{(t+1)} \end{matrix}] = [\begin{matrix} x_1(t) \\ x_2(t) \end{matrix}] + [\begin{matrix} v_1(t+1) \\ v_2(t+1) \end{matrix}]$$

$$= \text{X(t+1)} + \text{V(t+1)}$$

Example 2.2: This example demonstrates movement of a particle along minima (the global minima) on the search space for the Rosenbrock function. It is to be noted that for the two dimensional Rosenbrock function, the global minima occurs at (1, 1). Therefore the position vectors of the particles may be updated using the following two equations:

$$V(t+1) = \begin{bmatrix} v_1^{(t+1)} \\ v_2^{(t+1)} \end{bmatrix} = \alpha \begin{bmatrix} x_1(t) - 1.00 \\ x_2(t) - 1.00 \end{bmatrix}$$

$$X(t+1) = \begin{bmatrix} x_1^{(t+1)} \\ x_2^{(t+1)} \end{bmatrix} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} v_1(t+1) \\ v_2(t+1) \end{bmatrix}$$

Starting with $[x_1, x_2]$ = [4.5, 3.2] (say), choosing a = 2 and [x1*, x21 = [1, 1], we iteratively update the equations (2.5) and (2.6) in order and finally reach the global minima after 50 iterations. A schematic diagram describing the trajectory of motion of the particle is shown in figure 2.3.

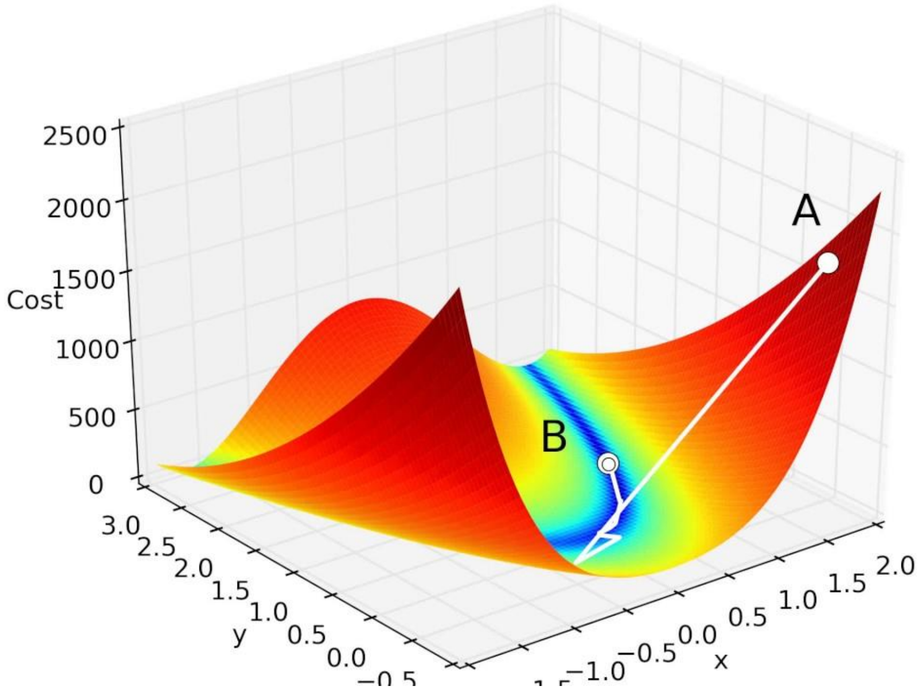


Fig. 2.3: Motion of a particle on Rosenbrock's banana valley

2.3 Particle swarm Optimization Algorithm :

A Fantastic algorithm can he originated by considering the above examples.There are two points to be referred here.
These are:

1) moving the particle towards the global minima and
2) attempting to move it further towards the local minima.

As stated before, PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b).

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) (a)
present[] = persent[] + v[] (b)

v[] is the particle velocity, persent[] is the current particle (solution). pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors. usually c1 = c2 = 2.

The pseudo code of the procedure is as follows

```
For each particle
    Initialize particle
END

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest) in history
            set current value as the new pBest
    End

    Choose the particle with the best fitness value of all the particles as the gBest
    For each particle
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    End
While maximum iterations or minimum error criteria is not attained
```

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax.

2.4 Mathematical Analysis of a Simplified PSO

This section is based on the seminal work done by Clerc and Kennedy [51. To gain a mathematical insight into the working principle of the algorithm, firstly one should strip the algorithm down to a simplest form. The particle swarm formula adjusts the velocity V j(t) by adding two terms to it as shown in (2.1). The two terms are of the same form, that is, φ(p —X_j(t)A, where P is the best position found so far, by the individual particle in the first term, or by any neighbor in the second term.

When the particle swarm operates on an optimization problem, the value of p is constantly updated, as the system evolves toward an optimum. In order to further simplify the system and make it understandable, p is set to a constant value in the following analysis. The system will also be more understandable if φ is assumed to be a constant as well; where normally it is defined as a random number between zero and a constant upper limit, the stochastic components are not involved in this discussion to keep things simple to the readers. The effect of (p on the system is very important, and much of the present paper is involved in analyzing its effect on the trajectory of a particle.

The system can be simplified even further by considering a 1-dimensional problem space, and again further by reducing the population to one particle. Thus we will begin by looking at a stripped-down particle by itself, e.g., a population of one, one-dimensional, deterministic particle, with a constant p.

Thus considering the reduced system:

v(t +1) = v(0+ φ (p— x(t))
x(t + 1) = x(t)+ v(t +1)

where, p and φ are constants. No vector notation is necessary, and there is no randomness.

Assuming $y_1 = p - x_t$ the simplified dynamic system can be expressed as

$$v_{t+1} = v_t + \varphi y_t$$

$$y_{t+1} = -v_t + (1 - \varphi) y_t$$

2.5 Extensions of the Classical PSO

Several extensions of the classical PSO algorithm have been introduced for different purposes. Some of them are described in this section.

2.5.1 Controlling the Convergence

Three approaches were proposed later to control the convergence of the classical PSO. They are described below.

2.5.1a The V_{max} Approach

The classical PSO is able to find at least the local optima of continuous, real-valued functions, if not the global one. The convergence of the entire swarm to an optimum is made possible by the introduced parameter V_{max}. If this parameter is not present in the algorithm, the dynamics would behave like the sinusoidal waves of increasing amplitudes, without being able to converge at the optima [6]. So it is obvious that V_{max} is necessary for the PSO to optimize the function.
But when V_{max}. is applied and the swarm may seem to settle on an optimum, the particles in the classical PSO do not actually converge towards a point. The use of V_{max}. imposes a maximum velocity step size on the particles, and this avoids divergence of the dynamics. A particle is most likely to hit a position nearest to (or equal to) the optimum, with the evaluations allocated more or less uniformly over the interval [f(t)— v f(t)± v 1. Hence, if we want to investigate the near neighborhood of f(t) for fine-tuning, then the V_{max}. approach is obviously not efficient.
This problem consideration has resulted in two new changes in the classical PSO, each of which can solve the problem. Today these changes have become an integrated part of the PSO model, because of the resulting performance improvements. These two models are described in the following sub-sections.

2.5.1b The Inertia-weight Approach

When a particle is in motion we usually consider its inertial effect to determine its next position. This is very rational in the sense that when a creature is in motion, it normally does not change its direction of motion abruptly, except in a few occurrences. This principle should was embedded in the design of the PSO dynamics by Eberhart and Shi [7]. In this inertia-weight approach, the velocity of the current time-step t is multiplied with a factor called the inertia-weight, o , in the velocity adaptation rule of the classical PSO.

2.5.1c The Constriction-factor. Approach

Maurice Clerc [5] introduced the concept of constriction factor in the classical PSO. In this approach, the right-hand side of the velocity-update formula is multiplied by a constriction factor x to constrict the velocity.

with 7c€ NA. This strategy reduces Vi (t) at every time-step, as compared to the original velocity-update formula. By setting x sufficiently small, we can assure convergence of the dynamics.

We see that the above two equations in effect are equivalent. The constriction-factor can mimic the inertia-weight, and vice versa.

The swarm will narrow down its search space with time by any of these two approaches, thus making the fine-tuning more efficient. However, the particles can move out of its orbit of convergence for a short time-span (which promotes more exploration of the search space to get better solutions) because of the randomness of the PSO-algorithm. But the inertia-weight (1) quickly pulls it back to continue its search around optimum, given that the local or global best position was not updated meanwhile.

2.5.2 Avoiding Premature Convergence

In this section we are concerned with the problem of premature convergence of the PSO dynamics to local optima. There are some extensions of the classical PSO which handle this problem by not only manipulating the velocity update formula, but also building up a genuinely new model.

In case of uni-modal problems, the convergence of the PSO dynamics should be as fast as possible, because there is no risk of being trapped on local minima. But when we deal with a multi-modal problem, which contains many more local minima, then the phenomenon of fast convergence becomes inadequate and unwanted. When there are many different local optima, we must spend more time on searching different solution areas before converging, simply to avoid getting stuck at a local optimum, i.e. to avoid premature convergence. This topic has been addressed in several papers, among which two are described below.

2.5.2a The Hybrid-PSO Model

Introduced a hybrid-PSO model which employs breeding between particles and sub-populations through arithmetic crossover. A parameter Pb is introduced to control the probability of breeding. Further, offspring replace parents and the population is divided into sub-populations to avoid premature convergence. The parameter Psb is also included to control the probability of breeding within sub-populations. From the experimental results, it is seen that the hybrid-PSO leads to a marginally faster convergence, and the best-found values are better on multi-modal problems, but worse on uni-modal ones.

2.5.2b Self-organized Criticality

The paper by LOvbjerg and Klink [91 introduces the concept of self-organized criticality (SOC). The SOC is used to control , the so-called "critical value", which is an individual new variable assigned to each particle. The inertia weight ω is then controlled dynamically by the expression

$$\omega = 0.2 + \frac{\text{criticalvalue}}{10}10$$

Criticality is added to the system when particles come too close to each other. When the critical value exceeds a limit, the particles are relocated using two different schemes: re-initialization of the particle and added velocity to the particle. Experimental results show that this approach leads to slightly faster convergence and better solutions than that achieved by using the classical PSO.

2.5.3 Speeding up the Dynamics

As discussed by Angeline [10] the convergence speed of the swarm towards optimum is an inherent force of the classical PSO. Hence, for uni-modal problems, the PSO converges to the optimum quickly, but it is more vulnerable to premature convergence on multi-modal problems. With the inclusion of to and x in the algorithm, the rapid convergence even lasts throughout the whole optimization process. So fine-tuning of solutions is necessary. So, the problem of having fast yet true convergence has been reduced to tuning w and x , rather than creating mechanisms which change the fundamental behavior of particles. Kennedy et al. [6] present a model that tries to improve the particle trajectories by approximating a number of cluster-centers. These centers might be nearer to the optimum, than the positions of the particles, thus substituting these centers for the local-best and global-best positions. This approach tries to improve the convergence within sub-clusters, which leads to faster convergence towards the optimum.

2.6 Parameter Selection of PSO

In this section we would like to address the important parameters for the PSO dynamics. The inertia-factor and the constriction factor have become fundamental part of the classical PSO. But there are other parameters considered in the classical PSO algorithm. The parameter-settings of the PSO determine how it optimizes the search space. One can apply a general setting that gives reasonable results, but seldom an optimal one. A useful setting for a general search is to set al (t). ag 0=2.0 and the inertia-weight w =0.8. But these settings cannot be used on many problems with optimal success; hence we must have the knowledge of the effect s of different parameter-settings. This section takes a look on how the parameters should be controlled in the PSO model.

2.6.1 Setting of Inertia Weight ω

In late 1990s when PSO was in its infancy co was presumed to be an identity matrix. Unfortunately, researchers noted that this has two-fold drawbacks. First of all, when a particle reaches the global minimum, i.e. the second and the third term of the RHS of expression (2.10) are zero; we note that the velocity of the particle Vj(t + 1) is forced to be equal to Vj(t) , which is still nonzero. To make the velocity profile of the particle zero when the particle reaches the global minimum, we should use a co matrix whose magnitude should gradually diminish to zero at steady state. When co <<1, only little momentum is preserved from the previous time-step, thus with this setting, the direction can quickly change. If co .0, the concept of previous velocity is completely lost, and the particle then moves each step without knowing the past velocity. When co >1, the particles can hardly change their direction and turn around, which implies a large exploration area as well as reluctance against convergence towards the optima.

The decreasing co -strategy allows the swarm to explore the search space at the beginning of the simulation, and manages to shift towards a local search when fine-tuning is needed. This is named as the PSO-TVIW method (PSO with Time Varying Inertia Weight).
In another paper by Eberhart and Shi uses a fuzzy controller to control co over time [12]. At each time step, the controller takes the "normalized current best performance evaluation" (NCBPE) and the current setting of (D as inputs and depending on which intervals each of the two inputs lie within, it outputs the probabilistic change in ω

2.6.2 Setting of Constriction Factor x

As mentioned earlier, the constriction factor model has the same effect as co . Basically x acts as w . Low values of X provides rapid convergence and little exploration whereas high values of provides slow convergence and more exploration. In the work by Maurice Clerc [5], it was mentioned that the constriction factor x can be set as a function of $\alpha^{l}(t)\ and\ \alpha^{g}(t)$, so that convergence is assured even without vmax• He has also introduced an additional parameter k, which controls the convergence speed of the particles to the point of attraction.

2.6.4 Setting of The Maximum Velocity V_{max}

Originally, $v_{max}$was introduced in the PSO algorithm to avoid divergence. But with the inclusion of w or x in the velocity update equation, v,,,,,, has become unnecessary to some degree, because now convergence can be assured without it [5]. In spite of this fact, the use of vii,a,, in the PSO algorithm can still improve the search.

2.6.5 Setting of The Neighborhood Topology

The fully connected (or g-best) topology is used in the classical PSO algorithm. Here, each particle is neighbor of other particles. Kennedy [14] described other topologies in a paper. The k-best topology connects each particle to its nearest k number of particles in the topological space. With k=2, this looks like the ring topology.

1)With k=swarmsize-1, it becomes the g-best topology.

2)In wheel topology, there is one central particle, and the connections are only form the central particle to other particles.

When exploring large problem spaces, optimization algorithms must effectively balance exploration and exploitation. Generally, it is wise to fist make a broad survey of the space, and then focus effort on the regions of the space that look most promising. So, Mark Richards and Dan Ventura [151 prompted a dynamic sociometry based on the topologies explored by Kennedy and Mendes. The swarm is initialized with a ring-type sociometry. Each particle is connected to just one other member of the swarm. Over time, additional links are added. Eventually, the network is fully connected in a star sociometly. The experimental results showed that this method achieved better convergence.

2.7 Some Recent Modifications of PSO

2.7.1 The PSO-RANDIW Method

It was seen that the PSO-TVIW method is not very efficient in tackling most of the dynamical problems of the real world. To track the dynamic systems, Eberhart and Shi proposed a new method called PSO-RANDIW method, where a random inertia weight factor is used and the acceleration coefficients are kept constant at 1.494.

2.7.2 The CPSO Method

A variation on the traditional PSO algorithm, called the cooperative particle swarm optimizer, or CPSO, employing cooperative behavior to significantly improve the performance of the original algorithm was introduced in [16]. This is achieved by using multiple swarms to optimize different components of the solution vector cooperatively. Application of the new PSO algorithm on several benchmark optimization problems shows a marked improvement in performance over the traditional PSO.

2.7.3 The HPSO-TVAC Method

Ratnaweera et al. [17] introduced a novel parameter automation strategy for the particle swarm algorithm and two further extensions to improve its performance after a predefined number of generations. Initially, to efficiently control the local search and convergence to the global optimum solution, time-varying acceleration coefficients (TVAC) are introduced in addition to the time-varying inertia weight factor in particle swarm optimization (PS0). From the basis of TVAC, two new strategies are discussed to improve the performance of the PSO. First, the concept of "mutation" is introduced to the particle swarm optimization along with TV AC (MPSO-TVAC), by adding a small perturbation to a randomly selected modulus of the velocity vector of a random particle by predefined probability. Second, we introduce a novel particle swarm concept "self-organizing hierarchical particle swarm optimizer with TVAC (HPSO-TVAC)." Under this method, only the "social"

part and the "cognitive" part of the particle swarm strategy are considered to estimate the new velocity of each particle and particles are reinitialized whenever they are stagnated in the search space.
In addition, to overcome the difficulties of selecting an appropriate mutation step size for different problems, a time-varying mutation step size was introduced. Further, for most of the benchmarks, mutation probability is found to be insensitive to the performance of MPSO TVAC method. On the other hand, the effect of reinitialization velocity on the performance of HPSO-TVAC method is also observed. Time-varying reinitialization step size is found to be an efficient parameter optimization strategy for HPSO-TVAC method. The HPSO-TVAC strategy outperformed all the methods considered in this investigation for most of the functions. Furthermore, it has also been observed that both the MPSO and HPSO strategies perform poorly when the acceleration coefficients are fixed at two.

2.7.4 Discrete PSO

Kennedy and Eberbart [18] proposed the first discrete version and Clerc [19] showed promising results on variants of the PSO specialized for some constrained optimization problem such as TSP. The similar method is used in Buthainah AI-Kazemi and Chiluduri K.Mohan 1201, which is called multi-phase discrete PSO.

2.7.5 Synergism of PSO with Other Evolutionary Computing Methods

In [8], a new crossover operator is defined to swap information between two individuals in order to determine their next position on the search landscape. Miranda et. al. in 1211 proposed a mutation operator On the parameters of the PSO dynamics and the position of the neighborhood best particle, so as to enhance the diversity of the particles, thereby increasing the chances of escaping local minima. in in the inertia weight is mutated and the particles are relocated when they are too close to each other. A further increase in the diversity of the population has been attained in [22]-[24] through introduction of a new collision-avoiding mechanism among the particles. Hendtlass et al. [25]combined Ant Colony Optimization with PSO to determine the neighborhood best of a particle from a list of best positions found so far by all the particles.
in [261, a differential operator (borrowed from differential evolution) was introduced in the velocity-update scheme of PSO and the resulting scheme was named as PSO-DV. The operator is invoked on the position vectors of two randomly

chosen particles (population-members), not on their individual best positions. Further, unlike the PSO scheme, a particle is actually shifted to a new location only if the new location yields a better fitness value, i.e., a selection strategy has been incorporated into the swarm dynamics. Morten LOvberg et al. [8] combined the traditional velocity and position update rules with the ideas of breeding and subpopulations, making PSO have the potential to achieve faster convergence and the potential to find a better solution. Andrew Lim et al. [27 j combined PSO with hill climbing to solve the Bandwidth minimization problem. Because the adaptive search heuristics are problem dependent, Thiemo Krink and Morten LOvberg 1281 introduced a hybrid approach called the Lifecycle model that simultaneously applied GA, PSO and stochastic hill climbing to create a generally well-performing search heuristics.

2.8 Applications of PSO

PSO being an evolutionary algorithm, finds extensive applications in intelligent search, machine learning [29] – [31] and in particular in optimization problems [321- [38]. Quite a large number of engineering problems including prediction, control, planning, pattern recognition [39] and scheduling can ultimately be transformed into one of the fundamental problems mentioned above. The fundamental approach of PSO in search, learning and optimization thus may be called generic applications.

2.8 Conclusion

Although PSO has been accepted widely as a potential global optimizing algorithm because of its convenience of realization and low constraints on the environment and objective function, there is still a great space for the research of the algorithm itself and its application. Despite of the promising results for many optimizing problems, the convergence property of PSO has been studied from the experiments by now. Many researchers are engaged in the work of fundamental theory, and research on the parameters is still going on, which aims balancing the convergence speed and convergence quality efficiently. In the application fields, it is most important to develop an effective parallel PSO to satisfy the requirement of large-scaled engineering optimization problems. There has been some works in this field, but it is far less.

References

[1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", in Proc. 6th Int. Symp. Micromachine Human Sci., vol. 1, Mar. 1995, pp. 39-43.

[2] F. van den Bargh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization", IEEE Trans. Evo. Comp., vol. 8, no. 3, June 2004.

[3]. J. Kennedy, R. Eberhart, "Particle swarm optimization", In Proceedings of IEEE International Conference on Neural Networks, (1995) 1942-1948.

[4] A. Konar and S. Das, Differential Evolution and Particle Swarm Optimization, Springer-Veriag, 2008 (to appear).

[5]M. Clem and J. Kennedy, "The Particle Swarm — Explosion, Stability, and Convergence
in a Multidimensional Complex Space", IEEE Transactions Computation, Vol. 6, No. 1, February, (2002): 58-73.
on Evolutionary

[6] J. Kennedy and R. C. Eberhart, Swarm Intelligence, ISBN 1-55860-595-9, Academic Press (2001).

[7]Y. Shi and R. C. Eberhart, "A modified Particle Swarm Optimizer", IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, (1998).

[8]M. LOvberg, T. K. Rasmussen, and T. Krink, "Hybrid Particle Swarm Optimiser with Breeding and Subpopulations", Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001).

[9] M. LOvberg and T. Krink, "Extending Particle Swarms with Self-Organized Criticality", appeared in: Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002).

[10] P. J. Angeline, "Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences", Evolutionary Programming VII, Lecture Notes in Computer Science 1447, Springer, (1998): 601-610.

[11]Y. Shi and R. C. Eberhart, "Empirical Study of Particle Swarm Optimization", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, IEEE Press, (1999): 1945-1950.

[12] Y. Shi and R. C. Eberhart, "Fuzzy Adaptive Particle Swarm Optimization", Proceedings of the 1999 Congress of Evolutionary Computation 2001, Seoul, Korea, IEEE Service Center, IEEE (2001): 101-106.

[13] .. Carlisle and G. Dozier, "An off-the-shelf PSO", Proceedings the workshop on particle svvarm optimization, Pardue School of engineering and technology, lndianapt I ku IN, (20(01).

[14] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, IEEE Press, (1999): 1931-1938.

[15] Mark Richards and Dan Ventura, "Dynamic sociometry in particle swarm optimization", International Conference on Computational Intelligence and Natural Computing, pp. 1557-1560, September 2003.

[1 6] Frans van den Bergh and Andries P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization", IEEE Trans. on Evo. Compu., vol. 8, no. 3, pp. 240-255, Jun. 2004.

[17] A. Ratnaweera, S. Halgamuge, and H. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients", IEEE Trans. on Evo. Compu., vol. 8, no. 3, pp. 240-255, Jun. 2004.

[18] J. Kennedy and R.C. Eberhart, "A discrete binary version of the particle swarm algorithm", Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, Piscataway, NJ, 1997.

[19]M. Clerc, Discrete particle swarm optimization, New Optimization Techniques in Engineering, Springer-Verlag. 2004.

[20] Buthainah Al-kazemi B. and Mohan CK, "Multi-phase discrete particle swarm optimization-, proceeding of Fourth International Workshop on Frontiers in Evolutionary Algorithms (FEA 2002), 2002.

[21] V. Miranda and N. Fonseca, "New evolutionary particle swarm algorithm (epso) applied to voltage/VAR control", The 14' Power Systems Computation Conference (PSCC'02), Seville, Spain, June, 2002.

[22] T. Blackwell, and P. J. Bentley, "Don't push me! Collision-avoiding swarms", IEEE Congress on Evolutionary Computation, Honolulu, Hawaii USA, 2002.

[23] T. Krink, J. S. VesterstrOm and J. Riget, "Particle swarm optimization with spatial particle explosion", Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002).

[24] X. Xie, W. Zhang and Z. Yang, "A dissipative particle swarm optimization", IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, USA, 2002.

[25] Hendtlass, T. and Randall, M., "A survey of ant colony and particle swarm metaheuristics and their application to discrete optimization problem", Proceedings of The Inaugural Workshop on Artificial Life (AL'01), pp-15-25, 2001

[26] S. Das, A. Konar, U. K. Chakraborty, "Improving Particle Swarm Optimization with Differentially Perturbed Velocity", in ACM-SIGEVO Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2005), Washington DC, June, 2005.

[27] A. Lim, Jing Lin and Fei Xiao, "Particle Swarm Optimization and Hill Climbing to Solve the Bandwidth Minimization Problem", The Fifth Metaheuristics International Conference (MIC2003), Kyoto-Japan, 2003.

[28] T. Krink and M. LOvberg, "The lifecycle model combining particle swarm optimisation, genetic algorithms and hill Climbers", 7th International Conference on Parallel Problem Solving from Nature, PPSN '2002. Granada, Spain, pp. 621-630, September 2002.

[29] van den Bergh F. and Engelbrecht A. P., "Cooperative learning in neural networks using particle swarm optimizers", South African Computer Journal, vol. 26, pp. 84-90.2000.

[30] Alex Conradie, Risto Miikkulainen and Christiaan Aldrich, "Adaptive Control Utilising Neural Swarming", GECCO 2002: 60-67.

[31] Eberhart R. C. and Hu X., "Human tremor analysis using particle swarm optimization", Proceedings of the IEEE Congress on evolutionary computation (CEC 1999). Washington D.C., pp. 1927-1930.1999.

[32]Parsopoulos K. E. and Vrahatis M. N., "Recent approaches to global optimization problems through particle swarm optimization," Natural Computing, vol. 1, no. 2-3, pp. 235-306.2002.

[33]Thomas Barr-Beielstein, Philipp Limbourg, Konstantinos E. Parsopoulos, Michael N. Vrahatis, Join Mehnen and Karlheinz, "Particle swarm optimizers for pareto optimization with enhanced archiving techniques", Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003), vol. 3, pp. 1780-1787, IEEE Press, Canberra, Australia, December 2003
.
[34] Hu X!, Eberhart R. C. and Shi Y., "Particle swarm with extended memory for multi-objective optimization", Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003). Indianapolis, Indiana, USA, pp. 193-197.2003.

[35] S. Mostaghim S. and J. Teich, "The role of e-donunance in multi-objective particle swarm optimization methods", Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), Canberra, Australia, pp. 1764-1771.2003.

[36] J. E. Fieklsend and S. Singh, "A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence", Proceedings of the 2002 U.K. Workshop on Computational Intelligence, Birmingham, UK, pp. 374,2002.

[37] S Mostaghim and J. Teich, "Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPS0)", The Proceedings of the IEEE 2003 Swarm Intelligence Symposium, Indiana, USA, pp 26-33, April 2003.

[38] H. Liu, "Dynamic population strategy assisted particle swarm optimization in multi-objective evolutionary algorithm design", IEEE Neural Network Society, IEEE NNS Student Research Grants 2002 — Final Reports, 2003.

[39]M. Omran, A. Engelbrecht, and A. Salman, "Particle swarm optimization method for image clustering", Int. J. Pattern Recognition Artificial Intelligence, vol. 19, no. 3, pp. 297-322, 2005.

Chapter 3

A Lyapunov-based Extension to Particle Swarm Dynamics for Continuous Function Optimization

This chapter proposes three alternative extensions to the classical global-best particle swarm optimization dynamics. and compares their relative performance with the classical particle swarm algorithm. The first extension. which readily follows from the well-known Lyapunov 's stability theorem, provides a mathematical basis of the particle dynamics with a guaranteed convergence at an optimum. The inclusion of local and global attractors to this dynamics leads to faster convergence speed and better accuracy than the classical one. The second extension augments the velocity adaptation equation by a negative randomly weighted positional term of individual particle, while the third extension considers the negative positional term in place of the inertial term. Computer simulations further reveal that the last two extensions outperform both the classical and the first extension in convergence speed and accuracy.

3.1. Introduction

Often, when performing optimization on complex non-linear functions. optima can be located more quickly using population-based algorithms than algorithms that consider only a single coordinate of the search space at a time. Population-based search methods can be defined as follows:

$$P' = m(f(P)) \qquad (3.1)$$

p is a multiset of positions in the search space, called the population ,f(.) a fitness function that a vector or values signifying the optimality of each population member and m is a population manipulation function that returns a new population from the old. Information either deduced explicitly from the parents or incorporated implicitly via the search dynamics acting on the population can often provide. Information unobtainable otherwise to an optimization technique .Particle Swarm Optimization is a population-based search method with the form of equation 3.1 where the manipulation function is based on models of insect swarm behavior. Each individual contains a current location in the search space. a current velocity. and the best position found by this individual up to this point in the search.

There exists tm extensive literature on improving the performance of the PSO algorithm. This has been undertaken by two alternative approaches. First. The researchers are keen to improve swarm behavior by selecting the appropriate form of the swarm dynamics. Alternatively. considering a given form of particle dynamics. researchers experimentally, or theoretically. attempted to find the optimal settings of the range of parameters to improve PSO behavior. In this chapter. we adopt the first policy to determine a suitable dynamics, and then attempted to empirically determine the optimal parameter settings.

The classical PSO dynamics adapts the velocity of individual particles by considering the inertia of the particle and the position of local and global attractors. The positions of the attractor, are also adapted over the iterations of the algorithm. The motion of the particles thus continues until most of do particles converge in the close vicinity of the global optima. In this paper, we consider different versions of the swarm dynamics to study the relative performance of the PSO algorithm both from the point of view of accuracy and convergence time.

The formal basis of our study originates from the well-known Lyapunov's theorem of classical control theory. The Lyapunov's theorem is widely used in nonlinear system analysis to determine the necessary conditions for stability of a dynamical system. In this paper. we indirectly used Lyapunov's stability theorem to determine a dynamics that necessarily converges to an optima of the Lyapunov-like search landscape. The principles of guiding particle dynamics towards the global and local optima, here too, is ensured by adding local and global attractor terms in the modified PSO dynamics the rationale of selecting a dynamics that converges at one of the optima on a multimodal surface and the principle of forcing the dynamic, to move toward its local and global optima together makes it attractive for use in continuous nonlinear optimization .

There are however, search landscapes that do not possess the necessary characteristics of a Lyapunov surface. This salts for an alternative dynamics, which maintains the motivation of this research , but can avoid the restriction on the objective function to necessarily be Lyanunov-like. A look at the dynamics constructed for Lyapunov-like benchmark functions essentially reveals an dark of a negative position term in the velocity adaptation rule. This prompted us to realize different variants of the classical PSO dynamics, such as

a)replacement of the inertial term by a negative partial derivative of the Lyapunov-like search landscape.

b) inclusion of a negative particle position in the velocity adaptation rule.

c) replacement of the inertial term by the negative positional term in the dynamics.

Computer simulations undertaken on a set of 8 benchmark functions reveals that the modifications in the PSO dynamics results in a significant improvement in the PSO algorithm with respect to both convergence speed and accuracy.

3.2 Classical g-best PSO Dynamics

The global-best (g-best) PSO dynamics for the j^{th}particle in the i^{th}dimension is given in equations 1 and 2.

$$v_{j,i}(t+1) = \omega v_{j,i}(t) + \alpha^{l}(t)(p^{l}_{j.i}(t) - x_{j,i}(t)) + \alpha^{g}(t)(p^{g}_{i}(t) - x_{j,i}(t)) \qquad (3.2)$$

$$x_{j,i}(t+1) = x_{j,i}(t) + v_{j,i}(t+1) \qquad (3.3)$$

where,

$v_{j,i}(t)$ is the i^{th} component of the velocity vector of particle j at t^{th} iteration,

$x_{j,i}(t)$is the i^{th} component of the position vector of particle j at t^{th} iteration,

$p^{l}_{j.i}(t)$ is the i^{th} component of the personal(local) best position of the particle j,so far achived until iteration t .

$p^{g}_{i}(t)$ is the i^{th} component of the global best position found so far by the entire swarm at iteration t .

ω is the inertia factor,

$\alpha^{l}(t)$ denotes local acceleration coefficient at time t,

$\alpha^{g}(t)$ denotes global acceleration coefficient at time t.

Empirically ,ω is a random no. in[0,1], $\alpha^{l}(t)$ and $\alpha^{g}(t)$are random coefficients in [0,2] and [0,4] respectively . Inertia factor ω is selected randomly only once in the PSO algorithm, whereas $\alpha^{l}(t)$ and $\alpha^{g}(t)$selected randomly in each iteration of the PSO algorithm.

The PSO algorithm attempts to determine the optima on a search landscape by allowing several panicles (agents) to explore en the surface with an ultimate aim to terminate at the global optima. The terminating condition usually includes an upper limit on the iterations or a lower limit to the unsigned successive difference in the best particle position. or whichever occurs earlier.

In the next section, we would look for a dynamics that has a tendency to move towards optima, which need not essentially be the global optima .This can be attained by identifying a suitable dynamics that ensures asymptotic stability in the vicinity of an optimum over the search landscape. This, of course, needs additional restriction on the surface to satisfy the necessary conditions to be Lyapunov-like [21].If a suitable dynamics ensuring the convergence to an optimum is identified, we can control the motion of the particles towards the global/local optima by adding global and local attractors in die dynamics as used in the PSO dynamics.

3.3. Identifying a stable Dynamics for a Lyapunov-like Surface

This section begins with a few definition , available in the standard literature in Non-linear Control Theory,to explain the methodology of determining a stable dynamics for a Lyapunov-like surface.

Definition 3.1

A point X= X_e, is called an equilibrium state, if the dynamics of the system is given by

$$\frac{d\,X}{dt} = f(X(t))$$

Become zero at X= X_e, for any t. The equilibrium state is also called equilibrium (stable), point in D-dimensional hyperspace- when the state $X_e$has D- components,

Definition 3.2

A scalar function V(X) is said to be positive definite with respect to the point X_e, in the region

$\|X - X_e,\| \leq K, if\ V(X) > 0\ at$ at all points of the region except at X_e, Where it Is zero.

Definition 3.3 .

A scalar function V(X) is said to be negative definite if - V((X) is positive definite.

Definition 3.4

A scalar function V(X) is said to be positive semi-definite with respect to the point X_e, in the region $\|X - X_e,\| \leq K$,if its value is positive at all points of the region except at finite number of points including origin where it is zero.

Definition 3.5

A scalar function V(X) is said to be negative semi definite if -V(X) is positive semi definite.

Definition 3.6

A scalar function V(X) is said to be indefinite in the region $\|X - X_e,\| \leq K$, if it assumes both positive and negative values within this region.

Definition 3.7

A scalar function V(X) is called a Lyapunov surface with respect to the origin 0 , if it satisfies the three conditions listed below:

i. V(0)=0
ii. ii. V(X) > 0 for X 0
iii. $\frac{\partial v}{\partial x_i}$ is a continuous function of x_i, where x_i, is the ith component of X .

Definition 3.8

A dynamics $\frac{dx}{dt}$= f(X(t)) is asymptotically stable at the equilibrium point Xe , if

a) it is stable in the sense of Lyapunov, i.e., for any neighborhood S(ε) surrounding Xe (S(ε) contains points X for which$\|X - X_e, \| \leq \varepsilon$) where there is a region S(δ) (S(δ)contains points $\vec{x}$ for which $\|X - X_e, \| \leq \delta$, $\delta < \varepsilon$ such that trajectories of the dynamics starting within S(δ) do not leave s(ε) as time t —> ∞ and

b) the trajectory starting within S(δ) converges to the origin as time t approaches infinity.

The sufficient condition for stability of a dynamics can be obtained from the Lyapunov's theorem, presented below.

Lyapunov's stability theorem [18]

Given a scalar function V(X) and some real number ε> 0, such that for all X in the region

$\|X - X_e, \| \leq \varepsilon$,the following conditions hold:

1) V(X_e)=0

2)V(X) > 0 for X #X_e, i.e. V(X) is positive definite.

3) V(X) has continuous first partial derivatives with respect to all components of X.

Then the equilibrium state X_e of the system$\frac{dX}{dt} = f(X(t))$ is

a) asymptotically stable $\frac{dV}{dt}$< 0, i.e. $\frac{dV}{dt}$ is negative definite, and

b) asymptotically stable in the large if $\frac{dV}{dt}$< 0 for X $\neq X_e$, and in addition,

V(X)→ ∞ as $\|X - X_e, \| \rightarrow \infty$

Example 3.1

Let

$$V(\mathbf{X}) = x_1^2 + x_2^2$$

be a Lyapunov energy function for the given dynamics

$$\frac{dx_1}{dt} = -x_1 \text{ and } \frac{dx_2}{dt} = -x_2$$

with the equilibrium point X = [0,0].
Then:

$$\frac{dV}{dt} = \frac{\partial V}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial V}{\partial x_2}\frac{dx_2}{dt}$$
$$= 2x_1(-x_1) + 2x_2(-x_2)$$
$$= -2\left(x_1^2 + x_2^2\right) < 0$$
i.e., negative definite.

Here, *V*(X) satisfies the first two criterions indicated in the theorem, and the partial derivatives *∂V/∂x* and *∂V/∂x* are also continuous functions of *x* and *x* . Consequently, the asymptotic stability of the dynamics is ensured as *dV/dt* is found to be negative definite for all points except at *x* = 0. Further, as

‖X‖ → ∞, *V*(X) also approaches infinity. Therefore, the asymptotic stability of the dynamics in the large is also ascertained.
The condition for asymptotic stability, as indicated in Theorem1, can be applied to the particle swarm optimization to ensure stability of the dynamics, thereby reducing the convergence time of the algorithm. When all the three underlying conditions of a Lyapunov function, indicated in Definition 2.4 are supported by the objective function, we would be interested to determine the dynamics that satisfies the necessary conditions for asymptotic stability of the dynamics. It follows from Lyapunov's Theorem that the asymptotic stability of an equilibrium state guided by the dynamics *dx /dt* is ascertained if:

$$\frac{df}{dt} = \sum_{i=1}^{D} \frac{\partial f}{\partial x_i}\frac{dx_i}{dt} < 0$$

The inequality (3) essentially holds when:

$$\frac{dx_i}{dt} = -\frac{\partial f}{\partial x_i}$$

It is indeed important to note that the condition (4) holds for the *i*-th dimension of a particle roaming over the Lyapunov-like surface for 1 ≤ *i*≤ *D*.

Example 3.2

In this example, we would like to determine a stable dynamics for a Lyapunov-like objective function. Consider for instance the Griewank function in D-dimension:

$$f(\mathbf{X}) = \frac{1}{4000}\sum\nolimits_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

In order to have asymptotic stability of the dynamics, we set

$$\frac{dx_i}{dt} = -\frac{\partial f}{\partial x_i} = \frac{x_i}{2000} + \frac{1}{\sqrt{i}} \sin\left(\frac{x_i}{\sqrt{i}}\right) \left[\prod_{j=1,j\neq i}^{D} \cos\left(\frac{x_j}{\sqrt{j}}\right)\right]$$

It is also apparent to note that the given function f (x) satisfies the three necessary conditions of a Lyapunov function, Now, if we replace the term involving inertia factor by the obtained value of in the PSO dynamics, then the PSO is expected to converge very quickly as the necessary at condition for asymptotic stability has been satisfied while deriving the dynamics

$$\frac{dx_i}{dt} = \frac{x_i}{2000} + \frac{1}{\sqrt{i}} \sin\left(\frac{x_i}{\sqrt{i}}\right) \left[\prod_{j=1,j\neq i}^{D} \cos\left(\frac{x_j}{\sqrt{j}}\right)\right]$$

3.4. Proposed Extensions of the Classical PSO Dynamics

We now define Lyapunov-based PSO dynamics (LyPSO) by adding the local and global attractor terms of classical PSO to the derived expression for asymptotically stable Lyapunov dynamics,

$$\boldsymbol{V}_j(t+1) = -\omega.\,\vec{\nabla} f(\boldsymbol{X}) + \alpha^l(t)\left(\boldsymbol{P}_j^l(t) - \boldsymbol{X}_j(t)\right) + \alpha^g(t)\left(\boldsymbol{P}_j^g(t) - \boldsymbol{X}_j(t)\right)$$

$$\mathrm{X}_j(t+1) = \mathrm{X}_j(t) + \mathrm{V}_j(t+1)$$

given in (3.4)- (3.5).

The 1st term in the right hand side of equation (3.4) ensures motion

while the second and third term controls the motion towards local and global optima respectively. It is apparent from Table 3.1 that $\frac{dx_i}{dt}$ obtained for different lyapunov like surface include a factor of (— xi). The condition. For $\frac{dx_i}{dt}$ = -ωx_i is tabulated for all the eight benchmark functions in Table 3.2. Consequently, instead of computing $\frac{dx_i}{dt}$ by the approach stated earlier, we can simply add a term -ωx_i to the ith component of the updated velocity in the classical PSO. The resulting dynamics then looks like

$$\boldsymbol{V}_j(t+1) = -\omega.\,\boldsymbol{X}_{j,i}(t) + \omega.\,\boldsymbol{V}_{j,i}(t) + \alpha^l(t)\,(\boldsymbol{P}_j^l(t) - \boldsymbol{X}_j(t)) \\ + \alpha^g(t)\,(\boldsymbol{P}_i^g(t) - \boldsymbol{X}_j(t))$$

$$X_j(t+1) = X_j(t) + V_j(t+1)$$

The dynamics given by is referred to as Position-based PSO (PPSO).

For the sake of completeness of our study, we consider a third category of the dynamics, where the inertial term is dropped from the PSO dynamics, indicated in The modified dynamics, called Steepest-PSO (SPSO) for its fast convergence, is formally given below.

$$\boldsymbol{V}_j(t+1) = -\omega.\,\boldsymbol{X}_j(t) + \alpha^l(t)\,(\boldsymbol{P}_j^l(t) - \boldsymbol{X}_j(t)) + \alpha^g(t)\,(\boldsymbol{P}_i^g(t) \\ - \boldsymbol{X}_j(t))$$

$$X_j(t+1) = X_j(t) + V_j(t+1)$$

Table 1.

The derived dynamics for the selected benchmark functions.

Function name	Functional form $f(\mathbf{X}(t))$	dx_i/dt
Sphere Function	$f(\mathbf{X}) = \sum_{i=1}^{D} x_i^2$	$-2x_i$
Rosenbrock's Function	$f(\mathbf{X}) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$-\left(400x_i^2 + 2\right) x_i + 2\left(1 + 200x_i x_{i+1}\right)$
Step Function	$f(\mathbf{X}) = \sum_{i=1}^{D} (\lvert x_i + 0.5 \rvert)^2$	$-2\lvert x_i + 0.5\rvert$
Schwefel's Problem 1.2	$f(\mathbf{X}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	$-2x_i \left(\sum_{j=1}^{i-1} x_j\right)$
Rastrigin's Function	$f(\mathbf{X}) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$-2x_i -20\pi \sin(2\pi x_i)$
Ackley's Function	$f(\mathbf{X}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})$ $- \exp(\frac{1}{D}\sum_{i=1}^{D} \cos 2\pi x_i) + 20 + e$	$-\frac{4}{D}\left(\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)\right) \frac{x_i}{\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}}$ $-\frac{2\pi}{D}\sin 2\pi x_i \left[\exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos 2\pi x_i\right)\right]$
Griewank's Function	$f(\mathbf{X}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-\frac{x_i}{2000} - \frac{1}{\sqrt{i}}\sin\left(\frac{x_i}{\sqrt{i}}\right)\left[\prod_{j=1,j\neq i}^{D} \cos\left(\frac{x_j}{\sqrt{j}}\right)\right]$
Salomon's function	$f(\mathbf{X}) = -\cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$ $+1$	$-2\pi x_i \left(\sum_{i=1}^{D} x_i^2\right)^{-\frac{3}{2}} \sin\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right)$ $-0.1x_i\left(\sum_{i=1}^{D} x_i^2 + 1\right)^{-\frac{3}{2}}$

Table 2.

Reduced form of dx_i/dt.

Function name	Reduced form of $\frac{dx_i}{dt}$	Condition for reduction
Sphere Function	$-2x_i$	Unconditional
Rosenbrock's Function	$-x_i\left(400x_i^2+2-400x_{i+1}\right)$	When $x_i x_{i+1} >> -\frac{1}{200}$
Step Function	$-2\|x_i\|$	When $x_i \gg 0.5$
Schwefel's Problem 1.2	$-2x_i\left(\sum_{j=1}^{i-1} x_j\right)$	Unconditional
Rastrigin's Function	$-x_i(2+4\pi^2)$	When x_i is very small.
Ackley's Function	$-x_i\left(\begin{array}{l}\frac{4}{D}\left(\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)\right)\frac{x_i}{\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} \\ +\frac{2\pi}{D}\left[\exp\left(\frac{1}{D}\right)\right]\end{array}\right)$	When x_i is very small.
Griewank's Function	$-x_i\left(\frac{1}{2000}+\frac{1}{i}\right)$	When x_i is very small
Salomon's function	$-x_i\left(\begin{array}{l}2\pi\left(\sum_{i=1}^{D} x_i^2\right)^{-\frac{3}{2}}\sin\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) \\ +0.1\left(\sum_{i=1}^{D} x_i^2+1\right)^{-\frac{3}{2}}\end{array}\right)$	Unconditional

3.5 The Rationale of Speed-up of the PPSO and SPSO Dynamics over Classical PSO

To compare the relative performance in Speedup and convergence of the proposed algorithm ,We study the the stability behavior of the proposed PPSO and SPSO dynamics, in absence ot the local and global attractors. This is performed by solving the first or difference equations. The condition for asymptotic stability and the location of the stable point can be ascertained from the solution of the dynamics. Theorems 1 to 2 provide interesting results, indicating asymptotic stability of the SPSO and PPSO dynamics to the origin irrespective of the search landscape, whereas Theorem 3 indicates asymptotic stability of the classical PSO to a stable point, which need not essentially be the origin. The rate at which the particle position approaches the origin further indicates that the speed of convergence of the SPSO algorithm far exceeds that of PPSO, while the speed of PPSO algorithms beats classical PSO.

Theorem 1: The dynamics of the j^{th} particle in the i^{th} dimension given by

$$v_{j,i}(t+1) = -\omega x_{j,i}(t)$$

has a stable point at the origin, when ω<i .

Proof. Let E be an extended difference operator,

such that

$$E(x_{j,i}(t)) = x_{j,i}(t) + \Delta x_{j,i}(t) = (1+\Delta)x_{j,i}(t) = x_{j,i}(t+1)$$

The equation (3,10) now can be approximately written as

$$\frac{x_{j,i}(t+1) - x_{j,i}(t)}{(t+1)-(t)} = -\omega x_{j,i}(t)$$

$$\Rightarrow x_{j,i}(t+1) - x_{j,i}(t) = -\omega x_{j,i}(t)$$

Replacing$x_{j,i}$ (t+1)by $E(x_{j,i}(t))$ we obtain:

$$E x_{j,i}(t) - (1-\omega) x_{j,i}(t) = 0$$
$$\Rightarrow (E - 1 + \omega) x_{j,i}(t) = 0$$
$$\therefore E = 1 - \omega$$

consequently, the solution of the dynamics is given by

$$x_{j,i}(t) = A(1-\omega)^t$$

where A is a constant.

Theorem 2:

The dynamics of the j^{th} particle in the i^{th}dimension given by

$$v_{j,i}(t+1) = \omega v_{j,i}(t) - \omega x_{j,i}(t)$$

is asymptotically stable with a stable point at the origin, when ω<1.

Proof We can rewrite equation as

$$\frac{x_{j,i}(t+1) - x_{j,i}(t)}{(t+1)-(t)} = -\omega x_{j,i}(t) + \omega x_{j,i}(t) - \omega x_{j,i}(t-1)$$

By Replacing ,we can get

$$\begin{aligned}
&\Rightarrow E x_{j,i}(t) - x_{j,i}(t) + \omega E^{-1} x_{j,i}(t) = 0 \\
&\Rightarrow \left(E^2 - 1 + \omega\right) x_{j,i}(t) = 0 \\
&\Rightarrow E^2 = 1 - \omega \\
&\Rightarrow E = \pm\sqrt{1-\omega}
\end{aligned}$$

So, the solution of the dynamics is given by

$$x_{j,i}(t) = A(\sqrt{1-\omega})^t - B(\sqrt{1-\omega})^t$$

where. A and B are constants.

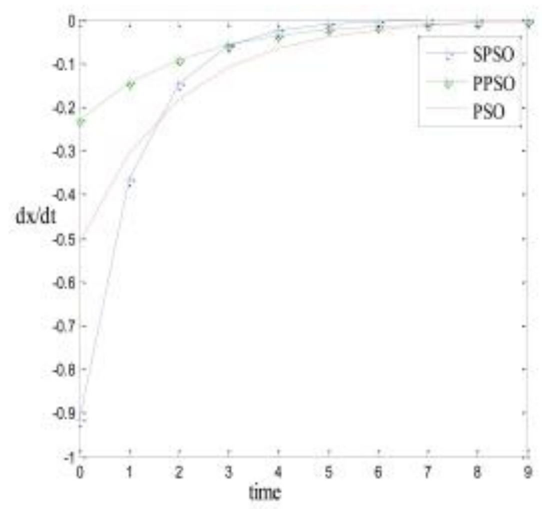


Fig 3.1

(Variation of $\frac{dx_i}{dt}$ $with\ respect\ to\ time$ **)**

For $\boldsymbol{\omega = 0.6}$

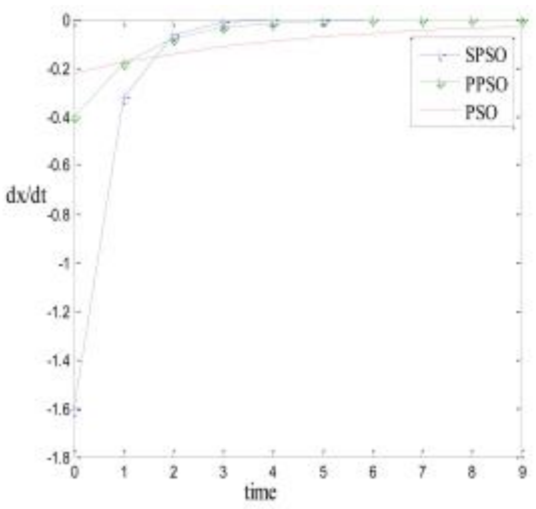


Fig 3.2

(Variation of $\frac{dx_i}{dt}$ $with\ respect\ to\ time$**)**

For $\boldsymbol{\omega = 0.8}$

3.6 Experimental Settings and Simulation Strategies for Benchmark Testing

3.6.1 Benchmarks

Eight well-known benchmarks were used to evaluate the performance of the proposed new developments, both in terms of the error after a predefined number Of iterations, and the time to converge to the optima. The performances of all new methods are then compared with the classical PS() method. The first three functions are simple unimodal functions, whereas the next five functions are multimodal functions designed with a considerable amount of local minima. All functions have the global minimum at the origin except the Rosenbrock function [17]. Simulations were carried out to find the global minima of each function. An benchmarks used are given in Table 3.1. The surface plots of eight benchmark functions are given in fig. 3.4 — fig. 1.11.

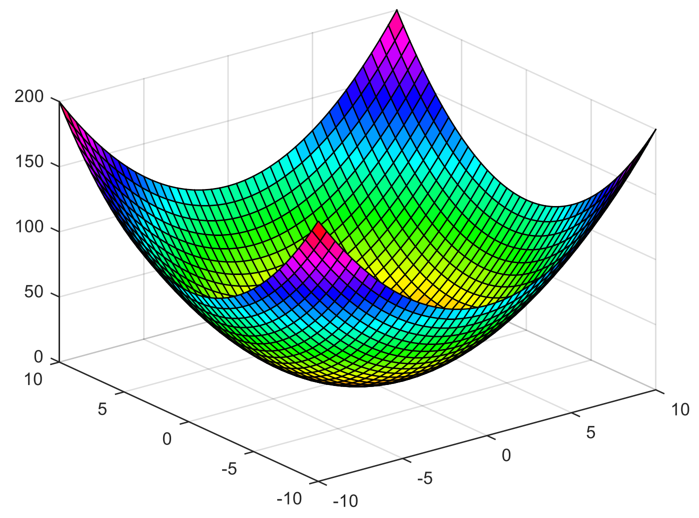


Fig 3.4: Surface plot of sphere function

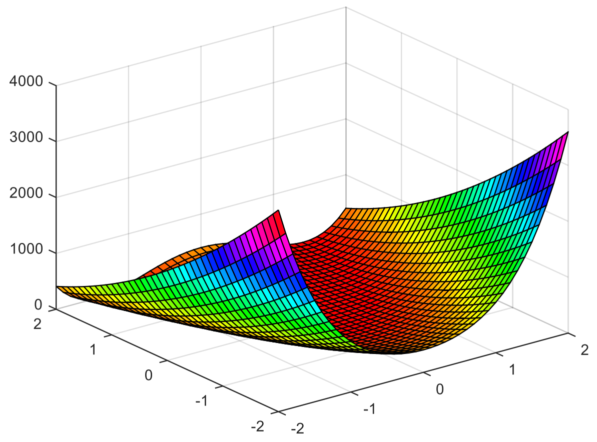


Fig 3.4: Surface plot of Rosenbrock's function

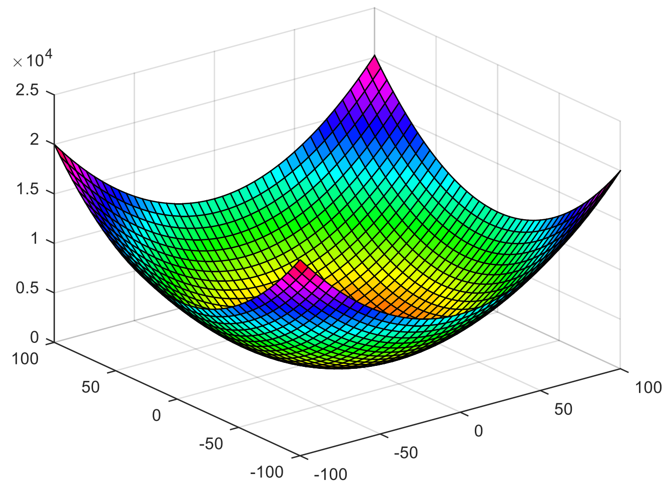


Fig 3.4: Surface plot of step function

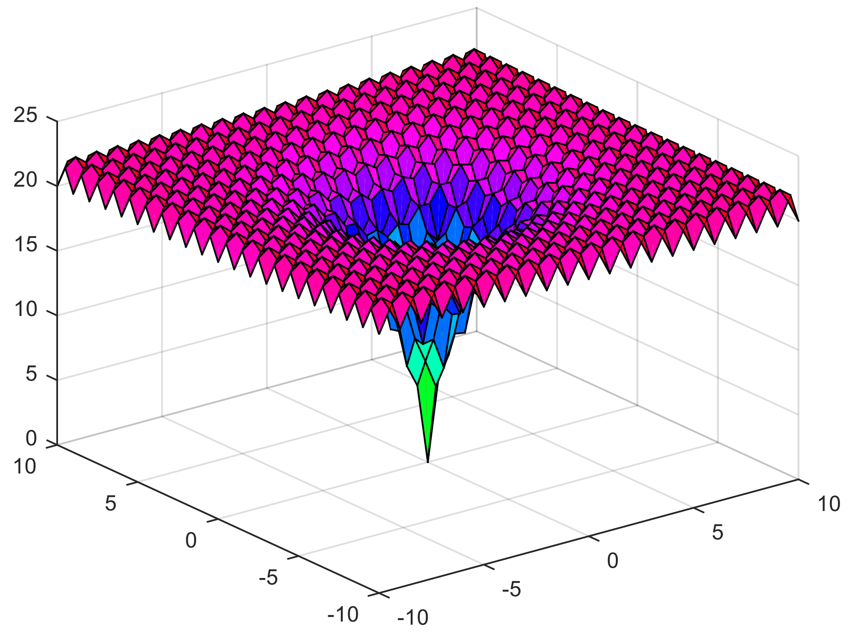


Fig 3.9:Surface plot of Ackley's Equation

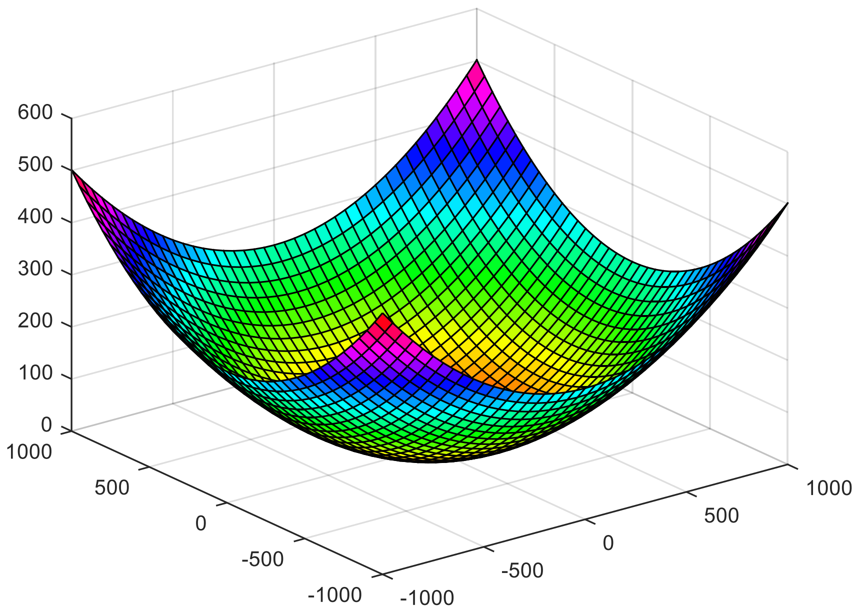


Fig 3.9:Surface plot of Griewank's Equation

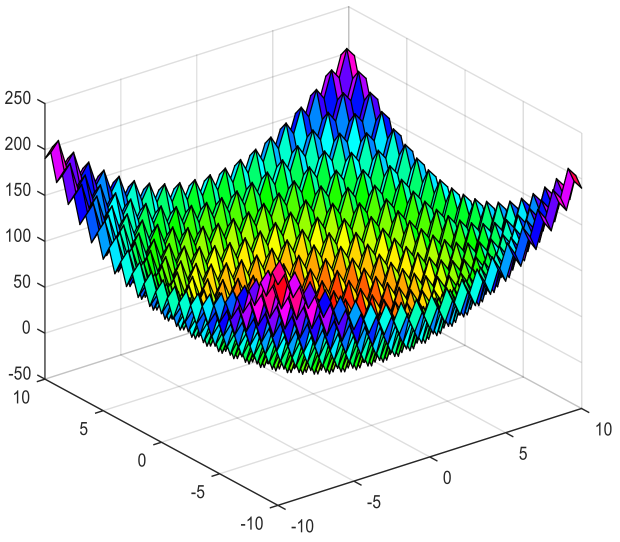


Fig 3.9:Surface plot of Rastrigin's Equation

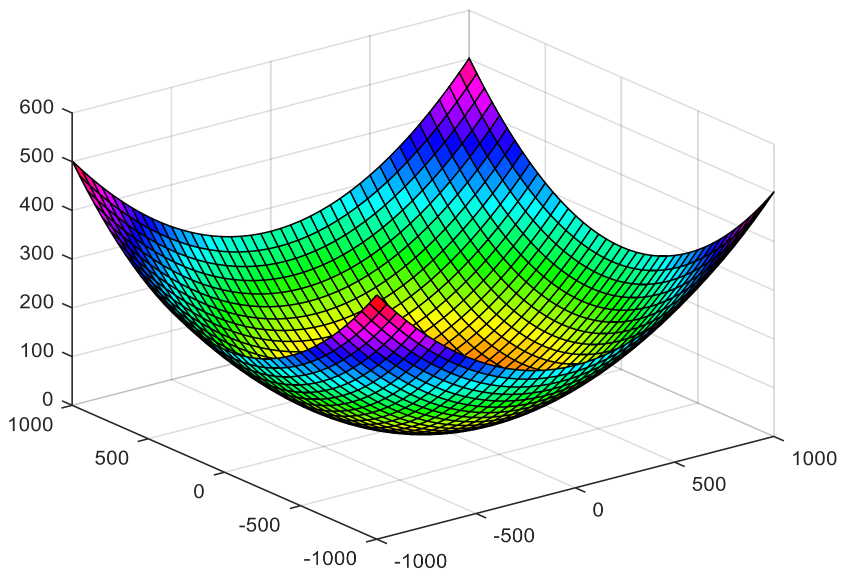


Fig 3.9:Surface plot of Schwefel's Equation

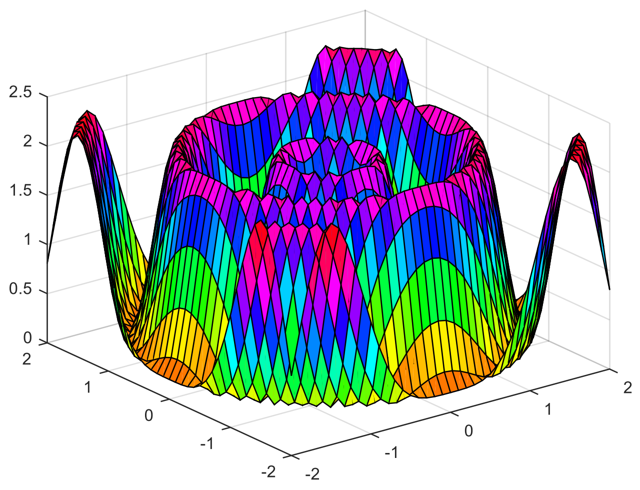


Fig 3.9:Surface plot of Salomon's Equation

3.6.2 Population Initialization

During the early stages of the development of the PSO algorithm, symmetric initialization was widely used, where the initial population is uniformly distributed in the entire search space. Later, Angeline[8] introduced .asymmetric initialization, in which the population is initialized only in a portion of the search space. Gehlhaar and Fogel [SI have shown that the typical initialization used io compare evolutionary computations can give false impressions of relative performance. In many comparative experiments, the initial population is uniformly distributed about the entire search space which is usually defined to he symmetric about the origin. in addition, many of the test functions are crafted in such a way as to have optima at or near the origin, including the test functions for this study. This method of initialization has two potential biases when considered. alone. First, if an operator is an averaging operator involving multiple parents, such as intermediate crossover often used in evolution strategies, recombining parents from opposite sides of the origin will naturally place the offspring close to the center of the initialization region. Second, given that the location of the optima is generally not known, there is no guarantee that any prescribed initialization method will include the optima.[5] suggests initializing in regions that expressly do not include the optima during testing to verify results obtained for symmetric initialization schemes.

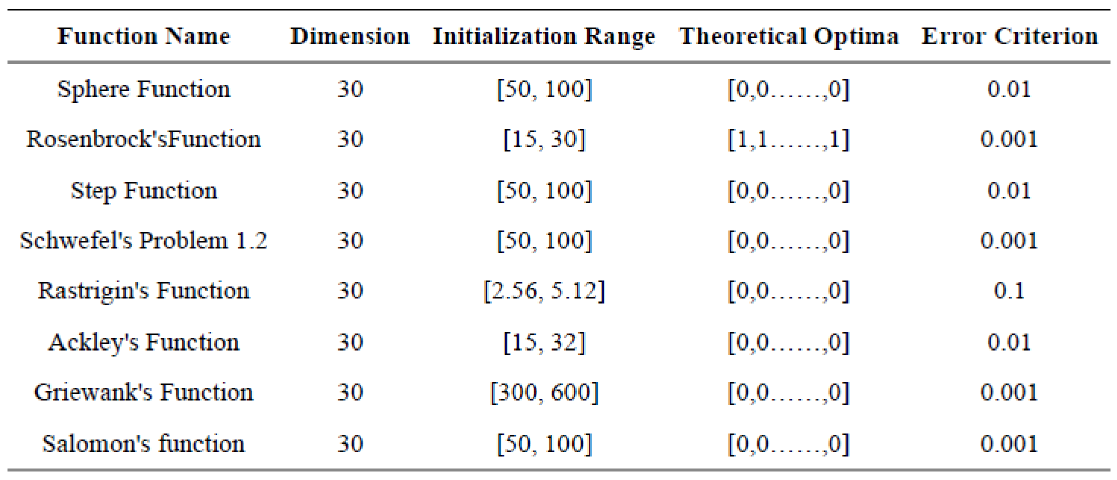
Since most of the benchmarks used here have the global minimum at the origin or very close to the origin of the search space, we use the asymmetric initialization method to observe the performance of the new dynamics. The most common dynamic ranges used in the literature for the benchmarks are used and the same dynamic range is used for all dimensions [11], [21]. Table 3.3 q-lows the range of population initialization and the dynamic range of the search for each function.

3.6.3 Simulation Strategies

Simulations were carried out to observe the rate of convergence and the quality of the optimum solution of the new dynamics introduced here in comparison with the classical PSO dynamics. Alf benchmarks were tested with dimensions 10, 15, 20, 25 and 30. For each function, 50 trials are carried out and the average error and the standard deviation are presented. Use of different stopping criteria for different benchmarks is reported in the literature [5], [20] Therefore different stopping criteria are used here.

Erhart and ,phi [12]indicated that the effect of population size on the performance of the PSO method is of minimum significance. However, it is quite common in PSO research to limit the number of particles to the range 20 to 60 [12] – [14]. Van den Bergh and Engelbrecht suggested that even though there •is a slight improvement of the optimal value with increasing swarm size, it increases the number of fitness function evaluations to converge to an error limit. Therefore, here all experiments were carried out with a population size of 40.

Parametric range of benchmark functions.



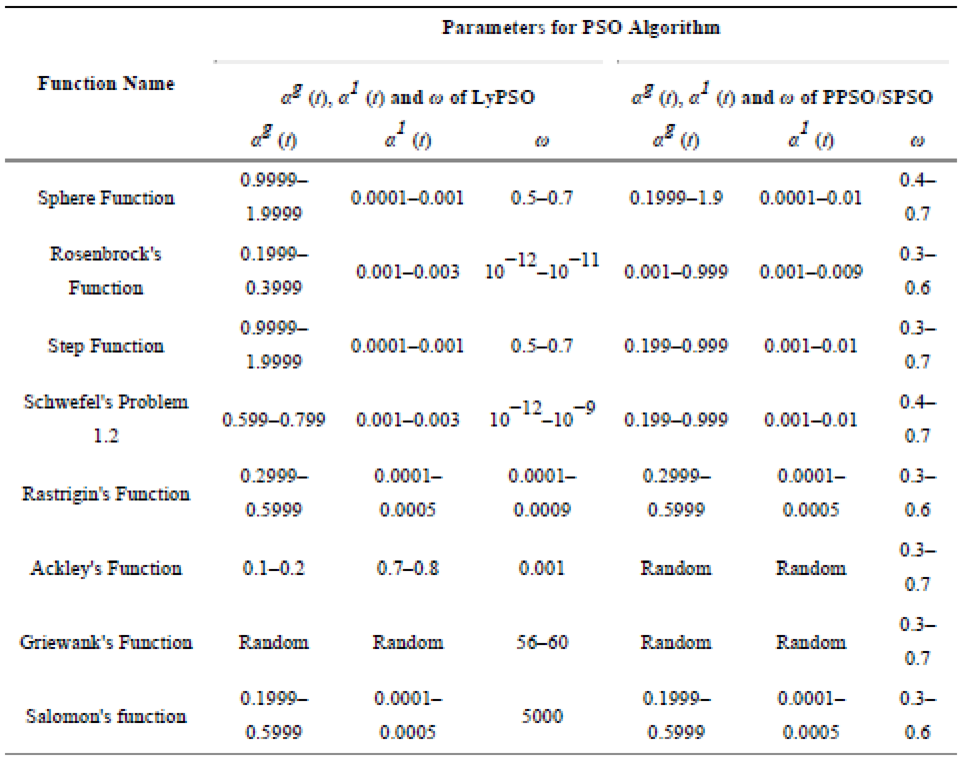
Function Name	Dimension	Initialization Range	Theoretical Optima	Error Criterion
Sphere Function	30	[50, 100]	[0,0……,0]	0.01
Rosenbrock'sFunction	30	[15, 30]	[1,1……,1]	0.001
Step Function	30	[50, 100]	[0,0……,0]	0.01
Schwefel's Problem 1.2	30	[50, 100]	[0,0……,0]	0.001
Rastrigin's Function	30	[2.56, 5.12]	[0,0……,0]	0.1
Ackley's Function	30	[15, 32]	[0,0……,0]	0.01
Griewank's Function	30	[300, 600]	[0,0……,0]	0.001
Salomon's function	30	[50, 100]	[0,0……,0]	0.001

3.5 Results from Benchmark Simulations

Parameter selection of the PSO dynamics also is a crucial issue for speed-up and accuracy of the PSO algorithm. For a given benchmark function, we initially took wider range of the PSO dynamics parameters: a a and co. The initial ranges selected in our simulation were a,, in [0, 4], al in [0, 2J and ω <1. Several hundred runs of the PSO programs with random parameter settings in the above ranges confirm that for a specific function, the best choice of parameters are restrictive as indicated in Table 3,4.

The following observations readily follow from Table 3.4.

Range of optimal values of $\alpha^g(t)$, $\alpha^l(t)$ and ω of LyPSO, PPSO and SPSO.



Function Name	Parameters for PSO Algorithm					
	$\alpha^g(t)$, $\alpha^l(t)$ and ω of LyPSO			$\alpha^g(t)$, $\alpha^l(t)$ and ω of PPSO/SPSO		
	$\alpha^g(t)$	$\alpha^l(t)$	ω	$\alpha^g(t)$	$\alpha^l(t)$	ω
Sphere Function	0.9999–1.9999	0.0001–0.001	0.5–0.7	0.1999–1.9	0.0001–0.01	0.4–0.7
Rosenbrock's Function	0.1999–0.3999	0.001–0.003	10^{-12}–10^{-11}	0.001–0.999	0.001–0.009	0.3–0.6
Step Function	0.9999–1.9999	0.0001–0.001	0.5–0.7	0.199–0.999	0.001–0.01	0.3–0.7
Schwefel's Problem 1.2	0.599–0.799	0.001–0.003	10^{-12}–10^{-9}	0.199–0.999	0.001–0.01	0.4–0.7
Rastrigin's Function	0.2999–0.5999	0.0001–0.0005	0.0001–0.0009	0.2999–0.5999	0.0001–0.0005	0.3–0.6
Ackley's Function	0.1–0.2	0.7–0.8	0.001	Random	Random	0.3–0.7
Griewank's Function	Random	Random	56–60	Random	Random	0.3–0.7
Salomon's function	0.1999–0.5999	0.0001–0.0005	5000	0.1999–0.5999	0.0001–0.0005	0.3–0.6

Observation 1:

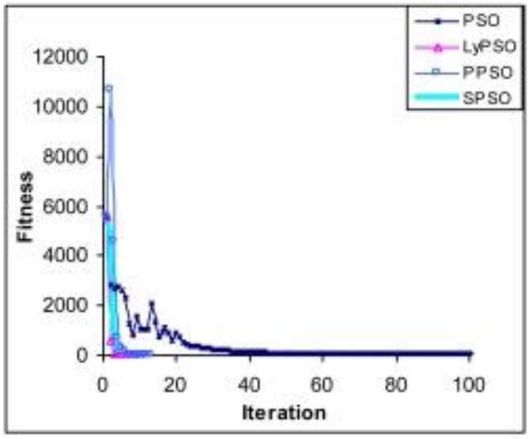
For each benchmark function, the parameter set of the dynamics including a al and co for LyPSO has a relatively restricted range than those of PPSO and SPSO.

Observation 2:

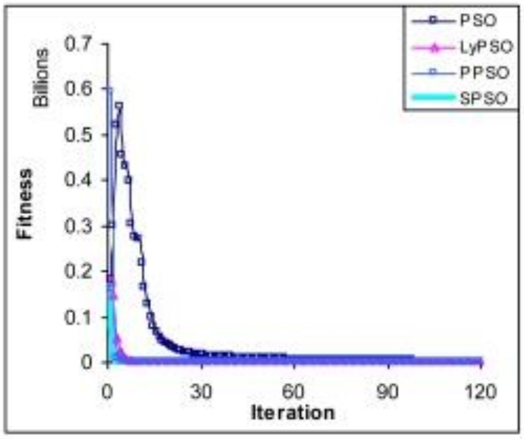
The parameter sets for most of the benchmark functions for the PPSO and SPSO dynamics, have a common range as listed below : α in [0.1999, 0.59991 and ω in [0.0001. 0,0011, The parameter sets for most of the benchmark functions for the PPSO and SPSO dynamics have a common range as listed below: α_g in [0.199, 0.999], α_l in [0.0001, 0.01] and ω in [0.3, 0.6].

The relative comparison of the convergence time of the three algorithms with respect to classical PSO are given in Fig. (3.12) — (3.19). It is observed from these figures that SPSO always outperforms PPSO in convergence time and accuracy. It is further revealed from these graphs that PPSO yields better performance in accuracy and convergence time with respect to both classical PSO and LyPSO. The performance of the four algorithms is summarized with a ' $\leq$' operator, where, x $\leq$ y indicates that performance of y is better than or equal to that of x. Relative performance:

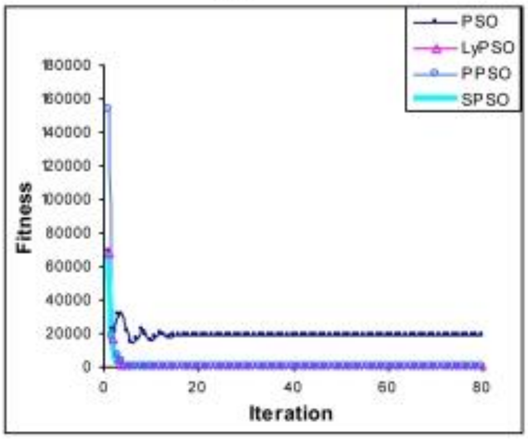
Classical PSO $\leq$ LyPSO $\leq$PPSO $\leq$ SPSO



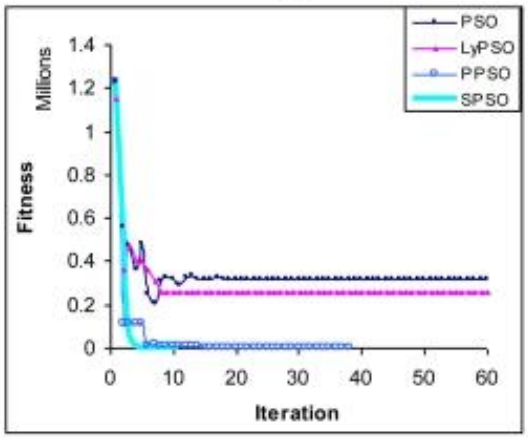
(a)



(b)



(c)



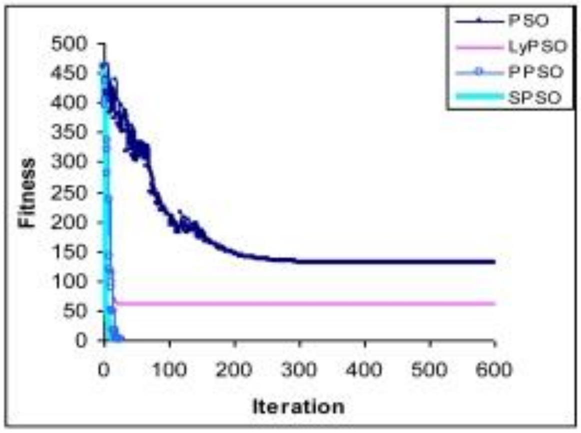
(d)

a)Progress toward the optima : sphere function

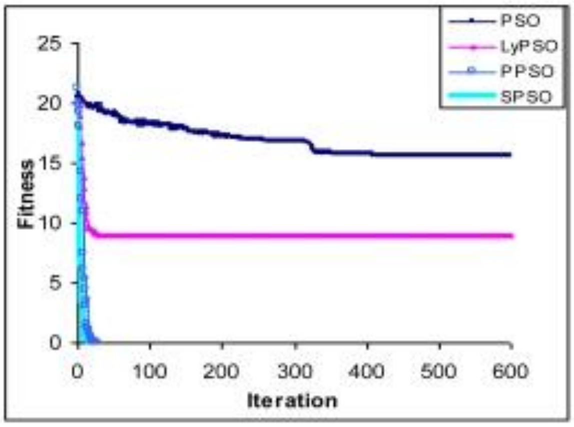
b) Progress toward the optima : Rosenbrock's Function

c) Progress toward the optima : Step Function

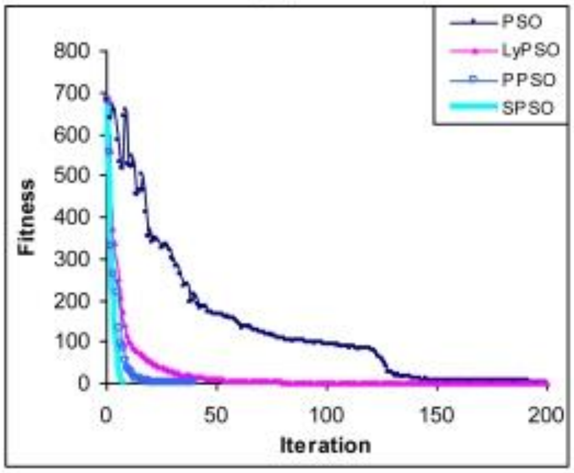
d) Progress toward the optima : Schwefel's Function



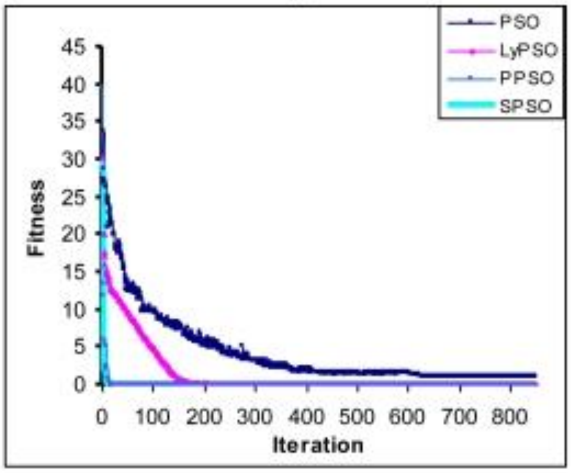
(e)



(f)



(g)



(h)

e) **Progress toward the optima : Rastrigin function**

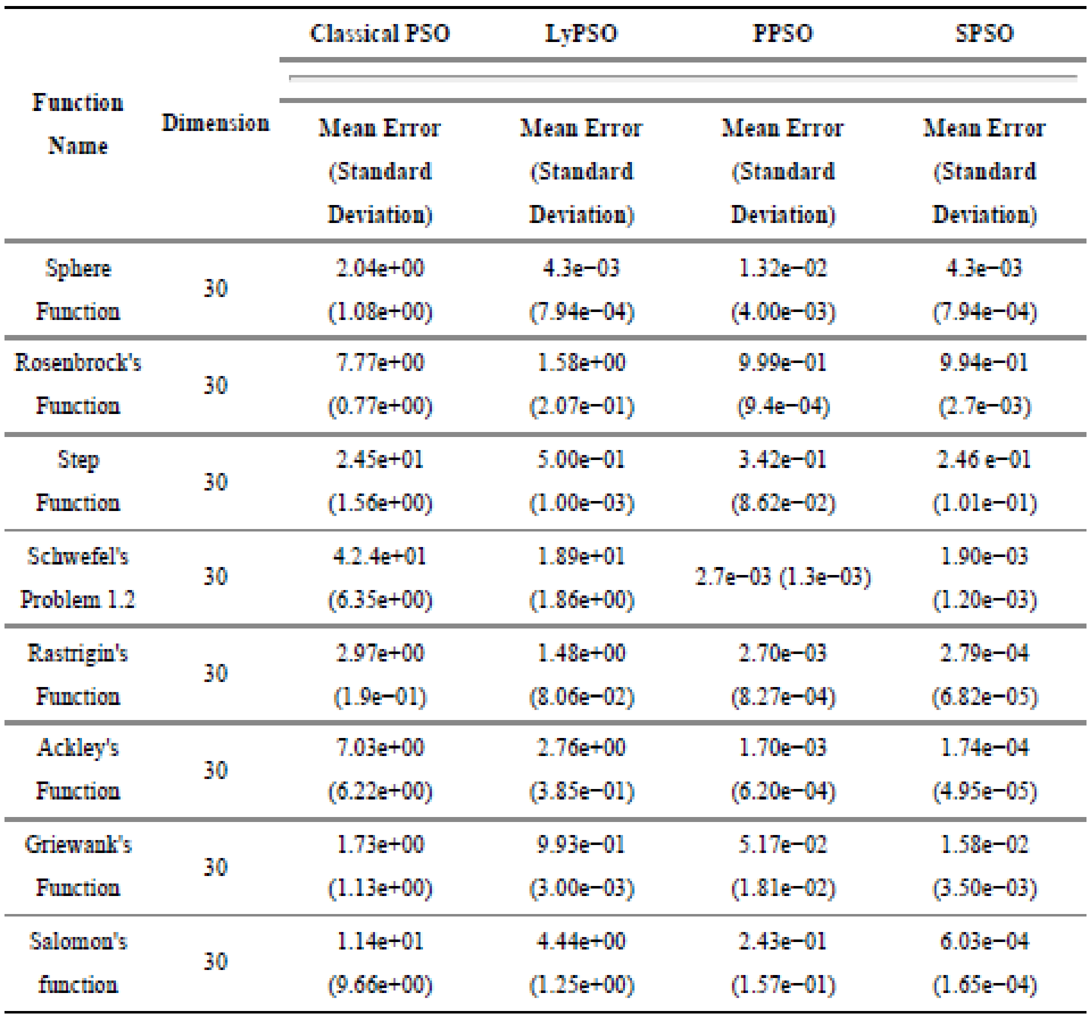
f) **Progress toward the optima : Ackley's function**

g) **Progress toward the optima : Griewank's function**

h) **Progress toward the optima : salomon's function**

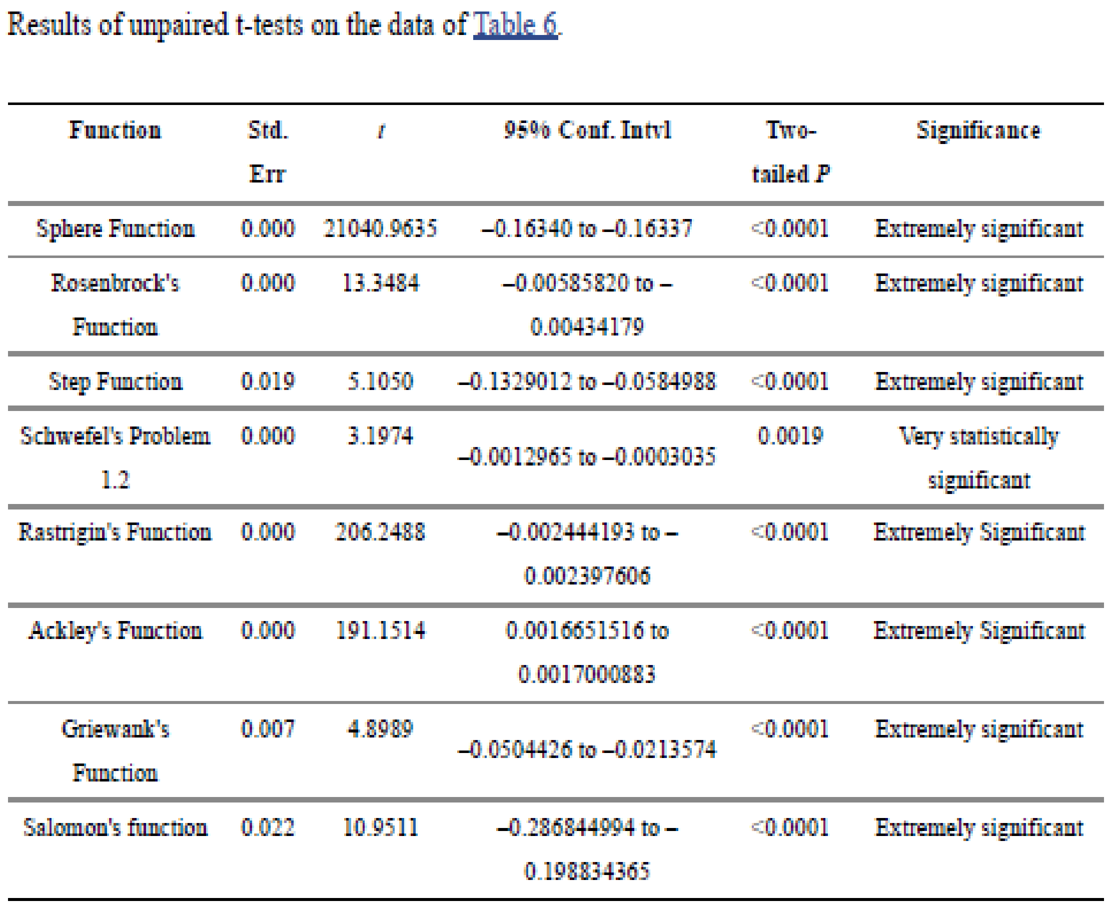
Table 3.5 provides the mean error and standard deviation for the globally best particle obtained by execution of the PPSO, SPSO, LyPSO and classical PS O over eight benchmark functions. The error was obtained by taking the Euclidean distance between the theoretical optima and the position of the best-fit particle for a given program run. The mean error designates the average of errors over 50 independent runs. In order to make the comparison fair enough, runs of all the algorithms were let start from the same initial population. The variance denotes the second moil-lent of the errors with respect to the mean error. It is clear from Table V that for mean error for the SPSO algorithm is comparable but less than that obtained by PPSO algorithm, and the mean error obtained by the PPSO algorithm is insignificantly less than that of LyPSO algorithm, further the mean error obtained by the LyPSO algorithm is less bin comparison to that of the classical PSO algorithm. This confirms that the SPSO algorithm outperforms the PPSO and LyPSO and definitely the classical PSO algorithm from the point of view of accuracy in solution.

Mean error and standard deviation over the benchmarks.



Function Name	Dimension	Classical PSO	LyPSO	PPSO	SPSO
		Mean Error (Standard Deviation)	Mean Error (Standard Deviation)	Mean Error (Standard Deviation)	Mean Error (Standard Deviation)
Sphere Function	30	2.04e+00 (1.08e+00)	4.3e−03 (7.94e−04)	1.32e−02 (4.00e−03)	4.3e−03 (7.94e−04)
Rosenbrock's Function	30	7.77e+00 (0.77e+00)	1.58e+00 (2.07e−01)	9.99e−01 (9.4e−04)	9.94e−01 (2.7e−03)
Step Function	30	2.45e+01 (1.56e+00)	5.00e−01 (1.00e−03)	3.42e−01 (8.62e−02)	2.46 e−01 (1.01e−01)
Schwefel's Problem 1.2	30	4.2.4e+01 (6.35e+00)	1.89e+01 (1.86e+00)	2.7e−03 (1.3e−03)	1.90e−03 (1.20e−03)
Rastrigin's Function	30	2.97e+00 (1.9e−01)	1.48e+00 (8.06e−02)	2.70e−03 (8.27e−04)	2.79e−04 (6.82e−05)
Ackley's Function	30	7.03e+00 (6.22e+00)	2.76e+00 (3.85e−01)	1.70e−03 (6.20e−04)	1.74e−04 (4.95e−05)
Griewank's Function	30	1.73e+00 (1.13e+00)	9.93e−01 (3.00e−03)	5.17e−02 (1.81e−02)	1.58e−02 (3.50e−03)
Salomon's function	30	1.14e+01 (9.66e+00)	4.44e+00 (1.25e+00)	2.43e−01 (1.57e−01)	6.03e−04 (1.65e−04)

Table 3.6 shows results of unpaired t-tests between the best and second best algorithms in each case (standard error Of difference of the two means, 95% confidence interval of this difference, the t value, and the two-tailed P value) . For all cases in Table 4, sample size = 50 and degrees of freedom=98 . It is interesting to see from Tables V and VI that one or more of the proposed PSO methods can always beat the classical PSO in a statistically significant way.

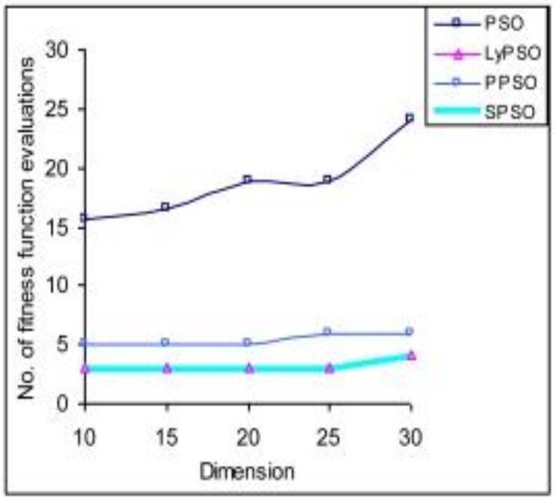


Results of unpaired t-tests on the data of Table 6.

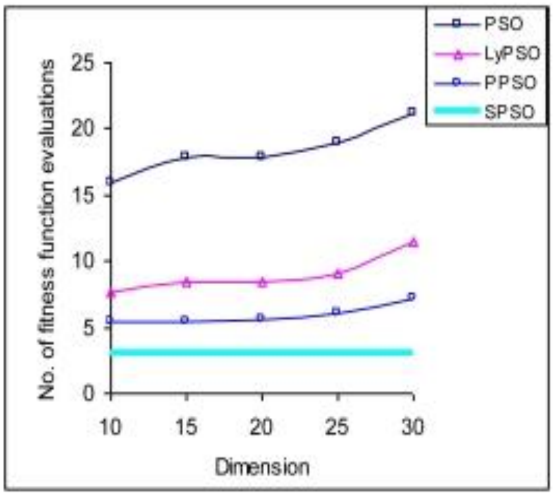
Function	Std. Err	t	95% Conf. Intvl	Two-tailed P	Significance
Sphere Function	0.000	21040.9635	−0.16340 to −0.16337	<0.0001	Extremely significant
Rosenbrock's Function	0.000	13.3484	−0.00585820 to −0.00434179	<0.0001	Extremely significant
Step Function	0.019	5.1050	−0.1329012 to −0.0584988	<0.0001	Extremely significant
Schwefel's Problem 1.2	0.000	3.1974	−0.0012965 to −0.0003035	0.0019	Very statistically significant
Rastrigin's Function	0.000	206.2488	−0.002444193 to −0.002397606	<0.0001	Extremely Significant
Ackley's Function	0.000	191.1514	0.0016651516 to 0.0017000883	<0.0001	Extremely Significant
Griewank's Function	0.007	4.8989	−0.0504426 to −0.0213574	<0.0001	Extremely significant
Salomon's function	0.022	10.9511	−0.286844994 to −0.198834365	<0.0001	Extremely significant

Table 3.6: Results of unpaired t-tests On the data of Table 3.5

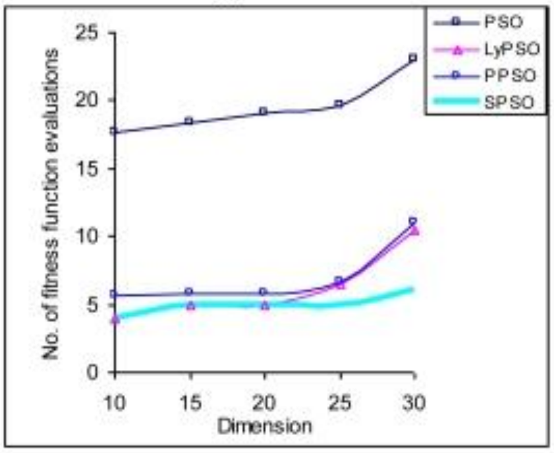
In order to compare the scalability Of the proposed PSO-variants against the growth of dimensionality of the search spaces we need to plot the no. of fitness function evaluations with dimension of the search landscape. The results shown in Figures are average over 50 independent runs of the PSO program, lt is clear from Fig 3.20 - 3.27 that the number of Fitness Function evaluations for PPSO and SPSO do not increase significantly in comparison to that of LyPS0 and classical PS O algorithms.


PSO
LyPSO
PPSO
SPSO
No. of fitness function evaluations
Dimension

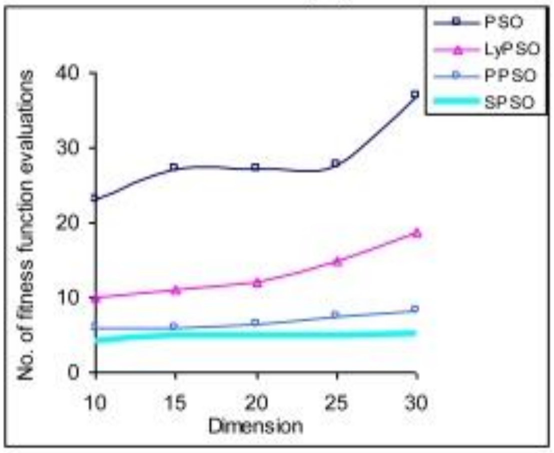
(a)


PSO
LyPSO
PPSO
SPSO
No. of fitness function evaluations
Dimension

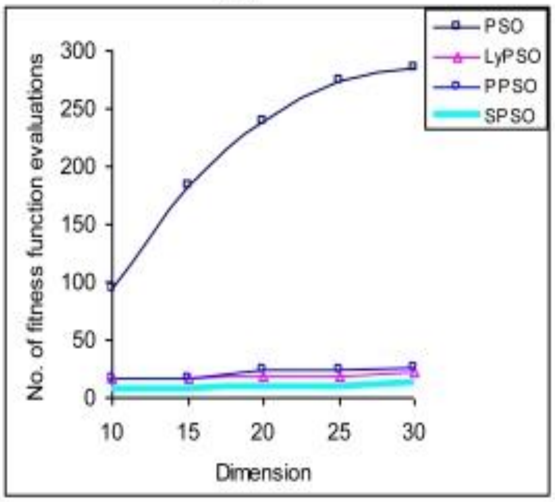
(b)


PSO
LyPSO
PPSO
SPSO
No. of fitness function evaluations
Dimension

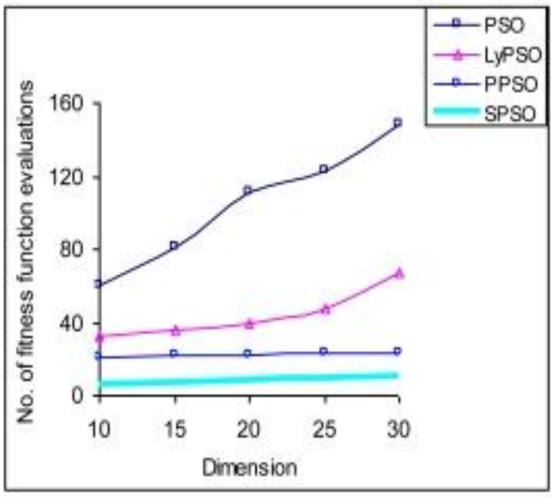
(c)


PSO
LyPSO
PPSO
SPSO
No. of fitness function evaluations
Dimension

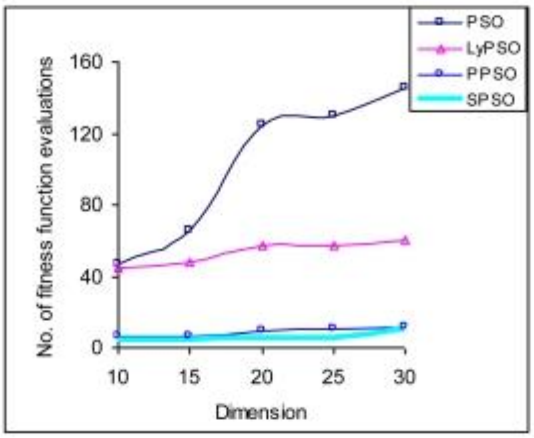
(d)


PSO
LyPSO
PPSO
SPSO
No. of fitness function evaluations
Dimension

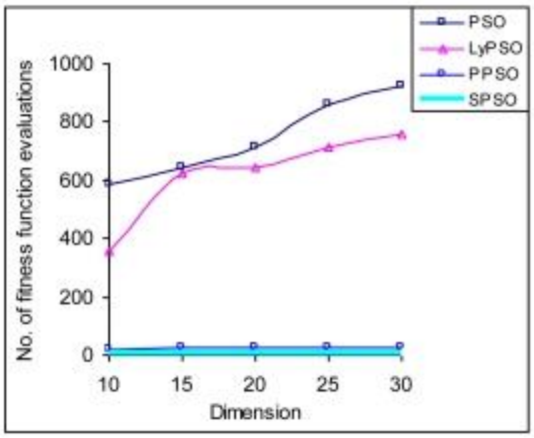
(e)


PSO
LyPSO
PPSO
SPSO
No. of fitness function evaluations
Dimension

(f)



(g)



(h)

Variation in number of fitness function evaluations with function dimension:

(a) Sphere function.

(b)Rosenbrock's function.

(c) Step function.

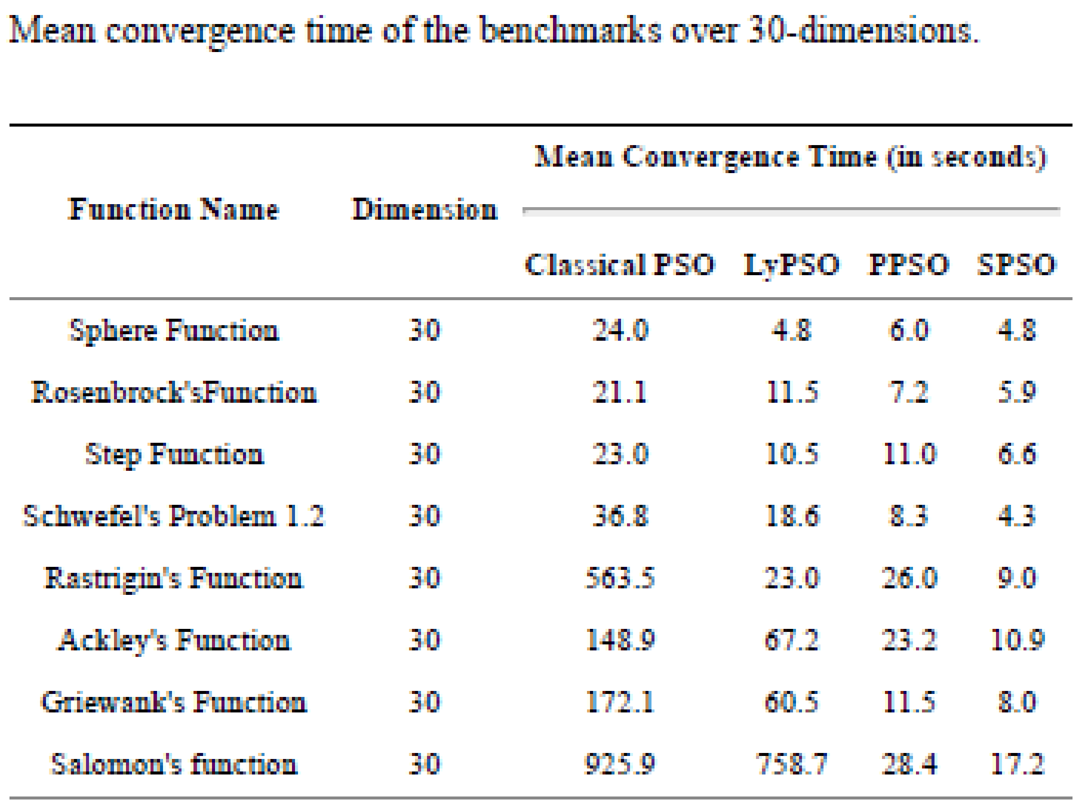
(d) Schwefel's Problem 1.2.

(e) Rastrigin's function.

(f) Ackley's function.

(g) Griewank's function.

(h) Salomon's function.


Mean convergence time of the benchmarks over 30-dimensions.

Function Name	Dimension	Mean Convergence Time (in seconds)			
		Classical PSO	LyPSO	PPSO	SPSO
Sphere Function	30	24.0	4.8	6.0	4.8
Rosenbrock'sFunction	30	21.1	11.5	7.2	5.9
Step Function	30	23.0	10.5	11.0	6.6
Schwefel's Problem 1.2	30	36.8	18.6	8.3	4.3
Rastrigin's Function	30	563.5	23.0	26.0	9.0
Ackley's Function	30	148.9	67.2	23.2	10.9
Griewank's Function	30	172.1	60.5	11.5	8.0
Salomon's function	30	925.9	758.7	28.4	17.2

Table 3.6: Mean Convergence Time of the Benchmarks over 30-dimensions

The PPSO, SPSO, LyPSO and PSO algorithms have been executed on eight benchmark functions, and for each algorithm the average of the convergence time for 50 independent runs to meet the error limit for individual function as specified in Table 3.3 is recorded. It is clear from this Table that the mean convergence time of the SPSO is less than that of PPSO. The mean convergence time of PPSO is less than that of LyPSO, and the latter is less than the mean convergence time of classical PSO. The above phenomena is true for all benchmark functions except the sphere, where the LyPSO and SPSO gives identical results because of same functional form in the SPSO and LyPS() dynamics.

3.6. Conclusion

Classical -hest PSO has a proven impact in optimization of multi-modal nonlinear objective functions However , for many nonlinear continuous multi-modal functions, where partial derivatives with respect to objective function variables exist, classical g-best PSO is not very efficient as it does not utilize gradient information of the search landscape . The paper bridges the gap between the gradient-free and gradient based optimization algorithm .It does not truly utilize gradient information of the search space, but it requires the background information that the gradient of the surface exists. When the prerequisite knowledge about the search space is, known. we extend the classical g-best PSO algorithm by the principles outlined in the paper.

Three alternative approaches to improve the speed of convergence of the PSO dynamics over continuous fitness landscapes is discussed in the paper. The fast approach attempted to re lace he inertial term in the dynamics by a factor that ensures asymptotic stability of the PSO dynamics. Construction of such dynamics presumes the characteristics of the surface being Lyapunov-like. This, however, is not a very restrictive assumption as many multi-modal surfaces support the conditions for Lyapunov function. On the contrary, the Lyapunov-based extension, even without local and global attractors, has a natural tendency to move towards optima on the surface. The convergence of the algorithm to local and global optima, however, is controlled by the presence of attractors in the PSO dynamics.

The second alternative approach to make the PSO smarter was derived from the Lyapunov-based formulation, just by noting that the Lyapunov-based dynamics includes a factor of negatively weighted position of the particle. Incorporation of this new term to the existing velocity adaptation rule classical PSO gives birth to the second alternative form of the extended PSO dynamics. The resulting dynamics has been found to have asymptotic stability for a selective range of $\omega < 1$, i.e. same as in classical PSO. The third extension lies in replacement of the inertial term by the negative position of the particle itself. A random factor is attached to this term to maintain explorative power of the PSO dynamics to avoid its premature convergence. Computer simulations undertaken ensure that the third alternative form of extended PSO dynamics results in significant improvement in convergence time and accuracy compared to the results obtained by the first and second attempt. However, all three approaches outperform the classical PSO dynamics from the point of view of the convergence time and accuracy.

References

1. Eberhart R.C., Kennedy J. A new optimizer using particle swarm theory. Proceedings of the 6th International Symposium on Micromachine Human Science; Nagoya, Japan. October, 1995; pp. 39–43.
[Google Scholar]

2. van den Bargh F., Engelbrecht A.P. A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comp. 2004;8:225–239. [Google Scholar]

3. Shi Y., Eberhart R.C. A modified particle swarm optimizer. Proceedings of IEEE Congress on Evolutionary Computing (CEC'98); Anchorage, AK, USA. May, 1998. [Google Scholar]

4. Clerc M., Kennedy J. The particle swarm-explosion, stability and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. 2002;6:58–73. [Google Scholar]
5. Eberhart R.C., Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings of IEEE Congress on Evolutionary Computing (CEC2000); San Diego, CA,
USA. July, 2000; pp. 84–89. [Google Scholar]

6. Angeline P.J. Using selection to improve particle swarm optimization. Proceedings of IEEE Congress on Evolutionary Computing; Anchorage, AK, USA. May, 1998; pp. 84–89. [Google Scholar]

7. Suganathan P.N. Particle swarm optimizer with neighborhood operator. Proceedings of IEEE Congress on Evolutionary Computing; Piscataway, NJ, USA. July, 1999; pp. 1958–1962. [Google Scholar]

8. Eberhart R.C., Hu X. Adaptive particle swarm optimization: detection and response to dynamic
systems. Proceedings of IEEE Congress on Evolutionary Computation (CEC2002); Honolulu, HI,
USA. May, 2002; pp. 1666–1670. [Google Scholar]

9. Parsopoulos K., Vrahatis M. On the computation of all global minimizers through particle swarm
optimization. IEEE Trans. Evol. Comput. 2004;8:211–224. [Google Scholar]
28/05/2019 A Lyapunov-Based Extension to Particle Swarm Dynamics for Continuous Function Optimization
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3267206/ 22/22

10. Ratnaweera A., Halgamuge S., Watson H. Self-organizing hierarchical particle swarm optimizer
with time-varying acceleration coefficients. IEEE Trans. Evol. Comput. 2004;8:240–255. [Google Scholar]

11. Løvbberg M., Rasmussen T., Krink T. Hybrid particle swarm optimizer with breeding and subpopulations. Proceedings of the 3rd Genetic and Evolutionary Computation Conference (GECCO'01); San Francisco, CA, USA. July, 2001; pp. 469–476. [Google Scholar]

12. Miranda V., Fonseca N. New evolutionary particle swarm algorithm (epso) applied to voltage/VAR
control. Proceedings of the 14th Power Systems Computation Conference (PSCC'02); Seville, Spain.
June, 2002. [Google Scholar]
13. Løvbberg M., Krink T. Extending particle swarms with self-organized criticality. Proceedings of
IEEE Congress on Evolutionary Computation (CEC2002); Honolulu, HI, USA. May, 2002. [Google Scholar]

14. Blackwell T., Bentley P.J. Don't push me! Collision-avoiding swarms. Proceedings of IEEE Congress on Evolutionary Computation; Honolulu, HI, USA. May, 2002; pp. 1691–1696. [Google Scholar]

15. Krink T., Vesterstrøm J. S., Riget J. Particle swarm optimization with spatial particle explosion.
Proceedings of IEEE Congress on Evolutionary Computation (CEC2002); Honolulu, HI, USA. May,
2002. [Google Scholar]

16. Xie X., Zhang W., Yang Z. A dissipative particle swarm optimization. Proceedings of IEEE Congress on Evolutionary Computation (CEC2002); Honolulu, HI, USA. May, 2002. [Google Scholar]

17. Hendtlass T., Randall M. A survey of ant colony and particle swarm metaheuristics and their application to discrete optimization problems. Proceedings of the Inaugural Workshop on Artificial
Life (AL'01); Adelaide, Australia. December, 2001; pp. 15–25. [Google Scholar]

18. Coello C.A.C., Lechuga M.S. MOPSO: A proposal for multiple objective particle swarm optimization. Proceedings of IEEE Congress on Evolutionary Computation (CEC2002); Honolulu, HI,
USA. May, 2002; pp. 1051–1056. [Google Scholar]

19. Coello C.A.C., Pulido G.T., Lechuga M.S. Handling Multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. 2004;8:240–255. [Google Scholar]

20. Agrawal S., Panigrahi B.K., Tiwari M.K. Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch. IEEE Trans. Evol. Comput. 2008;12:529–541. [Google Scholar]

21. Kuo B.C. Automatic Control Systems. Prentice-Hall; Englewood Cliffs, NJ, USA: 1987. [Google Scholar]
22. Kalman R.E., Bertram J.E. Control systems analysis and design *via* the second method of liapunov:
II, discrete time systems. Trans. ASME J. Basic Eng. D. 1960;3:371–400. [Google Scholar]

23. Hahn W. Theory and Application of Liapunov's Direct Method. Prentice-Hall; Englewood Cliffs,
NJ, USA: 1963. [Google Scholar]

24. Ogata K. Modern Control Engineering. Prentice-Hall; Englewood Cliffs, NJ, USA: 1990. [Google Scholar]

25. Angeline P. Evolutionary optimization verses particle swarm optimization: Philosophy and the
performance difference. Proceedings of 7th International Conference on Evolutionary Programming;
San Diego, CA, USA. March, 1998; pp. 600–610. [Google Scholar]

Chapter 4

Application of PSO, PPSO and SPSO age Segmentation

This chapter applies the classical PSO, PPSO and SPSO in image segmentation problem. The algorithm finds the centroids of a user-specified number of clusters, where each cluster groups patterns. The experiments are carried over our images and the performances similar with respect to a cluster validity index shows better results in case of PPSO and SPSO than the classical PSO. Experimental results taken in the form of graphs and tables also show that the mean convergence time of segmentation by PPSO and SPSO are less than that of classical PSO.

4.1 Introduction

Image segmentation or classification is the process of partitioning a digital image into meaningful object regions, so that information can be extracted from it. It is perhaps the most challenging and critical problem in the field of image processing and analysis. Research in this area will probably continue indefinitely long, since any single solution framework is unlikely to produce an optimal result for all possible application domains.

Image classification algorithms can be grouped into two main categories:

1)supervised

2)unsupervised

Supervised classification makes use of a supervised training step to compare the class of a pixel with a target class as provided by an external teacher.

Unsupervised classification, on the other hand ,has no external teacher to provide a target class, Unsupervised image classifiers examine the pixels of an image and group them into a number of clusters. The pixels within a cluster should have similar characteristics, whereas the pixels belonging to different clusters should have different characteristics and should be well-separated. The resultant clusters are referred to as spectral classes, since they are formed based on the natural groupings on the image values. These clusters have no identity, or class label, until compared with some reference data.

In this chapter, we are mainly concentrating on unsupervised image classification by PSO, PPSO and SPSO as introduced earlier in chapter 3.

4.2. Image Classification by Classical and Modified PSO Dynamics

The PSO algorithm consists of a swarm of particles flying through the search space [51. As described in chapter 2, each particle's position is a potential solution to the problem. Each particle's velocity is modified based on its distance from its personal best position and the global best position. In other words the particles move according to their experience and that of their neighbors which yields to the best fitness value. Each particle j maintains the following information [6]:

• X j (t) , the current position of the particle,

• V j (t), the current velocity of the particle,

•P (t) , the personal best position of the particle (pbest); the best position visited so far by the particle, and

• P(t), the global best position of the swarm (gbest); the best position visited so far by the entire swarm.

The objective function evaluates the positions of the particles. Personal best position (pbest) is then obtained as follows [6]:

$$\begin{aligned}\mathbf{P}_j^l(\mathrm{t}+1) &= \mathbf{P}_j^l(\mathrm{t}), &\quad \text{if } f(\mathbf{X}_j(\mathrm{t})) \geq f(\mathbf{P}_j^l(\mathrm{t})) \\ &= \mathbf{X}_j(\mathrm{t}), &\quad \text{if } f(\mathbf{X}_j(\mathrm{t})) < f(\mathbf{P}_j^l(\mathrm{t}))\end{aligned}$$

where f is the objective function. The global best position (gbest) is is obtained as follows [6]:

In chapter 3, we proposed three modified dynamics of the classical PSO, among which we are using the position-based PSO (PPSO) and the Steepest PSO (SPSO) for classification of images. In PPSO, we simply add a term -wx; to the ith component of the updated velocity in the classical PSO. The resulting dynamics then looks like (4.10) — (4.11).

$$v_{j,i}(t+1) = -\omega x_{j,i}(t) + \omega v_{j,i}(t) + \alpha^{l}(t)(p^{l}_{j,i}(t) - x_{j,i}(t)) + \alpha^{g}(t)(p^{g}_{i}(t) - x_{j,i}(t)) \qquad (4.10)$$

$$x_{j,i}(t+1) = x_{j,i}(t) + v_{j,i}(t+1) \qquad (4.11)$$

In SPSO, the inertial term is replaced by $-\omega x_i$. The resulting dynamics looks like (4.12) – (4.13).

$$v_{j,i}(t+1) = \omega v_{j,i}(t) + \alpha^{l}(t)(p^{l}_{j,i}(t) - x_{j,i}(t)) + \alpha^{g}(t)(p^{g}_{i}(t) - x_{j,i}(t)) \qquad (4.12)$$

$$x_{j,i}(t+1) = x_{j,i}(t) + v_{j,i}(t+1) \qquad (4.13)$$

4.3 The Similarity Measure

Clustering is the process of identifying natural groupings or clusters with multidimensional data based on some similarity measure. The most popular way to evaluate a similarity measure is the Euclidean distance. The Euclidean distance between two data points za , zb each having a dimension Nd is given by,

$$d(z_a, z_b) = \sqrt{\sum_{j=1}^{N_d} (z_{a,j} - z_{b,j})^2} = \| z_a - z_b \|$$

Since the image pixels are one dimensional representing their intensity or gray level values, for image segmentation application the similarity measure can be represented by simply the difference between two pixels.

4.4 The Fitness Function

In the context of image clustering, a single particle represents K cluster centroids. That is, each particle xi is constructed as xi = , mi,2 , , m) where m a refers to the k-th cluster centroid vector of the i-th particle. Therefore a swarm represents a number of candidate data clustering. The quality of each particle is measured using the fitness function,

$$f(\boldsymbol{x}_i, \boldsymbol{Z}_i) = \frac{d_{min}(\boldsymbol{x}_i)}{d_{max}(\boldsymbol{Z}_i, \boldsymbol{x}_i)} \tag{4.15}$$

where, $\boldsymbol{Z}_i$ is a matrix representing the assignment of patterns to the cluster of particle i. Each element $Z_{i,k,p}$ represents if pattern $\boldsymbol{Z}_p$ belongs to cluster $\boldsymbol{C}_k$ of particle i. Here,

$$d_{max}(\boldsymbol{Z}_i, \boldsymbol{x}_i) = \max_{k=1,2\cdots,K} \left\{ \frac{\sum_{\forall \boldsymbol{Z}_p \in \boldsymbol{C}_{i,k}} d(\boldsymbol{Z}_p, \boldsymbol{m}_{i,k})}{n_{i,k}} \right\} \tag{4.16}$$

i.e, the minimum Euclidean distance between any pair of clusters. A small value of the fitness function suggests compact and well-separated clusters.

4.5 Classification Algorithm by PSO

Omran et. al. [8] first used PSO in an image classification problem. The PSO clustering algorithm is summarized below:

```
1. Initialize each particle to contain $K$ randomly selected cluster centroids.
2. For $t=1$ to $t_{max}$
    (a) For each particle $i$
        i) For each pattern $Z_p$
            • Calculate $d(Z_p, m_{i,k})$ for all clusters using equation (4.16)
            • Assign $Z_p$ to $C_{i,k}$ where
                $d(Z_p, m_{i,k}) = \min_{\forall k=1,2,\cdots,K} \{ d(Z_p, m_{i,k}) \}$
        ii) Calculate the fitness function $f(x_i, Z_i)$
    (b) Find the personal best for each particle and global best solution.
3. Update the cluster centroids using equation (4.8) and (4.9).
```

In step 3 of the algorithm, the cluster centroids can be computed by equations (4.10) and (4.11) when using PPSO. Equations (4.12) and (4.13) are used to compute the cluster centriods when using SPSO.

4.6 Cluster Validity Measure

The main subject of cluster validation is "the evaluation of clustering results to find the partitioning that best fits the underlying data" [9]. Hence, cluster validity approaches are used to quantitatively evaluate the result of a clustering algorithm 01. These approaches have representative indices, called validity indices. The traditional approach to determine the "optimum" number of clusters is to run the algorithm repetitively using different input values and select the partitioning of data resulting in the best validity measure [9]. Two criteria that have been widely considered sufficient in measuring the quality of partitioning a data set into a number of clusters, are [9]

- **Compactness**: samples in one cluster should be similar to each other and different from samples in other clusters. An example of this would be the variance of a cluster.
- **Separation**: clusters should be well-separated from each other. An example of this criterion is the Euclidean distance between the centroids of clusters.

Some of the well-known indices available in the literature for fuzzy clustering are the partition coefficient [11], partition entropy [12], Xie-Beni index [13], Kwon's index [14], and the PBMF index [15].In the present work we have applied the famous Xie-Beni Index to measure the validity of the classification. The Xie-Beni index is given by:

$$XB_m = \frac{\sum_{i=1}^{K}\sum_{j=1}^{n} u_{ij}^2 \left\|\mathbf{Z}_j - \mathbf{C}_i\right\|^2}{n \times min_{i \neq j} \left\|\mathbf{C}_i - \mathbf{C}_j\right\|^2} \qquad (4.18)$$

where, μ_{ij} denotes the membership of the j-th element to the i-th cluster.

where, Ni denotes the membership of the j-th element to the i-th cluster.

4.7 Experimental Results and Computer Simulations

The PSO-based clustering algorithm has been applied to four images: Horse image, Car image, Cycle image and House image. These images are then used to compare the performance of PPSO and SPSO with that of classical PSO.

4.7.1 The Simulation Strategy

All the algorithms have been developed in Visual C++ on a Pentium IV, 1.2 GHz PC, with 512 KB cache and 2 GB of main memory with Windows Server 2003 environment. For each image data set, each run continues till 200 number of iterations for classical PSO, PPSO and SPSO. Twenty independent runs have been taken for all the algorithms. The results have been stated in terms of the mean best-of-run values and standard deviations over these 20 runs in each case. All the algorithms start from the same initial population each time. Comparisons are made in terms of three performance metrics:

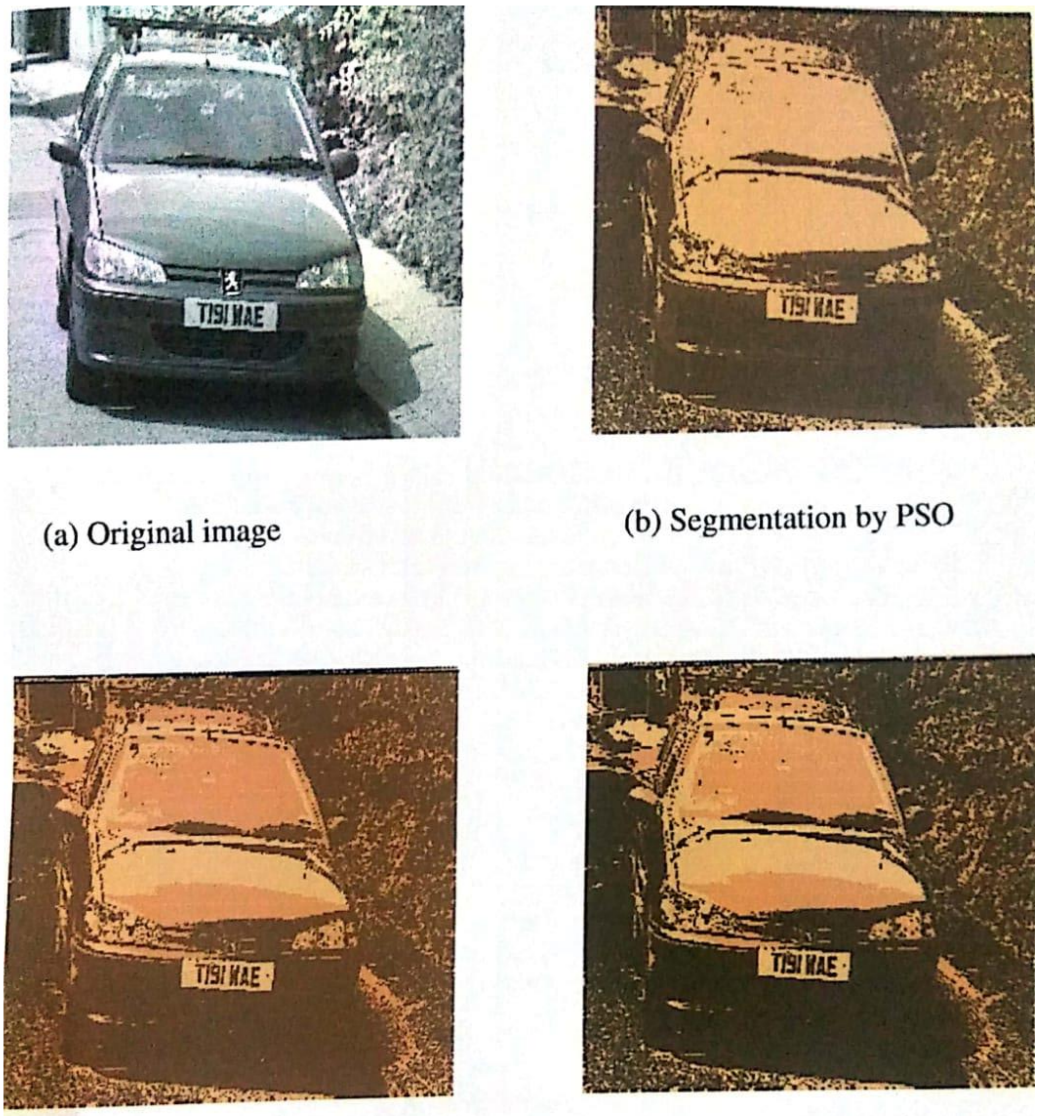
- quality of the solution as determined by the Xie-Beni index and t-test done over it,
- value of fitness function over 10 to 100 iterations for three algorithms, and
- convergence time.

Parameter selection of the PSO dynamics also is a crucial issue for speed-up and accuracy of the PSO algorithm. The values of parameters selected for classical PSO in our simulation were $\alpha^g(t) = \alpha^l(t) = 1.4494$and ω =0.732. For PPSO and SPSO, several hundred runs of the with random

parameter settings, we see that the best choice of parameters are restrictive as indicated in **Table 4.2.**

Image	**Parameters for PPSO/SPSO**		
	$\boldsymbol{\alpha^g(t)}$	$\boldsymbol{\alpha^l(t)}$	$\boldsymbol{\omega}$
Cycle Image	0.7	2.499	0.8
Horse Image	0.1	1.699	0.8
House Image	0.1	1.499	0.8
Car Image	0.1	1.499	0.8

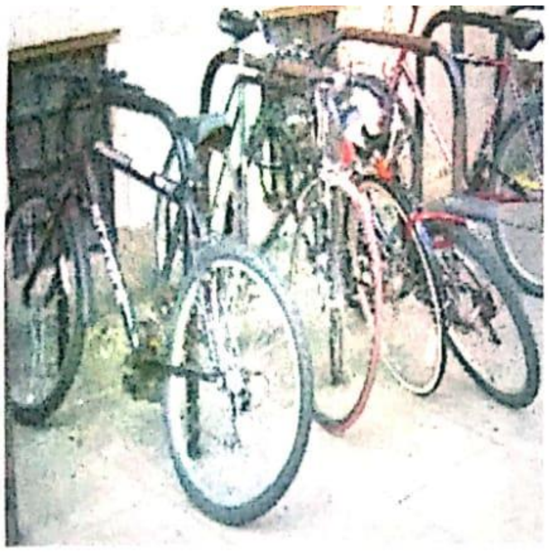
Parameter setting of PPSO/SPSO for classification of four Image



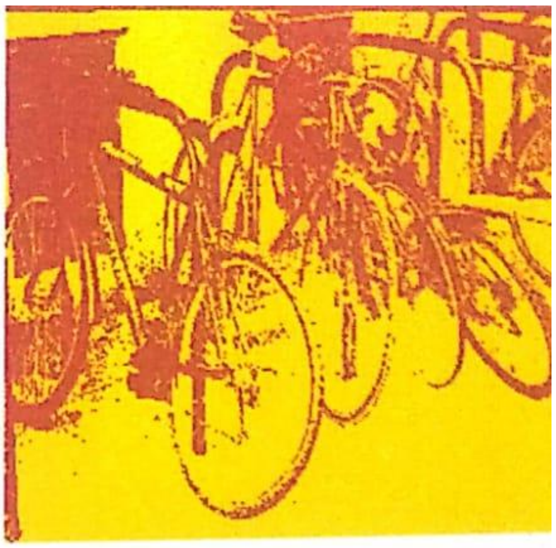
(a) Original image (b) Segmentation by PSO

(c) Segmentation by PPSO (d) Segmentation by SPSO

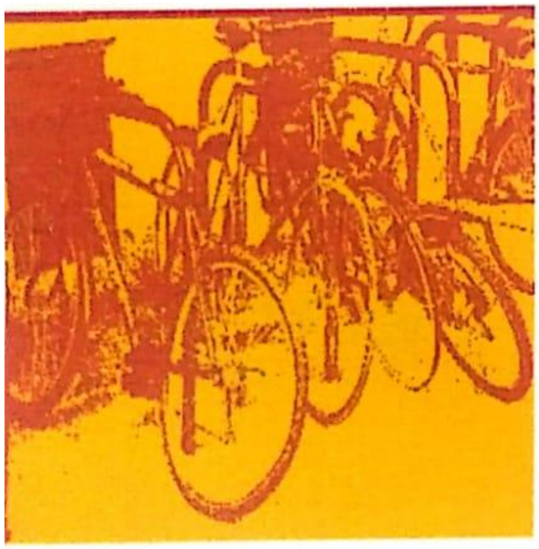
Car Image



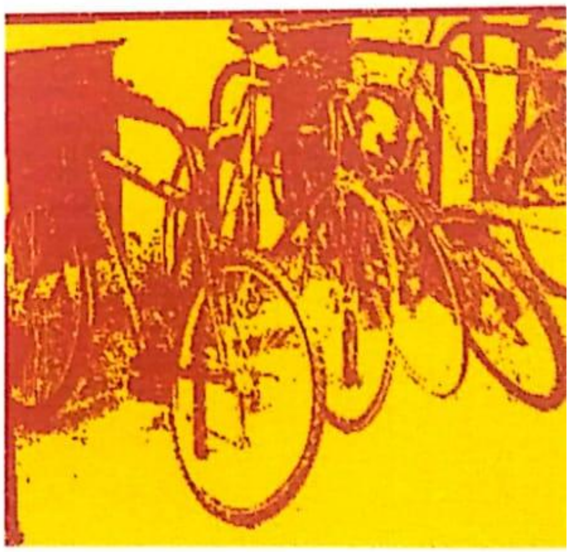
(a) Original image



(b) Segmentation by PSO

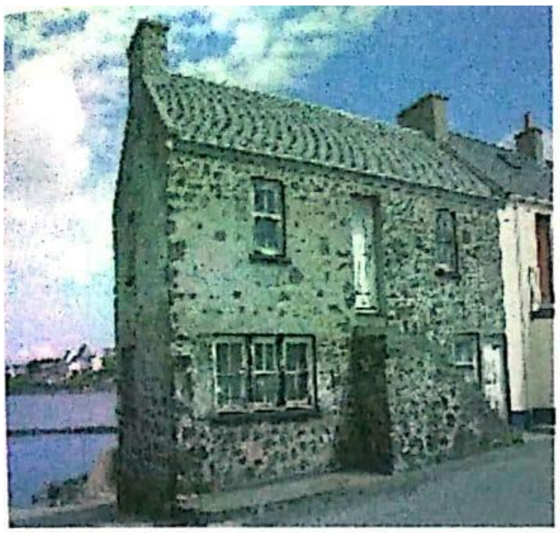


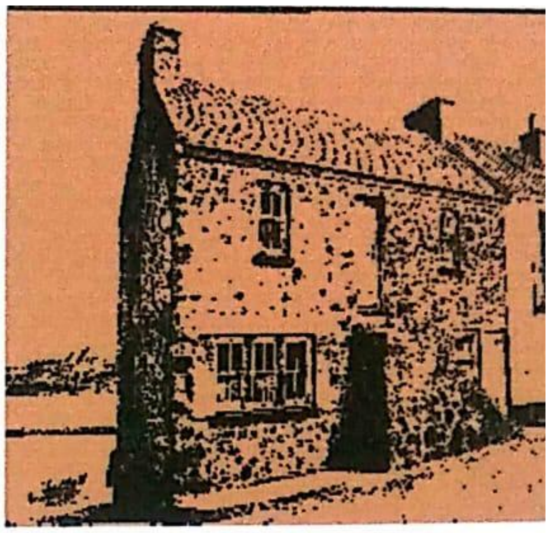
(c) Segmentation by PPSO

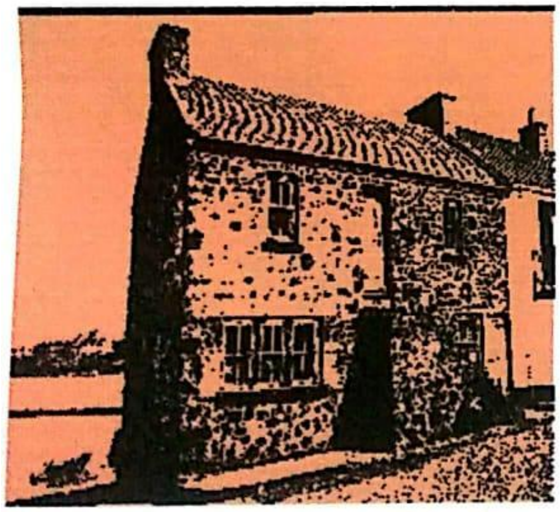


(d) Segmentation by SPSO

Fig. 4.4: Cycle Image


(a) Original image


(b) Segmentation by PSO


(c) Segmentation by PPSO

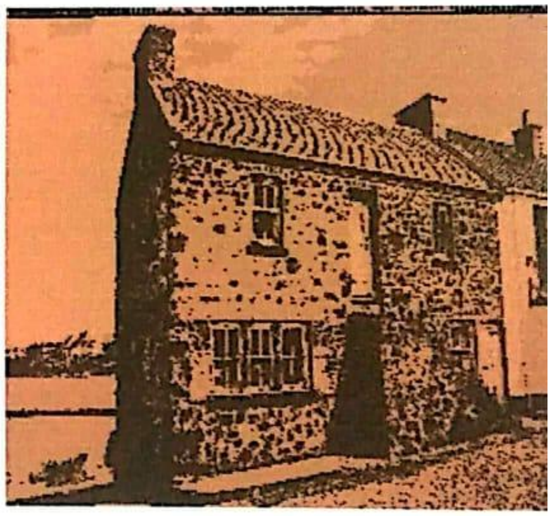
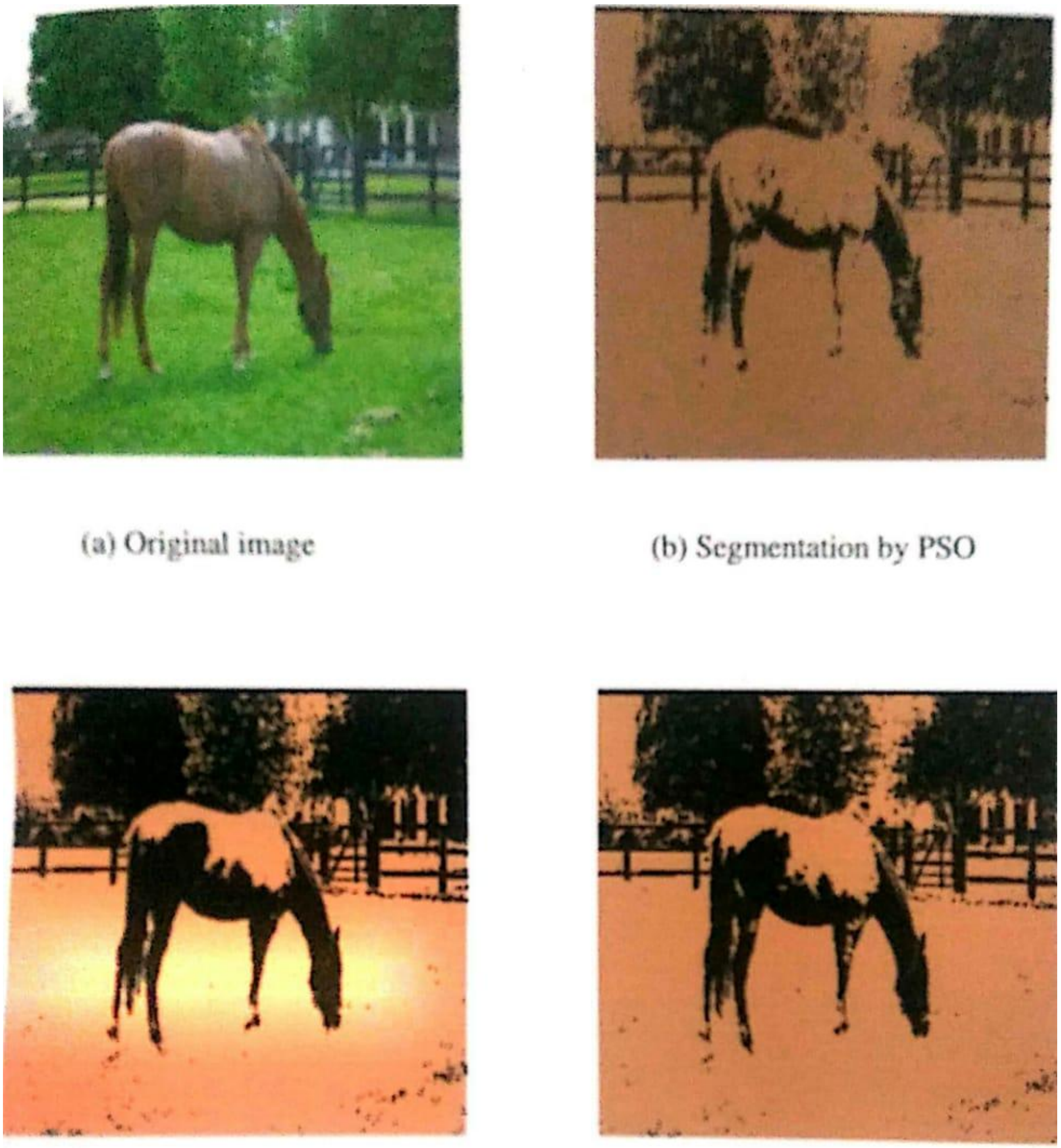

(d) Segmentation by SPSO

Fig. 4.5: House Image



(a) Original image

(b) Segmentation by PSO

(c) Segmentation by PPSO

(d) Segmentation by SPSO

Fig. 4.6: Horse Image

Image	Xie-Beni index for PSO	Xie-Beni index for PPSO	Xie-Beni index for SPSO
Car Image	0.9971 (1.0410)	0.8803 (0.6979)	0.4140 (0.4204)
Cycle Image	0.9770 (0.6120)	0.1924 (0.4751)	0.0370 (0.1012)
House Image	0.8560 (0.7770)	0.5394 (0.6055)	0.1539 (0.2522)
Horse Image	0.8804 (0.9634)	0.5578 (0.5606)	0.1540 (0.2817)

Table 4.3: Mean and standard deviation (in parentheses) of Xie-Beni index calculated on the final clustering results produced by classical PSO, PPSO and SPSO over 20 runs.

Image	Std. Err	t	95% Conf. Intvl	Two-tailed P	Significance
Car Image	0.182	2.5595	0.097493 to 0.835107	0.0146	Statistically significant
Cycle Image	0.1554	2.371	-0.2907 to -0.02013	0.0261	Significant
House Image	0.3855	2.628	0.0886 to 0.6824	0.0123	Significant
Horse image	0.4037	2.828	-0.6911 to -0.1163	0.0076	Very Significant

Unpaired t-test results on the data of the table

Iteration	Fitness by PSO	Fitness by PPSO	Fitness by SPSO
10	3.20e-03 (1.02e-02)	1.13e-04 (2.76e-04)	0.0016 (0.0031)
20	5.23e-05 (1.73e-04)	0.00 (0)	0.00 (0)
30	5.23e-05 (1.73e-04)	0.00 (0)	0.00 (0)
40	5.23e-05 (1.73e-04)	0.00 (0)	0.00 (0)
50	3.51e-05 (1.16e-04)	0.00 (0)	0.00 (0)
60	3.35e-05 (1.11e-04)	0.00 (0)	0.00 (0)
70	3.35e-05 (1.11e-04)	0.00 (0)	0.00 (0)
80	3.34e-05 (1.11e-04)	0.00 (0)	0.00 (0)
90	3.34e-05 (1.11e-04)	0.00 (0)	0.00 (0)
100	3.34e-05 (1.11e-04)	0.00 (0)	0.00 (0)

Table 4.5: Mean and standard deviation of fitness function of the car image over 10 to 100 iterations

Iteration	Fitness by PSO	Fitness by PPSO	Fitness by SPSO
10	3.99e-05 (8.84e-05)	3.73e-05 (1.49e-04)	0.00 (0)
20	1.29e-05 (2.76e-05)	1.00e-06 (3.16e-06)	0.00 (0)
30	1.19e-05 (2.79e-05)	1.00e-06 (3.16e-06)	0.00 (0)
40	1.09e-05 (2.81e-05)	1.00e-06 (3.16e-06)	0.00 (0)
50	1.90e-06 (4.12e-06)	9.00e-07 (2.85e-06)	0.00 (0)
60	1.90e-06 (4.12e-06)	0.00 (0)	0.00 (0)
70	8.00e-07 (1.39e-06)	0.00 (0)	0.00 (0)
80	8.00e-07 (1.39e-06)	0.00 (0)	0.00 (0)
90	8.00e-07 (1.3984e-06)	0.00 (0)	0.00 (0)
100	6.00e-07 (1.35e-06)	0.00 (0)	0.00 (0)

Table 4.8: Mean and standard deviation of fitness function of the horse image over 10 to 100 iterations

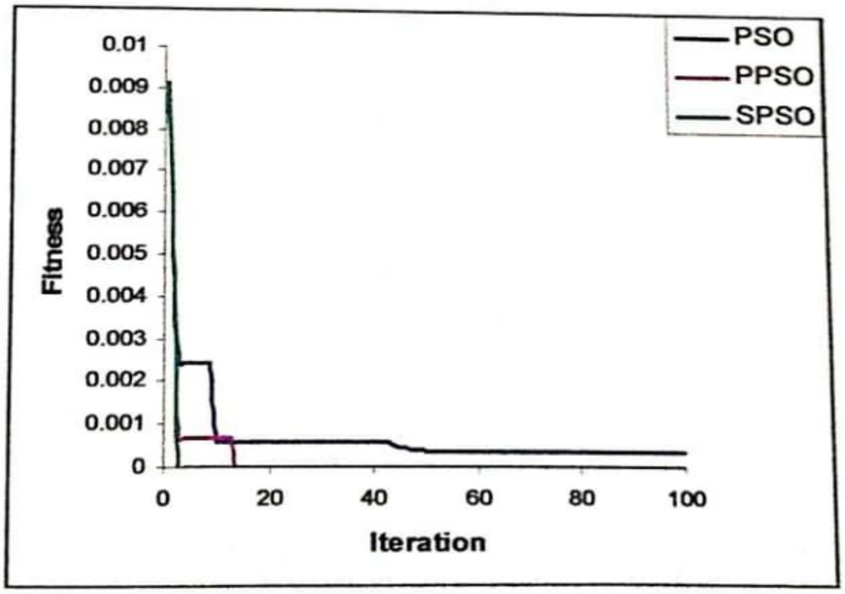


Fig. 4.7: Plot of fitness function vs. iterations for car image

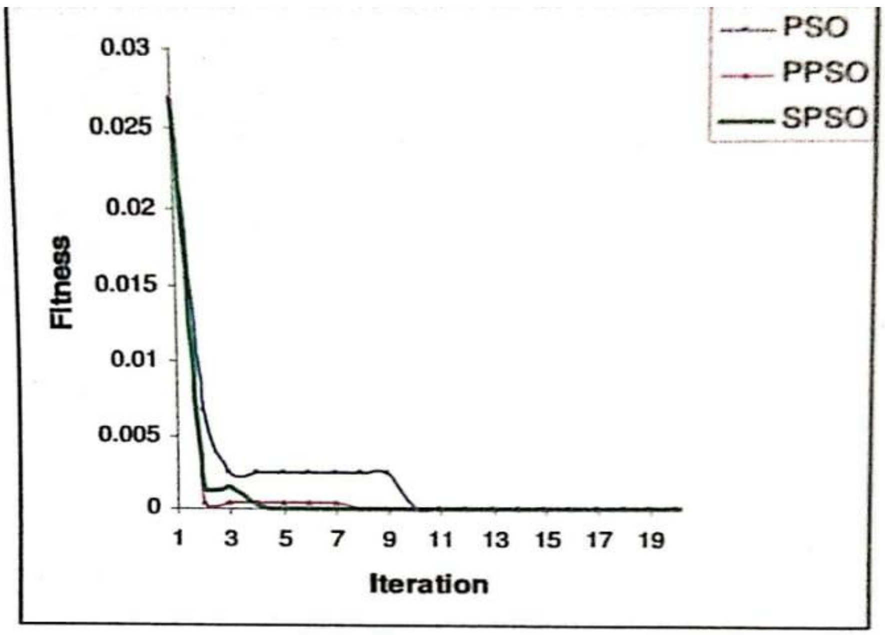


Fig. 4.8: Plot of fitness function vs. iterations for cycle image

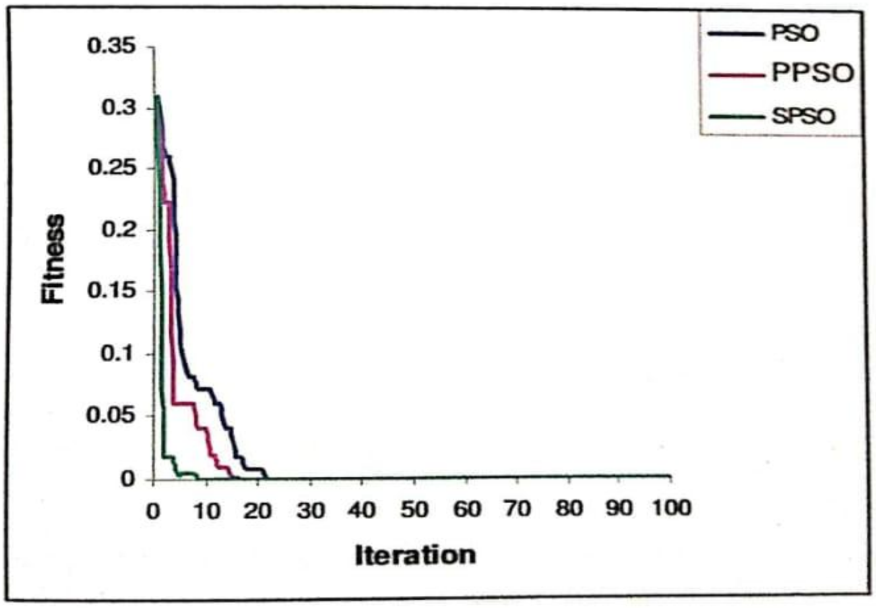


Fig. 4.9: Plot of fitness function vs. iterations for house image

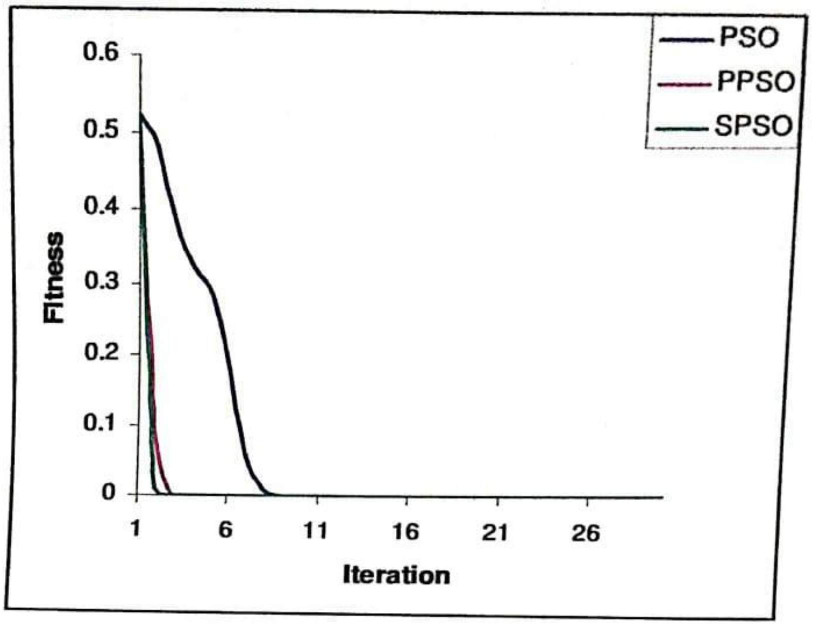


Fig. 4.10: Plot of fitness function vs. iterations for horse image

Image	Mean convergence time (in seconds)		
	PSO	PPSO	SPSO
Horse Image	13.9	7.2	4.4
Car Image	12.2	6.33	4.14
Car Image	23.33	6.9	6.3
House Image	28.0	9.5	9.0

Table 4.9: Mean Convergence Time (in seconds) for Four Images

Table 4.6: Mean and standard deviation of fitness function of the cycle image over 10 to 100 iterations

Tables 4.5 to 4.8 shows the mean and standard deviation of fitness function of the images for classical PSO, PPSO and SPSO over 10 to 100 iterations. The values in the tables indicate that segmentation done by PPSO and SPSO in most cases leads to a better accuracy than the classical PSO. It is also known from these tables that PPSO takes less time to converge than the classical PSO, whereas SPSO takes less time than PPSO. In short, both PPSO and SPSO beat the classical PSO in terms of convergence time and accuracy

We have also plotted the fitness function against number of iterations in figures 4.7 to 4.1.0. These four figures also confirm that the segmentation method done by SPSO converges faster than PPSO. Also we see that segmentation done by PPSO takes less time than the classical PSO. in all four cases, PPSO/SPSO performs better than the classical PSO.

Table 4.9 shows the mean convergence time for the segmentation of four images by the classical PSO, PPSO and SPSO respectively. We see from the table that the mean convergence time of PPSO is better than the classical PSO, whereas, the mean convergence time of SPSO is better than PPSO.

4.8 Conclusion

In this chapter, we classified four images by classical PSO, and two modified PSO dynamics (ppso and SPSO) proposed by us. The Xie-Beni index was chosen as the duster validity index. According to the mean and standard deviation of the Xie-Beni index and the segmented images for these three dynamics, we see that SPSO performs better than PPSO, and PPSO performs better than the classical PSO in terms of accuracy. The t-test done over the Xie-Beni index confirms this. We have also drawn tables containing the fitness function values over 10 to 100 iterations to show that PPSO converges faster than the classical PSO, and SPSO converges faster than PPSO. However, for the segmentation of all four images, segmentation based on our proposed dynamics outperforms the segmentation based on the classical PSO.

References

[1] C. Carpineto, G. Romano, A Lattice Conceptual Clustering System and Its

Application to Browsing Retrieval, Machine Learning, vol. 24(2), 95- 122, 1996.

[2] Bezdek, J. C., Keller, J. M., Krishnapuram, R. and Pal, N. R., Fuzzy Models and Algorithms for Pattern Recognition and Image Processing, Kluwer Academic, MA, 1999.

[3]Hathaway, R. J., Bezdek, J. C. and Hu, Y., "Generalized fuzzy c-means clustering strategies using Li, norm distances," IEEE Trans. on Fuzzy Systems, vol. 8, no. 5, October 2000.

[4]Krishnapuram, R., Segmentation, In Handbook of Fuzzy Computation, Ruspini, E. H., Bonissone, P. P.and Pedrycz, W. (Eds.), IOP Pub. Ltd., 1998.

[5]Kaewkamnerdpong, B. & Bentley, P.J., Perceptive Particle Swarm Optimisation: an investigation, IEEE Swarm Intelligence Symposium, pp. 169-176, Pasadena, CA, USA, June, 2005.

[6]van der Merwe, D.W. & Engelbrecht A.P. , Data Clustering Using Particle Swarm Optimization, 2003 Congress on Evolutionary Computation, 2003, pt. I , Vol. I, pp. 215- 220.

[7]y. Shi, R. Eberhart, Parameter Selection in Particle Swarm Optimization, Evolutionary Programming VII: Proceedings of EP 98, 591-600, 1998.

[8]Omran, M.; Salman, A. & Engelbrecht, A.P. (2002), Image Classification Using Particle Swarm Optimization, 2002.

[9]M. Halkidi, Y. Batistakis, M. Vazirgiannis, On Clustering Validation Techniques, Intelligent Information Systems Journal, Kluwer Pulishers, vol. 17(2-3), 107-145, 2001.

[10] M. Halkidi, M. Vazirgiannis, Clustering Validity Assessment: Finding the Optimal Partitioning of a data set, Proceedings of ICDM Conference, CA (USA), Nov. 2001.

[11] J. C. Bezdek, Numerical taxonomy with fuzzy sets, Journal of Math. Biol. 157-71, (1974).

[12] J. C. Bezdek, Cluster validity with fuzzy sets, Journal of Cybernetics, (3) 58-72, (1974).

[13] X. Xie and G. Beni, Validity measure .for fuzzy clustering, IEEE Trans. Pattern Anal. Machine Learning, Vol. 3, pp. 841-846, (1991).

[1 4] S. H. Kwon, Cluster validity index for fuzzy clustering, Electronic Letters 34 (22) (1998) 2176-2177.

[15] M. K. Pakhira, S. Bandyopadhyay and U. Maulik, A Study of Some Fuzzy Cluster Validity Indices, Genetic clustering And Application to Pixel Classification, Fuzzy Sets and Systems 155 (2005) 191-214.

Chapter 5

Conclusion

This chapter briefly highlights the findings and contributions of this thesis and discusses directions, far future research.

5.1 Summary

In this thesis, we have proposed three alternative approaches to modify the classical g -best Particle Swarm dynamics. The main issues of our research were improving accuracy and convergence speed in global search.

Here, we proposed three modified dynamics inspired by the Lyapunov energy function. The first approach attempted to replace the inertial term in the dynamics by a factor that ensures asymptotic stability of the PSO dynamics. The second alternative was to add the weighted negative position of the particle to the velocity update formula. The third approach was to replace the inertial term by the negative position of the particle itself. A random factor is attached to this term to maintain exploration of the PSO dynamics to avoid its premature convergence. Computer simulations undertaken on the optimization problem of 30 dimensional benchmark functions ensure that the third attempt results in significant improvement in convergence time and accuracy compared to the results obtained by the first and second attempt. However, all three approaches outperform the classical PSO dynamics from the point of view of the convergence time and accuracy.

We also applied the two modified dynamics (PPSO and SPSO) along with the classical PSO to the problem of unsupervised image segmentation. The objective of the algorithm was to minimize the fitness function and a well-known cluster validity index was used to determine the relative performance of these three dynamics. For investigating the performance of the dynamics, four images were taken. In all four cases, the final value of fitness function and the cluster validity index was less in the two modified dynamics than the classical PSO. Also, segmentation done by PPSO and SPSO takes less time than the classical PSO. The results suggest that the performances of the modified dynamics are better than the classical PSO in terms of accuracy and convergence time.

5.2 Future Research

The future research may focus upon an in depth mathematical analysis of the three modified dynamics proposed in this thesis and perhaps it may help us to get more insight into the working principle of these dynamics. We also wish to extend the application part of the proposed dynamics in the field of machine learning and bioinformatics. We are already in the process of testing these dynamics on constrained optimization problems.

A

Appendix A:

Statistical Methods Used

Particle Swarm Optimization algorithm is stochastic in nature. Therefore, it is important to analyze the results from several repeated runs by statistical methods to obtain empirical evidence of the capabilities of a given approach. The fundamental of our data analysis is the assumption that all test runs are independent; i.e, one run does not have any influence on subsequent runs. Appendix A includes a brief review of the statistical methods used to compare the performances of the algorithms.

A.1 Mean

Let, a series of n test runs yield the observations $x_1, x_2, \ldots\ldots x_n$ from the stochastic variables

$X_1, X_2, \ldots\ldots X_n$. Then the sample mean value is computed by

$\mu = x^- = \frac{l}{n}\sum_{i=l}^{n} x_i$

A.2 Standard Deviation

The mean value of n repetitions of an experiment is an indicator of what the expected outcome of this experiment will be. But this mean value does not express how much each of the repetitions is different from the expected "average outcome". For measuring this, we use standard deviation. It is given by

$$\sigma = \sqrt{\frac{l}{n-l}\sum_{i=l}^{n}(x_i - x^-)\hat{}2}$$

. where, x Is the sample mean as defined in A.1. (A.2)

A.3 Confidence Interval

A confidence interval is the probability that a measurement will fall within a given closed interval la; b]. If we assume that a given stochastic variable is Gaussian distributed with the (density distribution function N (μ , σ^2) , we can express the confidence interval in terms of p and .

A.4 t-tests

The t-test, like many statistical tests, assumes that we have sampled data from populations that follow a Gaussian bell-shaped distribution. The t-test assesses whether the means of two groups are statistically different from each other. When we are looking at the differences between scores for two groups, we have to judge the difference between their means relative to the spread or variability of their scores. The t-test does just this. The formula for the t-test is a ratio. The top part of the ratio is just the difference between the two means or averages. The bottom part is a measure of the variability or dispersion of the scores. This formula is essentially an example of the signal-to-noise metaphor in research: the difference between the means is the signal that, in this case, we think our program or treatment introduced into the data; the bottom part of the formula is a measure of variability that is essentially noise that may make it harder to see the group difference.

$$\text{t-value} = \frac{\text{Difference between group means}}{\text{Variability of groups}}$$

Here, ni (i = T or C) denotes number of observations. The t-value will be positive if the first mean is larger than the second and negative if it is smaller. In the t-test, the most important results are the P value and the confidence interval. If the P value is small, then it is unlikely that that the observed difference is due to a coincidence of random sampling. If the P value is large, the data do not give us any reason to conclude that the overall means differ. When the P value is larger than 0.05, the 95% confidence interval will start with a negative number (representing a decrease) and go up to a positive number (representing an increase). To interpret the results in a scientific context, we have to look at both ends of the confidence interval and ask whether they represent a difference between means that would be scientifically important or scientifically trivial.

B

Appendix B: Source Codes

All the source codes, implemented using Visual C++ and MATLAB 7.1 come with the accompanying CD-ROM. Sample runs are given in the Note Pad files bearing same name as that of the source file. The CD contains the following files

1. **PSO code.CPP and PSO_code.txt** — an implementation of the classical PSO algorithm.

2. **LyPSO.CPP and LyPSO.txt** — an implementation of the LyPSO scheme with 8 different benchmark functions.

3. **PPSO.CPP** — an implementation of the PPSO scheme.

4. **SPSO.CPP** — an implementation of the SPSO scheme.

5. **PSO jmage.CPP** — program to segment an image by classical PSO, PPSO or SPSO.

6. **MATLAB m-files implementing the visual plots of 8 benchmark functions in two dimensions.**