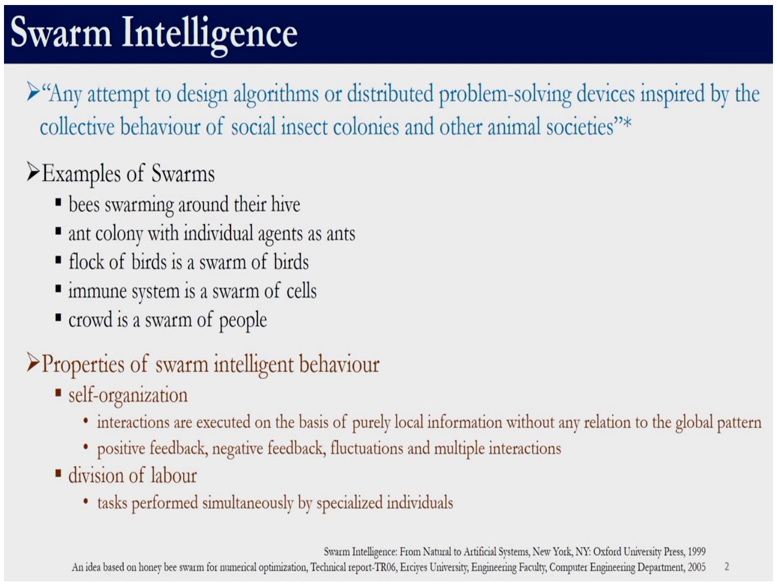
Computer Aided Applied Single Objective Optimization

Dr. Prakash Kotecha
Department of Chemical Engineering
Indian Institute of Technology, Guwahati

Lecture – 10
Particle Swarm Optimization

Welcome. In this session, we will be looking at Particle Swarm Optimization. This will be our second metaheuristic that we will be studying, the previous technique that we had studied was teaching learning based optimization. Particle swarm optimization is a swarm intelligence technique.

(Refer Slide Time: 00:48)



Swarm Intelligence

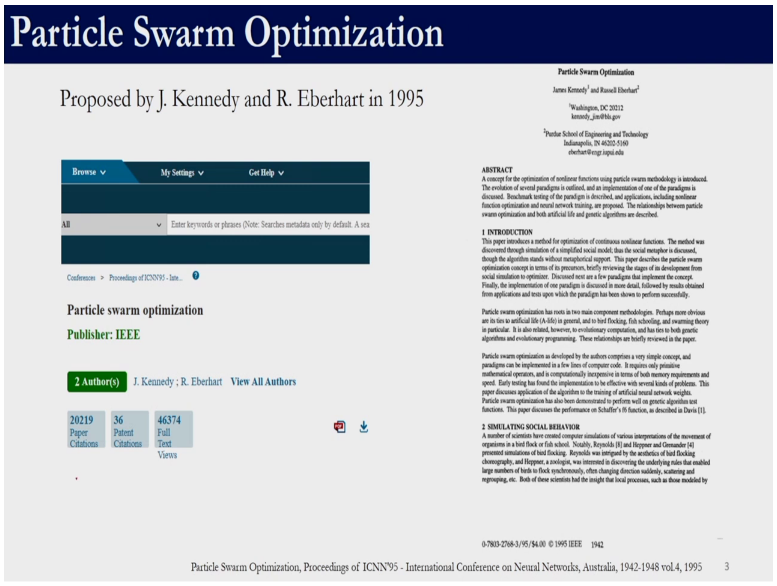
- "Any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies"*
- Examples of Swarms
  - bees swarming around their hive
  - ant colony with individual agents as ants
  - flock of birds is a swarm of birds
  - immune system is a swarm of cells
  - crowd is a swarm of people
- Properties of swarm intelligent behaviour
  - self-organization
    - interactions are executed on the basis of purely local information without any relation to the global pattern
    - positive feedback, negative feedback, fluctuations and multiple interactions
  - division of labour
    - tasks performed simultaneously by specialized individuals

Swarm Intelligence: From Natural to Artificial Systems, New York, NY: Oxford University Press, 1999
An idea based on honey bee swarm for numerical optimization, Technical report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005 2

Swarm intelligence is defined as any attempt to design algorithms or distributed problem solving devices that are inspired by the collective behavior of social insect colonies and other animal societies. So, particle swarm optimization gets its inspiration from the flocking of birds or what is called as fish schooling. Some of the examples of swarms are bees swarming around their hive, ant colony with individual agents as ants, flock of birds is a swarm of birds, immune system is a swarm of cells and a crowd can be considered as a swarm of people.

The two important properties of swarm intelligent behavior are self organization and division of labor. By self organization we mean that the interactions are executed on the basis of purely local information without any relation to the global pattern. Self organization consists of positive feedback, negative feedback, fluctuations and multiple interactions. By division of labor we mean that the tasks that are to be performed are performed simultaneously by specialized individuals. This is what constitutes as swarm intelligence.

(Refer Slide Time: 02:01)



Particle Swarm Optimization

Proposed by J. Kennedy and R. Eberhart in 1995

Browse
My Settings
Get Help
All
Enter keywords or phrases (Note: Searches metadata only by default. A sea
Conferences > Proceedings of ICNN'95 - Inte...

Particle swarm optimization

Publisher: IEEE

2 Author(s) J. Kennedy ; R. Eberhart View All Authors

20219 Paper Citations | 36 Patent Citations | 46374 Full Text Views

Particle Swarm Optimization

James Kennedy[1] and Russell Eberhart[2]

[1]Washington, DC 20212
kennedy_jim@bls.gov

[2]Purdue School of Engineering and Technology
Indianapolis, IN 46202-5160
eberhart@engr.iupui.edu

ABSTRACT
A concept for the optimization of nonlinear functions using particle swarm methodology is introduced. The evolution of several paradigms is outlined, and an implementation of one of the paradigms is discussed. Benchmark testing of the paradigm is described, and applications, including nonlinear function optimization and neural network training, are proposed. The relationships between particle swarm optimization and both artificial life and genetic algorithms are described.

1 INTRODUCTION
This paper introduces a method for optimization of continuous nonlinear functions. The method was discovered through simulation of a simplified social model; thus the social metaphor is discussed, though the algorithm stands without metaphorical support. This paper describes the particle swarm optimization concept in terms of its precursors, briefly reviewing the stages of its development from social simulation to optimizer. Discussed next are a few paradigms that implement the concept. Finally, the implementation of one paradigm is discussed in more detail, followed by results obtained from applications and tests upon which the paradigm has been shown to perform successfully.

Particle swarm optimization has roots in two main component methodologies. Perhaps more obvious are its ties to artificial life (A-life) in general, and to bird flocking, fish schooling, and swarming theory in particular. It is also related, however, to evolutionary computation, and has ties to both genetic algorithms and evolutionary programming. These relationships are briefly reviewed in the paper.

Particle swarm optimization as developed by the authors comprises a very simple concept, and paradigms can be implemented in a few lines of computer code. It requires only primitive mathematical operators, and is computationally inexpensive in terms of both memory requirements and speed. Early testing has found the implementation to be effective with several kinds of problems. This paper discusses application of the algorithm to the training of artificial neural network weights. Particle swarm optimization has also been demonstrated to perform well on genetic algorithm test functions. This paper discusses the performance on Schaffer's f6 function, as described in Davis [1].

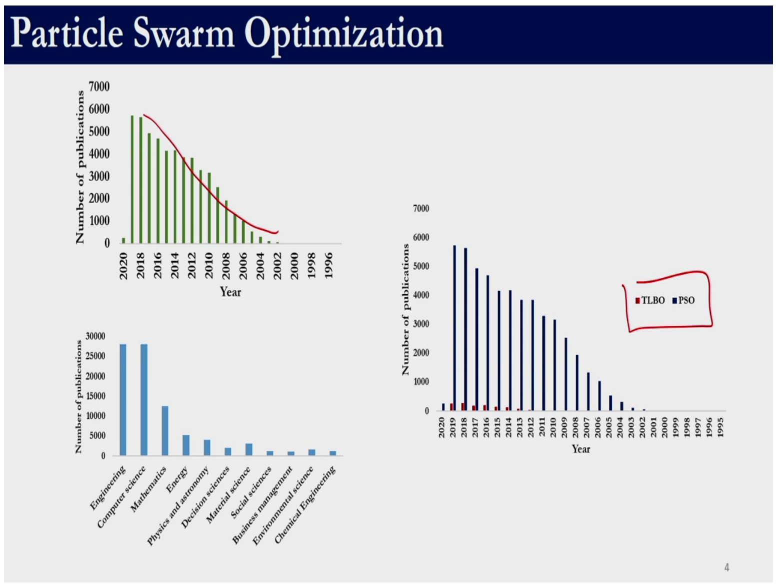
2 SIMULATING SOCIAL BEHAVIOR
A number of scientists have created computer simulations of various interpretations of the movement of organisms in a bird flock or fish school. Notably, Reynolds [8] and Heppner and Grenander [4] presented simulations of bird flocking. Reynolds was intrigued by the aesthetics of bird flocking choreography, and Heppner, a zoologist, was interested in discovering the underlying rules that enabled large numbers of birds to flock synchronously, often changing direction suddenly, scattering and regrouping, etc. Both of these scientists had the insight that local processes, such as those modeled by

0-7803-2768-3/95/$4.00 © 1995 IEEE 1942

Particle Swarm Optimization, Proceedings of ICNN'95 - International Conference on Neural Networks, Australia, 1942-1948 vol.4, 1995 3

Particle swarm optimization was proposed by Kennedy and Eberhart in 1995. It was presented in the International Conference of Neural Networks in Australia. Ever since it has gained a lot of attention so, right now if we see it has more than 20000 papers citations right.

(Refer Slide Time: 02:13)



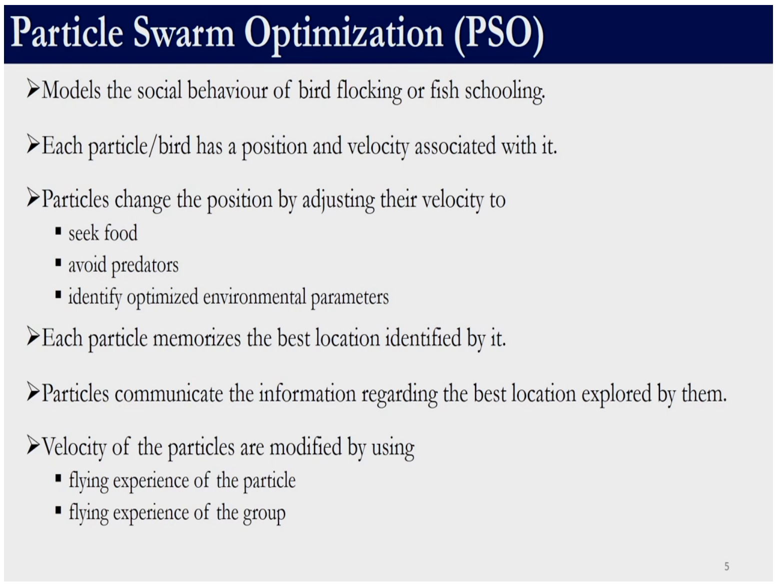
Particle Swarm Optimization

So, this shows over the years how people have taken to particle swarm optimization. So, as we can see it is exponential increase over here in the number of publications that have cited particle swarm optimization.

So, again particle swarm optimization as it is a meta heuristic technique, we have been telling it multiple times that it does not matter which domain we are working in. Meta heuristic techniques can always be used as and when we have an optimization problem, especially when the problems are non-linear or mixed integer. non-linear programming or even when our problem is a black box optimization problem. So, as we can see particle swarm optimization has been used in engineering, computer science, decision sciences and also in social sciences.

This plot shows the comparison between a TLBO and PSO. So,; obviously, it is unfair to compare given that TLBO has been recently proposed whereas, particle swarm optimization was proposed wayback in 95, but still what we can see is that people have started to use TLBO as can be seen over here.

(Refer Slide Time: 03:20)



Particle Swarm Optimization (PSO)

- Models the social behaviour of bird flocking or fish schooling.
- Each particle/bird has a position and velocity associated with it.
- Particles change the position by adjusting their velocity to
  - seek food
  - avoid predators
  - identify optimized environmental parameters
- Each particle memorizes the best location identified by it.
- Particles communicate the information regarding the best location explored by them.
- Velocity of the particles are modified by using
  - flying experience of the particle
  - flying experience of the group

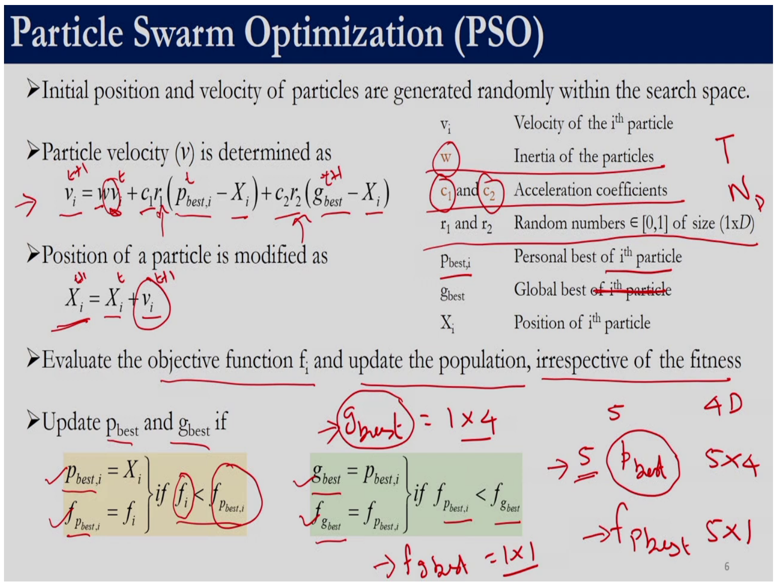
5

So, particle swarm optimization models the social behavior of bird flocking or fish schooling. So, the inspiration is bird flocking and fish schooling. So, the solution what we call it as a solution in optimization is known as particle or bird in particle swarm optimization. So, each particle or bird has a position and velocity associated with it; in real life this particles keep changing their position by adjusting their velocity. So, they primarily do this either to seek food or to avoid predators or to identify optimize environmental parameters. There can be more than one reason for which the particles may be changing their positions.

A significant difference between TLBO and PSO is that each particle memorize the best location identified by it. So, it is more like a we keep track of our own best. So, each particular keeps track of its best location identified by it. The particle will have that position in it is memory and will still keep exploring the search place, but it nevertheless memorizes its best location. So, particles communicate the information regarding the best location explored by them. So, all the particles communicate their own best location and from this the best location the best of the individual particles can be located that would be the global best.

So, velocity of that particles are modified by using flying experience of the particular particle. So, every particle has a velocity associated with it. So, how do we modify this velocity? By using the flying experience of that particular particle as well as the entire group.

(Refer Slide Time: 04:52)



Particle Swarm Optimization (PSO)

- Initial position and velocity of particles are generated randomly within the search space.
- Particle velocity (v) is determined as

$$\rightarrow \overset{t+1}{v_i} = \overset{t}{w v_i} + c_1 r_1 (\overset{t}{p_{best,i}} - X_i) + c_2 r_2 (\overset{t+1}{g_{best}} - X_i)$$

- Position of a particle is modified as

$$\overset{t+1}{X_i} = \overset{t}{X_i} + \overset{t+1}{v_i}$$

v_i	Velocity of the ith particle	
w	Inertia of the particles	T
c_1 and c_2	Acceleration coefficients	N_p
r_1 and r_2	Random numbers ∈ [0,1] of size (1xD)	
$p_{best,i}$	Personal best of ith particle	
g_{best}	Global best ~~of ith particle~~	
X_i	Position of ith particle	

- Evaluate the objective function f_i and update the population, irrespective of the fitness
- Update p_{best} and g_{best} if

$$\left.\begin{array}{l} p_{best,i} = X_i \\ f_{p_{best,i}} = f_i \end{array}\right\} if\ f_i < f_{p_{best,i}} \qquad \left.\begin{array}{l} g_{best} = p_{best,i} \\ f_{g_{best}} = f_{p_{best,i}} \end{array}\right\} if\ f_{p_{best,i}} < f_{g_{best}}$$

$\rightarrow g_{best} = 1 \times 4$

5 4D

$\rightarrow 5\ p_{best}\ 5 \times 4$

$\rightarrow f_{gbest} = 1 \times 1$

$\rightarrow f_{pbest}\ 5 \times 1$

6

So, in particle swarm optimization the first step is to initialize the position and velocity of the particles. So, these are generated randomly within the search space similar to what we did in teaching learning based optimization. So, this will be our particles collection of our particles or more familiarly what we will be calling as population in metaheuristic techniques. So, each particle will be associated with it is velocity. So, these are the two equations which will govern the generation of new solution right.

So, here w is known as the inertia of the particles; c 1 and c 2 are acceleration coefficients right. So, these three parameters have to be provided by the user in addition to the termination criteria or the number of iterations or generations or cycles whatever we choose to call it and the population size N P. So, in addition to the these two parameters the user is also supposed to give the inertia and the acceleration coefficient there are two acceleration coefficients right.

So, here v i is the velocity of the particular particle right. So, this equation gives us a procedure to update the velocity right. So, the current velocity it will be a function of the previous velocity. So, that is why this term velocity appears over here right. So, this r 1 and r 2 are random numbers right random numbers that belong in the space 0 to 1 right they will be of 1 cross D. So, it will have one row and d columns where D is the number of decision variables. So, for each decision variable we will require one random number for this r 1 and one random number for this r 2 right. So, basically for every decision variable we will require two random numbers one over here, one over here right.

P best of i corresponds to the best location identified by the particle so far; not its current position, but the best position that it has identified till any given point of time. X i is its current position; g best is the global best right it is the best among all. So, there is no g best for every individual particle right. g best is for the entire population whereas, p best is for every individual particle member and X i is again the position of the particle in the previous iteration right. So, how do we update?

So, we determined this velocity and use this velocity and the current position right, this newly determined velocity from this equation and the current position to find out the new position. So, here i does not indicate iteration, i indicates the i-th particular right. So, if you want to include iteration over here. So, in the t plus 1th iteration; this is what is in the t-th iteration and this is what was determined in the t plus 1th iteration.

So, this will be t plus 1 and this will be t this will be of t t-th iteration, this will be also what is found in t plus 1th iteration right; t plus 1th iteration because it may happen that the fourth solution might have updated the g best. So, whatever is the updated g best that is what we will be using right.

So, p best obviously, will come from the previous iteration, that is how we can include the iteration in these equations right. To keep it simple we did not include the iteration over here. So, as we will do one example you will be able to better understand that. Once the position of a new particle has been identified we need to evaluate its objective function or the fitness function and update the population.

Remember, we need to update the population irrespective of the fitness there is no greedy search involved over here. The position is definitely going to be updated right whether the solution is good or bad does not matter. It will be included in the population right. This is unlike TLBO wherein we employed a greedy selection mechanism if and only if the new solution was good, it was taken into the population. In PSO it is not like that, new the new position is always going to be taken into the population right. So, that is one major difference between TLBO and particle swarm optimization.

The next step is to update the p best and g best. Given that we have generated a new solution over here right this solution may be better than its own best or it may even be better than the global best. So, we check this condition that the new solution which we have generated if it is fitness function is denoted by f of i and if it is less than the fitness function of the best position of the i-th particle right if this condition is satisfied then we update the p best of the i-th solution and f p best of the i-th solution. So, b p best stores the values of the decision variable and f f of p best stores the value of the fitness function right.

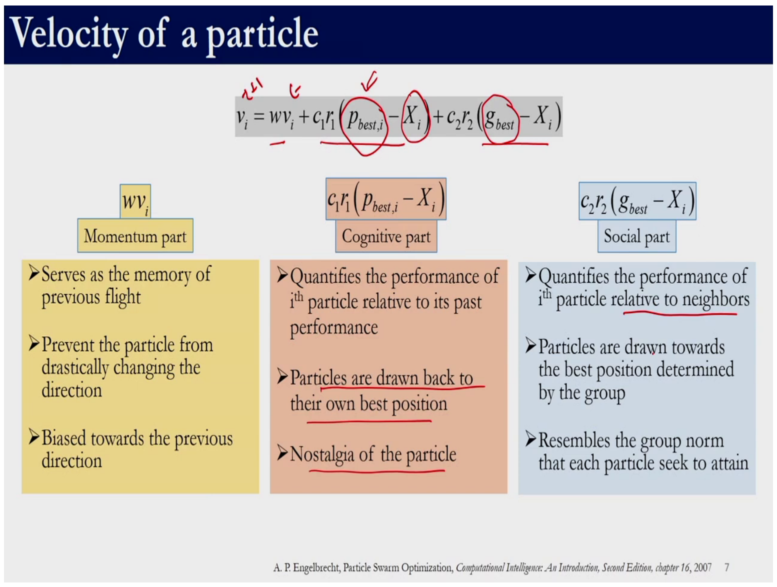
Here it is more like a greedy selection strategy that once we generate a new solution if the solution is good or bad it is definitely going to be included in the population, but the p best and the g best may or may not be updated. Over here we say that if the p best of the i-th particle right so, first we update the p best and if we check it with the g best right. So, if the p best has been updated there may be a possibility that the g best also can be improved right. So, in this case if this satisfies that the p best of the i-th particle if it is better right, we will update the g best and f f of g best. Again, g best corresponds to the decision variable and f of g best corresponds to our fitness function value.

So, let us say if I have five population members right so, there will be 5 p best and if it is a 4-dimensional problem it will be a 5 cross 4 matrix, we can keep it as a 5 cross 4 matrix f of p best will be 5 cross 1 because each solution will have one fitness function value. So, this is what, but g best at any given point of time g best will be g best indicates the decision variables right. So, it will be 1 row and 4 columns corresponding to the four decision variables whereas, f of g best will be a one 1 cross 1 scalar right.

So, you need to remember that g best and p best indicate the solutions whereas, f of g best and f of p best indicates the fitness function value. There is one only one global best the very names global suggests that that there is only one global best if there are five particles there will be 5 p best corresponding to each particle right. So, that is how we need to understand p best and g best.

So, there is a small mistake in this slide right. So, as we have discussed global best would be only one right for the entire population there is only one global best. So, this is not of i-th particle right personal best is of for every particle we have a personal best, but for the global best we have only one global best for the entire population right it is not global base of the i-th particle. So, this is actually a typo over here.

(Refer Slide Time: 11:58)


Velocity of a particle

$$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$$

wv_i Momentum part	$c_1 r_1 (p_{best,i} - X_i)$ Cognitive part	$c_2 r_2 (g_{best} - X_i)$ Social part
➢Serves as the memory of previous flight	➢Quantifies the performance of i^{th} particle relative to its past performance	➢Quantifies the performance of i^{th} particle relative to neighbors
➢Prevent the particle from drastically changing the direction	➢Particles are drawn back to their own best position	➢Particles are drawn towards the best position determined by the group
➢Biased towards the previous direction	➢Nostalgia of the particle	➢Resembles the group norm that each particle seek to attain

A. P. Engelbrecht, Particle Swarm Optimization, *Computational Intelligence: An Introduction, Second Edition, chapter 16*, 2007 7

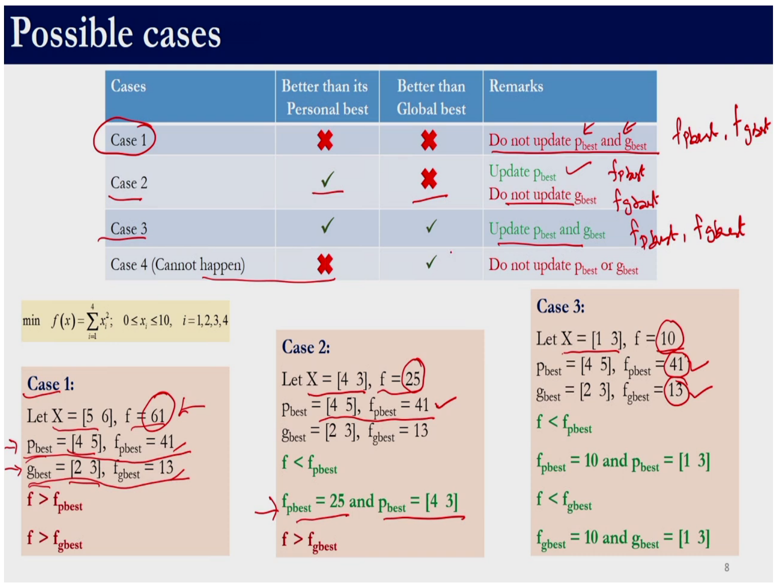
So, this equation which we used to update the velocity right if you see it consists of three parts – part 1, part 2 and part 3. So, the first part is known as the momentum part right it serves as the memory of the previous flight right. So, if I am going to find out v of t plus 1 right this is going to be of t-th iteration the previous iteration. So, that way it serves as the memory of the previous flight right. This component prevents the particle from drastically changing their direction right and it is biased towards the previous direction because since it is dependent on v v i of t it is biased towards the previous direction right.

The second part is known as the cognitive part right it quantifies the performance of the i-th particle relative to its own past performance right. So, for each particle we have a p best right. So, here we are taking the difference between the current position of the particle indicated by x i and its best location found in any generation. So, assume that we are in 25th generation, the first particle might have encountered it is best location in the third generation right. So, this will be the value in the third generation and this will be it is value in the previous generation right.

So, this will be the value whenever it has obtained its best location right whereas, X i indicates its current position by the use of the cognitive part particles are drawn back to their own best position because we are comparing it with its own best right. So, this is more like nostalgia of the particle right they remember their best and they are always comparing with their own best. The third term is known as the social part right; social part because this g best is coming from the global right.

So, it quantifies the performance of the i-th particle relative to its to the neighbors. So, here we are considering the entire population to be neighbors that is why we are updating g best with the help of all the p best. So, using the social part the particles are drawn towards the best position determined by the group and this resembles the group norm that each particle seeks to attain right.

(Refer Slide Time: 14:00)



Possible cases

Cases	Better than its Personal best	Better than Global best	Remarks
Case 1	✗	✗	Do not update p_{best} and g_{best} f_{pbest}, f_{gbest}
Case 2	✓	✗	Update p_{best} f_{pbest} Do not update g_{best} f_{gbest}
Case 3	✓	✓	Update p_{best} and g_{best} f_{pbest}, f_{gbest}
Case 4 (Cannot happen)	✗	✓	Do not update p_{best} or g_{best}

$$\min \quad f(x)=\sum_{i=1}^{4} x_i^2; \quad 0 \le x_i \le 10, \quad i=1,2,3,4$$

Case 1:

Let $X = [5 \;\; 6]$, $f = 61$

$p_{best} = [4 \;\; 5]$, $f_{pbest} = 41$

$g_{best} = [2 \;\; 3]$, $f_{gbest} = 13$

$f > f_{pbest}$

$f > f_{gbest}$

Case 2:

Let $X = [4 \;\; 3]$, $f = 25$

$p_{best} = [4 \;\; 5]$, $f_{pbest} = 41$

$g_{best} = [2 \;\; 3]$, $f_{gbest} = 13$

$f < f_{pbest}$

$f_{pbest} = 25$ and $p_{best} = [4 \;\; 3]$

$f > f_{gbest}$

Case 3:

Let $X = [1 \;\; 3]$, $f = 10$

$p_{best} = [4 \;\; 5]$, $f_{pbest} = 41$

$g_{best} = [2 \;\; 3]$, $f_{gbest} = 13$

$f < f_{pbest}$

$f_{pbest} = 10$ and $p_{best} = [1 \;\; 3]$

$f < f_{gbest}$

$f_{gbest} = 10$ and $g_{best} = [1 \;\; 3]$

8

So, when we generate a new solution three cases are possible. So, the first case is that we have generated a new solution which is not better than its personal best right and it is not better than the global best right. So, for example, let us say for case 1 let us say the newly generated solution is 5 and 6 and the objective function is 61 right. Let us also assume that the p best before we got this new solution and its fitness function was 41 and the g best was 2 comma 3, with its f of g best as 13.

So, now if we see this new solution has an objective function of 61 and we are talking about now a minimization problem. So, this 61 is neither better than 41 nor better than 13 right. So, this solution will enter the population right, but we will not update the p best or the g best because this is not the best position obtained by this particle so far because 4 5 was a better position right. So, we will retain this p best and f p best and similarly since it is not better than this g best we will retain the current g best and f best.

So, in this case we are not supposed to update the f best or g best. Similarly, since we are not update updating the solution we are not supposed to update f of p best and f of g best. And, so, this is a one case wherein the solution is neither better than its personal best nor better than its global best.

So, case 2 the solution is better than its own personal, but poorer than its global best. So, in this case we will only update the p best, we will not update the g best right because it is not better than the global it is only better than its best position so far. So, here if we see let us say the new solution which we generated is 4 comma 3 and its fitness function let it be 25. So, and the p best let us assume that the current p best and f of p best is 4 comma 5 and 41 right.

So, now, the solution which we have obtained 25 is actually better than this 41 right. So, we can update f p best and p best. Remember, we need to update the objective function value or the fitness function value as well as the decision variables right whereas, now if we compare this 25 with 13 right, 25 is not better than 13 because we are talking about a minimization problem. So, we do not need to update the g best right. Similarly, what if we are updating p best we are also supposed to update f of p best right since we are not updating g best we do not update f of g best also right. So, this is the second condition.

The third the third case may be that the newly generated solution is actually better than its personal best as well as its better than its global best. So, for example, take this case X is equal to 1 1 comma 3 right. So, it is like we had a solution, for that solution we generated a velocity and then we determined a new position. If the new position is 1 comma 3 and if the fitness function is 10, if we compare p best previously we had 41, now we have 10 of this particular particle right. So, p best remember there are multiple p best. We need to compare with the p best of this first particle. If X is the first particle we need to compare it with the part first p best right.

So, since here if we see the solution 10 which we have obtained is better than the p best as well as than f of g best. So, in this case we will update both of them the positions p best and g best also we will update the respective objective function value f of p best and f of g best right. So, both have to be updated. So, we will never come across this case wherein a solution is not better than its personal best, but it is better than its global best.

So, this condition will not ever happen right because if something has to be better than global best then obviously, it is a bettering either it is the same as its personal best or it is better than the previous personal best. So, this case can never happen, the case 4 can never happen where in the solution is not better than its personal best, but it is better than the global best because global best is the best of the personal best. So, this case can never happen wherein a solution is not improving it is personal best, but it is improving it is global best that that would not happen.

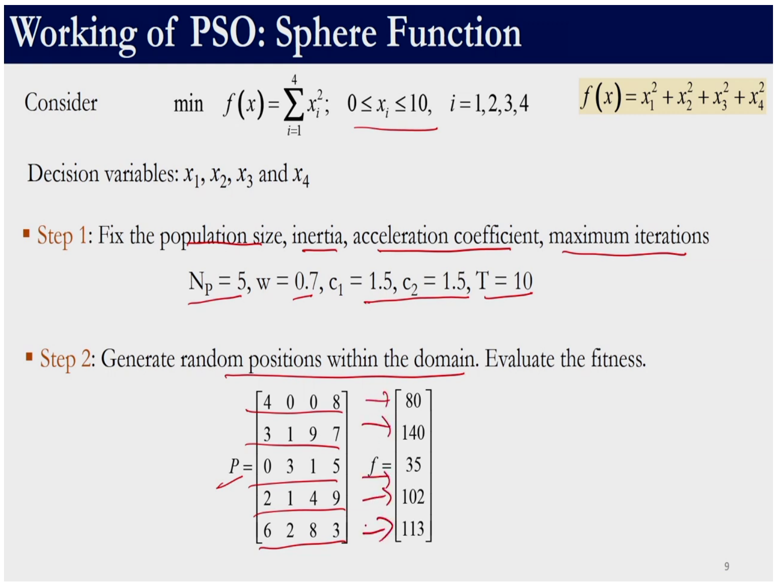
So, as we can see a particle swarm optimization is a fairly simple algorithm, I would say that it is even simpler than teaching learning based optimization right. Despite that fact we have taken particle swarm optimization after teaching learning based optimization is because of the tuning parameters. So, in teaching learning based optimization we required only two user defined parameters; one was the population size and the second was termination criteria.

In PSO we need both of them in addition to that we need the acceleration coefficients which are two in number c 1 and c 2 one associated with the cognitive part and one another one associated with the social part and also the inertia associated it with its velocity right. So, for anyone to execute a particle swarm optimization in addition to the details of the optimization problem they also need to specify five parameters – inertia, two cognitive factors, the population size and the termination criteria right. So, particle swarm optimization requires five parameters from the users.

It is not always easy to specify this parameters for an arbitrary problem though there are some basic guidelines, but it is not guaranteed that they will always work right. So, one will have to solve the same problem with different settings right and as we know for a stochastic algorithm we anyway need to run multiple times right. Now, we not only have to run multiple times for a single setting, but we will also have to execute the problem for multiple settings and for each setting we need to do multiple runs. So, that is a benefit of teaching learning based optimization that we did not require multiple user defined parameters, we only required two, here we require 5 right.

Now, that we have understood particle swarm optimization let us know solve a small problem right so, that will help us to clarify the concepts further right. So, for this we will take the same sphere function with four decision variables. So, the objective function is x 1 square plus x 2 square plus x 3 square plus x for square and the domain let us take the domain also the to be the same what we considered in TLBO between 0 and 10.

(Refer Slide Time: 20:36)



Working of PSO: Sphere Function

Consider min $f(x)=\sum_{i=1}^{4} x_i^2;\quad 0 \le x_i \le 10,\quad i=1,2,3,4$ $\quad f(x)=x_1^2+x_2^2+x_3^2+x_4^2$

Decision variables: x_1, x_2, x_3 and x_4

- Step 1: Fix the population size, inertia, acceleration coefficient, maximum iterations

  $N_P = 5$, $w = 0.7$, $c_1 = 1.5$, $c_2 = 1.5$, $T = 10$

- Step 2: Generate random positions within the domain. Evaluate the fitness.

$$P=\begin{bmatrix}4 & 0 & 0 & 8\\ 3 & 1 & 9 & 7\\ 0 & 3 & 1 & 5\\ 2 & 1 & 4 & 9\\ 6 & 2 & 8 & 3\end{bmatrix} \quad f=\begin{bmatrix}80\\ 140\\ 35\\ 102\\ 113\end{bmatrix}$$

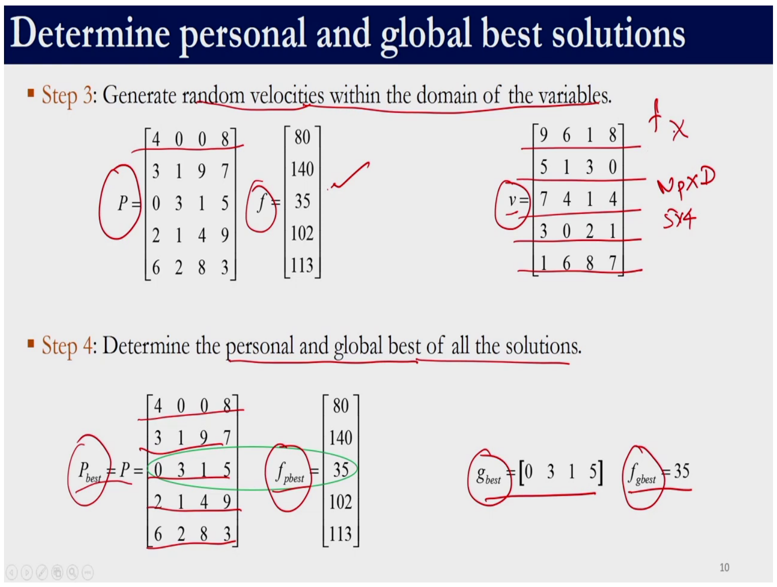
9

So, the first step is to fix the population size inertia acceleration coefficient maximum iterations right. So, let us say the population size is 5, the inertia to be 0.7, the acceleration coefficients to be 1.5 and 1.5 respectively and that we would like to do 10 iterations right.

So, the next step is to generate random positions within the domain of the decision variables. So, we know the domain of the decision variable is 0 to 10. So, in this domain we need to generate five solutions right. So, here we have taken integers so, that it is easier to calculate right. So, these are the five positions right.

So, for optimization we will be generally calling it as a population, but these are particles because in particle swarm optimization these are considered to be five particles and these values are its position right. So, these are the positions which is nothing, but the value of the decision variables right and this is the fitness function value for each of the solution right.

(Refer Slide Time: 21:36)



Determine personal and global best solutions

- Step 3: Generate random velocities within the domain of the variables.

$$P = \begin{bmatrix} 4 & 0 & 0 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 6 & 2 & 8 & 3 \end{bmatrix} \quad f = \begin{bmatrix} 80 \\ 140 \\ 35 \\ 102 \\ 113 \end{bmatrix} \quad v = \begin{bmatrix} 9 & 6 & 1 & 8 \\ 5 & 1 & 3 & 0 \\ 7 & 4 & 1 & 4 \\ 3 & 0 & 2 & 1 \\ 1 & 6 & 8 & 7 \end{bmatrix}$$

f x
$N_p \times D$
5×4

- Step 4: Determine the personal and global best of all the solutions.

$$P_{best} = P = \begin{bmatrix} 4 & 0 & 0 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 6 & 2 & 8 & 3 \end{bmatrix} \quad f_{pbest} = \begin{bmatrix} 80 \\ 140 \\ 35 \\ 102 \\ 113 \end{bmatrix} \quad g_{best} = [0 \;\; 3 \;\; 1 \;\; 5] \quad f_{gbest} = 35$$

10

So, the next step is to initialize random velocities right. So, we will also generate random velocities within the domain of the decision variable right. So, domain of the decision variable we know 0 to 10, right. So, within a 0 to 10 we need to generate a velocity right; velocity for each particle right. So, for each particle means we have this five rows right, this is the velocity of particle 1, this is the velocity of particle 2, particle 3, particle 4 and particular 5 right and it is also a 5 cross 4 because this is N p cross D; D is the number of decision variable in our problems right. So, remember velocity is not a scalar value right.

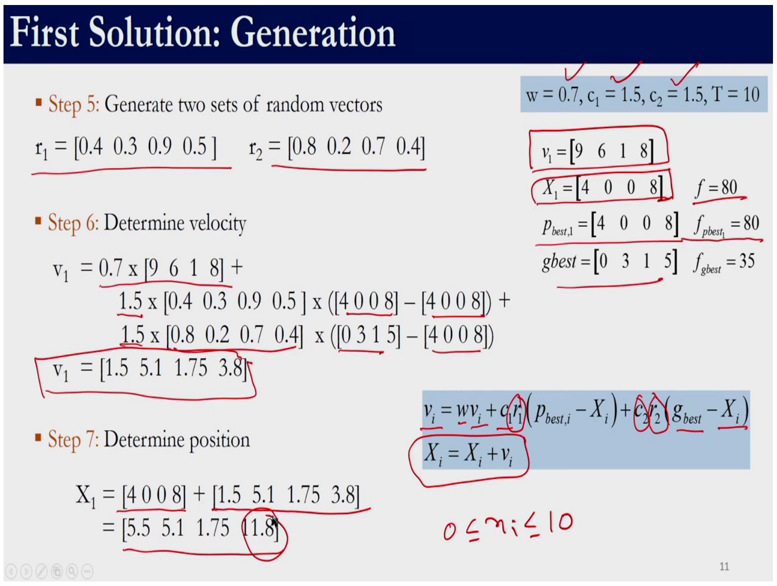
So, again it we might feel that it is required to calculate the fitness function of this velocity, but no fitness function of the velocity is not required, so only the fitness function value of the positions are required which we have already calculated over here right. So, it is not necessary to calculate the fitness of this velocity right. So, it is sufficient to calculate the fitness of the positions right.

So, next step is to determine the personal and global best of all the solutions. So, we have five particles right. So, for each particle since it is the first iteration the best pose the personal best is considered to be the solution itself right. So, for the first one 4 0 0 8, 4 0 0 8 is the personal best because in the first iteration we have we do not have any historical information. So, the current solution itself is the current direction itself has taken as the best direction right. So, this is only for the first iteration. We will see that for the second iteration this will change right.

So, these are the p best for each particle. So, p best for the first particle is 4 0 0 8; p best for the second particle is 3 1 9 7; p best for the third particle is 0 3 1 5; p best for the fourth particle is 2 1 4 9 and p best for the fifth particle is 6 2 8 3 9 and similarly we take the same objective function value. We do not need to re recalculate them because we will get the same answers right. So, we directly assign p as p best and f as f of p best only for the first iteration right. After this first iteration is complete you will see that the personal best values and the current positions can be different.

So, once we are done with this right we need to identify the global best right. So, the global best if we see now in this vector of fitness 35 is the global best right. So, that is f of g best and the solution corresponding to it is 0 3 1 5. So, this is the position, these are the velocities, these are the personal best, this is the g best right. So, for position we have the fitness function for p best we have a f of p best, for g best we have f of g best. There is no need to calculate the fitness function of the velocities.

(Refer Slide Time: 24:30)



First Solution: Generation

- Step 5: Generate two sets of random vectors

$r_1 = [0.4\ \ 0.3\ \ 0.9\ \ 0.5\,]$ $r_2 = [0.8\ \ 0.2\ \ 0.7\ \ 0.4]$

- Step 6: Determine velocity

$v_1 = 0.7 \times [9\ \ 6\ \ 1\ \ 8] +$
$1.5 \times [0.4\ \ 0.3\ \ 0.9\ \ 0.5\,] \times ([4\,0\,0\,8] - [4\,0\,0\,8]) +$
$1.5 \times [0.8\ \ 0.2\ \ 0.7\ \ 0.4] \times ([0\,3\,1\,5] - [4\,0\,0\,8])$
$v_1 = [1.5\ \ 5.1\ \ 1.75\ \ 3.8]$

- Step 7: Determine position

$X_1 = [4\,0\,0\,8] + [1.5\ \ 5.1\ \ 1.75\ \ 3.8]$
$= [5.5\ \ 5.1\ \ 1.75\ \ 11.8]$

w = 0.7, c_1 = 1.5, c_2 = 1.5, T = 10

$v_1 = [9\ \ 6\ \ 1\ \ 8]$
$X_1 = [4\ \ 0\ \ 0\ \ 8]$ $f = 80$
$p_{best,1} = [4\ \ 0\ \ 0\ \ 8]$ $f_{pbest_1} = 80$
$gbest = [0\ \ 3\ \ 1\ \ 5]$ $f_{gbest} = 35$

$$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$$
$$X_i = X_i + v_i$$

$0 \le x_i \le 10$

11

The next step is to generate the velocities right. So, we are now in the first iteration so, the first solution in the first iteration. So, we need to find out it is updated velocity right. So, updated velocity v i is going to be w which we have fixed 0.7 into v i; v i we already have generated it randomly so, that is 9618, c 1 it is a user defined constant we have taken it as 1.5, r 1 – r 1 let us take it to be 0.4, 0.3, 0.9 and 0.5 right and this vector. So, the difference between the p best so, the current p best is 4 0 0 8 and its f best is 80 and the current position is also 4 0 0 8 and obviously, that will also have the fitness of 80.

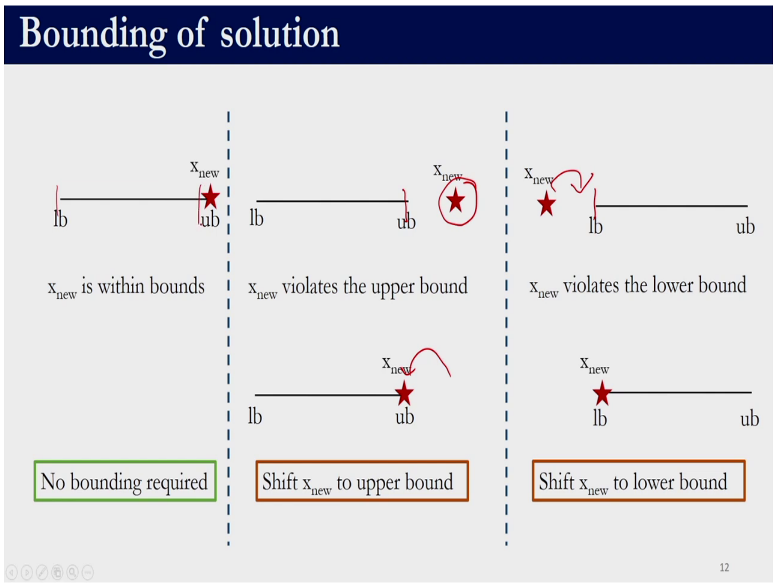
So, this is the p best of 1 minus X of 1 right plus c 2 again that is a user defined parameter; in this case it has been set to 1.5 we need another set of random numbers. So, let that be 0.8, 0.2, 0.7 and 0.4, we require g best which we have identified that it is 0 3 1 5 and the current position which is again 4 0 0 8 right. So, if we plug those values into this. So, 0.7 into 9 6 1 8, in the first iteration p best and the solution are the same the current direction and the current p best are the same right. So, this will contribute to 0.

This is going to happen as you will see this will happen for all the five solutions, but remember this will happen only in the first iteration. In the second iteration this these two can be different right this p best of i and X i X of i can be different. So, this is going to contribute 0 as of now right the third part is c 2 into the random numbers which we have selected right global best 0 3 1 5 and the current solution 4 0 0 8. So, if we evaluate this we get this velocity right.

Once we have found the velocity we will use this equation the second equation to update the position. So, the current position of the first particle is 4 0 0 8 right plus the newly determined velocity is 1.5, 5.1, 1.75 and 3.8 right. So, the new position is 5.5, 5.1, 1.75 and 11.8 right. So, this is the new position from the perspective of optimization is this is a new solution right. So, since this is a new solution we will have to see whether if the solution is within the bounds.

So, in this case the bound that we are working with is for all the variables that it has to be below between 0 and 10 right. So, this particular variable is actually violating the bounds right. So, we need to bring it back into the bound.

(Refer Slide Time: 27:16)

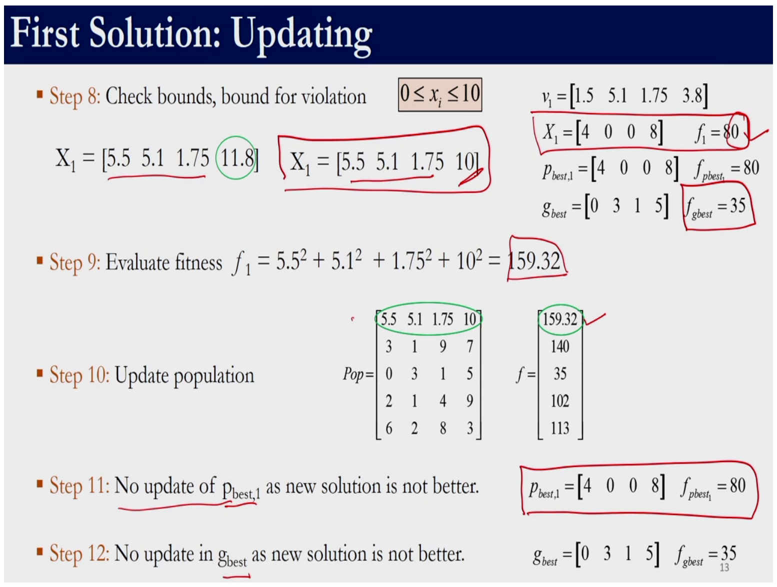


Bounding of solution

x_{new}
lb
ub
x_{new} is within bounds
No bounding required
x_{new}
lb
ub
x_{new} violates the upper bound
x_{new}
lb
ub
Shift x_{new} to upper bound
x_{new}
lb
ub
x_{new} violates the lower bound
x_{new}
lb
ub
Shift x_{new} to lower bound
12

So, again as we discussed previously we will use the corner bounding strategy right. Corner bounding strategies specifies that if something is violating the upper bound so, if particular variable is violating the upper bond so, the upper bound is here if it is violating the upper bond it has to be brought back to the upper bound right. Same thing for the lower bound that if it is violating the lower bound it is to be pulled back to the lower bound and if a solution happens to be between upper and lower bound, then we do not need to do anything right.

(Refer Slide Time: 27:45)



First Solution: Updating

- Step 8: Check bounds, bound for violation $\boxed{0 \le x_i \le 10}$

  $X_1 = [5.5\ \ 5.1\ \ 1.75\ \ 11.8]$ $\boxed{X_1 = [5.5\ \ 5.1\ \ 1.75\ \ 10]}$

  $v_1 = [1.5\ \ 5.1\ \ 1.75\ \ 3.8]$

  $X_1 = [4\ \ 0\ \ 0\ \ 8]$ $f_1 = 80$

  $p_{best,1} = [4\ \ 0\ \ 0\ \ 8]$ $f_{pbest_1} = 80$

  $g_{best} = [0\ \ 3\ \ 1\ \ 5]$ $\boxed{f_{gbest} = 35}$

- Step 9: Evaluate fitness $f_1 = 5.5^2 + 5.1^2 + 1.75^2 + 10^2 = 159.32$

- Step 10: Update population

$$Pop = \begin{bmatrix} 5.5 & 5.1 & 1.75 & 10 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 6 & 2 & 8 & 3 \end{bmatrix} \quad f = \begin{bmatrix} 159.32 \\ 140 \\ 35 \\ 102 \\ 113 \end{bmatrix}$$

- Step 11: No update of $p_{best,1}$ as new solution is not better. $\boxed{p_{best,1} = [4\ \ 0\ \ 0\ \ 8]\ \ f_{pbest_1} = 80}$

- Step 12: No update in g_{best} as new solution is not better. $g_{best} = [0\ \ 3\ \ 1\ \ 5]$ $f_{gbest} = 35$

13

So, in this case this if we apply corner bounding these three values will survive as such only this 11.8 will be converted to 10, right. So, this is a solution right we can evaluate the objective function for this solution so, that turns out to be 159.32. Remember, there is no greedy selection over here, whether the solution is better or not does not matter it is to be included in the population or in the position the. So, the position is to be definitely updated right.

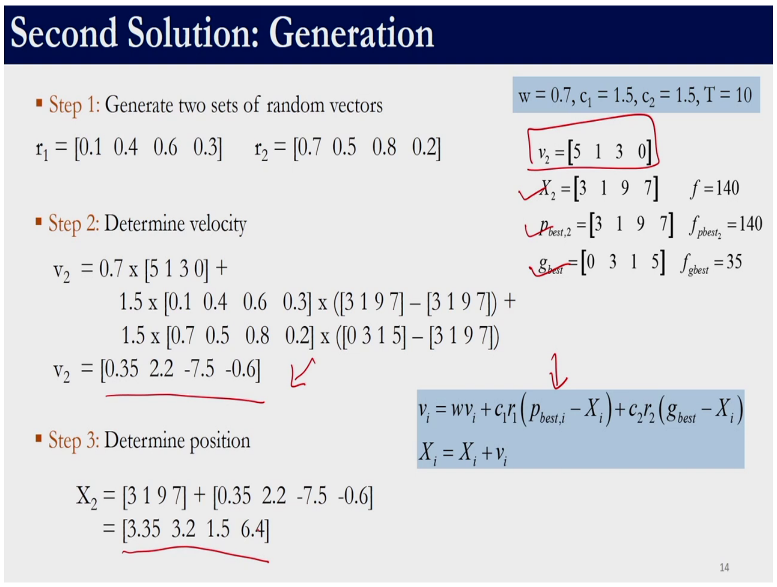
So, the solution which we use to generate is this one. It has a fitness function of 80 right whereas, here the new solution which we have got is 159.32. Remember, we are working with a minimization problem. So, actually this 4 0 0 8 is better than a 5.5, 5.1, 1.75 and 10 despite that fact we will update the population right. So, the population is to be updated right. So, that is a fundamental difference between TLBO and particle swarm optimization that for a solution to come into the population we do not need to perform a greedy selection right. So, this solution enters the population.

Now, we need to see if we need to update the p best or the g best right. So, since it is the first solution we can only update p best 1 if it is a better solution right. So, what we have got is 159.32 and what we had was 80 right. So, between these two 80 is a better solution. So, that will be p best of 1 right the solution in the population has changed, but the p best is not changed.

So, for this particle currently it has a position with the fitness of 159.32, but in the past it had a solution 4 0 0 8 for which the fitness was 80 right. So, the personal best for the first particle is not to be updated why because the fitness that we had previously is better right. So, we do not update this personal best one right.

So, now if we compare with the global best also you will realize that global best cannot be updated because global best what we have is 35 right and this solution has a fitness of 159.32. So, no update in g best right.

(Refer Slide Time: 30:07)



Second Solution: Generation

- Step 1: Generate two sets of random vectors

$r_1 = [0.1 \;\; 0.4 \;\; 0.6 \;\; 0.3]$ $r_2 = [0.7 \;\; 0.5 \;\; 0.8 \;\; 0.2]$

- Step 2: Determine velocity

$v_2 = 0.7 \times [5\;1\;3\;0] + 1.5 \times [0.1 \;\; 0.4 \;\; 0.6 \;\; 0.3] \times ([3\;1\;9\;7] - [3\;1\;9\;7]) + 1.5 \times [0.7 \;\; 0.5 \;\; 0.8 \;\; 0.2] \times ([0\;3\;1\;5] - [3\;1\;9\;7])$

$v_2 = [0.35 \;\; 2.2 \;\; -7.5 \;\; -0.6]$

- Step 3: Determine position

$X_2 = [3\;1\;9\;7] + [0.35 \;\; 2.2 \;\; -7.5 \;\; -0.6]$
$= [3.35 \;\; 3.2 \;\; 1.5 \;\; 6.4]$

$w = 0.7, c_1 = 1.5, c_2 = 1.5, T = 10$

$v_2 = [5 \;\; 1 \;\; 3 \;\; 0]$

$X_2 = [3 \;\; 1 \;\; 9 \;\; 7]$ $f = 140$

$p_{best,2} = [3 \;\; 1 \;\; 9 \;\; 7]$ $f_{pbest_2} = 140$

$g_{best} = [0 \;\; 3 \;\; 1 \;\; 5]$ $f_{gbest} = 35$

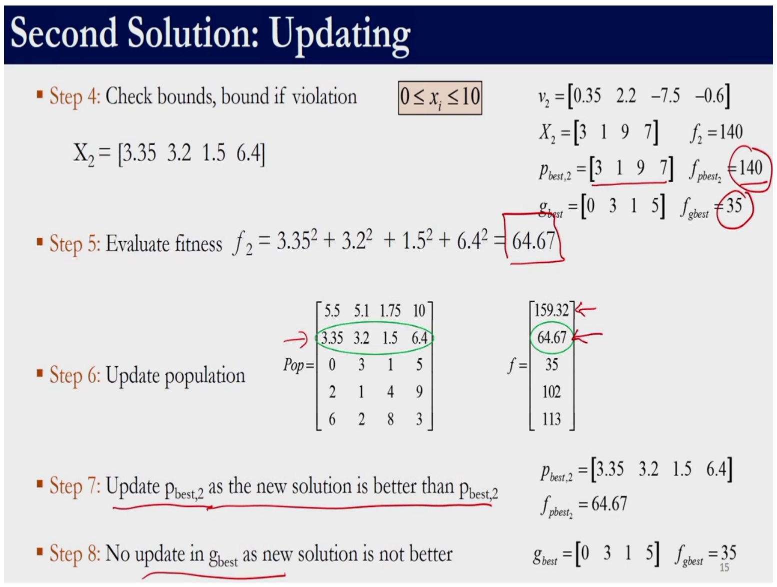
$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$
$X_i = X_i + v_i$

14

So, coming to the second solution so, the velocity is we had randomly generated this is a our X 2 right, this is our p best 2 is the same as X 2 for the first generation at least right and this is our g best. So, if we apply this equation and calculate its velocity right we see that we get 0.35 2.2 minus 7.5 and minus 0.6.

Remember, we are not supposed to bound the velocity right the velocities can though it as a 1 cross 4 vector which is similar to our position vector we are not supposed to bound this right. We will bound only the positions before calculating the objective function right. So, in this case if we find the position it is 3.35 3.2 1.5 6.4 which is well within the bounds so, we do not need to bound it right.

(Refer Slide Time: 30:51)



Second Solution: Updating

- Step 4: Check bounds, bound if violation $\boxed{0 \le x_i \le 10}$

  $X_2 = [3.35\ \ 3.2\ \ 1.5\ \ 6.4]$

$v_2 = [0.35\ \ 2.2\ \ -7.5\ \ -0.6]$

$X_2 = [3\ \ 1\ \ 9\ \ 7]$ $\quad f_2 = 140$

$p_{best,2} = [3\ \ 1\ \ 9\ \ 7]$ $\quad f_{pbest_2} = 140$

$g_{best} = [0\ \ 3\ \ 1\ \ 5]$ $\quad f_{gbest} = 35$

- Step 5: Evaluate fitness $f_2 = 3.35^2 + 3.2^2 + 1.5^2 + 6.4^2 = \boxed{64.67}$

- Step 6: Update population

$$Pop = \begin{bmatrix} 5.5 & 5.1 & 1.75 & 10 \\ 3.35 & 3.2 & 1.5 & 6.4 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 6 & 2 & 8 & 3 \end{bmatrix} \qquad f = \begin{bmatrix} 159.32 \\ 64.67 \\ 35 \\ 102 \\ 113 \end{bmatrix}$$

- Step 7: Update $p_{best,2}$ as the new solution is better than $p_{best,2}$

$p_{best,2} = [3.35\ \ 3.2\ \ 1.5\ \ 6.4]$

$f_{pbest_2} = 64.67$

- Step 8: No update in g_{best} as new solution is not better

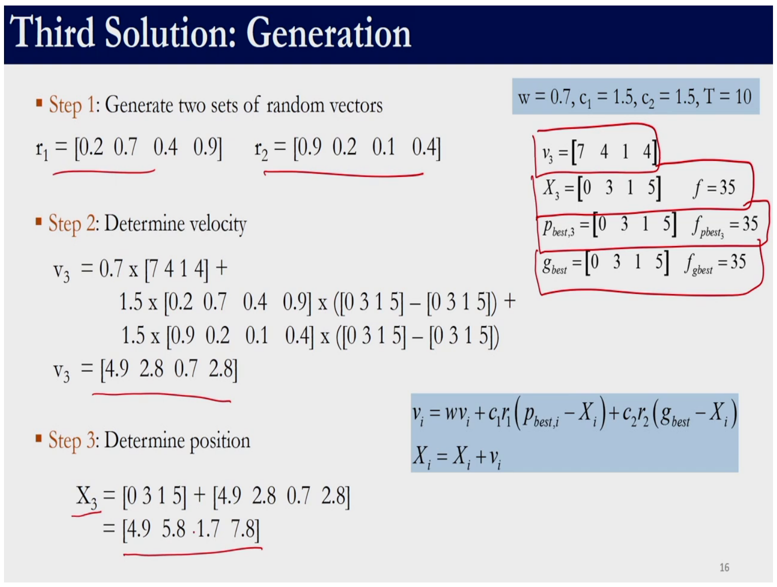
$g_{best} = [0\ \ 3\ \ 1\ \ 5]$ $\quad f_{gbest} = 35$

15

And, if we calculate the fitness of it happens that we get a fitness of 64.7, we are not supposed to compare. Remember, for the first solution we did not compare. So, similarly for the second solution also we are not going to compare with anything to update the population. The second solution is updated into that population right. Now, we have before going to the third solution, we need to see if we have to update the p best and g best.

So, in this case if we see the p best that we have is 3 1 9 7 with the fitness of 140 right, but now I have a position which is better than this 140. So, we will update it right update p best since the new solution is better than p best 2, right. So, for every solution we have its corresponding p best right so, but if we compare with g best g best is still 35 and the solution which we have obtained a 64.67. So, obviously, 64.67 is not better than 35. So, we do not need to update the g best.

(Refer Slide Time: 31:53)



Third Solution: Generation

$w = 0.7, c_1 = 1.5, c_2 = 1.5, T = 10$

- Step 1: Generate two sets of random vectors

$r_1 = [0.2\ \ 0.7\ \ 0.4\ \ 0.9]$ $r_2 = [0.9\ \ 0.2\ \ 0.1\ \ 0.4]$

$v_3 = [7\ \ 4\ \ 1\ \ 4]$

$X_3 = [0\ \ 3\ \ 1\ \ 5]$ $f = 35$

$p_{best,3} = [0\ \ 3\ \ 1\ \ 5]$ $f_{pbest_3} = 35$

$g_{best} = [0\ \ 3\ \ 1\ \ 5]$ $f_{gbest} = 35$

- Step 2: Determine velocity

$v_3 = 0.7 \times [7\ 4\ 1\ 4] +$
$1.5 \times [0.2\ \ 0.7\ \ 0.4\ \ 0.9] \times ([0\ 3\ 1\ 5] - [0\ 3\ 1\ 5]) +$
$1.5 \times [0.9\ \ 0.2\ \ 0.1\ \ 0.4] \times ([0\ 3\ 1\ 5] - [0\ 3\ 1\ 5])$

$v_3 = [4.9\ \ 2.8\ \ 0.7\ \ 2.8]$

$$v_i = wv_i + c_1 r_1 \left(p_{best,i} - X_i\right) + c_2 r_2 \left(g_{best} - X_i\right)$$
$$X_i = X_i + v_i$$

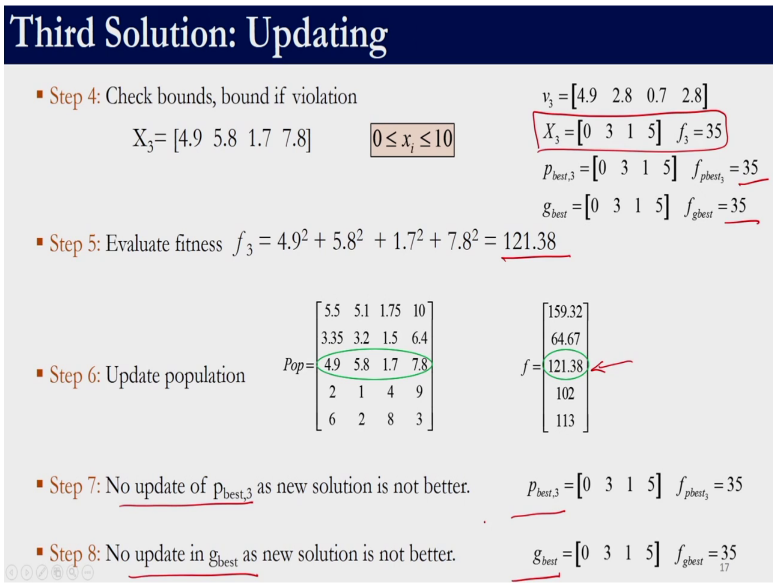
- Step 3: Determine position

$X_3 = [0\ 3\ 1\ 5] + [4.9\ \ 2.8\ \ 0.7\ \ 2.8]$
$= [4.9\ \ 5.8\ \ 1.7\ \ 7.8]$

16

So, coming to the third solution similarly we need two sets of random numbers between 0 and 1, we generate it. The velocity v 3 was generated randomly we know the position current position for the first iteration it happens that the current position and the p best are the same, we also have the g best over here right. So, if we apply this equation we get the velocity, we calculate the position and then we look at this position. If the position is violating the bounds we update the position not the velocity.

(Refer Slide Time: 32:23)



Third Solution: Updating

- Step 4: Check bounds, bound if violation

  $X_3 = [4.9\ \ 5.8\ \ 1.7\ \ 7.8]$ $\boxed{0 \le x_i \le 10}$

$v_3 = [4.9\ \ 2.8\ \ 0.7\ \ 2.8]$

$X_3 = [0\ \ 3\ \ 1\ \ 5]\ \ f_3 = 35$

$p_{best,3} = [0\ \ 3\ \ 1\ \ 5]\ \ f_{pbest_3} = 35$

$g_{best} = [0\ \ 3\ \ 1\ \ 5]\ \ f_{gbest} = 35$

- Step 5: Evaluate fitness $f_3 = 4.9^2 + 5.8^2 + 1.7^2 + 7.8^2 = \underline{121.38}$

- Step 6: Update population

$$Pop = \begin{bmatrix} 5.5 & 5.1 & 1.75 & 10 \\ 3.35 & 3.2 & 1.5 & 6.4 \\ 4.9 & 5.8 & 1.7 & 7.8 \\ 2 & 1 & 4 & 9 \\ 6 & 2 & 8 & 3 \end{bmatrix} \qquad f = \begin{bmatrix} 159.32 \\ 64.67 \\ 121.38 \\ 102 \\ 113 \end{bmatrix}$$

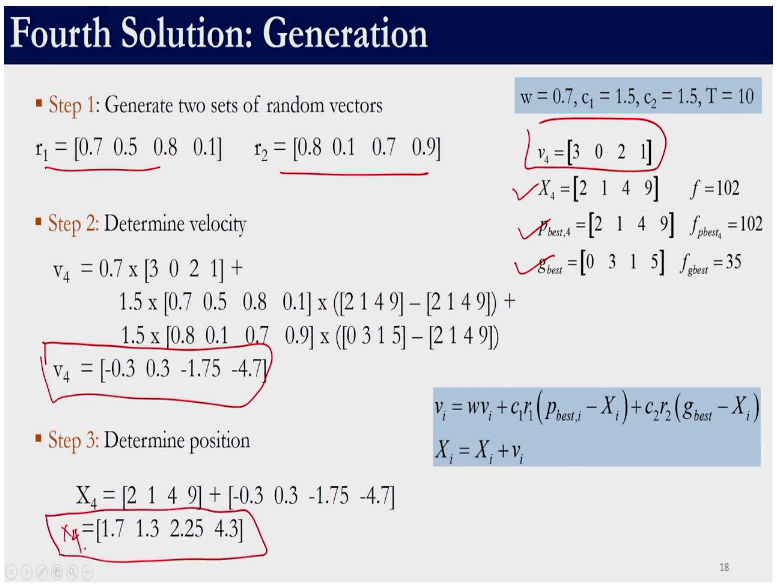
- Step 7: No update of $p_{best,3}$ as new solution is not better. $p_{best,3} = [0\ \ 3\ \ 1\ \ 5]\ \ f_{pbest_3} = 35$

- Step 8: No update in g_{best} as new solution is not better. $g_{best} = [0\ \ 3\ \ 1\ \ 5]\ \ f_{gbest} = 35$

17

So, we determined the objective function value to be 121.38, right. So, for the third solution if we see what we started was with 35 right; what we have ended up with is 121.38, but still it will enter the population right, but there would not be any update of p best 3 or g best 3 because p best is 35; g best is also 35 and both of them are better than 121.38. So, I do not need to update p best 3 or g best. So, the previous value is retained. So, that is for the third solution.

(Refer Slide Time: 32:57)



Fourth Solution: Generation

- Step 1: Generate two sets of random vectors

$r_1 = [0.7\ \ 0.5\ \ 0.8\ \ 0.1]$ $\quad r_2 = [0.8\ \ 0.1\ \ 0.7\ \ 0.9]$

- Step 2: Determine velocity

$v_4 = 0.7 \times [3\ 0\ 2\ 1] +$
$1.5 \times [0.7\ \ 0.5\ \ 0.8\ \ 0.1] \times ([2\ 1\ 4\ 9] - [2\ 1\ 4\ 9]) +$
$1.5 \times [0.8\ \ 0.1\ \ 0.7\ \ 0.9] \times ([0\ 3\ 1\ 5] - [2\ 1\ 4\ 9])$

$v_4 = [-0.3\ \ 0.3\ \ -1.75\ \ -4.7]$

- Step 3: Determine position

$X_4 = [2\ \ 1\ \ 4\ \ 9] + [-0.3\ \ 0.3\ \ -1.75\ \ -4.7]$

$X_4 = [1.7\ \ 1.3\ \ 2.25\ \ 4.3]$

$w = 0.7, c_1 = 1.5, c_2 = 1.5, T = 10$

$v_4 = [3\ \ 0\ \ 2\ \ 1]$

✓ $X_4 = [2\ \ 1\ \ 4\ \ 9] \quad f = 102$

✓ $p_{best,4} = [2\ \ 1\ \ 4\ \ 9] \quad f_{pbest_4} = 102$

✓ $g_{best} = [0\ \ 3\ \ 1\ \ 5] \quad f_{gbest} = 35$

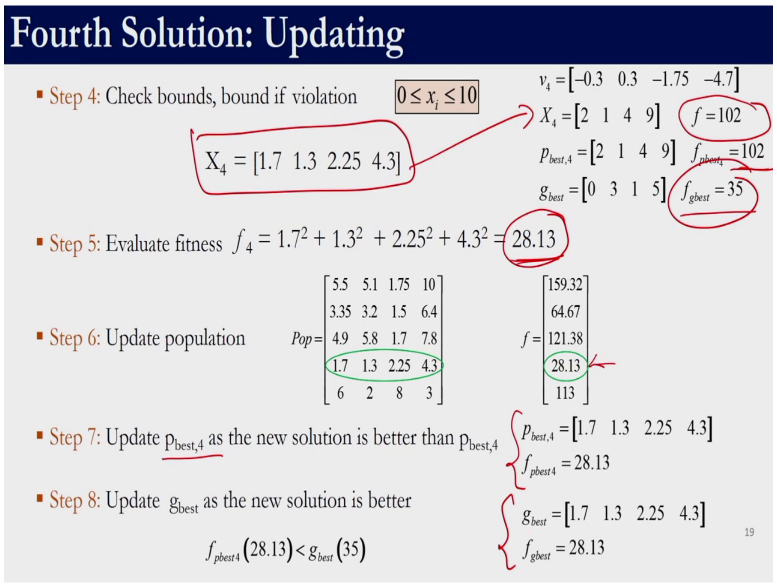
$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$

$X_i = X_i + v_i$

18

So, for the fourth solution, the velocity is again given, the position is known, the p best is known, the g best known. If you apply this equation with these two sets of random numbers you are you will get a velocity of this right and then we find new solution or new position. Again, we can find the objective function of this.

(Refer Slide Time: 33:19)



Fourth Solution: Updating

- Step 4: Check bounds, bound if violation $\boxed{0 \le x_i \le 10}$

$$X_4 = [1.7\ \ 1.3\ \ 2.25\ \ 4.3]$$

$$v_4 = [-0.3\ \ 0.3\ \ -1.75\ \ -4.7]$$
$$X_4 = [2\ \ 1\ \ 4\ \ 9] \quad f = 102$$
$$p_{best,4} = [2\ \ 1\ \ 4\ \ 9] \quad f_{pbest4} = 102$$
$$g_{best} = [0\ \ 3\ \ 1\ \ 5] \quad f_{gbest} = 35$$

- Step 5: Evaluate fitness $f_4 = 1.7^2 + 1.3^2 + 2.25^2 + 4.3^2 = 28.13$
- Step 6: Update population

$$Pop = \begin{bmatrix} 5.5 & 5.1 & 1.75 & 10 \\ 3.35 & 3.2 & 1.5 & 6.4 \\ 4.9 & 5.8 & 1.7 & 7.8 \\ 1.7 & 1.3 & 2.25 & 4.3 \\ 6 & 2 & 8 & 3 \end{bmatrix} \qquad f = \begin{bmatrix} 159.32 \\ 64.67 \\ 121.38 \\ 28.13 \\ 113 \end{bmatrix}$$

- Step 7: Update $p_{best,4}$ as the new solution is better than $p_{best,4}$

$$p_{best,4} = [1.7\ \ 1.3\ \ 2.25\ \ 4.3]$$
$$f_{pbest4} = 28.13$$

- Step 8: Update g_{best} as the new solution is better

$$f_{pbest4}(28.13) < g_{best}(35)$$

$$g_{best} = [1.7\ \ 1.3\ \ 2.25\ \ 4.3]$$
$$f_{gbest} = 28.13$$

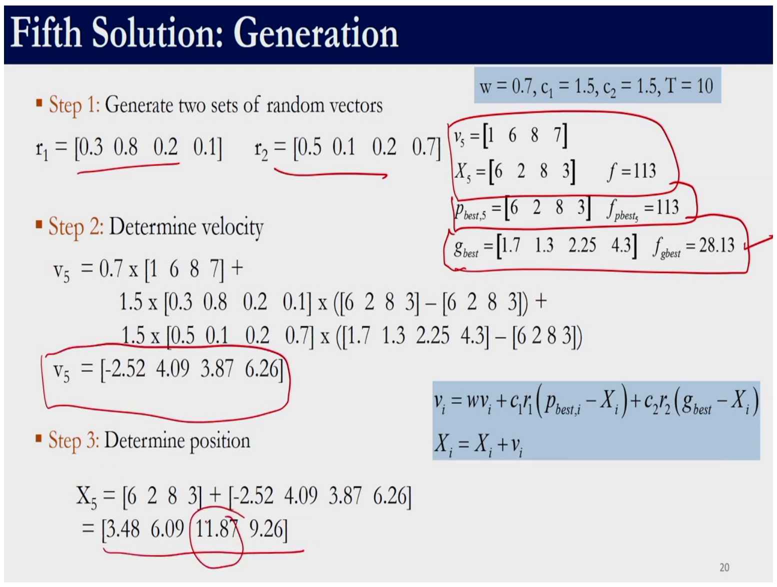
19

So, if you find the objective function value of this we get 28.13. So, the current solution which was used to generate this X 4 is this one 2 1 4 9 right. It has a fitness of 102 it has 28.3 28.13. However, even if it had been 128.13 we will update it right. As we have been stressing it time and again the new solution will always be updated in the population there is no greedy selection right. So, the solution enters the population right. So, that is done.

Now, we need to compare the p best right. So, the p best currently is 102 and the solution which we have is 28.3. So, it is better right. So, we update the p best and f best right. Similarly, in this case it happens that the newly generated solution right is also better than g best. So, does the g best was 35 and what we have obtained is 28.3 28.13. So, 28.13 is less than 35. So, we also update the g best right. So, if you see we have tried to capture all the possible scenarios.

One solution wherein no p best update no g best update is required, another solution required p best update, another solution required g best up update right. So, the example has been carefully constructed so that we can we come across all the possible cases right.

(Refer Slide Time: 34:47)



Fifth Solution: Generation

$w = 0.7, c_1 = 1.5, c_2 = 1.5, T = 10$

- Step 1: Generate two sets of random vectors

$r_1 = [0.3\ \ 0.8\ \ 0.2\ \ 0.1]$ $r_2 = [0.5\ \ 0.1\ \ 0.2\ \ 0.7]$

$v_5 = [1\ \ 6\ \ 8\ \ 7]$

$X_5 = [6\ \ 2\ \ 8\ \ 3]$ $f = 113$

$p_{best,5} = [6\ \ 2\ \ 8\ \ 3]$ $f_{pbest_5} = 113$

$g_{best} = [1.7\ \ 1.3\ \ 2.25\ \ 4.3]$ $f_{gbest} = 28.13$

- Step 2: Determine velocity

$v_5 = 0.7 \times [1\ \ 6\ \ 8\ \ 7] +$
$1.5 \times [0.3\ \ 0.8\ \ 0.2\ \ 0.1] \times ([6\ \ 2\ \ 8\ \ 3] - [6\ \ 2\ \ 8\ \ 3]) +$
$1.5 \times [0.5\ \ 0.1\ \ 0.2\ \ 0.7] \times ([1.7\ \ 1.3\ \ 2.25\ \ 4.3] - [6\ 2\ 8\ 3])$

$v_5 = [-2.52\ \ 4.09\ \ 3.87\ \ 6.26]$

$$v_i = wv_i + c_1 r_1 \left(p_{best,i} - X_i\right) + c_2 r_2 \left(g_{best} - X_i\right)$$
$$X_i = X_i + v_i$$

- Step 3: Determine position

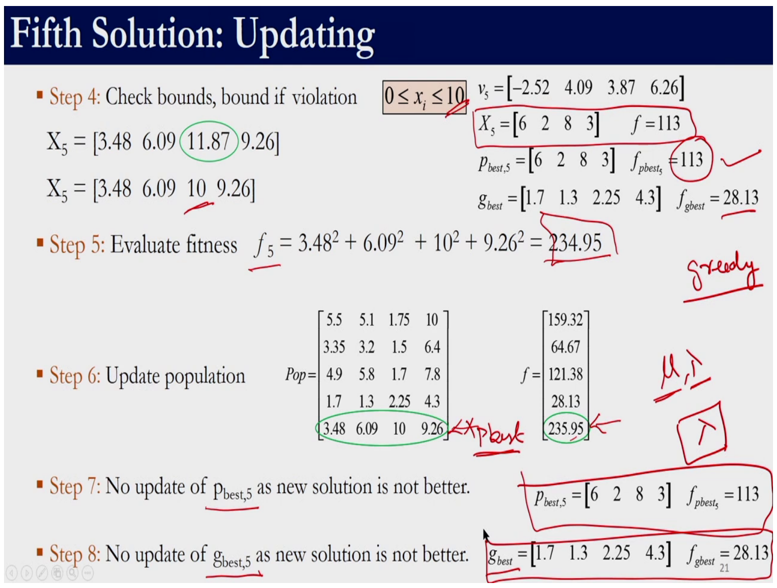
$X_5 = [6\ \ 2\ \ 8\ \ 3] + [-2.52\ \ 4.09\ \ 3.87\ \ 6.26]$
$= [3.48\ \ 6.09\ \ 11.87\ \ 9.26]$

20

So, similarly fifth case you can try it out right. So, we know the velocity the position the current best the p best of the fifth particle is known, the g best is known. Remember, this g best is not 35 which we which was the case when we started right, but in the previous solution for the fourth solution the g best was updated.

Since we have not updated g best we will take into consideration the updated g best right. So, with these two random numbers if we calculate the velocity will turn out to be this right and then we can find out the position. So, here in this case 11.87 is out of bounds.

(Refer Slide Time: 35:24)



Fifth Solution: Updating

- Step 4: Check bounds, bound if violation $\boxed{0 \le x_i \le 10}$

  $X_5 = [3.48\ \ 6.09\ \ 11.87\ \ 9.26]$

  $X_5 = [3.48\ \ 6.09\ \ 10\ \ 9.26]$

$v_5 = [-2.52\ \ 4.09\ \ 3.87\ \ 6.26]$

$X_5 = [6\ \ 2\ \ 8\ \ 3]\quad f = 113$

$p_{best,5} = [6\ \ 2\ \ 8\ \ 3]\quad f_{pbest_5} = 113$

$g_{best} = [1.7\ \ 1.3\ \ 2.25\ \ 4.3]\quad f_{gbest} = 28.13$

- Step 5: Evaluate fitness $f_5 = 3.48^2 + 6.09^2 + 10^2 + 9.26^2 = 234.95$

- Step 6: Update population

$$Pop = \begin{bmatrix} 5.5 & 5.1 & 1.75 & 10 \\ 3.35 & 3.2 & 1.5 & 6.4 \\ 4.9 & 5.8 & 1.7 & 7.8 \\ 1.7 & 1.3 & 2.25 & 4.3 \\ 3.48 & 6.09 & 10 & 9.26 \end{bmatrix} \qquad f = \begin{bmatrix} 159.32 \\ 64.67 \\ 121.38 \\ 28.13 \\ 235.95 \end{bmatrix}$$

- Step 7: No update of $p_{best,5}$ as new solution is not better. $p_{best,5} = [6\ \ 2\ \ 8\ \ 3]\quad f_{pbest_5} = 113$

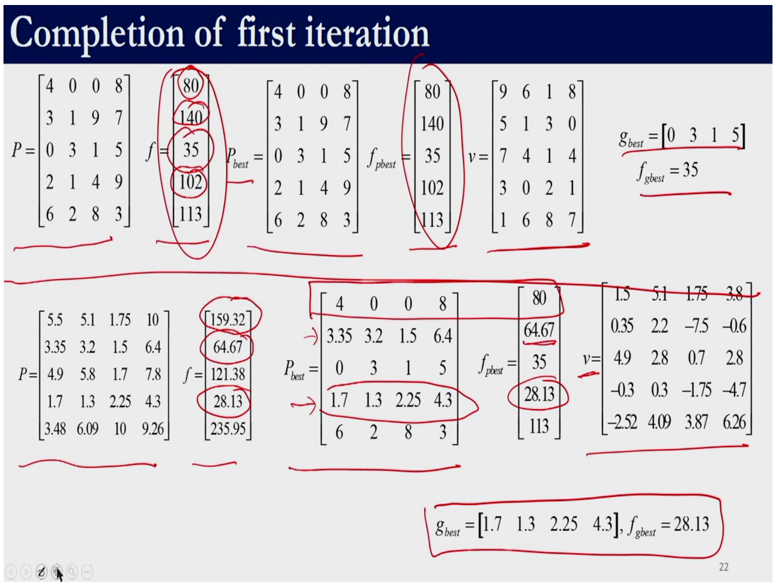
- Step 8: No update of $g_{best,5}$ as new solution is not better. $g_{best} = [1.7\ \ 1.3\ \ 2.25\ \ 4.3]\quad f_{gbest} = 28.13$

So, we will bound it to 10 using corner bounding because our upper bound is 10 right. Now, we can find out the fitness function value of the fifth solution; in this case it works out to be 234.95, right. The solution which underwent this iteration was fifth solution which had 113, right. So, what we have got is a poor solution despite of that fact the poor solution will enter because this is the newly generated solution. So, this strategy is commonly known as mu comma lambda that mu is a solution which we had and lambda is a solution which we generated. So, among mu comma lambda, lambda will survive. It will survive even if it is bad right.

So, this is not greedy selection; in greedy selection only the a solution which was better. So, among mu and lambda the one that was better survived right in mu comma lambda here we have only 1, 1 solution we are comparing the first solution with first solution; the second solution with the secondly generated solution. So, in this mu comma lambda, lambda always survives right we it is not a greedy selection mechanism right. So, we have this 235.95.

Now, we need to compare with the p best. Now, we need to decide whether p best and g best are to be updated. So, p best currently is 113 and what we got is 235.95. So, obviously, 113 is better so, that is retained right, this is not updated as p best. So, this is not p best right though it is in the population as the name itself in the indicates it is the best position of the particle so far right. Similarly, we will check for g best g best that we currently have is 28.13 and the one that we obtained is 235.95, since 28.13 is better we will not update g best right. So, this completes one iteration of particle swarm optimization.

(Refer Slide Time: 37:13)



Completion of first iteration

$$P=\begin{bmatrix}4&0&0&8\\3&1&9&7\\0&3&1&5\\2&1&4&9\\6&2&8&3\end{bmatrix}\quad f=\begin{bmatrix}80\\140\\35\\102\\113\end{bmatrix}\quad P_{best}=\begin{bmatrix}4&0&0&8\\3&1&9&7\\0&3&1&5\\2&1&4&9\\6&2&8&3\end{bmatrix}\quad f_{pbest}=\begin{bmatrix}80\\140\\35\\102\\113\end{bmatrix}\quad v=\begin{bmatrix}9&6&1&8\\5&1&3&0\\7&4&1&4\\3&0&2&1\\1&6&8&7\end{bmatrix}$$

$$g_{best}=[0\quad 3\quad 1\quad 5]$$
$$f_{gbest}=35$$

$$P=\begin{bmatrix}5.5&5.1&1.75&10\\3.35&3.2&1.5&6.4\\4.9&5.8&1.7&7.8\\1.7&1.3&2.25&4.3\\3.48&6.09&10&9.26\end{bmatrix}\quad f=\begin{bmatrix}159.32\\64.67\\121.38\\28.13\\235.95\end{bmatrix}\quad P_{best}=\begin{bmatrix}4&0&0&8\\3.35&3.2&1.5&6.4\\0&3&1&5\\1.7&1.3&2.25&4.3\\6&2&8&3\end{bmatrix}\quad f_{pbest}=\begin{bmatrix}80\\64.67\\35\\28.13\\113\end{bmatrix}\quad v=\begin{bmatrix}1.5&5.1&1.75&3.8\\0.35&2.2&-7.5&-0.6\\4.9&2.8&0.7&2.8\\-0.3&0.3&-1.75&-4.7\\-2.52&4.09&3.87&6.26\end{bmatrix}$$

$$\boxed{g_{best}=[1.7\quad 1.3\quad 2.25\quad 4.3],\ f_{gbest}=28.13}$$

22

So, this was the population which we are which we began with this was the fitness function value, this was initially since there is no memory the position of the particle themselves where considered to be the their personal best. The fitness function of the best position so far would be the same right this both would be the same in the first iteration and we generated this velocity randomly within the bounds right and this was our g best because this was the best solution that we had right.

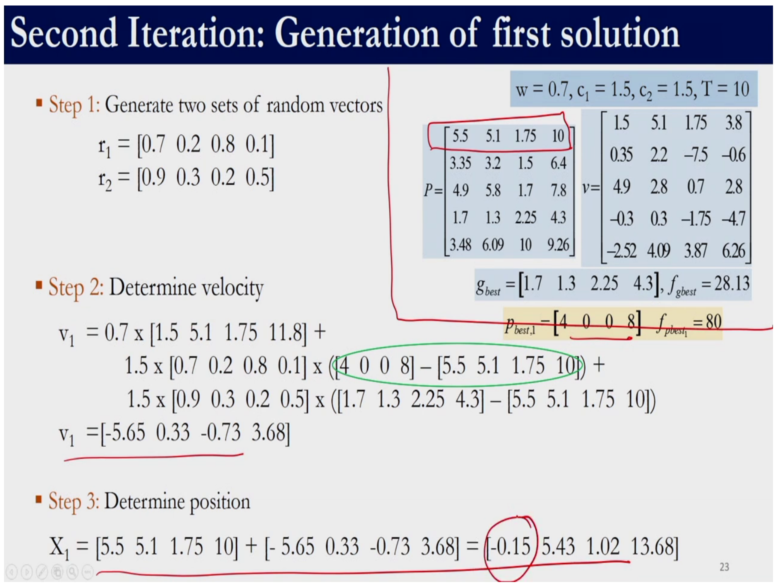
So, we started with this at the end of first iteration this was our population. So, every member would get updated. So, here if you see all the five rows are different from what we started right that is because newly generated solution is always taken into the population. We do not care if it is better or not right, updating of the population is with the new solution.

So, now, if we see p best; p best some in some cases right we have been able to update it right, the other three rows remain the remained the same right. The first row, the third row and the fifth row remain the same, because in this iteration when we perform the first iteration we got two solutions which were better than their past right. So, for example, here we had 140 and the new one which we have generated is 64.67. So, obviously, this is the personal best. For the second solution the best solution which it has encountered so far is 64.67, right.

Similarly, for the fourth solution we had 102 and we were able to generate a solution of 28.13 right. So, that has been updated over here whereas, for the first solution if you see initially we started with the fitness function value of 80 and we generated another solution which was 159.32 right. So, the one that we generated is actually bad. So, the personal best for the first member remains the same for 0 0 8 and 80 right.

And, this velocity every time we determine the velocity so, that velocity is to be used for the second generation or second iteration. So, this is the velocity that we had in generated for each and every solution and this is our g best right; g best is what you what is the best in the p best. So, p best we have five solutions; the personal best of each is known. So, the global best is the minimum among that since we are solving a minimization problem. So, that completes one iteration right.

(Refer Slide Time: 39:43)



Second Iteration: Generation of first solution

$w = 0.7, c_1 = 1.5, c_2 = 1.5, T = 10$

$$P = \begin{bmatrix} 5.5 & 5.1 & 1.75 & 10 \\ 3.35 & 3.2 & 1.5 & 6.4 \\ 4.9 & 5.8 & 1.7 & 7.8 \\ 1.7 & 1.3 & 2.25 & 4.3 \\ 3.48 & 6.09 & 10 & 9.26 \end{bmatrix} \quad v = \begin{bmatrix} 1.5 & 5.1 & 1.75 & 3.8 \\ 0.35 & 2.2 & -7.5 & -0.6 \\ 4.9 & 2.8 & 0.7 & 2.8 \\ -0.3 & 0.3 & -1.75 & -4.7 \\ -2.52 & 4.09 & 3.87 & 6.26 \end{bmatrix}$$

$g_{best} = [1.7 \;\; 1.3 \;\; 2.25 \;\; 4.3], f_{gbest} = 28.13$

$p_{best,1} = [4 \;\; 0 \;\; 0 \;\; 8] \;\; f_{pbest_1} = 80$

- Step 1: Generate two sets of random vectors

  $r_1 = [0.7 \;\; 0.2 \;\; 0.8 \;\; 0.1]$

  $r_2 = [0.9 \;\; 0.3 \;\; 0.2 \;\; 0.5]$

- Step 2: Determine velocity

  $v_1 = 0.7 \times [1.5 \;\; 5.1 \;\; 1.75 \;\; 11.8] +$
  $1.5 \times [0.7 \;\; 0.2 \;\; 0.8 \;\; 0.1] \times ([4 \;\; 0 \;\; 0 \;\; 8] - [5.5 \;\; 5.1 \;\; 1.75 \;\; 10]) +$
  $1.5 \times [0.9 \;\; 0.3 \;\; 0.2 \;\; 0.5] \times ([1.7 \;\; 1.3 \;\; 2.25 \;\; 4.3] - [5.5 \;\; 5.1 \;\; 1.75 \;\; 10])$

  $v_1 = [-5.65 \;\; 0.33 \;\; -0.73 \;\; 3.68]$

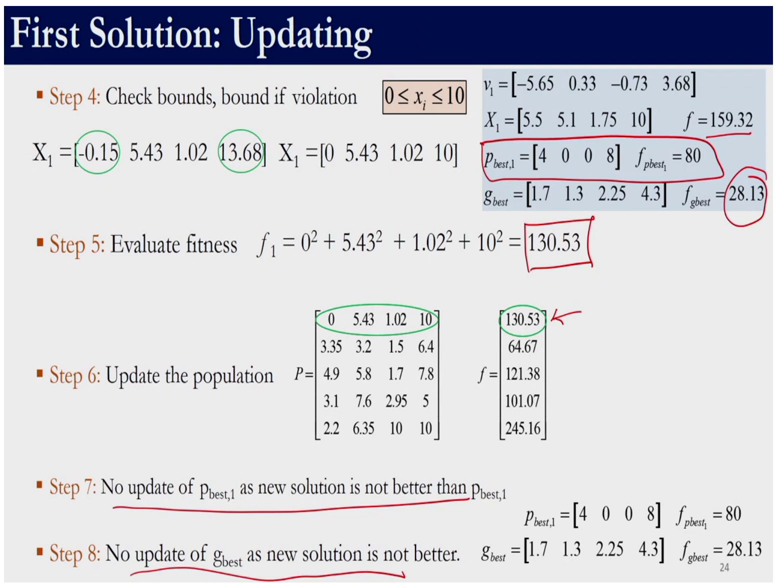
- Step 3: Determine position

  $X_1 = [5.5 \;\; 5.1 \;\; 1.75 \;\; 10] + [-5.65 \;\; 0.33 \;\; -0.73 \;\; 3.68] = [-0.15 \;\; 5.43 \;\; 1.02 \;\; 13.68]$

23

So, second iteration the same value is whatever you have seen previously have been given over here. So, the second iteration if we perform so, here you will be able to see in the first situation this term was canceling out right p best minus X i was canceling out right, but in the second iteration now we have the personal best to be different right. The personal best is 4 0 0 8 and the solution is 5.5 5.1 1.7 10 right. So, this term does not always go to 0, only in the first iteration it goes to 0, right. So, we find the velocity we find the new position since this particular variable has a value which is out of its bound, we will bound it, we will calculate its objective function.

(Refer Slide Time: 40:25)



First Solution: Updating

- Step 4: Check bounds, bound if violation $\boxed{0 \le x_i \le 10}$

$X_1 = [-0.15 \;\; 5.43 \;\; 1.02 \;\; 13.68] \quad X_1 = [0 \;\; 5.43 \;\; 1.02 \;\; 10]$

$v_1 = [-5.65 \;\; 0.33 \;\; -0.73 \;\; 3.68]$

$X_1 = [5.5 \;\; 5.1 \;\; 1.75 \;\; 10] \quad f = 159.32$

$p_{best,1} = [4 \;\; 0 \;\; 0 \;\; 8] \quad f_{pbest_1} = 80$

$g_{best} = [1.7 \;\; 1.3 \;\; 2.25 \;\; 4.3] \quad f_{gbest} = 28.13$

- Step 5: Evaluate fitness $f_1 = 0^2 + 5.43^2 + 1.02^2 + 10^2 = \boxed{130.53}$

- Step 6: Update the population

$$P = \begin{bmatrix} 0 & 5.43 & 1.02 & 10 \\ 3.35 & 3.2 & 1.5 & 6.4 \\ 4.9 & 5.8 & 1.7 & 7.8 \\ 3.1 & 7.6 & 2.95 & 5 \\ 2.2 & 6.35 & 10 & 10 \end{bmatrix} \qquad f = \begin{bmatrix} 130.53 \\ 64.67 \\ 121.38 \\ 101.07 \\ 245.16 \end{bmatrix}$$

- Step 7: No update of $p_{best,1}$ as new solution is not better than $p_{best,1}$

$p_{best,1} = [4 \;\; 0 \;\; 0 \;\; 8] \quad f_{pbest_1} = 80$

- Step 8: No update of g_{best} as new solution is not better.

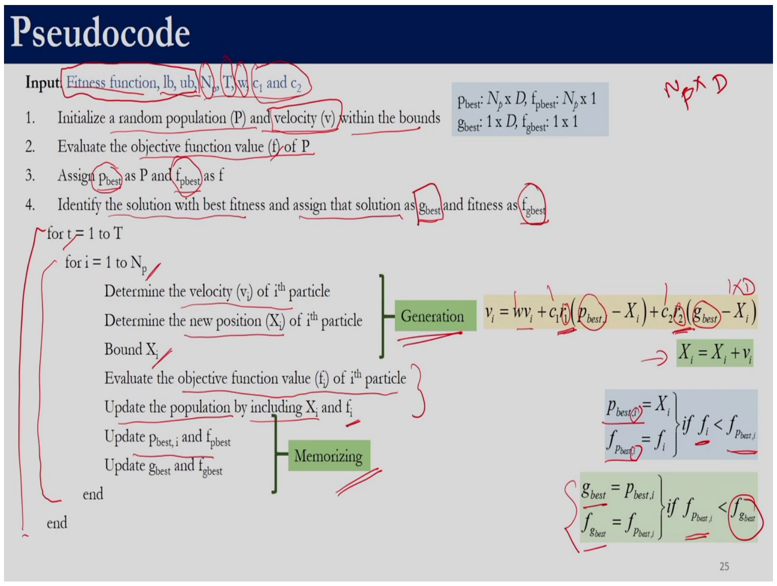
$g_{best} = [1.7 \;\; 1.3 \;\; 2.25 \;\; 4.3] \quad f_{gbest} = 28.13$

24

It is 130.53 what we started was with was 159.32 what we have got is 130.53, irrespective of whether it is better or not it will find place in the population right. So, it finds a place in the population right. For the first member the history so far says that 80 was its personal best what we have now currently got is 130.53 which is still bad than 80 right. So, we do not need to update. So, we do not update. Similarly, there is no update in g best because the g best which we have is 28.13 and the solution which we have generated has a fitness of 130.53, right.

So, similarly you can proceed for the other solutions. So, as you can see it is a extremely simple algorithm right. Only thing that we need to keep in mind is that the newly generated solution is always going to be accepted into the population, it does not matter whether it is better or bad. So, the new solution will always be used in to the population and for each member we keep a track of its personal best right and the global best is the solution which is the best among all the personal best. So, that concludes this example you can go and try out the couple of more iterations and see if you are able to implement it right.

(Refer Slide Time: 41:57)



Pseudocode

```
Input: Fitness function, lb, ub, $N_p$, T, w, $c_1$ and $c_2$
1.  Initialize a random population (P) and velocity (v) within the bounds
2.  Evaluate the objective function value (f) of P
3.  Assign $p_{best}$ as P and $f_{pbest}$ as f
4.  Identify the solution with best fitness and assign that solution as $g_{best}$ and fitness as $f_{gbest}$
for t = 1 to T
    for i = 1 to $N_p$
        Determine the velocity ($v_i$) of i^th particle
        Determine the new position ($X_i$) of i^th particle
        Bound $X_i$
        Evaluate the objective function value ($f_i$) of i^th particle
        Update the population by including $X_i$ and $f_i$
        Update $p_{best,i}$ and $f_{pbest}$
        Update $g_{best}$ and $f_{gbest}$
    end
end
```

p_{best}: N_p x D, f_{pbest}: N_p x 1
g_{best}: 1 x D, f_{gbest}: 1 x 1

$N_p \times D$

Generation

$1 \times D$

$$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$$

$$X_i = X_i + v_i$$

Memorizing

$$\left.\begin{aligned} p_{best,i} &= X_i \\ f_{p_{best,i}} &= f_i \end{aligned}\right\} if\ f_i < f_{p_{best,i}}$$

$$\left.\begin{aligned} g_{best} &= p_{best,i} \\ f_{g_{best}} &= f_{p_{best,i}} \end{aligned}\right\} if\ f_{p_{best,i}} < f_{g_{best}}$$

25

Now, as we have understood the working of particle swarm optimization let us have a look at it is pseudo code right. So, for us to execute particle swarm optimization obviously, we need an objective function value or the fitness function value. We need to know the number of decision variables and their corresponding lower and upper bounds right; we need to fix the particle size.

So, these three things come as part of the problem. As and when someone says that they have an optimization problem to solve they need to know the decision variables the lower and upper bound and their fitness function. So, that is part of the problem whereas, this rest five parameters are to be given by the user and so, we need to specify the population size, the termination criteria in this case the number of iterations, the inertia w and acceleration coefficient c 1 and c 2 right.

So, then our first step is to initialize a random population within the bounds and also initialize the velocity right. So, both of this are N p cross D right since our population size is N p the number of decision variable let it be D. So, D will be the length of a either lb or ub right. So, N p cross D. So, population as well as will be a matrix of N p cross D right and since we are generating we will straight away generate within their bounds right. Once we have the population we need to evaluate it is objective function value. Once we have evaluated the objective function value of p we can assign p best and f best.

So, p best will be nothing, but the random population and the fitness of the corresponding p best would be nothing, but the fitness of the objective function value. So, we do not need to go in determine the fitness of p best again right because it is this it will be the same as the fitness function of the population right. And, remember we are not required to determine the fitness of the velocity right, though it is N p cross D we do not require it is fitness right.

So, the next step is to identify the solution with best fitness and assign that solution as g best and f f g best. So, looking at this f of p best we will find what is the least value so, that least value will be nothing, but f of g best and the solution corresponding to that least value will be our g best. So, this is the global best solution and this is the fitness corresponding to that global best solution.

So, the next step is to perform the iterations right. So, for i equal to 1 to T and N. So, whatever is there within this loop will be performed t times right. So, if our each iteration we need to perform the operation for each member. So, we have another for loop for i is equal to 1 to N p, where N p is the number of particles.

So, the next step is to determine the velocity v i of the particle using this equation. So, initial w is a constant; c 1, c 2 have been user defined. So, we need to generate r 1, r 2 those are random numbers between 0 and 1 and their dimension will be 1 cross D. So, we will require D

D D random numbers here and D random numbers here, one for each variable. We know p best i, we know g best. So, we will be able to apply this equation to determine the velocity.

Once the velocity is determined we need to find the new position using this equation right. So, first find the velocity and then find the position right. So, after finding the position we need to bound it if required right. So, if it is violating the upper or lower bound we need to bound it and then determine the objective function value of the particle right.

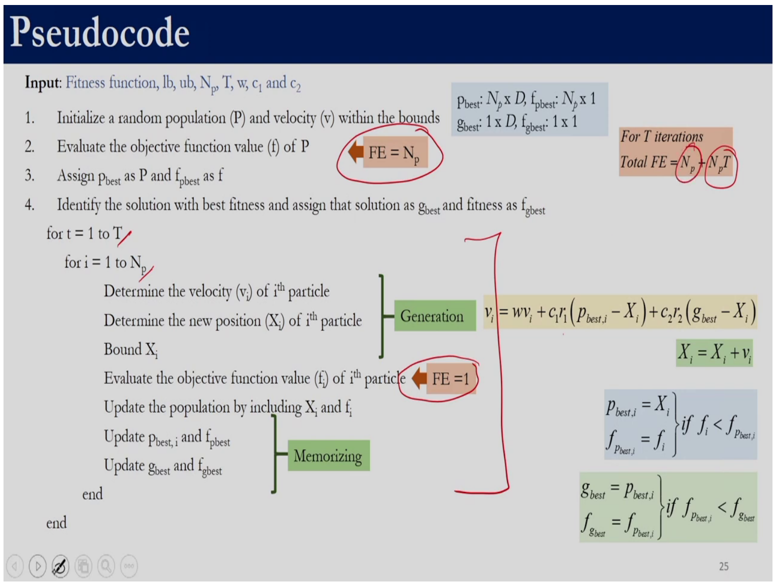
So, once this is determined it will be updated in the population. So, cures the newly generated position will be updated inside the population. So, remember as we have said multiple times there is no greedy selection over here, the position of that particle is updated. We need to update the population by including X i and also replace f i in the vector f.

The next step is to far to check whether the newly generated solution if it is better than the p best right. So, we will be employing this condition that if the newly generated solution if it has a fitness of f i if it is lower than f of p best i, then we will update it. We will update the p best and f best right else we will not update the p best and f best. So, this i corresponds to the particle, it is not corresponding to the iteration; iteration is t.

Similarly, we will check for the g best right if we have been able to update the p best then we actually see if the p best of the current member is actually less than the global best which we have determined so far. If that is the case then we will update it else we will retain the current g best and f of g best right. So, this is the pseudocode of particle swarm optimization as we can see it, it would be fairly simple to implement this code ok.

So, here these steps these three steps correspond to the generating solution right and this is the memorizing, right. We are checking if it is better than the current p best and the current g best if it is better, then we will update else we will not update. So, that is the memorizing portion.

(Refer Slide Time: 47:06)



Pseudocode

Input: Fitness function, lb, ub, N_p, T, w, c_1 and c_2

```
1. Initialize a random population (P) and velocity (v) within the bounds
2. Evaluate the objective function value (f) of P                         FE = $N_p$
3. Assign $p_{best}$ as P and $f_{pbest}$ as f
4. Identify the solution with best fitness and assign that solution as $g_{best}$ and fitness as $f_{gbest}$
   for t = 1 to T
      for i = 1 to $N_p$
         Determine the velocity ($v_i$) of i-th particle          ⎫
         Determine the new position ($X_i$) of i-th particle      ⎬ Generation
         Bound $X_i$                                              ⎭
         Evaluate the objective function value ($f_i$) of i-th particle   FE =1
         Update the population by including $X_i$ and $f_i$
         Update $p_{best,i}$ and $f_{pbest}$                      ⎫ Memorizing
         Update $g_{best}$ and $f_{gbest}$                        ⎭
      end
   end
```

p_{best}: $N_p \times D$, f_{pbest}: $N_p \times 1$

g_{best}: $1 \times D$, f_{gbest}: 1×1

For T iterations

Total FE $= N_p + N_p T$

$$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$$

$$X_i = X_i + v_i$$

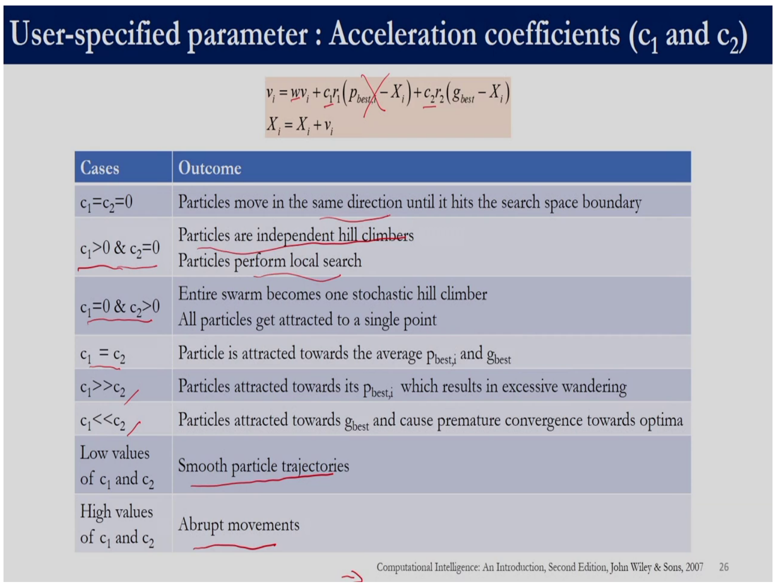
$$\left.\begin{aligned} p_{best,i} &= X_i \\ f_{p_{best,i}} &= f_i \end{aligned}\right\} \text{if } f_i < f_{p_{best,i}}$$

$$\left.\begin{aligned} g_{best} &= p_{best,i} \\ f_{g_{best}} &= f_{p_{best,i}} \end{aligned}\right\} \text{if } f_{p_{best,i}} < f_{g_{best}}$$

25

Now, let us look at the fitness function evaluation right. So, in how many places are we evaluating the fitness function right. So, over here initially we evaluate the fitness function for N p times right, we regenerate N p random population members. So, we need to evaluate their fitness. So, we will be using the fitness function evaluation N p times and we will be using one time for every member right and we have N p members and we will do it for t times. So, for this iteration loop we will be performing it for N p in to t times and we have initial N p valuation; if you see it is different from teaching learning based optimization.

(Refer Slide Time: 47:48)



User-specified parameter : Acceleration coefficients (c_1 and c_2)

$$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$$
$$X_i = X_i + v_i$$

Cases	Outcome
$c_1=c_2=0$	Particles move in the same direction until it hits the search space boundary
$c_1>0$ & $c_2=0$	Particles are independent hill climbers Particles perform local search
$c_1=0$ & $c_2>0$	Entire swarm becomes one stochastic hill climber All particles get attracted to a single point
$c_1 = c_2$	Particle is attracted towards the average $p_{best,i}$ and g_{best}
$c_1>>c_2$	Particles attracted towards its $p_{best,i}$ which results in excessive wandering
$c_1<<c_2$	Particles attracted towards g_{best} and cause premature convergence towards optima
Low values of c_1 and c_2	Smooth particle trajectories
High values of c_1 and c_2	Abrupt movements

Computational Intelligence: An Introduction, Second Edition, John Wiley & Sons, 2007 26

So, now let us see what is the significance of this parameter w, c 1, c 2 right. So, there are various cases. So, if c 1 and c 2 are 0 right so, the typical outcome that we expect is that the particles will move in the same direction. So, here if we see if we eliminate the second and the third term if that is eliminated because c 1 and c 2 is equal to 0, then if we see w into v will be in the same direction it will not change the direction right.

So, if it keeps moving in the same direction then either it will hit the lower bound or the upper bound and then we will do a corner bounding strategy to bring it back to the bounds right that is what will happen if c 1 and c 2 are 0 right. If c 1 is greater than 0 and c 2 is equal to 0, so, in this case we are not relying on the global best; we are only relying on the particle best. So, what would happen is the particles will become independent hill climber. So, the global knowledge do not have, they only have their own personal best and they will get stuck around their own personal best. So, that is they are looking around their own personal best. So, that is more like performing a local search in the in that particular region right.

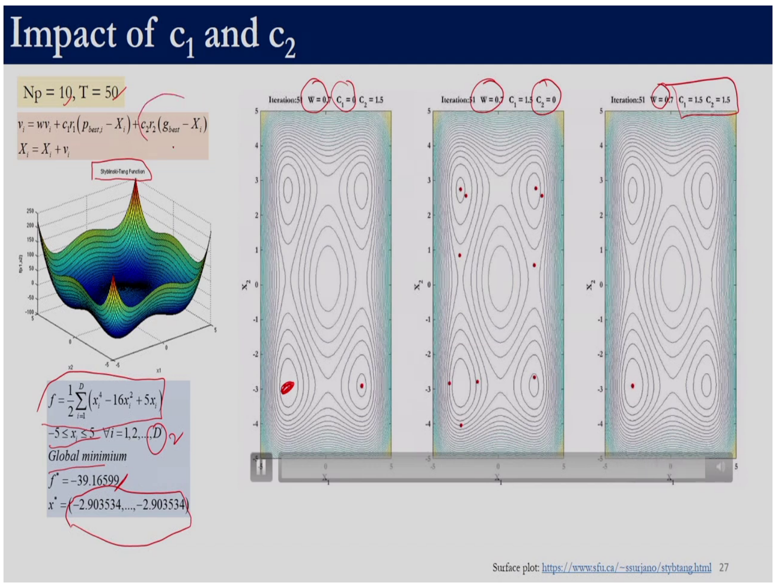
Again, the next case is wherein c 1 is equal to 0 and c 2 is greater than 0 in that case we are not relying on the personal best we are relying only on the global best. So, in this case what happens is that entire swarm becomes one stochastic hill climber and all the particles will get attract to the single point.

Now, the single point may or may not be the global optima right. It is like we are just trusting the solution which is the best so far right. So, all the particles will try to reach that particular solution because we are not considering p best into account right. So, that is what will happen if c 1 is equal to 0 and c 2 is greater than 0.

So, if c 1 is equal to c 2, then the particle is set to be attracted towards the average of p best and g best right. Similarly, if c 1 is much greater than c 2 right then the particles are attracted towards its p best and this will result in excessive wandering right. So, this is similar to the second case though it is not 0, but it is insignificant when compared to c 1. And, in the case c 1 being significantly lower than c 2 the particle will get attracted towards g best and this usually causes premature convergence towards optima right.

For low values of c 1 and c 2 there will be minor perturbations. So, the particle trajectories are smooth; for high values of c 1 c and c 2 will have abrupt moments. So, this slide is primarily based on from this book Computational Negligence: An Introduction from John Wiley and Sons. For further details on this you can look into this book ok.

(Refer Slide Time: 50:34)



Impact of c_1 and c_2

Np = 10, T = 50

$$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$$
$$X_i = X_i + v_i$$

$$f = \frac{1}{2}\sum_{i=1}^{D}(x_i^4 - 16x_i^2 + 5x_i)$$

$-5 \le x_i \le 5 \ \forall i = 1, 2, \ldots, D$ 2

Global minimium

$f^* = -39.16599$

$x^* = (-2.903534, \ldots, -2.903534)$

Surface plot: https://www.sfu.ca/~ssurjano/stybtang.html 27

Styblinski-Tang Function
f(x1,x2)
x1
x2
Iteration:51 W = 0.7 C1 = 0 C2 = 1.5
Iteration:51 W = 0.7 C1 = 1.5 C2 = 0
Iteration:51 W = 0.7 C1 = 1.5 C2 = 1.5
X1
X2

So, now let us actually see the impact of the various parameters. So, this is a function known as tilt bank tank function it is actually a scalable function right, but we will be showing it for two variables right. So, all the variables have the domain of minus 5 to 5 right. In this case in our case for this discussion we will be taking D as 2. So, we have two decision variable and this is the objective function right half summation of i equal to 1 to D x 1 power 4 minus 16x 1 squared plus 5x 1 plus again x 2 to the power 4 minus 16x 2 square plus 5x 2 right.

So, the global minima of this function irrespective of the number of decision variable is located at this point minus 2.903 comma minus 2.903 for a 2 variable problem and the objective function value is minus 39.166 right. So, if we see these are the contour plots right. So, we are considering three cases in one case wherein c 1 is equal to 0 right; this case is c 2 equal to 0 and in this case we have c 1 and c 2 equal to 1.5 and the inertia in all three cases is 0.7 right.

So, these are the three cases we are studying. So, this is the initial population before we begin the iteration. So, with the population size of 10 and for the termination criteria of 50 iterations we will see what will happen right. So, the optima if we see it is located somewhere over here minus 2.9 comma minus 2.9. So, it will be somewhere over here the optima right. So, we will see in on all three conditions what happens.

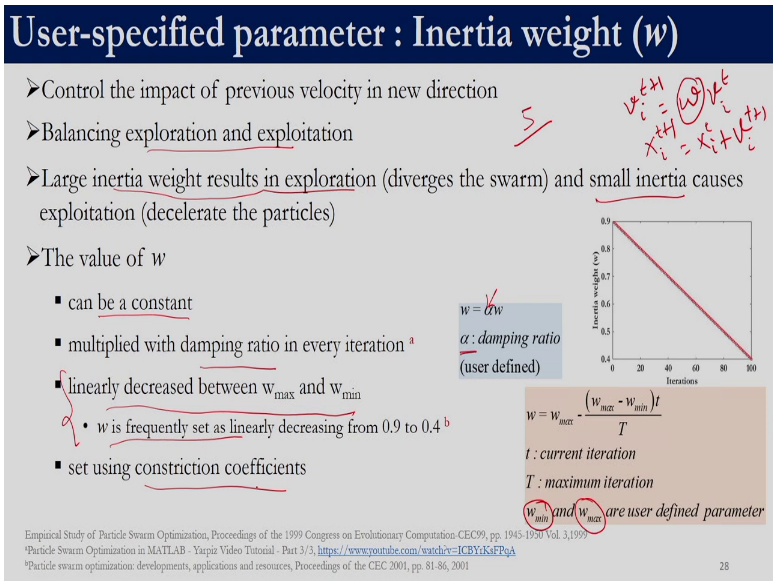
So, so, let us first look at the first figure right. So, the particles are moving right. So, the optima is over here right. So, however, the solutions are converging over here. So, when c 1 is equal to 0 what we are saying is the global best is not considered right. So, here we have given the equation. So, it relies only on the g best right. So, it happens that one particle one particular particle had this as its personal best and that become its g best and all particles converged over here in this in this particular region, despite the fact that the optima is over here right. So, that is what happens when we rely only on the global best and not on the personal best.

So, let us see in the second case what happens in the second case right. So, in the second case if we see over here in this plot right so, we see all the particles are individually moving in their own region they do not have the global knowledge. So, if c 2 is equal to 0; that means, we are ignoring in this term. So, all the particles are relying on their own personal base. So, in that case if we see at the end of 50 iterations none of the particles are over here right and all of the particles where individually moving. So, we can see that once again. So, right.

So, the particles are never converging, they are just following their own peppers. So, right now the particles are not even moving properly ok. So, the third case is wherein c 1 is equal to 1.5, c 2 is equal to 1.5 which means that both of these terms are active. The p best term is also active the g best term is also active if we run this again.

So, if we see the third case all the particles are converging towards the global optima which is located at minus 2.9 comma minus 2.9 right. So, all the particles are indeed reaching that particular global optima or almost close to the global optima the values might be slightly different. So, this shows that the personal best and the g best contribute to reach the optimal solution right.

(Refer Slide Time: 54:36)



User-specified parameter : Inertia weight (w)

- Control the impact of previous velocity in new direction
- Balancing exploration and exploitation
- Large inertia weight results in exploration (diverges the swarm) and small inertia causes exploitation (decelerate the particles)
- The value of w
  - can be a constant
  - multiplied with damping ratio in every iteration [a]
  - linearly decreased between w_{max} and w_{min}
    - w is frequently set as linearly decreasing from 0.9 to 0.4 [b]
  - set using constriction coefficients

5

$v_i^{t+1} = \boxed{w} v_i^t$

$x_i^{t+1} = x_i^t + v_i^{t+1}$

$w = \alpha w$

α : damping ratio

(user defined)

$$w = w_{max} - \frac{(w_{max} - w_{min})t}{T}$$

t : current iteration

T : maximum iteration

$\boxed{w_{min}}$ and $\boxed{w_{max}}$ are user defined parameter

Empirical Study of Particle Swarm Optimization, Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, pp. 1945-1950 Vol. 3,1999

[a]Particle Swarm Optimization in MATLAB - Yarpiz Video Tutorial - Part 3/3, https://www.youtube.com/watch?v=ICBYrKsFPqA

[b]Particle swarm optimization: developments, applications and resources, Proceedings of the CEC 2001, pp. 81-86, 2001

28

So, next we will be looking into this factor inertia weight. It is used to control the impact of previous velocity in new directions right. So, the velocity in this iteration is a function of velocity of previous iteration and this w will help us to control the impact of previous velocity in the new direction because this velocity is again going to be added to our position right. So, that is why we say that it helps us to control the impact of previous velocity in new direction right. So, it is used to balance exploration and exploitation.

So, a large inertia weight will result in exploration right, the swarm will diverge and small values of inertia will cause exploitation that is it will decelerate the particles there would not be huge changes, but there will be just minor changes. So, the value of w can be a constant.

So, very often it is taken to be a constant. It can also be varied iteration to iteration. So, in this case alpha is a damping factor which is to be given by the user.

So, now, in addition to the population size the number of iterations w which is the inertia of weight and the acceleration coefficient we also need to provide the damping ratio right. So, what this will do is that it will decrease w as the iteration progresses it will decrease the w depending upon this damping ratio. So, that is one scheme.

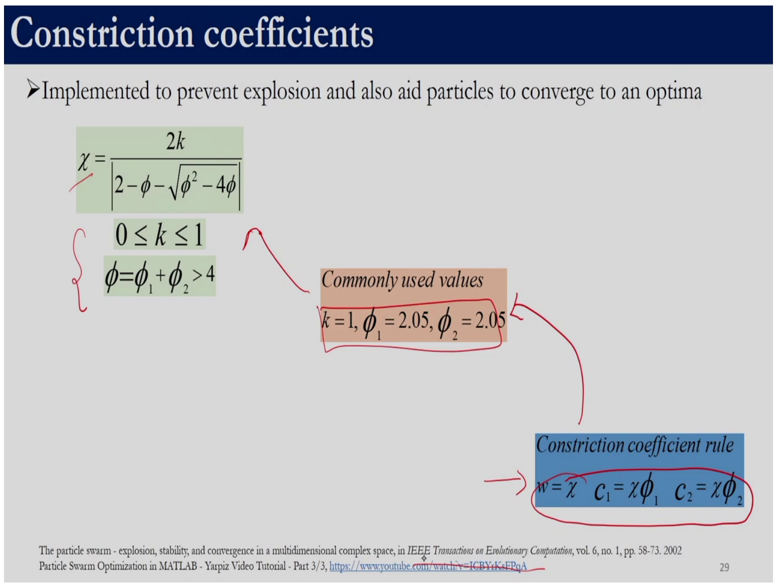
Another scheme is wherein w is linearly decreased between a w max and w min right. So, if we are going to use that is this scheme then in addition to the five parameters that we need to give we will also have to give w min and w max right. With this relation if you see as the number of iterations progress w will decrease right from w max to w min right it will be truncated at w min it will not go below w min right. So, it will be truncated at w min. So, as iteration progress the inertia weight is linearly decreased. So, very often people use this scheme too right.

And, then we have another scheme called as the use of constriction coefficients. So, now, in addition to this five tuning parameters we will also have to decide on how we are going to use w. Are we going to use a constant value or are we going to employ a damping ratio in every iteration or are we going to linearly decrease it, if we are going to linearly decrease it we need to decide on w max w min and if you use a constriction coefficient you will see that there will be few more parameters that we will have to specify right.

So, as you can see we will have to provide a large number of user defined parameters in addition to the two which was common to teaching learning based optimization. And, it is difficult to specify which one of this scheme will work for an arbitrary problem right and for many people who use optimization as a black box tool, it even more becomes difficult for them to tune this parameters that what value of w will best work for them they since they may not even know what is happening inside the algorithm it becomes very difficult for them to tune this right.

So, very often this is why some algorithms which might be very good when the authors propose it is not widely accepted by the user community especially those who are applying optimization in a black box sense.

(Refer Slide Time: 58:07)



Constriction coefficients

- Implemented to prevent explosion and also aid particles to converge to an optima

$$\chi = \frac{2k}{\left|2-\phi-\sqrt{\phi^2-4\phi}\right|}$$

$$0 \le k \le 1$$

$$\phi = \phi_1 + \phi_2 > 4$$

Commonly used values

$k = 1, \phi_1 = 2.05, \phi_2 = 2.05$

Constriction coefficient rule

$w = \chi \quad c_1 = \chi\phi_1 \quad c_2 = \chi\phi_2$

The particle swarm - explosion, stability, and convergence in a multidimensional complex space, in *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73. 2002
Particle Swarm Optimization in MATLAB - Yarpiz Video Tutorial - Part 3/3, https://www.youtube.com/watch?v=ICBYrKsFPqA

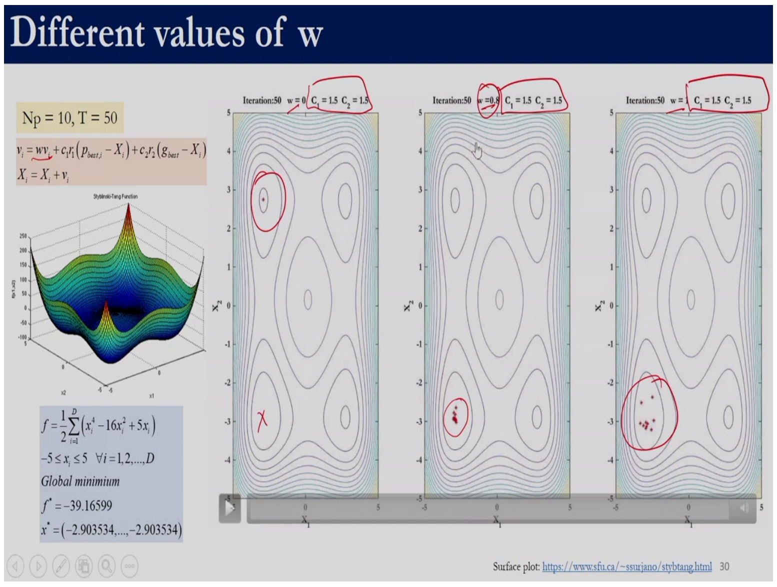
29

So, let us look into the constriction coefficients. So, this constriction coefficients are implemented to prevent explosion and it also aids in converging to an optima. So, constriction coefficient rule specifies that we can set w to be chi c 1 as chi into phi 1 and c 2 as chi into phi 2 right. So, the usual values of k phi 1 and phi 2 are taken as 1, 2.05 and 2.05. So, these values which we take should satisfy these two relations that k should be between 0 and 1 and phi should be greater than 4 and phi is nothing, but summation of phi 1 and phi 2 right and chi can be calculated using this formula.

So, once we calculate chi we can find out the value of w c 1 and c 2 right. So, in this case we can say that we have a scheme to determine the value of w c 1 c 2. So, in, but we need to specify these three values then right and these three values which you specify should satisfy this relation. So, we may not have to directly specify these values, but we will have to specify k phi 1 and phi 2 right. So, it is still the number of tuning parameters is still the same.

So, you can further read on this constriction coefficients in this article which appeared in I triple E transactions on Evolutionary Computation, also you can look at this a YouTube video in which the demonstrate the working of a constriction coefficients.

(Refer Slide Time: 59:39)



Different values of w

Np = 10, T = 50

$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$

$X_i = X_i + v_i$

$f = \frac{1}{2}\sum_{i=1}^{D}(x_i^4 - 16x_i^2 + 5x_i)$

$-5 \le x_i \le 5 \quad \forall i = 1,2,\ldots,D$

Global minimum

$f^* = -39.16599$

$x^* = (-2.903534,\ldots,-2.903534)$

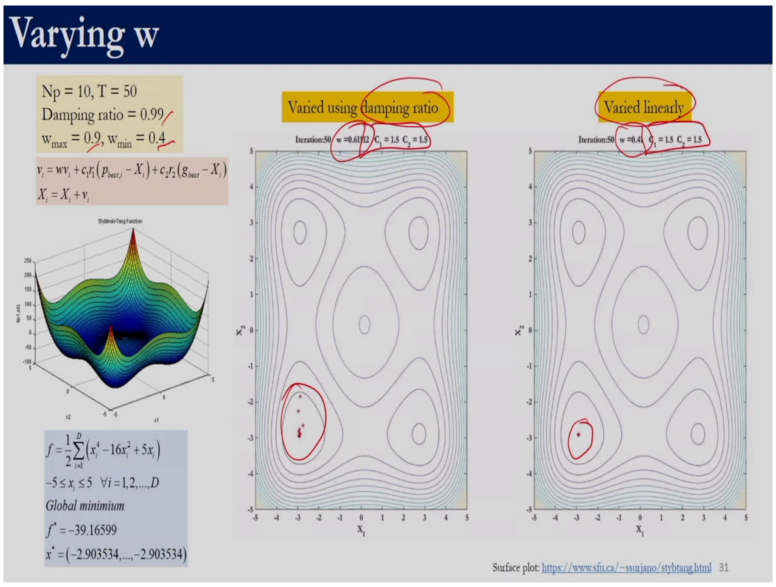
Surface plot: https://www.sfu.ca/~ssurjano/stybtang.html 30

So, now let us study the impact of the inertia weight right. So, here we have three cases for the same problem which we have discussed right. So, c 1 and c 2 are constant for all the three cases it is set at 1.5. The inertia weight is also set to three constant values w is equal to 0, w is equal to 0.8 and w is equal to 1; right now we are not varying w with respect to the iteration it is a it is a constant value right. So, now, if we see in this three contour plots so, remember the optimized somewhere over here minus 2.9 comma minus 2.9.

So, in the first case with w is equal to 0 we see that since the particle do not have this w into v i component they tend to get stuck at a local optima as shown in this first one right. In the second one we can see in this figure we can see when w is equal to 0.8 the solutions are scattered over here right it is close to the global it is close to the globally optimal solution. And, in the third case also if we see it is also closer over here.

So, this basically shows us that w does have an impact on the performance of the algorithm right. So, with 0.8 we see that it is converged it is more closer to the global optima when compared to once. So, w has to be appropriately specified because it has an impact on the performance of the algorithm right.

(Refer Slide Time: 61:01)



Varying w

Np = 10, T = 50

Damping ratio = 0.99

$w_{max} = 0.9$, $w_{min} = 0.4$

$$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$$
$$X_i = X_i + v_i$$

$$f = \frac{1}{2}\sum_{i=1}^{D}(x_i^4 - 16x_i^2 + 5x_i)$$
$$-5 \le x_i \le 5 \quad \forall i = 1,2,...,D$$

Global minimium

$$f^* = -39.16599$$
$$x^* = (-2.903534,...,-2.903534)$$

Varied using damping ratio

Iteration:50 w =0.61112 C_1 = 1.5 C_2 = 1.5

Varied linearly

Iteration:50 w =0.41 C_1 = 1.5 C_2 = 1.5

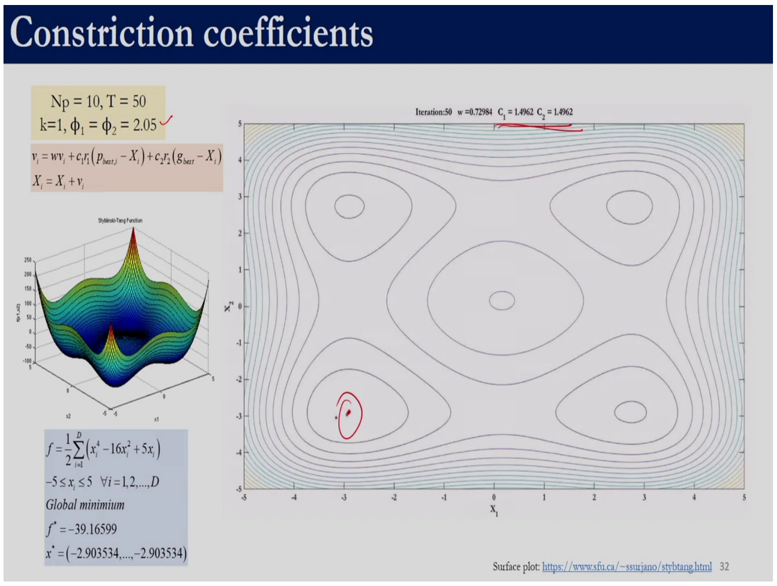
Surface plot: https://www.sfu.ca/~ssurjano/stybtang.html 31

Styblinski-Tang Function
x1
x2
x_1
x_2

So, in this case we show two strategies to vary w right against c 1 and c 2 is still we are deal dealing with the same problem. c 1 and c 2 are fixed at 1.5, 1.5. In this case we will be using a damping ratio in the first case we will be using a damping ratio in the second case will be varying the inertia of weight linearly from 0.9 to 0.4 right and here we have used the damping ratio of 0.99. For this function and this settings if we execute a particle from optimization we can see here that w is decreasing right.

So, over here we can see that for linearly varying w, we have converged to an optima fairly quickly when compared to this damping ratio right. So, we can just once again have a look at it. So, we can see at what iteration is it becoming closer. So, for 23th or 20 23rd iteration things where not even appearing over here when we are using damping ratio right. So, by pretty early we are able to get the optimal solution in a when we vary the inertia weight linearly.

(Refer Slide Time: 62:12)



Constriction coefficients

Np = 10, T = 50

k=1, $\phi_1 = \phi_2 = 2.05$ ✓

$v_i = wv_i + c_1 r_1 (p_{best,i} - X_i) + c_2 r_2 (g_{best} - X_i)$

$X_i = X_i + v_i$

$f = \frac{1}{2}\sum_{i=1}^{D}\left(x_i^4 - 16x_i^2 + 5x_i\right)$

$-5 \le x_i \le 5 \quad \forall i = 1,2,\ldots,D$

Global minimium

$f^* = -39.16599$

$x^* = (-2.903534,\ldots,-2.903534)$

Surface plot: https://www.sfu.ca/~ssurjano/stybtang.html 32

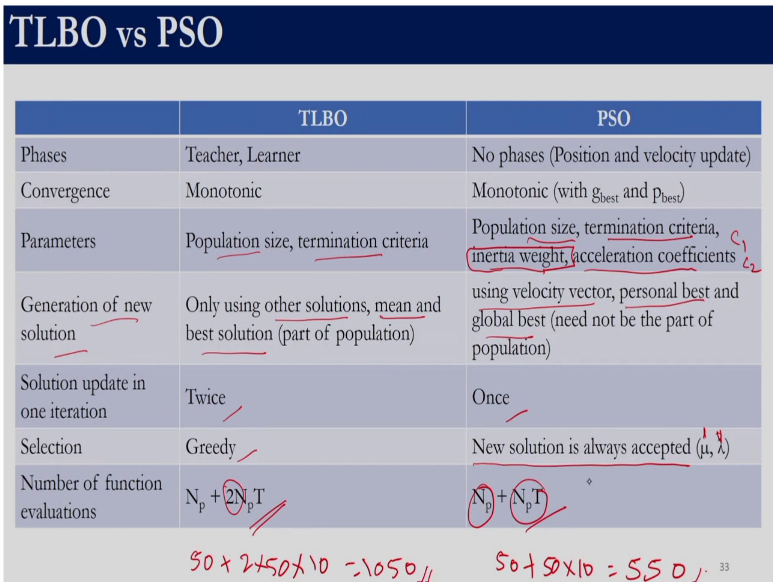
This shows the use of constriction coefficients. So, here these values of c 2 and c 2 have been derived using the suggested value in literature. Phi 1 is equal to phi 2 is equal to 2.05 and k is equal to 1. We know the constriction equations right if I plug this into the chi equation we will be able to get the value of chi and using chi we can set all these three parameters – w, c 1 and c 2 right. So, over here if we see it performs fairly better it will be able to reach the globally optimal solution which is somewhere here right minus 2.92 to minus 2.92.

So, 50 iterations at least some of the solutions have been able to come closer to the global optima. So, that concludes the study of the impact of inertia weight c 1 and c 2 right. So, these are these do have an impact on the performance of the algorithm, but it is not straightforward to set these values we have only showed you whatever is usually considered in literature.

There are further studies on how to tune this values right. So, that is why we started with TLBO because the number of tuning parameters is less. So, that is one of the advantage of TLBO that a user need not spend time in tuning the parameters, there are only two and termination criteria is fairly straight forward – like someone could terminate based on the number of iterations or someone can terminate based on the amount of time that they have to solve that particular optimization problem right.

So, only two parameters are needed in TLBO whereas, here we have five parameters and for tuning c 1, c 2 and w there are schemes available, but then there are multiple schemes, which scheme has to be used for the problem at hand is not clear right. So, though PSO is simpler than TLBO in its implementation, the tuning of these factors can become very difficult for many problems.

(Refer Slide Time: 64:34)



TLBO vs PSO

	TLBO	PSO
Phases	Teacher, Learner	No phases (Position and velocity update)
Convergence	Monotonic	Monotonic (with g_{best} and p_{best})
Parameters	Population size, termination criteria	Population size, termination criteria, inertia weight, acceleration coefficients c_1 c_2
Generation of new solution	Only using other solutions, mean and best solution (part of population)	using velocity vector, personal best and global best (need not be the part of population)
Solution update in one iteration	Twice	Once
Selection	Greedy	New solution is always accepted (μ, λ)
Number of function evaluations	$N_p + 2N_pT$	$N_p + N_pT$

$50 + 2 \times 50 \times 10 = 1050$ $\quad$ $50 + 50 \times 10 = 550$

33

So, before concluding particle swarm optimization let us quickly make a comparison between a teaching learning based optimization and particle swarm optimization.

So, in teaching learning based optimization we had two phases teacher phase and learner phase whereas, in particle swarm optimization as such we did not have any phase, it is just that we had to update the position and the velocity. In TLBO as well as in PSO the convergence is monotonic right; in TLBO it is monotonic because we employ a greedy selection strategy that any member which enters the population should be better than the member which is living the population right. So, that way a monotonic convergence is ensuring teaching learning based optimization.

In particle swarm optimization we do not have a greedy selection to update the population, but we do greedy mechanism for updating g best and p best. Only if a solution is better than g best we will be update g best. So, g best will at any iteration g best will correspond to the best solution obtained so far. So, if we are going to plot a iteration versus a g best value it will be monotonic monotonically converging right.

So, the parameters in TLBO was only two parameters population size and termination criteria right; over here we need to specify population size, termination criteria, inertia weight, acceler and acceleration coefficient. So, there are two acceleration coefficients remember – c 1 and c 2. As we have looked into inertia weight it can be kept a constant or it can be linearly varied or we can use a damping mechanism right.

So, again the choice of that becomes a user defined value right as to which a scheme has to be used to update inertia weight whether we want to update the inertia weight or not and if we want to update the inertia weight in every iteration then what scheme has to be employed. So, that becomes another user specified value right.

So, in TLBO when we are generating new solutions we were using a mean and the best solution in teachers phase right and in partner phase we were using another solution. So, the other solution and best solution were part of the population itself; whereas, in particle swarm optimization we generate a solution using the velocity vector. Again, were will calculating the velocity we use the personal best as well as the global best. These personal best and global best need not be part of the population right because we are not employing a greedy scheme to update the population. The personal best and global best may not even be a part of population.

So, in TLBO we generated a new solution in the teacher phase as well as in the learner phase. So, in one iteration for one population member we explore two new solutions whereas, in particle warm optimization for one iteration for every member we explore only one new solution right. So, that is why we say the change in this these values right N p plus 2N p T that is the number of total functional evaluations in a TLBO whereas, in particle swarm optimization the number of functional evaluation is given by N p plus N p T right. So, since here we do it twice the this two appears in TLBO whereas, here it does not appear.

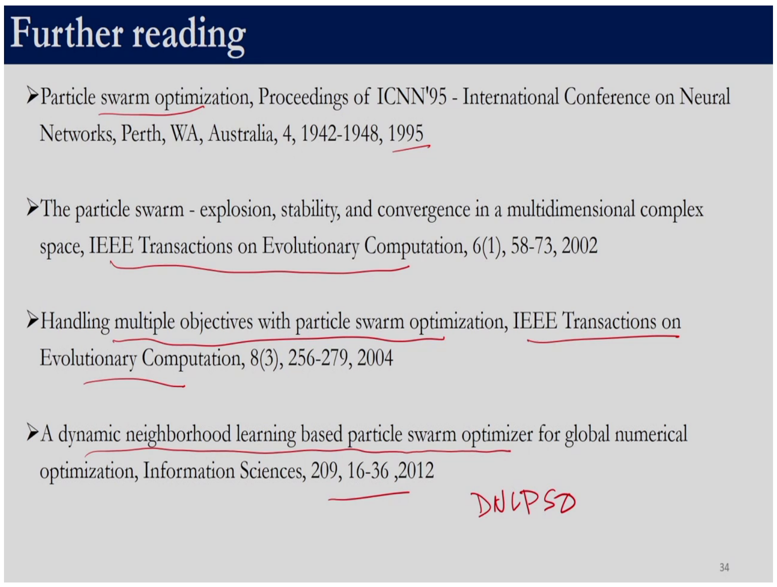
So, the selection mechanism used in TLBO was a greedy selection mechanism right the population was updated using a greedy mechanism whereas, in particle swarm optimization the new solution was always accepted it is more like a mu comma lambda; again, mu being a single solution and lambda being a single solution right. So, in both of them the newly generated solution was always accepted.

So, a number of functional evaluation as we just discussed that we evaluate the solution once in teacher phase and once in learner phase for every members that gives us 2 N p for every iteration. So, for T iteration we get 2N p T and then we have initially N p evaluations ok. This is the expression that we have previously determined; over here for every iteration for very population member we only explore one new solution right. So, this N p T is for that and this N p is initial population member.

So, now, we know two algorithms right. So, you now you can also appreciate the fact that why algorithms should not be compared with same number of iterations. So, performing 50 num 50 iterations of TLBO requires a different number of functional evaluations than 50 iterations of a particle swarm optimization right. So, if you take a population size of let us say 50 right and let us say we are doing 10 iterations right. So, this is 2 in to 50 into 10 in this case right. So, this will be 500, 1000; 1050 valuations right. Over here if we see 50 plus 50 into 10. So, it will be a 550 right.

So, if we want this to be equal right if we want the total number of functional evaluations to be equal then we will have to perform twice the number of generations.

(Refer Slide Time: 69:30)



Further reading

- Particle swarm optimization, Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 4, 1942-1948, 1995
- The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, 6(1), 58-73, 2002
- Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation, 8(3), 256-279, 2004
- A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization, Information Sciences, 209, 16-36 ,2012

DNLPSO

34

So, this is some of the further reading on PSO right you can read the actual paper Particle swarm optimization published in 1995. You can also look into this paper in IEEE Transactions on Evolutionary Computation wherein the study in detail about the explorer explosion stability and convergence. So, that would give you a better idea on how to select the values of w, c 1 and c 2. Remember this is an introductory course on optimization particularly on metaheuristic techniques. So, we are not going into details about how to tune it, but you should know that it has to be tuned appropriately right.

So, how to handle multiple objectives in particle swarm optimization again it appeared in IEEE Transaction on Evolutionary Computation. You can look into it for solving multi objective optimization and this is a more recent paper called as DNLPSO – Dynamic neighborhood learning based particle swarm optimization. So, this is a recent variant you can if you are interested you can look into that and see what are the modifications that have been done by the authors.

So, with that we conclude this session on particle swarm optimization I hope you would have found it interesting. In the next session we will use MATLAB to implement particle swarm optimization, again it is going to be a fairly simple one once you know the pseudocode well we will be able to quickly implement it.

So, thank you.