IIT – Bombay
Department of Aerospace Engineering
Supervised Learning Project (SLP)
Spring semester
(January – April' 2012)

Implementation and critical parametric analysis of multi-target swarm optimization

by

Pushkar Godbole

Project guide

Prof. Ashok Joshi

Acknowledgement

With utmost sense of gratitude, I acknowledge the guidance extended by Prof. Ashok Joshi, Department of Aerospace engineering, Indian Institute of Technology Bombay. I would also like to deeply thank the authors of the papers and reports I referred to, without which the work would not have been possible.

Department of Aerospace Engineering,

IIT Bombay

Pushkar Godbole

(09D01005)

Abstract

Various swarming techniques have been and are being developed in recent years to solve real life problems in multiple fields, ranging from objective optimization, crowd simulations, networking to collaborative controls. Chief inspiration has often been derived from natural and biological systems. This report deals with one of such swarming algorithms : Particle Swarm Optimization (PSO). It initially explains the working and implementation of the algorithm; and the significance of its formulating parameters in presence of single/multiple target(s). Later, the report contains experimental results of the PSO simulations for investigating the relation between the variation of the acceleration coefficients (tuning parameters) and the performance. Further, the report attempts to formulate this behavior mathematically. Finally it exhibits the key features of the algorithm and its formulation and concludes with the wisdom gained about the choice of parameters for desired performance.

Table of Contents

Nomenclature

PSO : Particle Swarm Optimization

boid : A swarming entity in a PSO instance (equivalent to a bird)

w : Inertia weight

c_best : Individual best weight

gc_best : Global best weight

lc_best : Local best weight

$x_j(i)$: x position of 'j'th boid in 'I'th iteration

$v\_x_j(i)$: x velocity of 'j'th boid in 'i'th iteration

x_best_j : Best x position achieved by boid 'j' yet

gx_best : Best x position achieved by the entire flock yet

lx_best_j : Best x position achieved by the chunk of boids in the locality of boid 'j'

IC 1 : Initial condition 1 : **Boids : (-500, 500), (500, 500), (500, -500), (-500, -500); food : (0,0)**

IC 2 : Initial condition 2 : **Boids : (500, 50), (500, -200), (-50, 500), (-500, -100); food : (0,0)**

IC 3 : Initial condition 3 : **Boids : (500, 500), (500, -500), (-500, 500), (-500, -500) , (400, 0), (0, 400)**
food : (0,0)

List of Figures

1. Introduction

Swarm intelligence (SI) is the collective behaviour of decentralized, self-organized systems, natural or artificial. The concept was primarily employed in work on artificial intelligence. But it is gaining prominent significance in a wide variety of applications. Various Swarm Intelligence algorithms have been developed in recent years. The primary inspiration has often been nature. Some of the notable ones include : Ant Colony Optimization (ACO) which employs the intelligent techniques ants utilize to collaboratively locate food. Artificial Bee Colony (ABC) algorithm, which implements techniques used by bees in searching and collecting nectar. Particle Swarm Optimization (PSO), which uses communication methods used by birds and other swarms to exchange information about food location, and a wide plethora of swarming algorithms rapidly being developed today.

PSO is a relatively new technique that has come to prominence in 1995. But since then, a lot of work on its improvement has been widely carried out. It has exhibited better performance over many other optimization and evolutionary techniques. Its chief advantages are, minimal requirement of initial condition information, discrete optimization, excellent performance and relative ease of implementation.

PSO uses the cumulative information collected by its swarming entities called ‘boids’ to reach the optimum (food, in the case of birds). This information mainly includes the best position of the individual boids, the best position in the entire flock, the local best position in a chunk of the flock, and velocities and positions of the boids at an instant. The boids use this information to collectively reach a point of optimum in the sample space (a map containing food at certain places, in case of birds). Almost the entire performance of the PSO depends on the weights allotted to various criteria mentioned above. The report mainly focus on the qualitative and quantitative performance of the PSO based on these weights (acceleration coefficients) in presence of single and multiple targets.

2. PSO – The algorithm

PSO is one of the Artificial Life based algorithms that employs techniques that base their heart in properties of real life. The natural entities emulated are birds in search of food on a map. PSO was introduced to the world by Dr. Eberhart and Dr. Kennedy in 1995 in their paper "Particle Swarm Optimization". Although initially implemented for simulating social behaviour, it was observed to behave as an excellent optimizer.

Using the basic principles birds flocks/fish schools use to locate food, PSO can work out an optimum for a multi variable constraint optimization problem. The simulation starts with randomly/equally distributed initial positions of boids in the sample space. These boids then exchange information and iteratively edge towards a better position. The conventional PSO primarily uses three criteria to reach the food (optimum).

2.1. Inertia

This is the tendency of a boid to stay in the same state of motion as previous irrespective of the information received. This criterion is given a weight ***w***. *w* is the multiplier of the current velocity of a boid. Setting w to very high will make the system stubborn and indifferent to the new information collected. Whereas setting it too low will kill the continuity in the simulation and hence will lead to loss of information. Inertia weight is thus in a way the level of information retention capacity of the system.

$(Inertia)_j = w \times v_j$ [Inertia for a boid 'j']

2.2. Individual best

This is the tendency of a boid to approach the best point it has reached thus far. It is the conservative nature of a bird that is modelled in this criterion. The weight given to this is called ***c_best***. *c_best* is the multiplier of the present distance of a boid from the its individual best point. Higher the distance, higher will be the contribution. Initially the best position of every boid is its current position. As the boid explores the sample space it finds new and better points. It is then that the individual best point of the boid is updated from the initial to the new better point discovered. Setting c_best too high will kill the explorative behaviour of the boids and the simulation as a whole will tend to get stranded. Setting this criterion too low will kill the importance of the individual best position for a boid and it will then tend to follow a complete herd mentality without having an individual contribution to the entire process.

$(Individual\ best\ contribution)_j = c\_best \times$ *(Distance from the current individual best point for boid 'j')*

2.3. Global best

This is the tendency of a boid to approach the best position yet achieved by the flock as a whole. That is, the best position reached by a boid globally in the entire flock. This is the converging tendency of the flock. This component is responsible for the swarming behaviour of the algorithm. The information exchanged at the end of every iteration of the PSO is this. The co-ordinates of the best position reached so far by the flock. The weight given to this criterion is called ***gc_best***. *gc_best* is the multiplier of the distance of the present position of a boid from the best point reached yet by the entire flock. Higher the distance, higher will be the contribution. Setting gc_best too high will lead to premature convergence of the PSO to the point which is initially the best. Setting it too low will kill the swarming nature of the boids destroying the essence of the algorithm.

$$(\textit{Global best contribution})_j = gc\_best \times (\textit{Distance from the current global best point for boid 'j'})$$

2.4. Formulation

Above three criteria cumulatively define the behaviour of the boids on the map. The sum of the three terms constitutes the velocity of a boid. This velocity is calculated individually for each boid and updated in every iteration. In case of multidimensional functions (map), the velocities for the dimensions are independently computed to avoid interference. Thus in a 2D map, the velocities and distances are computed independently for the x and y axes.

For x axis :

$$v\_x_j(i+1) = w \times v\_x_j(i) + c\_best \times (x\_best_j - x_j(i)) + gc\_best \times (gx\_best - x_j(i))$$

$$x_j(i+1) = x_j(i) + v\_x_j(i+1)$$

The velocities and positions for other dimensions are also computed similarly and updated until all boids reach the food location.

The above formulation works best for single optimum functions (single food location on the map). But real life problems often have multiple optima (multiple food locations). The global-best PSO tends to converge to an optimum too early once it is found, thus diminishing the possibility of finding some potential optima in the unexplored regions of the map.

A candidate solution to this problem is the local-best PSO which introduces a local search tendency in the boids thus enhancing the explorative behaviour of the flock as a whole. The global-best PSO now just becomes a degenerate case of the local-best PSO as we will see below.

2.5. Local best

This is the tendency of a boid to approach the best position achieved by the boids in its locality including itself. This requires defining 'locality'. A locality is defined as the closest 'n' boids to the boid under consideration. This number may vary from the boid itself to the total number of boids. When 'n' is set to total number of boids in simulation, the local-best tendency boils down to global-best.

This component is responsible for local search behaviour of a chunk of the flock. The weight given to this criterion is called ***lc_best***. *lc_best* is the multiplier of the distance of the present position of a boid from the best point reached yet by the local cluster of boids.

$$(Local\ best\ contribution)_j = lc\_best \times (Distance\ from\ the\ current\ local\ best\ point\ for\ boid\ 'j')$$

Implementing local and global-best tendencies simultaneously leads to conflicts in the decisions of individual boids and failure of convergence. Local and global-best are complementary tendencies and need to be implemented exclusive to each other.

The new velocity formulation looks like :

$$v\_x_j(i+1) = w \times v\_x_j(i) + c\_best \times (x\_best_j - x_j(i)) + gc\_best \times (gx\_best - x_j(i)) + lc\_best \times (lx\_best_j - x_j(i))$$

Often the velocity formulation is coupled with some randomness introduced as a multiplier to all the terms. Introducing this randomness increases the explorative tendencies and a healthy deviation from the process flow. The velocity equation with randomness contribution :

$$v\_x_j(i+1) = w \times rand_1 \times v\_x_j(i) + c\_best \times rand_2 \times (x\_best_j - x_j(i)) + gc\_best \times rand_3 \times (gx\_best - x_j(i)) + lc\_best \times rand_4 \times (lx\_best_j - x_j(i))$$

Where $rand_1$, $rand_2$, $rand_3$, $rand_4$ are random numbers between 0 and 1. Also the velocities of boids in the PSO commonly go on increasing in every iteration. This lead to some boids speedily moving away from the solution space. This is avoided by setting an upper bound on the velocity, as has been done in the following implementation.

$$v\_x_j(i+1) = min(v\_x_j(i+1), 2)$$ [For instance, 2 units has been set as the ceiling velocity for the boid in every iteration]

Thus the multi target PSO has been formulated. The best positions computed in the algorithm depend on the fitness function (which is the function to be optimized). This function in case of bird flock simulation is the distance from the food itself. Replacing this distance function with another function will lead to the boids converging at the optima of that function. Thus PSO acts as an optimizer. The behaviour of the algorithm entirely depends on the acceleration coefficients *w, c_best, gc_best* and *lc_best*. This behaviour will be analysed and formulated further in the report.

3. Implementation

The algorithm was implemented in MATLAB. With boid positions, velocities and food position initiated at the beginning. Evaluating the behaviour of the simulation was facilitated by choosing a known optimum (food location). The positions and velocities of boids were updated and dynamically plotted in every iteration. Following simplified flow chart will explain the process :

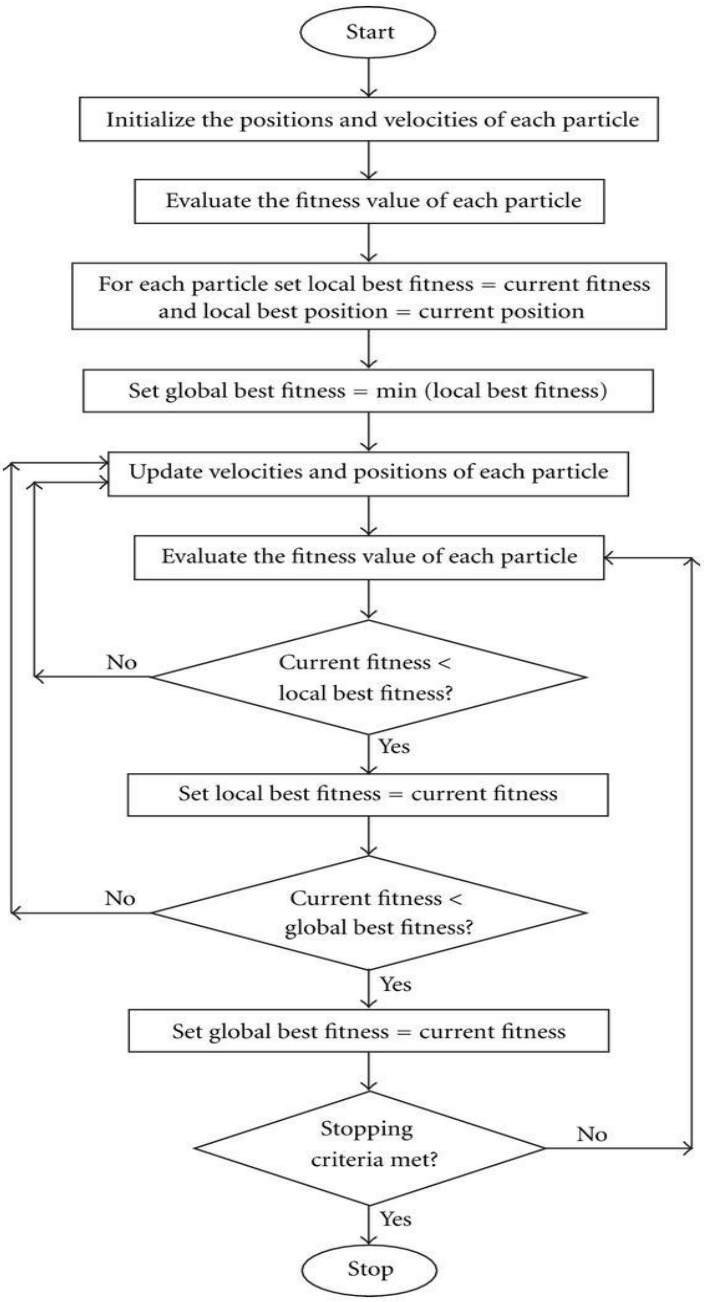


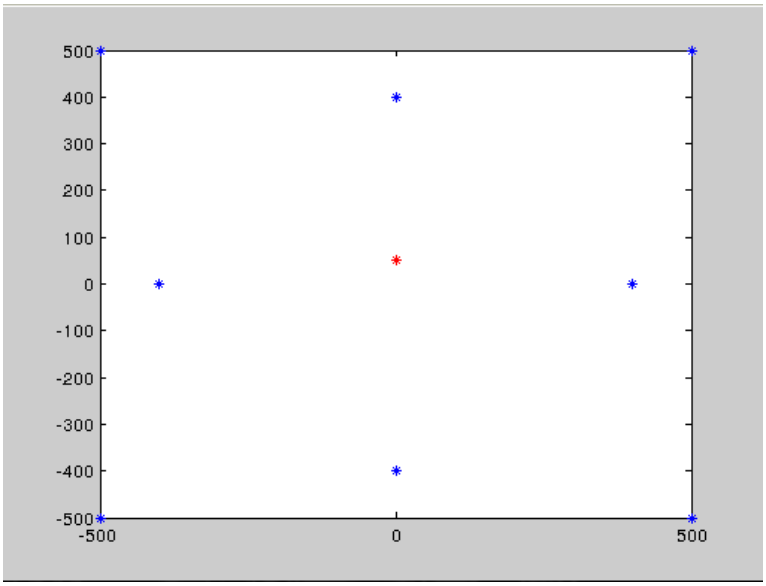
Fig 3.1 : PSO algorithm flow chart

The stopping criterion in this case was the limiting value of 'i' set to 5000 iterations.

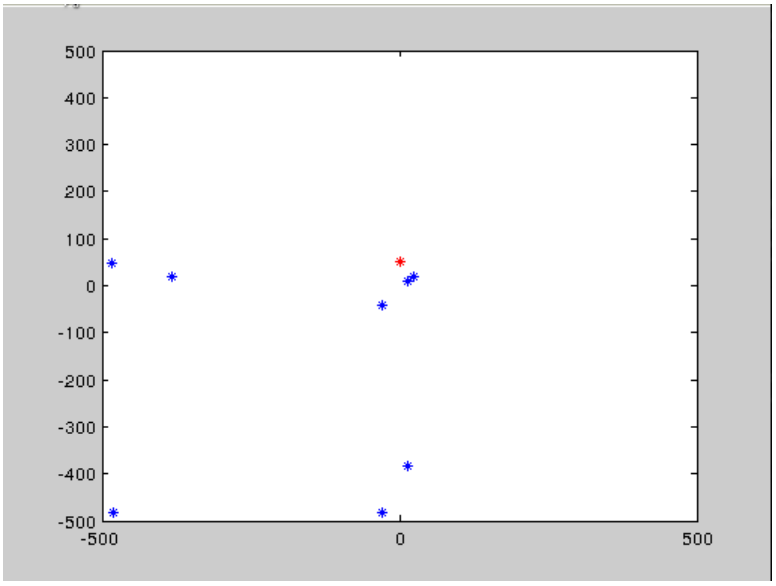
The fitness function chosen in this implementation is the distance from the food. Boid with the least distance from food will thus be the best boid. Initial velocity for all boids is kept at zero to avoid offsets and for better analysis.

3.1. Global best implementation

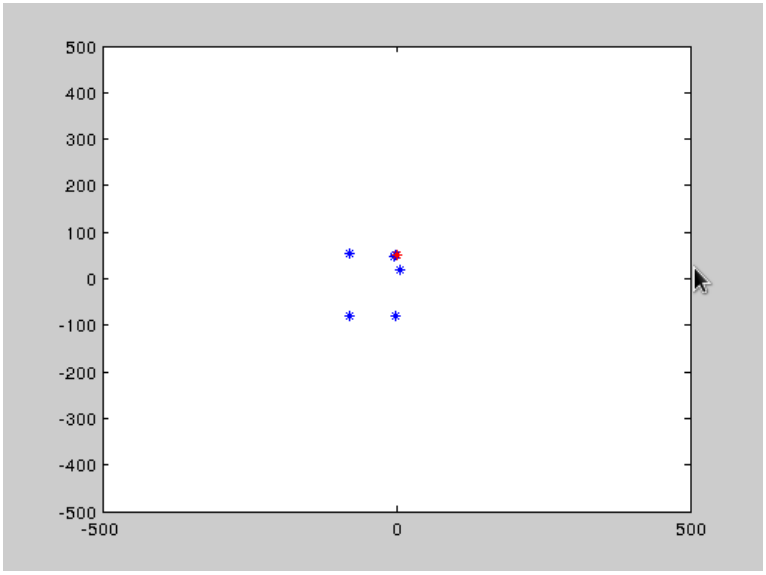
This is a global-best PSO implementation with single food location. Following 4 plots show the progress of PSO with time. The red * indicates the food position and the blue *s are the boids. We can observe that the boids tend towards the food and finally converge around it.



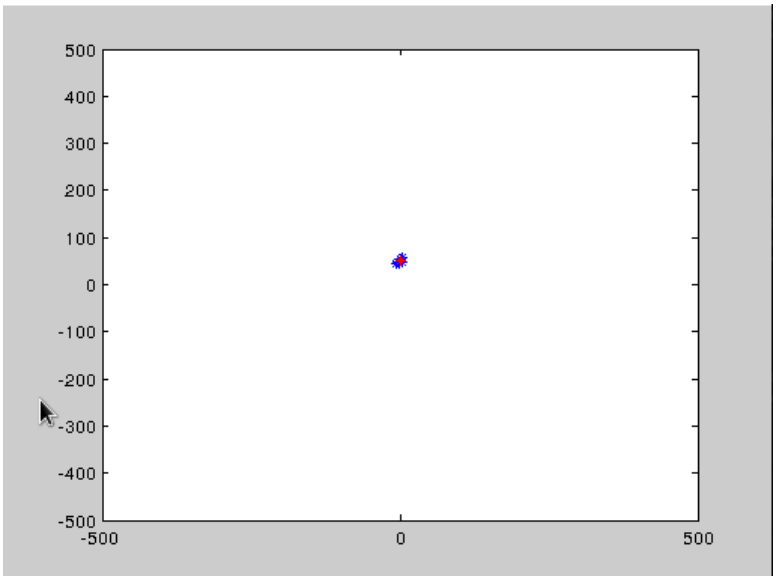
1



2



3



4

Fig 3.2 : Global-best PSO on single food map

Configuration :

w = 1 c_best = 0.05 gc_best = 0.05

Implementing this global-best PSO algorithm on a multiple food location map, leads either to all boids converging to one of the two food locations or some boids not converging. Which implies that, even the boids close to a food but not close enough, will tend to track the food presently tracked by the global-best boid thus travelling a larger distance than necessary. In the following 4 plots with two food locations, we see that the boids on the right also finally converge to the food on the left since that is tracked by the global-best boid. The non-converging boid has its individual best near the food on the right. This is causing the boid to have tendencies to approach this point as well as the gloal best point on the left simultaneously. It has thus settled to an equilibrium point between the two.

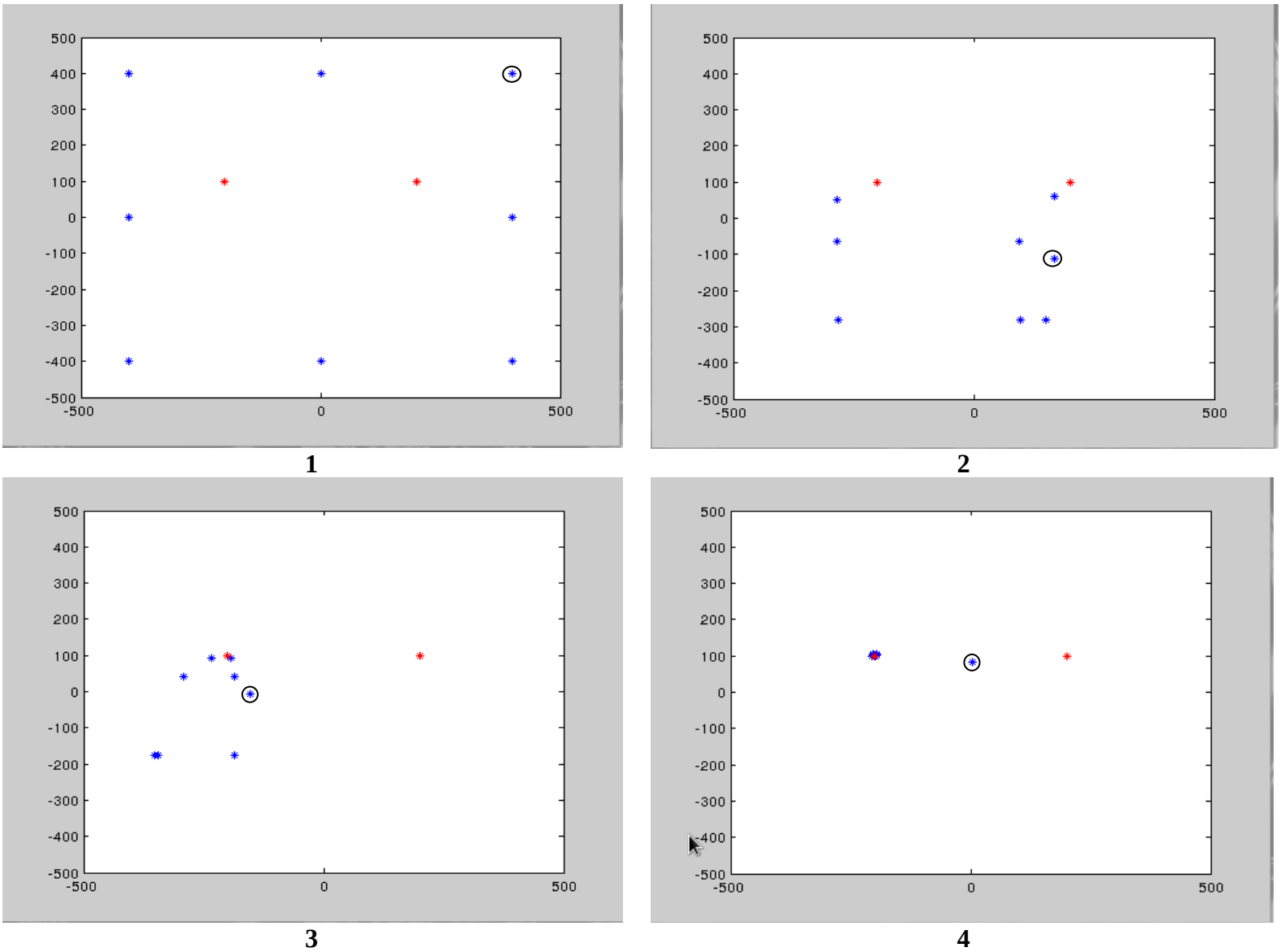


Fig 3.3 : Global best PSO on multiple food map

Configuration :

w = 1 **c_best = 0.05** **gc_best = 0.05**

3.2. Local best implementation

Implementing the local-best PSO along with a global-best contribution leads to conflict between the local-best and the global-best tendencies. The local-best tries to drive a boid towards the best point achieved locally. But a conflict arises when the local best point is not good enough to become a global best point. Thus the boid has tendencies to move towards both points simultaneously. This leads to an incomplete convergence of the boid to either of the two when the weights to these criteria are comparable. The following 4 plots show this dual behaviour leading to non-convergence.

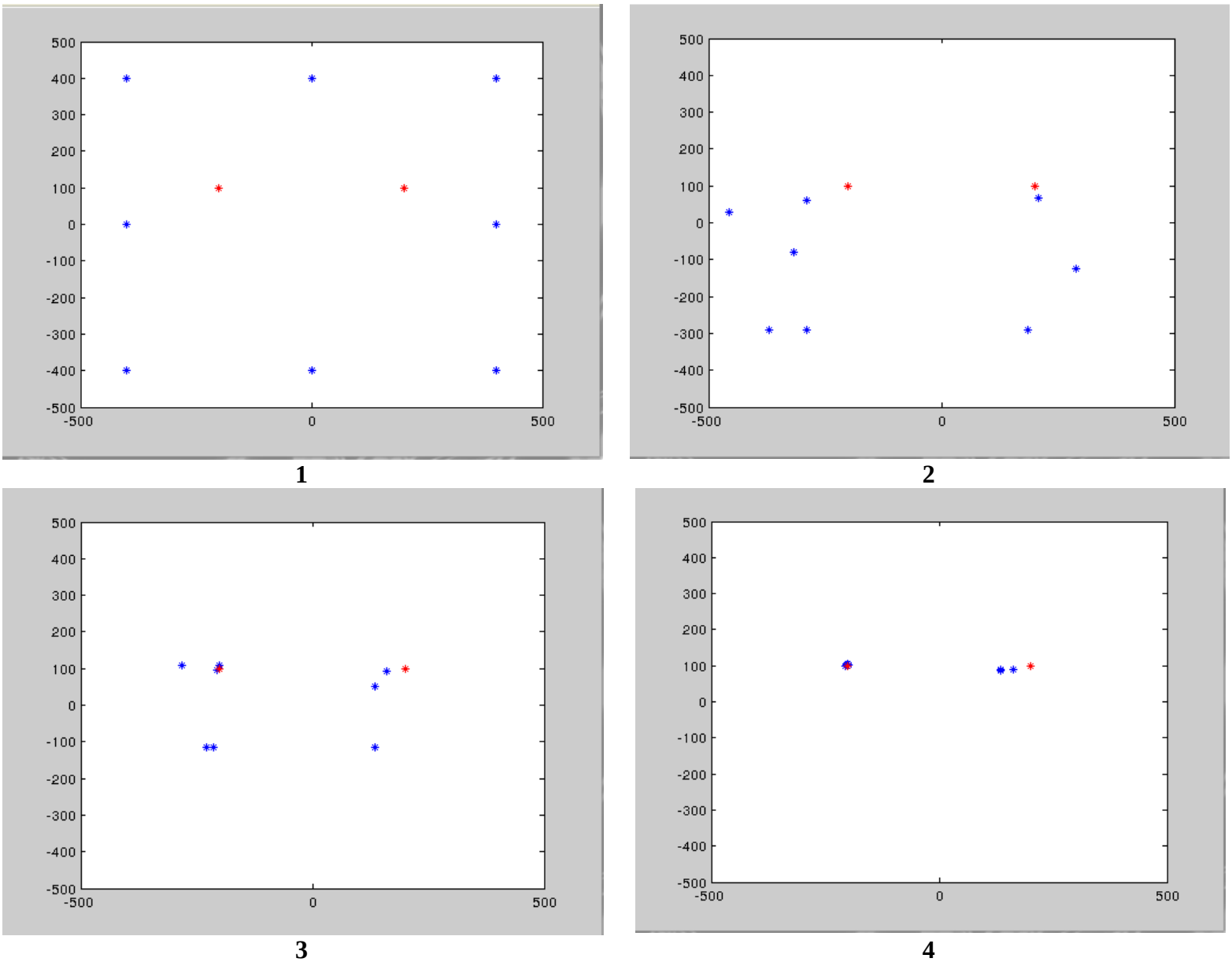


Fig 3.4 : Local-best PSO with global-best contribution on multiple food map

Configuration :

w = 1 c_best = 0.1 gc_best = 0.01 lc_best = 0.1 local = 2

Implementing the local-best PSO alone bypasses these conflicts and leads to complete convergence. The boids on the left now tend to track the left food, while those on the right track the right one. The search space is basically localized to the surroundings of the boids. In the implementations, no weight has been assigned to the food. So both food locations are the global optima. The 'only local-best' implementation fails to converge at the global optimum in case of multiple optima. The boids have no tendency to track the better optimum. We thus observe that neither of the methods are completely suitable individually, for the best PSO convergence. The following 4 plots show the 'only local-best' implementation.

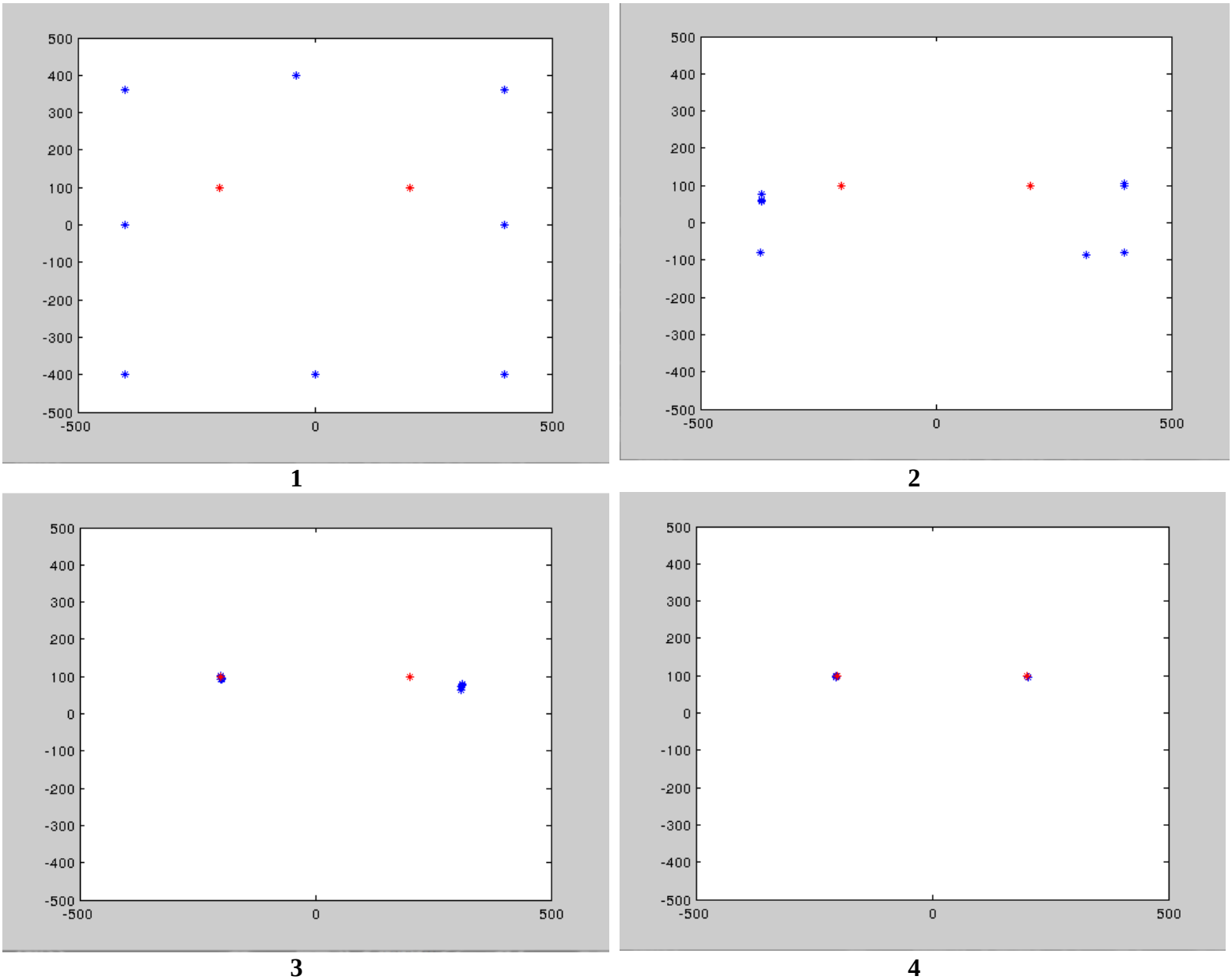


Fig 3.5 : Only local-best PSO on multiple food map

Configuration :

w = 1 c_best = 0.05 gc_best = 0.0 lc_best = 0.05 local = 2

4. Analysis

As observed, complete behaviour of PSO depends on the acceleration coefficients. Since PSO has lesser tuning parameters compared to other evolutionary algorithms, it is easier to implement. But this also assigns high significance to these parameters. So much so that, wrongly setting one of them can lead the PSO to not even initialize. Many attempts have been made to analyse the behaviour and efficiency of a PSO implementation based on these parameters. These attempts have also lead to many favourable modifications to the basic PSO algorithm. The most significant of them include the variable acceleration coefficient implementations. This chapter will comprehensively investigate the impact of the acceleration coefficient values on the behaviour.

We will first consider the single optimum (food location) case and extrapolate it to the multi-target case. Many literatures and papers traditionally suggest that, setting the *c_best* and *gc_best* around 2 give a good and efficient performance on the PSO. Some suggest that maintaining the sum of *c_best* and *gc_best* around 4 lead to a good convergence. Although, critical investigation in this domain is necessary.

Before analysis, we will define what efficiency of convergence means to us. We will evaluate our convergence based on following two criteria :

1) Quality of convergence
2) Speed of convergence.

4.1. Quality of convergence

PSO will never converge to a single point on the map, on account of the continuous and additive nature of its formulation. We may observe that, the inertia term in the formulation maintains the initial velocity of a boid in every iteration. Any modification done to its velocity is over and above this initial. The boids will essentially remain in a proximity region around the optimum, once an equilibrium is attained. We will define this proximity region by its radius. The **quality of convergence** of PSO will thus be defined by the radius of this proximity region. We define this radius as the distance of the farthest boid after the equilibrium is reached.

In the implementation, it is taken as the maximum of the farthest distance in every iteration, over the last 1000 iterations. The boids keep oscillating in this proximity region about the best position they have located.

Following is a plot of the distance of the farthest boid in every iteration for :
c_best = 0.1 and ***gc_best = 0.5***.
8 boids initialized symmetrically from corners and edges of a square :
(-400, 400), (0, 400), (400, 400), (400, 0), (400, -400), (0, -400), (-400, -400), (-400, 0)
and food placed at **(0,0)**.

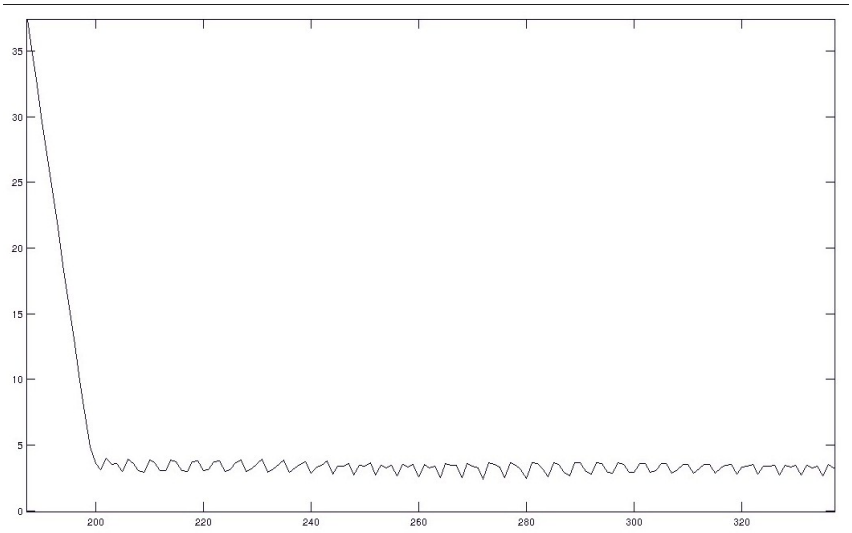


Fig 3.6 : Farthest distance plot for an instance of global-best PSO

In the figure, we see that the boids converge in a proximity region with approximate radius of 3 units. The distance of the farthest boid keeps oscillating within this radius. An analogy of planetary capture can be give in this context. The food here acts as the planet attracting the boids. When they come in proximity of the food, their initial velocities drive their oscillatory behaviour around the food.

To investigate the effect of *c_best* and *gc_best* on the convergence radius, this radius was calculated for multiple implementations of the global-best PSO varying the values of *c_best* and *gc_best* in the process. Both ***c_best*** and ***gc_best*** were varied from **0 to 5 in steps of 0.05**, 4 boids were initialized from 4 corners of a square : **(-500, 500), (500, 500), (500, -500), (-500, -500)** with food at **(0,0)**; and the convergence radius was calculated in each case. **(Initial condition 1)**. Following is the plot of the convergence radius v/s *c_best* and *gc_best* :

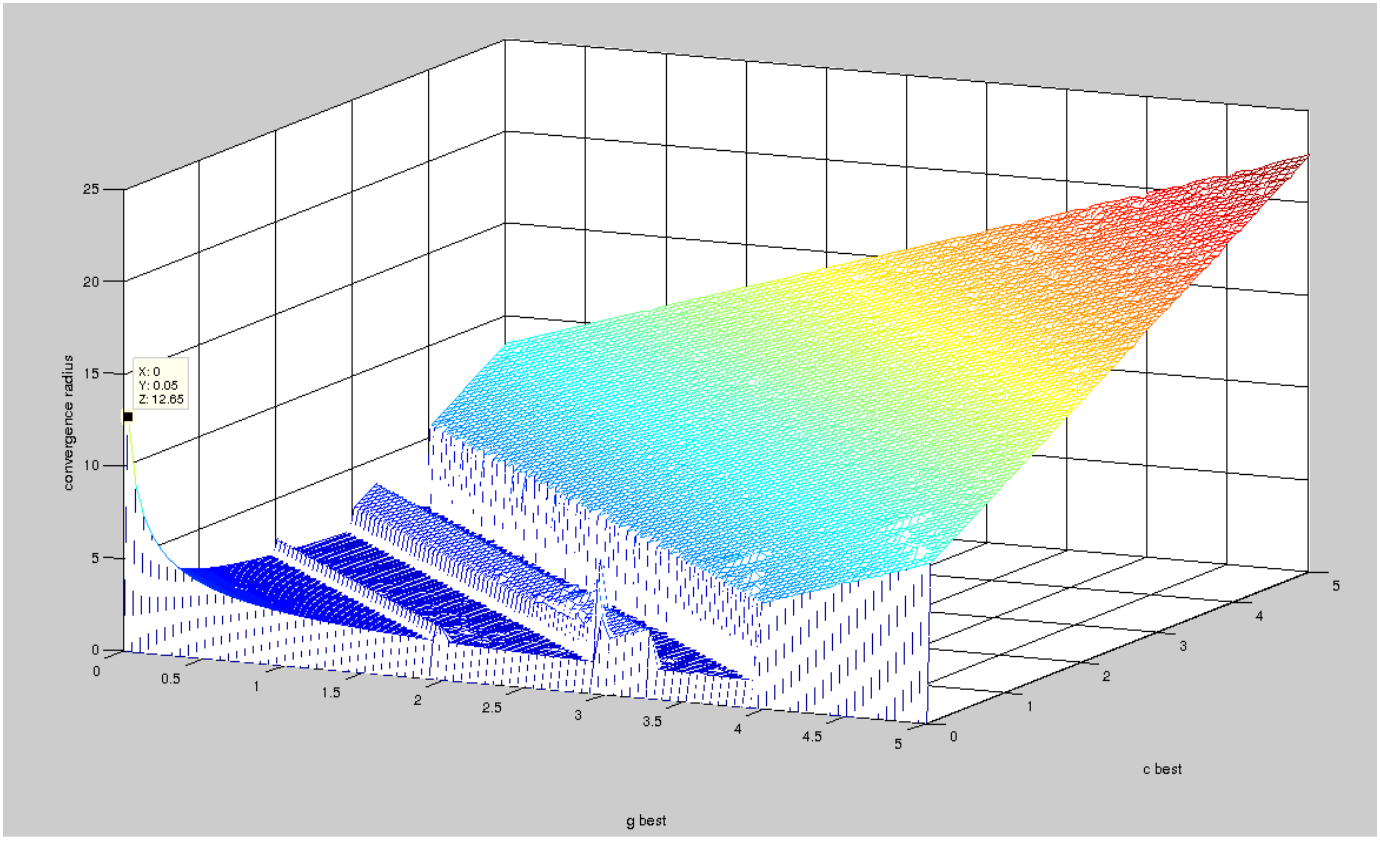


Fig : 3.7 : Convergence radius v/s *c_best* and *gc_best* *(Initial condition 1)*

It needs to be noted that, setting *gc_best* to 0, would kill the tendency of the boids to move in the first place. Since initial velocities for all boids have been set to zero, the boids will remain in their original position. This original position will also be their individual best position and the contribution from *c_best* too will remain 0. Without a driving factor, the PSO will be dead. Therefore a non-zero *gc_best* is necessary for a meaningful PSO implementation. The convergence radius calculated by matlab for all *gc_best* = 0 implementations was thus the initial maximum distance i.e. $500\sqrt{2}$ = 707.11. Those were made equal to 0 for better viewing the plot.

In order to ensure that the convergence radius is independent of the initial conditions, it was implemented for multiple initial configurations.

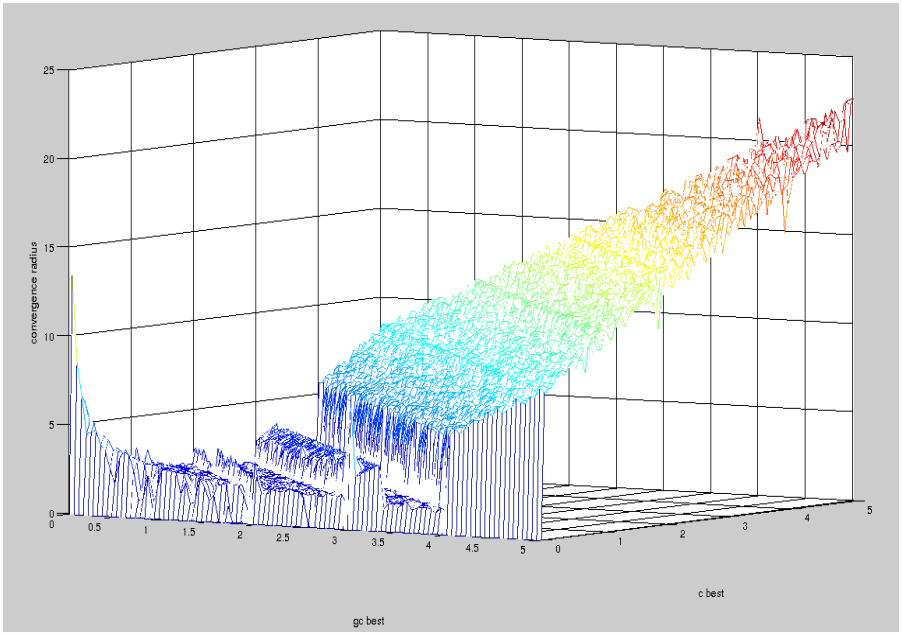


Fig 3.8 : Convergence radius v/s *c_best* and *gc_best* *(Initial condition 2)*

4 boids **(Initial condition 2)**

Initial boid positions :

(500, 50), (500, -200), (-50, 500), (-500, -100)

Food = **(0, 0)**

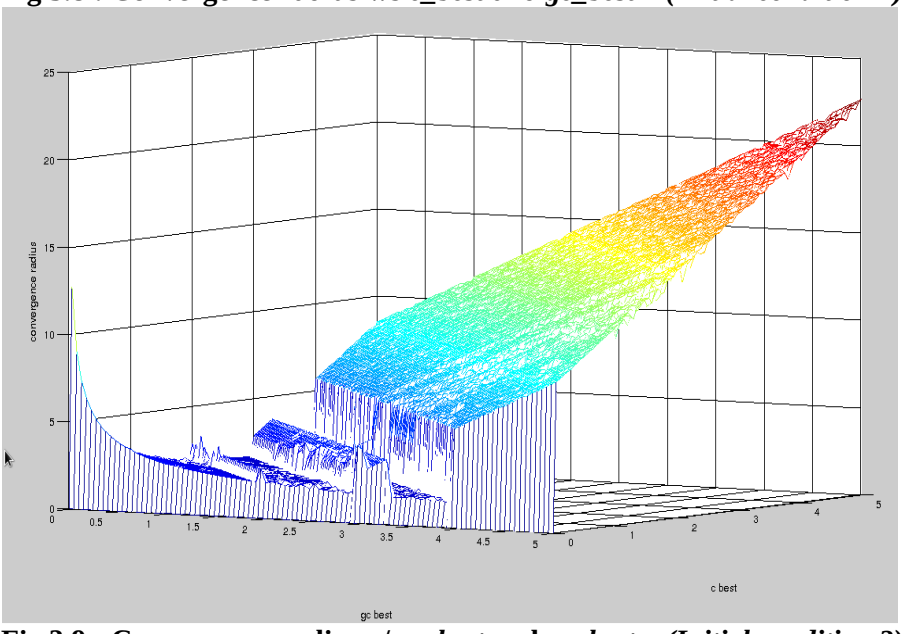


Fig 3.9 : Convergence radius v/s *c_best* and *gc_best* *(Initial condition 3)*

6 boids **(Initial condition 3)**

Initial boid positions :

(500, 500), (500, -500), (-500, 500), (-500, -500) , (400, 0), (0, 400)

Food = **(0, 0)**

Formulation :

It can be observed from the above plots that, the convergence radius appears to be a function of the sum of *c_best* and *gc_best.* The plot is symmetric about y = x plane.

It can be seen that, the plot has an abrupt variation at the line x + y = 4. The convergence radius, changes its behaviour beyond *c_best* + *gc_best* = 4. Up till *c_best* + *gc_best* = 4, the convergence radius decreases with (*c_best* + *gc_best)* with offsets for *c_best* + *gc_best* = 2 and between 3 to 3.3. Excluding these offsets, the convergence radius appears to relate inversely to some power of (*c_best* + *gc_best)*. Beyond *c_best* + *gc_best* = 4, the convergence radius appears to be linearly related to *c_best* + *gc_best.*

Curve fitting was used and the function for Initial condition 1 was observed to be :

$$z = 2.8 \times (x + y)^{-0.5} + 2.5 \times 10^{-6}$$

(for x + y < 4 – [2, (3 : 3.3)])

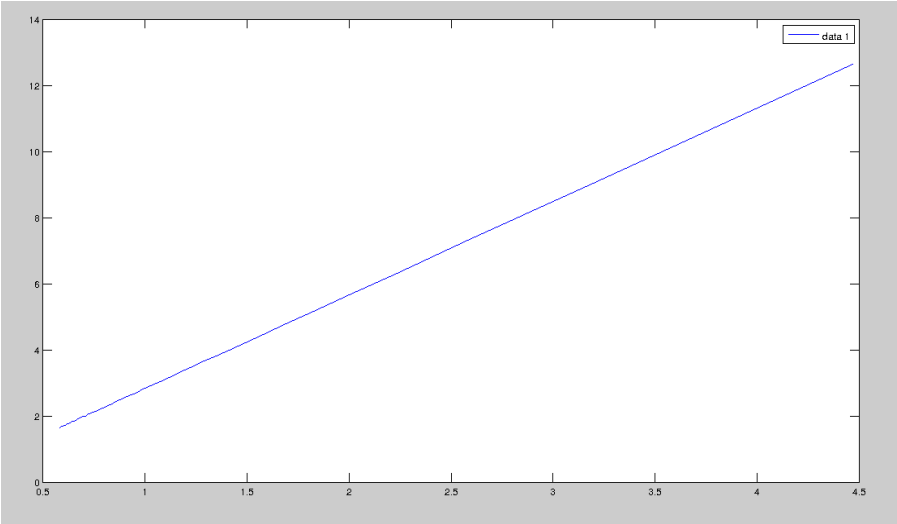


Fig 3.10 : Convergence radius v/s (*c_best* + *gc_best)*$^{-0.5}$ *(x+y < 4)* [IC 1]

$$z = 2.8 \times (x + y) - 5.6$$

(for x + y > 4)

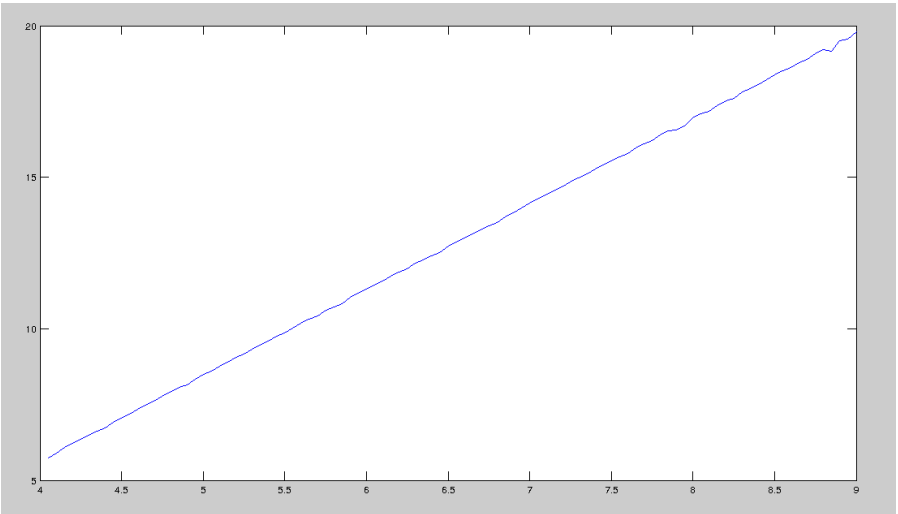


Fig 3.11 : Convergence radius v/s (*c_best* + *gc_best)* *(x+y > 4)* [IC 1]

Curve fitting was also used for the other two initial conditions. Following were the functions obtained :

Initial condition 2 :

$z = 3.1 \times (x+y)^{-0.5} - 0.54$ *(for $x + y < 4 – [2, (3 : 3.3)]$)*

$z = 2.7 \times (x+y) - 5.4$ *(for $x + y > 4$)*

Initial condition 3 :

$z = 2.9 \times (x+y)^{-0.5} - 0.09$ *(for $x + y < 4 – [2, (3 : 3.3)]$)*

$z = 2.8 \times (x+y) - 5.7$ *(for $x + y > 4$)*

We can observe that, for $x + y < 4 – [2, (3 : 3.3)]$, z is proportional to $(x+y)^{-0.5}$, with an approximate factor of 3. While for $x + y > 4$, z is linearly proportional to $x + y$, with an approximate factor of 2.8. There is an offset from this behaviour for $x + y$ in $[2, (3 : 3.3)]$. The value of z *appears to be linearly increasing with* $x + y$ between 3 to 3.3.

Since the z intercept for $x + y < 4$, appears to be very near to zero, we will approximate it to be zero. Taking the proportionality factor to be 2.8, the formulation of convergence radius (*r_conv*) will be :

$r\_conv = 2.8 \times (c\_best + gc\_best)^{-0.5}$ *(for $c\_best + gc\_best < 4 – [2, (3 : 3.3)]$)*

The above formulation appears to satisfy the *r_conv* observed for :
$(c\_best + gc\_best) > 0.0001$
As *c_best + gc_best* approaches 0, *r_conv* will decrease and will equal the initial distance of the farthest boid at *c_best + gc_best = 0.*

The z intercept for $x + y > 4$, appears to be approximately around -5.6 while the proportionality factor approximates 2.8. The formulation of convergence radius (*r_conv*) will be :

$r\_conv = 2.8 \times (c\_best + gc\_best) – 5.6$ *(for $c\_best + gc\_best > 4$)*

4.2. Speed of convergence

With the convergence radius formulated, we now move on to our second criteria for evaluating the efficiency, namely the speed of convergence. We will measure the speed of convergence of the PSO, based on the number of iterations required by the PSO to converge. Now with the convergence radius data in hands, we may utilize this to detect the convergence. We will say that, the boids have converged to the solution when all have entered the convergence radius. But for *c_best* and *gc_best* configurations giving a very high *r_conv*, this definition of convergence will be meaningless. The boids need to be in some minimum range from the food before we can say that they have converged.

We define this minimum to be 5 units in this case. Thus we will define convergence as :

min(r_conv, 5)

With this definition, the simulation was run to detect the boid's entry into this newly define convergence region, for varying values of *c_best* and *gc_best*. Just as above, both ***c_best*** and ***gc_best*** were varied from **0 to 5 in steps of 0.05.** Simulation was run for IC 1, and following was the convergence response obtained.

The number of iterations for convergence have been plotted against *c_best* and *gc_best*.

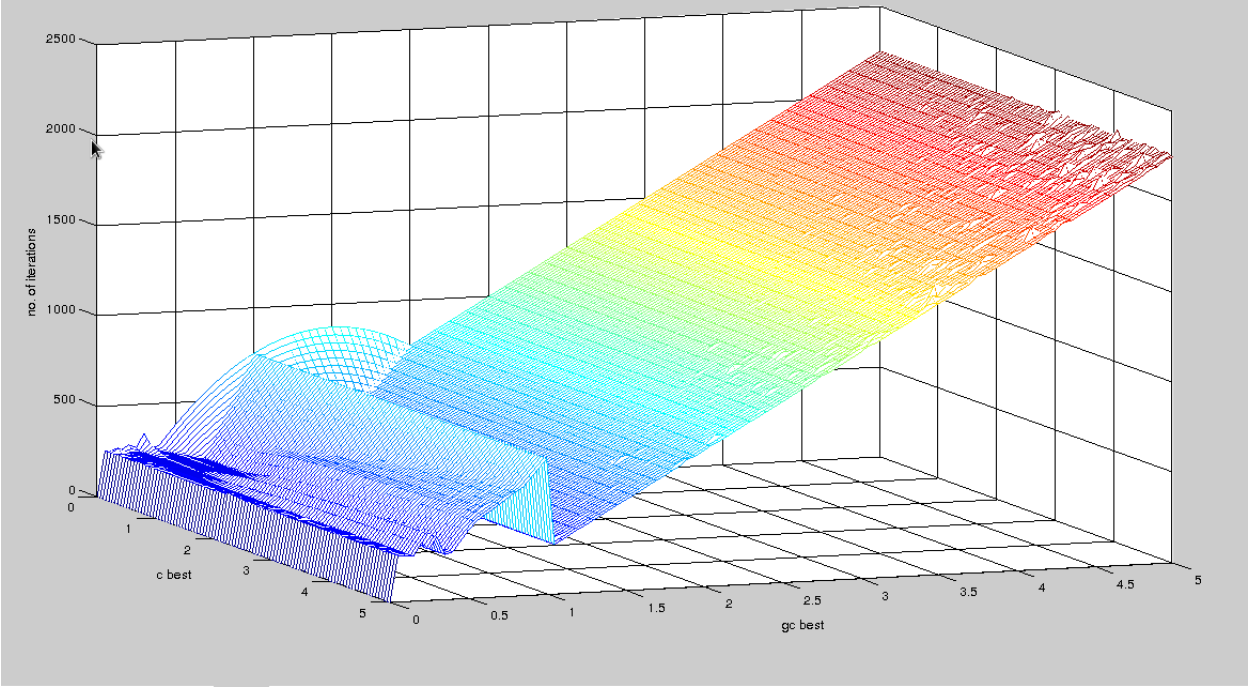


Fig 3.12 : Number of iterations v/s *c_best* and *gc_best* *(Initial condition 1)*

The no. of iterations for convergence, for *gc_best* = 0 are 5000, because the PSO essentially does not converge for *gc_best* = 0 as explained before. Those have been set to zero for better viewing the graph.

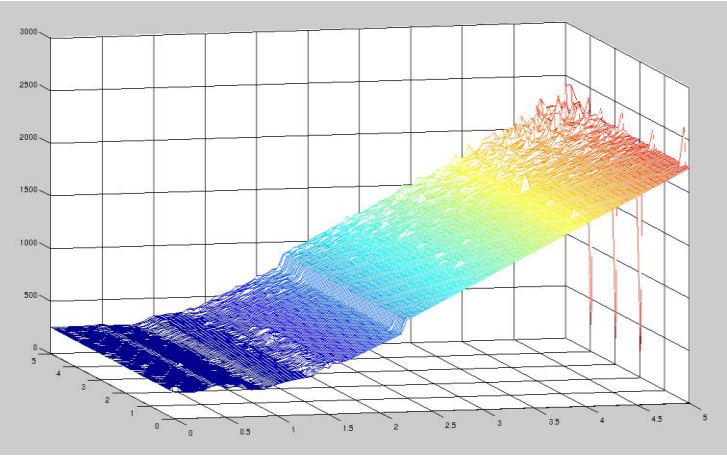


Fig 3.13 : Number of iterations v/s *c_best* and *gc_best* *(Initial condition 2)*

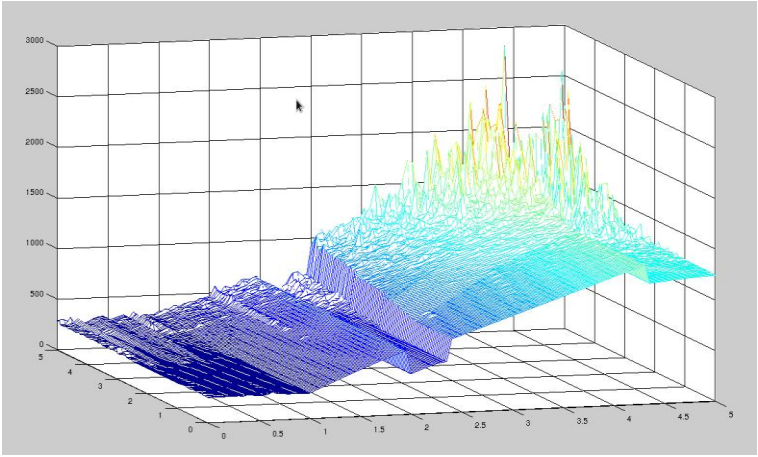


Fig 3.14 : Number of iterations v/s *c_best* and *gc_best* *(Initial condition 3)*

Formulation :

The speed of convergence of the PSO appears to be a function of *gc_best* only. The number of iterations needed for convergence can be broadly said to be increasing with the value of *gc_best*.

An explicit expression for the number of iterations required for convergence as a function of *gc_best* could not be produced. As, the behaviour of the function is also governed by the initial conditions. The range of values of the function depends on the initial distance of the farthest boids from the food.

A general trend of constant number of iterations needed for convergence, was observed for values of *gc_best* below 1.0. Beyond this the function was broadly observed to increase linearly, with intermediate offsets. The slope of the increasing portion was observed to be around 500.

The function can thus be approximated to :

$$\textbf{no. of iterations} \approx c_1$$ *[for gc_best < 1.0 and c_1 is a constant dependent on the initial distance of the farthest boid]*

$$\textbf{no. of iterations} \approx 500 \times \textbf{gc\_best} + c_2$$ *[for gc_best > 1.0 and c_2 is a constant which varies for different ranges of gc_best and initial boid positions]*

4.3. Multi target analysis :

In the presence of multiple targets, implementing the local-best PSO with global-best contribution, only led the simulation to stagnate amidst the process leading to non-convergence of some or all the boids. local-best and global-best are mutually opposing criteria creating tendencies in the boids, to approach to different points on the map simultaneously. Thus the simulation exhibited a non-convergent behaviour.

Implementing the only local-best PSO, displayed similar behaviour to the global-best PSO. The relation between *c_best* and *gc_best* in case of global-best PSO, replicated itself to the relation between *c_best* and *lc_best* in case of the only local-best PSO.

Mathematically, the 'only local-best' PSO is same as the global-best PSO and all the formulae for the convergence radius and number of iterations *w.r.t. c_best* and *gc_best* may be applied to the only local-best PSO, replacing every *gc_best* by *lc_best* in the formulation.

5. Wisdom

As aforementioned, neither of the PSO implementations individually perform well in the multi-target scenario. Tweaking the algorithm to blend it with multi-target optimization is necessary. This chapter will firstly attempt to logically reason out the results of the analysis in the above chapter. Secondly, it will exhibit the key features of the PSO formulation and use the results and observations to propose a strategy to bypass the problem we are currently facing with, multi-target optimization.

5.1. Global best PSO

Some intriguing facts about the workings of PSO were explored in the analysis of its convergence with respect to its acceleration coefficients. Here we will try to deduce the reasons behind these observations.

Quality of convergence :

The quality of convergence (convergence radius) of the boids was observed to be a function of (*c_best* + *gc_best*). To understand this, we must note that convergence radius is measured when the boids have attained equilibrium around the food location. After this, the boids will keep oscillating in this region. This region essentially has to be small compared to the sample space and hence the position tracked by an individual boid and the flock as a whole. i.e. the x_best_i and *gx_best* will not be spatially far apart. These positions will approximately lie on the food itself with a slight offset.

This causes an additive effect on the two terms -
$c\_best \times (x\_best_j - x_j(i))$ and $gc\_best \times (gx\_best - x_j(i))$
as, $(x\_best_j - x_j(i))$ and $(gx\_best - x_j(i))$ are almost the same and the above expression may very well be replaced by : $(c\_best + gc\_best) \times (gx\_best - x_j(i))$.

As to the relation between *(c_best + gc_best)* and *r_conv*, two behaviours were observed. The inverse relation for $(c\_best + gc\_best) < 4$ can be attributed to the inertia present in the formulation. With lower values of *c_best* and *gc_best*, the incremental change $(c\_best + gc\_best) \times (gx\_best - x_j(i))$ is insufficient to counter the inertia already present in the velocity and exhibit instantaneous changes in position. Thus the change in velocity in the desired direction is gradual and while the complete effect is realized, the boid passes over the best position and the same cycle repeats.

As *(c_best + gc_best)* increases above 4, the change in velocity $(c\_best + gc\_best) \times (gx\_best - x_j(i))$ exhibits opposite behaviour as the boid now covers excess distance and passes over the best position it was tracking, within a single iteration. The boid is now on the opposite side of its last position with respect to the best position. The boid thus follows this jittery oscillatory trajectory whose amplitude increases as *(c_best + gc_best)* increases.

Speed of convergence :

The speed of convergence was observed to depend broadly only on *gc_best*. It appears to remain constant and then decrease as *gc_best* increases. The number of iterations needed for convergence thus increase linearly as *gc_best* increases.

In PSO, the conservative behaviour is introduced by *c_best*. This increases the tendency of the boid to deviate from the flocking behaviour introduced by *gc_best*. Hence, *c_best* apparently does not contribute significantly to speed of convergence, while *gc_best* is the driver of collaborative tendency of the boids attracting them to a point on the map.

An explicit relation between speed of convergence and *gc_best* cannot however be established. This is because, the number of iterations needed for convergence also depend on the initial distance of the farthest boid on the map. It can intuitively be discerned that, farther the boids from food on the map, higher will be the number of iterations they require to reach the food.

Although, we may argue that the increment in velocity $gc\_best \times (gx\_best - x_j(i))$ is proportional to the distance between *gx_best* and $x_j(i)$ which will be higher for larger sample spaces, in which case the boid could travel faster initially. But we need to critically note that, there is also a clamping on the maximum velocity a boid can attain (set to 2 units in the implementation). Hence initially, if the v_x_j comes out to be higher than 2 due to larger distance, this will be set to 2 and our conclusion stands.

5.2. Local-best and Global-best PSO

A local-best PSO employs local search to scan the sample space. Chunks of the flock of boids explore their locality to find optima (food locations). The global-best PSO works on a wider scope taking the entire flock as a chunk to track the optimum. The global-best PSO as mentioned before is a special case of local-best PSO with the locality defined to include the entire set of boids in the simulation.

Number of boids :

The same formulation implemented for global-best PSO was employed for the local-best PSO with *lc_best* replacing *gc_best*. We can evidently see that, global-best PSO is a special case of local-best PSO. But to use the formulation of global-best PSO for local-best implementation, needs justification.

The justification lies in the impact of the total number of boids, on the behaviour of individual boids. The formulation for velocity of a boid includes sum of three terms :

$w \times v\_x_j(i)$, $c\_best \times (x\_best_j - x_j(i))$ and $gc\_best \times (gx\_best - x_j(i))$

We can observe that, the first term is independent of the number of boids and only depends on the velocity of the boid in the previous iteration. Also, the x_best_j and *gx_best* are single terms which track the individual best and the global best and are independent of the number of boids considered. Although the performance of the flock as a whole varies and actually improves with the number of boids present, it doesn't affect the behaviour of individual boids. They only may track better points with increase in the number of boids. Hence, considering a chunk or the entire flock to locate the best position doesn't carry a significant difference.

Shortcomings in multi-target optimization

As exhibited in the implementation, both local-best and global-best PSO encounter shortcomings in their exploration and optimization.

In global-best PSO, the entire flock works in collaboration to locate the optimum. In multi-target case, gobal-best PSO fails to track the global optimum by either converging too early or not converging completely. Since all boids track the same global best position, their explorative tendencies are reduced and might miss out on some possibly better optima present in the unexplored region.

Local-best PSO has an advantage over global-best of the ability to track multiple optima by partitioning itself into smaller chunks. Thus the boids are better able to explore the solution space and converge to an optimum although locally. Now the absence of global-best tendency kills the ability of the swarm to distinguish between two optima to find the better of the two. A local-best PSO is only capable of locating the local optima. IT stagnates thereafter.

But, implementing the local-best and global-best criteria simultaneously leads to conflicts between the two tendencies as exhibited in the implementation. Hence a better strategy needs to be devised to suit PSO for multi-target optimization.

5.3. New strategy for multi-target PSO

Many modifications and strategies have been developed over the years to suit various conditions. We observed that, the basic PSOs fail to work on multi-target optimization case. Also, the local-best and global-best PSOs have their individual advantages. To make proper use of these advantages we implement them mutually exclusively.

The local-best PSO is best at better exploring the sample space by tracking multiple local optima. We utilize this ability by initializing the PSO with zero contribution from global-best criteria. i.e. setting *gc_best* to zero and setting a small locality size for the local search.

Next, we can now utilize the convergence radius information we have to detect convergence as we have defined it. Once all boids have converged locally, we need to compare these local optima. This can be done using two strategies.

1) Kill the local-best tendency completely by setting *lc_best* to zero and raising *gc_best* to a non-zero value. Now the boids at the global optimum will remain unmoved while all other boids at the local optima will track this global optimum and converge.
2) Instead of replacing local-best behaviour completely by global-best, we slowly transform the local-best tendency to global-best by increasing the locality size. Once the convergence for a set value of locality size has been detected, increase the locality size by 1, continuing till the locality size becomes equal to the total number of bois in the simulation. Although slower, this strategy will give higher chance to the boids to explore more of the sample space with every increment of locality size.

With this new strategy, the problem of multi-target optimization using PSO has been resolved. The capacities of various criteria have been efficiently used to achieve the desired end.

6. Conclusion

The PSO algorithm was implemented and tested for single and multiple target scenarios. The working of PSO based on its tuning parameters was critically studied. Attempt was made to formulate the behaviour of the PSO with respect to the acceleration coefficients. The traditional values of acceleration coefficients (*c_best = 2* and *gc_best = 2*), generally set in PSO implementations showed sufficient compliance with the formulation. But it also exhibited that better performance may be achieved for other values as well. The analysis of the simulations and formulation was logically reasoned out. This exposed key features of the PSO which make it a brilliant optimizer. Finally a strategy to minimally tweak the conventional PSO to suit multi-target optimization was proposed and tested to success.

With finer quantitative knowledge of the working of PSO, better strategies can be designed to cater to specific requirements. Many advancements in the algorithm are already being made and various versions of PSO to improve the performance in diverse environments exist.

References

[1] Prateek P. Deo; PSO "Algorithm for trajectory planning in Re-usable launch vehicles". Dual Degree Project report, 2011.

[2] Kennedy J, Eberhart R. ; "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks, 1995.

[3] Jian Zhang; "Experimental Parameter Investigations on Particle Swarm Optimization Acceleration Coefficients". IJACT, Vol. 4, No. 5, 2012.

[4] Pu Sun, Hua Sun, et al.; "A Study of Acceleration Coefficients in Particle Swarm Optimization Algorithm Based on CPSO". IEEE, 2010.

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

[6] http://www.particleswarm.info