

Computer Aided Applied Single Objective Optimization
Dr. Prakash Kotecha
Department of Chemical Engineering
Indian Institute of Technology, Guwahati

Lecture – 26

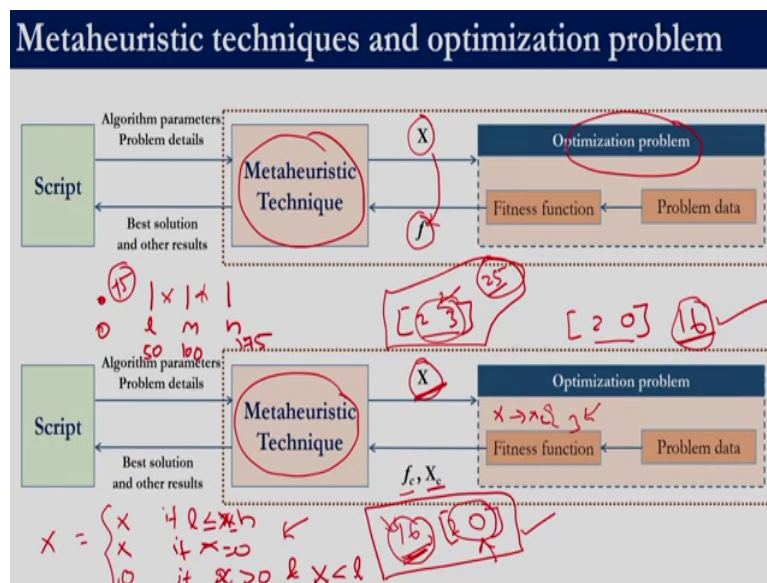
Constraint-Handling using Correction Approach Case Study: Production Planning

Welcome, in the previous session when we were talking about Constraint Handling. So, we only discussed on penalty approach right. So, as and when a metaheuristic techniques provides the solution, we check for the constraints right if that constraints are not satisfied, we assign an appropriate penalty right. So, the penalty can be either hard penalty or it can be determined based on the amount of violation right. So, previously we had solve production planning problem using this approach wherein we had used a hard penalty approach for handling the domain constraint, right.

So, any solution which gave a production which is non zero, but it is less than 1 of the penalized that solution with a constant value of 10^5 right. Whereas, for budget and investment constraint, we determine what is the extent of violation and we assigned a penalty appropriately. Another way to handle constraint is to correct the solution right. So, based on the nature of the problem; whatever solution we receive from the algorithm can be corrected right. Based on the domain specific knowledge right and the corrected solution can be returned back to the metaheuristic technique right.

So, it is a very simple approach that approach can provide significant improvement in the results right. So, that is the approach we will be looking in the session right and we will demonstrate it on the production planning problem. So, that you can realize the benefit of employing your correction approach rather than a penalty approach.

(Refer Slide Time: 01:49)



If you recollect whatever we have been discussing so far. This was the communication between the metaheuristic technique and the optimization problem which we had right. So, the metaheuristic technique will pass the solution X to the optimization problem and the optimization problem when in turn will provide the fitness function of this X right.

If this X is an infeasible solution and if we are solving a minimization problem this f would be very high value right. Whereas, in the correction approach, what we do is; the metaheuristic techniques still passes the solution X right. The optimization problem does not merely determine the fitness function of X right, but checks for the violation.

And if there are any violation in any of those constraint right, if there is an inbuilt mechanism right. So, if we employ some mechanism right which will help to improve the solution x right

then that improved solution is passed back to the algorithm along with the improved solution we also pass the fitness function of the improved solution right.

So, it is not correct to convert X to X_c let us consider we have a two variable problem and the decision variable passed are 2 and 3. And let us say there is a mechanism in the optimization problem which looks into the solution and corrects the solution to 2 0 right. And the fitness function of this 2 0 is let us say 16 right. For some evaluation of the objective function and the other constraints, the fitness function value turns out to be 16 and for this 2 and 3, let us say it is 25 right. So, what we obtain from the algorithm is 2 3, it has a fitness of 25 and it violates some constraint right.

There is some mechanism in this fitness function which corrects the solution right. The solution is corrected to 2 0. And the fitness function of this 2 0 is let us say it is 16. The solution is better than this solution and this betterment is not directly because of the metaheuristic technique, but because of the mechanism which we had employed over here to correct the solution right.

So, in this case to what the optimization problem is supposed to communicate to the metaheuristic technique is not the value 16, but also the solution 2 and 0 right. So, what should happen in this metaheuristic technique is; this 2 and 3 should be replaced with this 2 and 0 and the corresponding fitness function value right

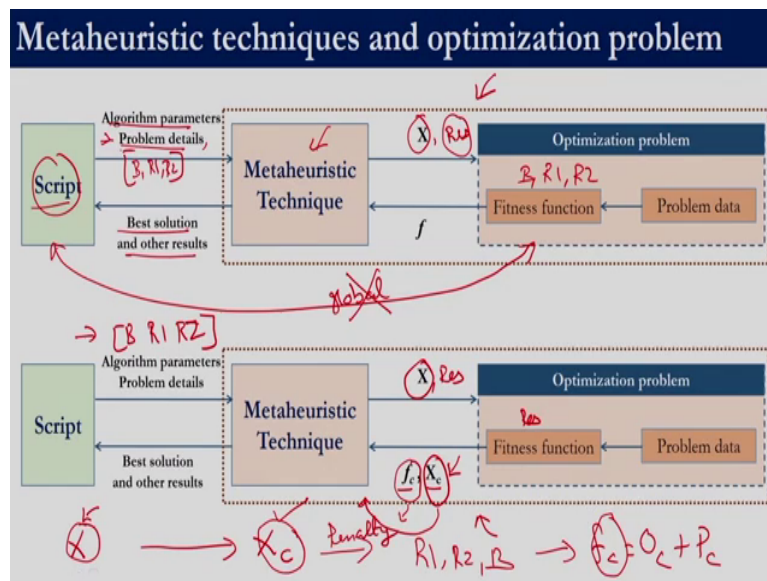
So, this is the correction approach wherein we get a solution from the metaheuristic technique, we employ some mechanism which is problem specific right to correct the solution and we will send back the corrected solution in this case 2 0 along with the fitness function value of the corrected solution right.

So, for example, if we consider teaching learning based optimization; we generate a new solution. That new solution let us say it is 2 3. So, when it is communicated to the optimization problem; the optimization problem corrects it to 2 0 and this 2 0 is sent back to the metaheuristic technique along with the fitness function value of 16.

So, the metaheuristic technique or TLBO in this case, should consider 2 0 as the newly generated solution along with its fitness function. So, it is this solution which has to be now used for greedy selection right. So, this correction approach as you can see it is a very straightforward approach only thing is that we should be able to develop a mechanism for the problem at hand right.

So, it is very problem specific, the correction approach is not part of the metaheuristic technique, but it is a part of the optimization problem. But it is implemented in the fitness function evaluation right. There are couple of things which you need to remember while employing a correction approach right.

(Refer Slide Time: 05:28)



The first thing is that the fitness function should be calculated for the corrected solution right. So, what we get from the algorithm is the set of decision variable let us call it as X right. So,

that is corrected to a new solution let us say X_c right. So, penalty is not supposed to be calculated for this X right. So, penalty is supposed to be calculated after correcting the solution. This is because when we are conveying the fitness function value right. We should convey the fitness function value of the solution which we are passing to the algorithm.

So, the solution that which we are passing to the algorithm is X_c right. So, the fitness function should correspond to this corrected solution right. So, for corrected solution; if you think about it at least for this production planning problem if you see that, it will not have any violation of domain constraint right. So, this will have a better fitness than X right.

So, there is no need to calculate the fitness function of X itself right because, we are anyway correcting the solution. Once we correct the solution; we can find out the penalty for that right. So, in this case there would not be any violation with respect to domain constraint, but there maybe violation with respect to raw material 1, raw material 2 and the budget constraint.

So, the penalty is to be calculated for the corrected solution and then fitness function is to be calculated for the corrected solution. Because, this is the solution which we are conveying back to the metaheuristic technique plus penalty of the corrected solution. So, this will be the fitness which we need to pass through the algorithm. So, here you need to remember that whenever we are employing a correction approach. We are doing something on top of what the algorithm is doing.

So, the algorithm let us say it suggested a solution 1, 2, 3 right. So, for some reason we are correcting that solution in the optimization algorithm let us say that solution 1, 2, 3 becomes 3, 8, 7 right. So, we have done something on top of the algorithms. So, in some sense you can say that we are disturbing the algorithm. For many problems depending upon the correction approach that you employ you may get a better result, but there may be cases where in a correction approach does not enable you to get a better solution.

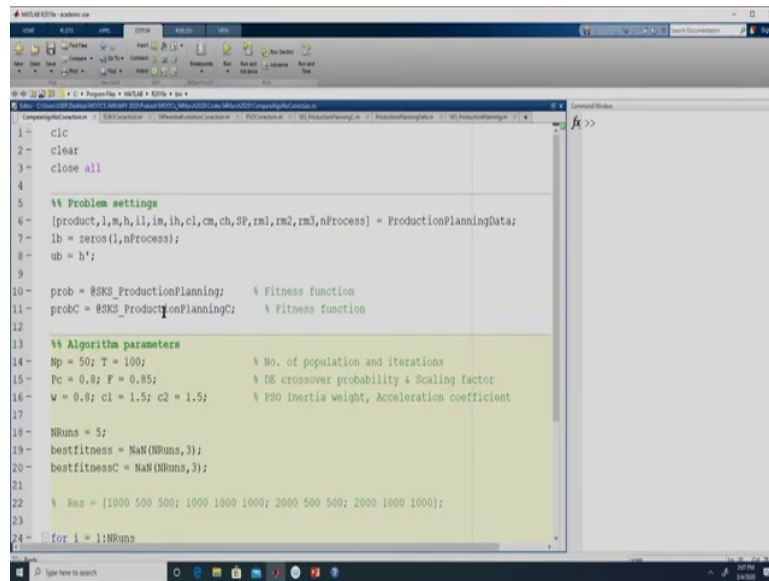
So, in the case of production planning problem. So, this was l this was h right and let us say m . Let us say l is 50, m is 100 and h is let us 175 and this 0 is a feasible solution right. So, if we

get any solution over here right. It does not violate domain constraint. So, you are not adding penalty.

If the solution is here, we again do not add any penalty right or if the solution is 0, we do not add penalty. So, what we were doing previously is that if any solution x was over here which is greater than 0, but less than 50. Let us say it was 45 right. So, what we did for this solution 45 is that; we assigned a penalty for it right. So, instead of assigning penalty what we can do as part of the correction approach for production planning problem is that any solution which is greater than 0, but less than 1 can be converted to 0 right. So, this is some mechanism that we are employing right.

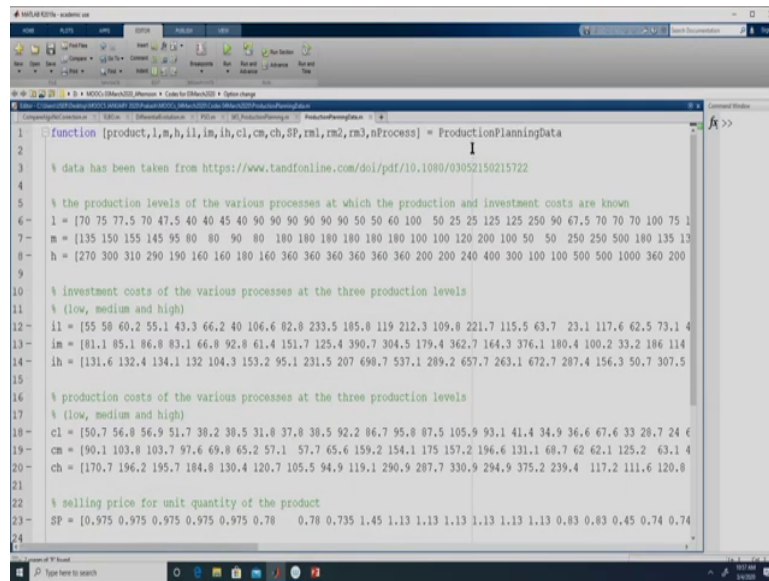
So, solution X remains X right. If this X is greater than or equal to 1 and less than or equal to h , remains X if X is 0 right. And what we are saying is; we will convert it to 0. The algorithm gave us some value, but we will convert it to 0 if this X is greater than 0 and it is less than 1 right. So, this is the correction approach that we are employing right. So, we will correct the X and we will no longer assign the penalty. Because, we have converted the solution in such a way that it no longer violates the domain constraint. So, we will no longer penalize it right.

(Refer Slide Time: 09:15)



```
1 - clc
2 - clear
3 - close all
4
5 %% Problem settings
6 - [product,l,m,h,il,lm,lb,c1,cm,ch,SP,rm1,rm2,rm3,nprocess] = ProductionPlanningData;
7 - lb = zeros(1,nprocess);
8 - ub = h';
9
10 - prob = @SKS_ProductionPlanning; % Fitness function
11 - probC = @SKS_ProductionPlanningC; % Fitness function
12
13 %% Algorithm parameters
14 - Np = 50; T = 100; % No. of population and iterations
15 - Pc = 0.8; F = 0.85; % DE crossover probability & Scaling factor
16 - w = 0.8; c1 = 1.5; c2 = 1.5; % PBO Inertia weight, Acceleration coefficient
17
18 - NRuns = 5;
19 - bestfitness = NaN(NRuns,3);
20 - bestfitnessC = NaN(NRuns,3);
21
22 % Res = [1000 500 500; 1000 1000 1000; 2000 500 500; 2000 1000 1000];
23
24 - for i = 1:NRuns
```

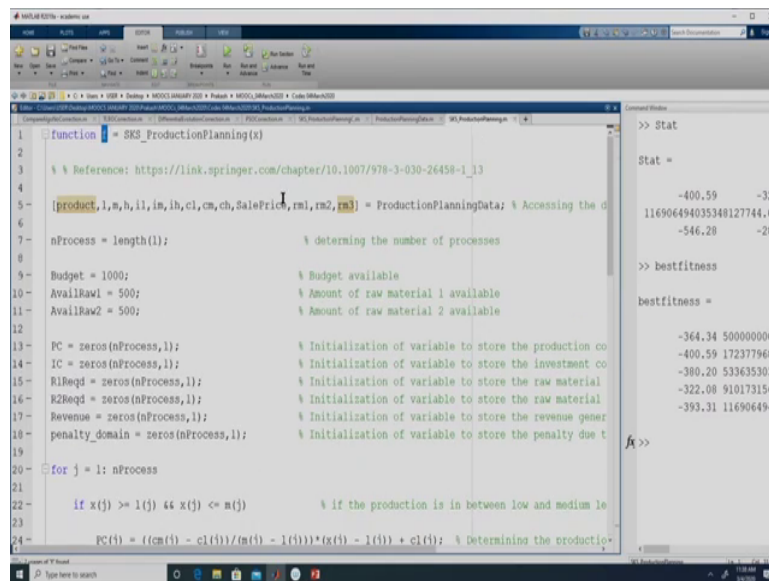
(Refer Slide Time: 09:16)



```
1 function [product,l,m,h,il,im,ih,cl,cm,ch,SP,rm1,rm2,rm3,nProcess] = ProductionPlanningData
2
3 % data has been taken from https://www.tandfonline.com/doi/pdf/10.1080/030521502150215722
4
5 % the production levels of the various processes at which the production and investment costs are known
6- l = [70 75 77.5 70 47.5 40 40 45 40 90 90 90 90 90 50 50 60 100 50 25 25 125 125 250 90 67.5 70 70 100 75 1
7- m = [135 150 155 145 95 80 80 90 80 180 180 180 180 180 100 100 120 200 100 50 250 250 500 180 135 13
8- h = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 200 200 240 400 300 100 100 500 500 1000 360 200
9
10 % investment costs of the various processes at the three production levels
11 % (low, medium and high)
12- il = [55 50 60.2 55.1 43.3 66.2 40 106.6 82.8 233.5 185.8 119 212.3 109.8 221.7 115.5 63.7 23.1 117.6 62.5 73.1 4
13- im = [81.1 85.1 86.8 83.1 66.8 92.8 61.4 151.7 125.4 390.7 304.5 179.4 362.7 164.3 376.1 180.4 100.2 33.2 186 114
14- ih = [131.6 132.4 134.1 132 104.3 153.2 95.1 231.5 207 698.7 537.1 289.2 657.7 263.1 672.7 287.4 156.3 50.7 307.5
15
16 % production costs of the various processes at the three production levels
17 % (low, medium and high)
18- cl = [50.7 56.0 56.9 51.7 39.2 38.5 31.0 37.0 30.5 92.2 86.7 95.0 87.5 105.9 93.1 41.4 34.9 36.6 67.6 33 28.7 24 6
19- cm = [90.1 103.8 103.7 97.6 69.8 65.2 57.1 57.7 65.6 159.2 154.1 175 157.2 196.6 131.1 68.7 62 62.1 125.2 63.1 4
20- ch = [170.7 196.2 195.7 184.8 130.4 120.7 105.5 94.9 119.1 290.9 287.7 330.9 294.9 375.2 239.4 117.2 111.6 120.8
21
22 % selling price for unit quantity of the product
23- SP = [0.975 0.975 0.975 0.975 0.975 0.78 0.78 0.735 1.45 1.13 1.13 1.13 1.13 1.13 1.13 0.83 0.83 0.45 0.74 0.74
24
```

So, to better understand let us consider an example let us look into the production planning data.

(Refer Slide Time: 09:21)



```
function ok = SKS_ProductionPlanning(x)
% Reference: https://link.springer.com/chapter/10.1007/978-3-030-26458-1_13
[product, l, m, h, il, im, ih, cl, cm, ch, SalePrice, rml, rm2, rm3] = ProductionPlanningData; % Accessing the data
nProcess = length(l); % determining the number of processes
Budget = 1000; % Budget available
AvalRaw1 = 500; % Amount of raw material 1 available
AvalRaw2 = 500; % Amount of raw material 2 available
PC = zeros(nProcess,1); % Initialization of variable to store the production cost
IC = zeros(nProcess,1); % Initialization of variable to store the investment cost
R1Reqd = zeros(nProcess,1); % Initialization of variable to store the raw material 1 required
R2Reqd = zeros(nProcess,1); % Initialization of variable to store the raw material 2 required
Revenue = zeros(nProcess,1); % Initialization of variable to store the revenue generated
penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penalty due to domain violation
for j = 1: nProcess
    if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low and medium level
        PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determines the production cost
    else
        PC(j) = 1000000000; % Penalty due to domain violation
    end
end
% Calculating the fitness function
f = sum(PC) - sum(SalePrice.*product); % Calculating the fitness function
% Calculating the raw material requirements
R1Reqd = sum(product.*rml);
R2Reqd = sum(product.*rm2);
% Calculating the investment cost
IC = sum(product.*h);
% Calculating the revenue
Revenue = sum(product.*SalePrice);
% Calculating the total fitness function
f = f + IC - Revenue;
% Calculating the best fitness
bestfitness = f;
% Calculating the corrected solution
ok = x;
end
```

Now, that we have designed a correction approach. We can implement that. So, this fitness function will receive the solution X, but it will return not only f, but it will also return the corrected solution ok. So, this is done. We are no longer going to have any variable which is going to violate the domain constraint right, because we are going to correct it right. So, if X lies between l and m; we do not need to do anything, because no penalty was calculated.

(Refer Slide Time: 09:42)

```

22 if x(j) >= l(j) && x(j) <= m(j) % if the production is in between
23
24 PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)))*(x(j) - l(j)) + cl(j); % Determir
25 IC(j) = ((lm(j) - ll(j))/(m(j) - l(j)))*(x(j) - l(j)) + ll(j); % Determir
26 R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1
27 R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2
28 Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generat
29
30
31 elseif x(j) > m(j) && x(j) <= h(j) % if the production is in between
32
33 PC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determir
34 IC(j) = ((lh(j) - lm(j))/(h(j) - m(j)))*(x(j) - m(j)) + lm(j); % Determir
35 R1Reqd(j) = x(j)*rm1(j); % Determining the raw material 1
36 R2Reqd(j) = x(j)*rm2(j); % Determining the raw material 2
37 Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generat
38
39
40 elseif 0 < x(j) && x(j) < l(j) % if the production is greater th
41
42 % penalty domain(j) = 10^5; % Assigning penalty for violatir
43 x(j) = 0;
44 end
45

```

```

>> Stat
Stat =
    -400.59    -322.08    -372.10
116906494035348127744.00  9101731564140766
    -546.28    -285.63    -420.77

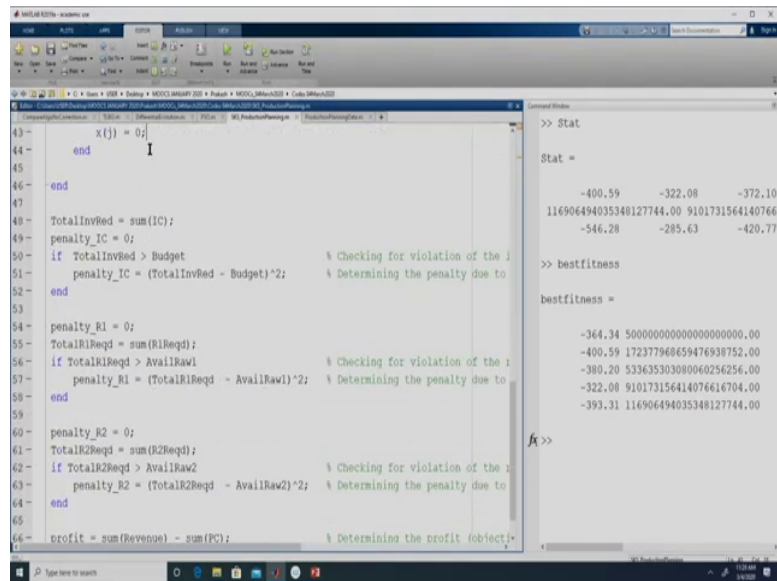
>> bestfitness
bestfitness =
    -364.34  5000000000000000000.00
    -400.59  172377968659476938752.00
    -380.20  533635303080060256256.00
    -322.08  910173156414076616704.00
    -393.31  116906494035348127744.00

```

If X lies between m and h again, we do not do anything because it was already in a domain right. So, previously what we had done is; if a solution is greater than 0 right and less than l; we were assigning a penalty. So, right now what we will do is; we will not assign a penalty, but we will merely correct the solution right. So, we will say X of j is equal to 0 right. So, no matter where the solution lies right. If it is greater than 0, but less than l, it does not satisfy the constraint. Since it does not satisfy the constraint previously we were assigning a penalty.

Now, we will not assign a penalty, but we will merely correct the variable. So, the corrected value of the variable is 0 because, 0 we know for sure does not violate the domain constraint right. So, we assign it to 0 as discussed earlier right you can also assign it the value of l. As discussed earlier you could even choose it to assign it a value of l right or you can generate a random number and see if the random number is greater than 0.5; you can assign it to 0, if it is less than 0.5; you can assign it to l right. So, here we are the demonstrating one of the scheme, you can try out the other two schemes right.

(Refer Slide Time: 10:58)



```
43- x(j) = 0;
44- end
45-
46- end
47-
48- TotalInvReqd = sum(IC);
49- penalty_IC = 0;
50- if TotalInvReqd > Budget % Checking for violation of the
51-     penalty_IC = (TotalInvReqd - Budget)^2; % Determining the penalty due to
52- end
53-
54- penalty_R1 = 0;
55- TotalR1Reqd = sum(R1Reqd);
56- if TotalR1Reqd > AvailRaw1 % Checking for violation of the
57-     penalty_R1 = (TotalR1Reqd - AvailRaw1)^2; % Determining the penalty due to
58- end
59-
60- penalty_R2 = 0;
61- TotalR2Reqd = sum(R2Reqd);
62- if TotalR2Reqd > AvailRaw2 % Checking for violation of the
63-     penalty_R2 = (TotalR2Reqd - AvailRaw2)^2; % Determining the penalty due to
64- end
65-
66- profit = sum(Revenue) - sum(PC); % Determining the profit (objecti
```

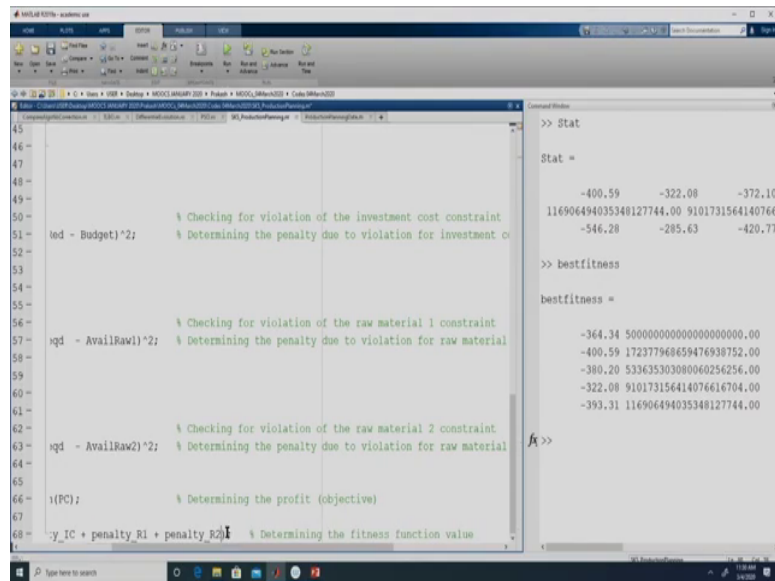
```
>> Stat
Stat =
    -400.59    -322.08    -372.10
 116906494035348127744.00  9101731564140766.
    -546.28    -285.63    -420.77

>> bestfitness
bestfitness =
    -364.34  5000000000000000000.00
    -400.59  17237796859476938752.00
    -380.20  533635303080060256256.00
    -322.08  910173156414076616704.00
    -393.31  116906494035348127744.00

fx>>
```

So, this x of j is equal to 0, we do not need to change anything with respect to calculating the investment cost or the raw material requirement of 1 and 2 right.

(Refer Slide Time: 11:11)

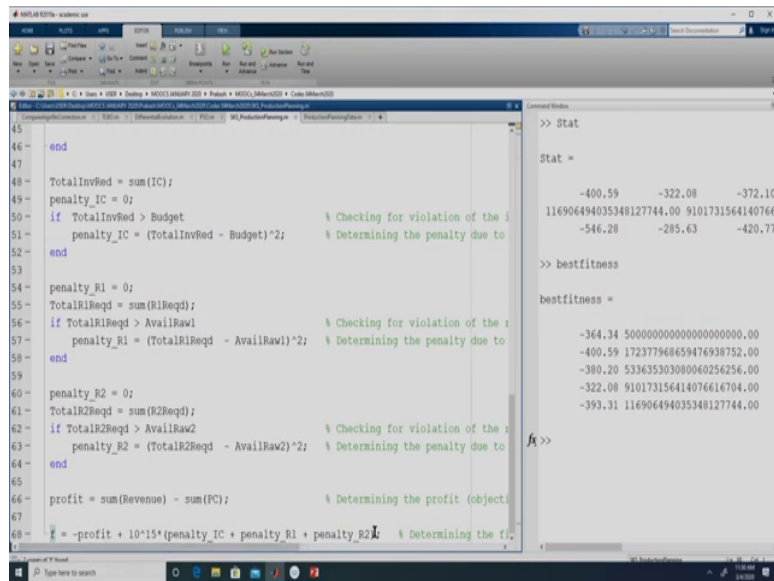


```
45  
46  
47  
48  
49  
50 % Checking for violation of the investment cost constraint  
51 led - Budget)^2; % Determining the penalty due to violation for investment c  
52  
53  
54  
55  
56 % Checking for violation of the raw material 1 constraint  
57 sqd - AvailRaw1)^2; % Determining the penalty due to violation for raw material  
58  
59  
60  
61  
62 % Checking for violation of the raw material 2 constraint  
63 sqd - AvailRaw2)^2; % Determining the penalty due to violation for raw material  
64  
65  
66 f(PC); % Determining the profit (objective)  
67  
68 y_IC + penalty_R1 + penalty_R2; % Determining the fitness function value  
69  
70
```

```
>> Stat  
Stat =  
-400.59 -322.08 -372.10  
116906494035340127744.00 9101731564140766.  
-546.28 -285.63 -420.77  
  
>> bestfitness  
bestfitness =  
-364.34 5000000000000000000.00  
-400.59 17237796859476938752.00  
-380.20 533635303080060256256.00  
-322.08 910173156414076616704.00  
-393.31 116906494035340127744.00  
f >>
```

So, everything will remain same except that this term is no longer needed. Because, there is no penalty with respect to violation of the domain constraint, because if any solution violated the domain constraint, we corrected it right.

(Refer Slide Time: 11:23)



```
45 -
46 - end
47 -
48 - TotalInvReqd = sum(IC);
49 - penalty_IC = 0;
50 - if TotalInvReqd > Budget % Checking for violation of the i
51 -     penalty_IC = (TotalInvReqd - Budget)^2; % Determining the penalty due to
52 - end
53 -
54 - penalty_R1 = 0;
55 - TotalR1Reqd = sum(R1Reqd);
56 - if TotalR1Reqd > AvailRaw1 % Checking for violation of the i
57 -     penalty_R1 = (TotalR1Reqd - AvailRaw1)^2; % Determining the penalty due to
58 - end
59 -
60 - penalty_R2 = 0;
61 - TotalR2Reqd = sum(R2Reqd);
62 - if TotalR2Reqd > AvailRaw2 % Checking for violation of the i
63 -     penalty_R2 = (TotalR2Reqd - AvailRaw2)^2; % Determining the penalty due to
64 - end
65 -
66 - profit = sum(Revenue) - sum(PC); % Determining the profit (objecti
67 -
68 - f = -profit + 10^15*(penalty_IC + penalty_R1 + penalty_R2); % Determining the fi
```

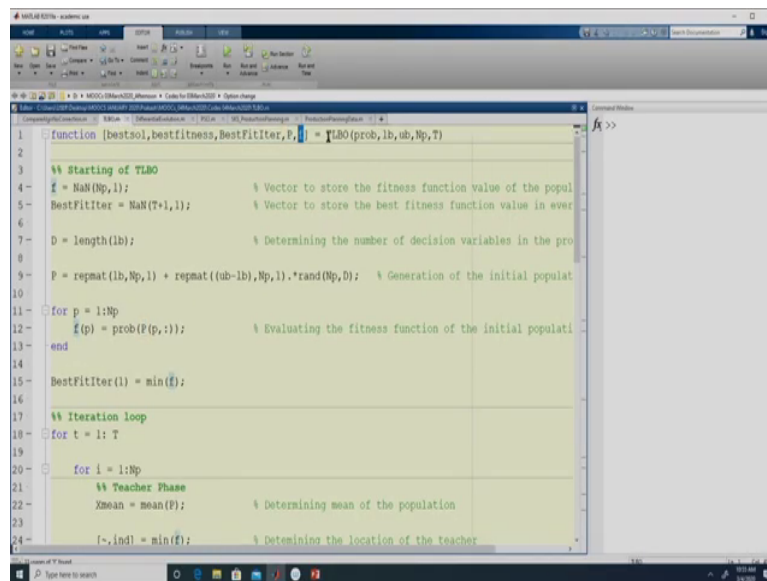
```
>> Stat
Stat =
    -400.59    -322.08    -372.10
116906494035348127744.00  9101731564140766.
    -546.28    -285.63    -420.77

>> bestfitness
bestfitness =
    -364.34  5000000000000000000.00
    -400.59  172377968659476938752.00
    -380.20  533635303080060256256.00
    -322.08  910173156414076616704.00
    -393.31  116906494035348127744.00

f >>
```

So, over here we have removed that thing right. So, now, our objective function is compatible.

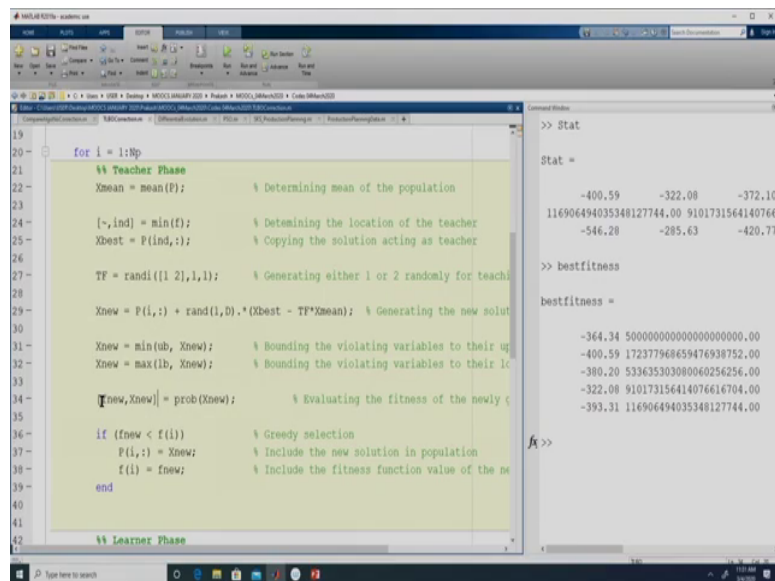
(Refer Slide Time: 11:29)



```
1 function [bestsol,bestfitness,BestFitter,P] = TLBO(prob,lb,ub,Np,T)
2
3 %% Starting of TLBO
4 f = NaN(Np,1); % Vector to store the fitness function value of the popul
5 BestFitter = NaN(T+1,1); % Vector to store the best fitness function value in ever
6
7 D = length(lb); % Determining the number of decision variables in the pro
8
9 P = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D); % Generation of the initial populat
10
11 for p = 1:Np
12     f(p) = prob(P(p,:)); % Evaluating the fitness function of the initial populati
13 end
14
15 BestFitter(1) = min(f);
16
17 %% Iteration loop
18 for t = 1:T
19     for i = 1:Np
20         %% Teacher Phase
21         Xmean = mean(f); % Determining mean of the population
22         [-,ind1] = min(f); % Determining the location of the teacher
```

But if you look at our algorithms our algorithms were designed in such a way that it can receive only the fitness function right. So, what we will do is; we will make a copy of this. So, that we can compare both the approaches right. So, let me say this is TLBO correction right and over here, we will receive not only the fitness function, but we will also replace the member which we sent right. So, the member which we are sending is the p th row of the population right. So, that we are replacing with the solution provided by the problem right.

(Refer Slide Time: 12:09)



```
19
20 for i = 1:Np
21     %% Teacher Phase
22     Xmean = mean(F); % Determining mean of the population
23
24     [~,ind] = min(f); % Determining the location of the teacher
25     Xbest = P(ind,:); % Copying the solution acting as teacher
26
27     TF = randi([1 2],1,1); % Generating either 1 or 2 randomly for teacher
28
29     Xnew = P(i,:) + rand(1,D).*(Xbest - TF*Xmean); % Generating the new solution
30
31     Xnew = min(ub, Xnew); % Bounding the violating variables to their upper bound
32     Xnew = max(lb, Xnew); % Bounding the violating variables to their lower bound
33
34     [fnew, Xnew] = prob(Xnew); % Evaluating the fitness of the newly generated solution
35
36     if (fnew < f(i)) % Greedy selection
37         P(i,:) = Xnew; % Include the new solution in population
38         f(i) = fnew; % Include the fitness function value of the new solution
39     end
40
41
42     %% Learner Phase
```

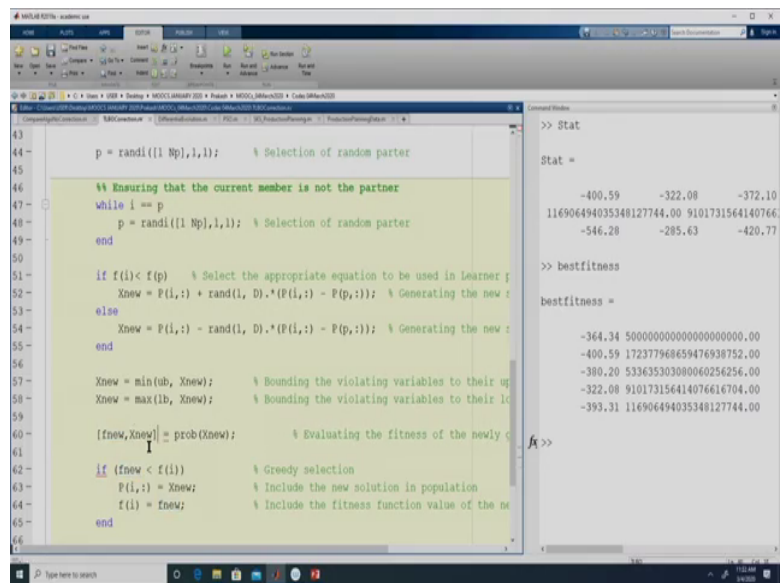
```
>> Stat
Stat =
    -400.59    -322.08    -372.10
116906494035348127744.00  9101731564140766.
    -546.20    -285.63    -420.77

>> bestfitness
bestfitness =
    -364.34  5000000000000000000.00
    -400.59  172377968659476938752.00
    -380.20  533635303080060256256.00
    -322.08  910173156414076616704.00
    -393.31  116906494035348127744.00

f>>
```

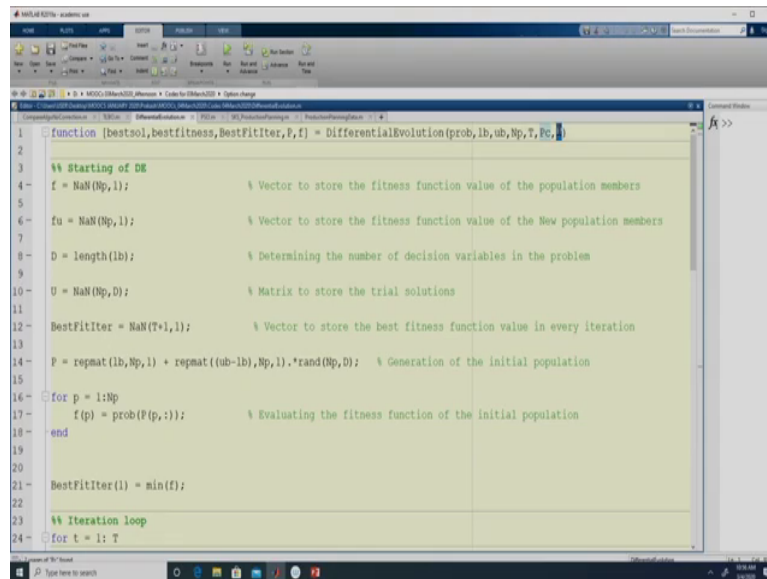
So, this line has become compatible now right. In two other places we would be calling the objective function; one is after generating the new solution in teacher phase right, so over here. So, here we will say f_{new} comma X_{new} right. So, we now receive not only the fitness function, but also the corrected solution right. So, if the solution is already within the domain nothing is going to change right, we will not be modifying the correction approach does not help, but if the solution is violating the domain constraint it is corrected in such a way that the solution no longer violates the domain constraint right.

(Refer Slide Time: 12:41)



```
43
44 p = randi([1 Np],1,1); % Selection of random parter
45
46 %% Ensuring that the current member is not the partner
47 while i == p
48     p = randi([1 Np],1,1); % Selection of random parter
49 end
50
51 if f(i) < f(p) % Select the appropriate equation to be used in Learner p
52     Xnew = P(i,:) + rand(1, D) * (P(i,:) - P(p,:)); % Generating the new c
53 else
54     Xnew = P(i,:) - rand(1, D) * (P(i,:) - P(p,:)); % Generating the new c
55 end
56
57 Xnew = min(lb, Xnew); % Bounding the violating variables to their up
58 Xnew = max(lb, Xnew); % Bounding the violating variables to their lo
59
60 [fnew, Xnew] = prob(Xnew); % Evaluating the fitness of the newly c
61
62 if (fnew < f(i)) % Greedy selection
63     P(i,:) = Xnew; % Include the new solution in population
64     f(i) = fnew; % Include the fitness function value of the ne
65 end
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
265
```

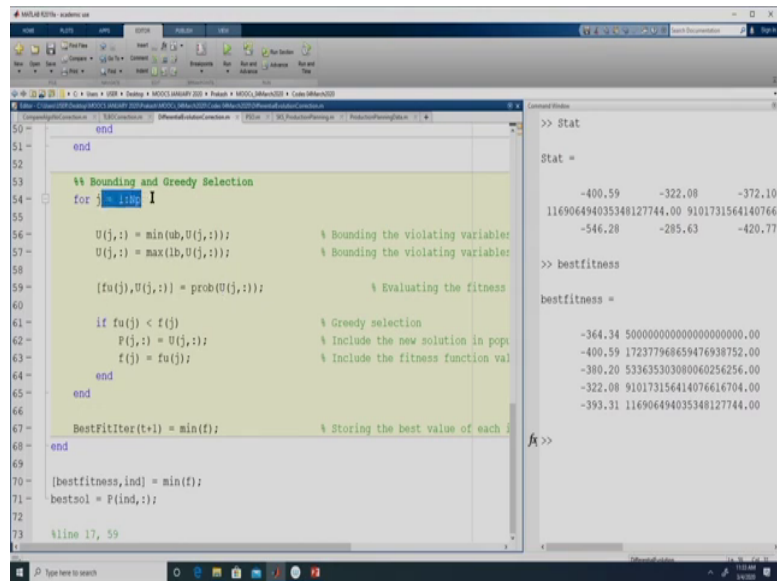

(Refer Slide Time: 12:57)



```
1 function [bestsol,bestfitness,BestFitter,F,f] = DifferentialEvolution(prob,lb,ub,Np,T,Pc,Fc);
2
3 %% Starting of DE
4 f = NaN(Np,1); % Vector to store the fitness function value of the population members
5
6 fu = NaN(Np,1); % Vector to store the fitness function value of the New population members
7
8 D = length(lb); % Determining the number of decision variables in the problem
9
10 U = NaN(Np,D); % Matrix to store the trial solutions
11
12 BestFitter = NaN(T+1,1); % Vector to store the best fitness function value in every iteration
13
14 P = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D); % Generation of the initial population
15
16 for p = 1:Np
17     f(p) = prob(P(p,:)); % Evaluating the fitness function of the initial population
18 end
19
20
21 BestFitter(1) = min(f);
22
23 %% Iteration loop
24 for t = 1: T
```

Similarly, let us convert for differential evolution right. So, this is differential evolution, it employs correction approach right. So, remember we are able to do this because we have complete control over the algorithm as well as the problem right. So, since the algorithms we had coded it ourself since we know this algorithms; we can employ this correction approach, we can easily employ this correction approach. So, over here we are again receiving the p th member of the population right and then we will be evaluating the objective function over here right.

(Refer Slide Time: 13:29)



```
50 - end
51 - end
52 -
53 - %% Bounding and Greedy Selection
54 - for i = 1:np
55 -
56 -     U(j,:) = min(ub,U(j,:)); % Bounding the violating variables
57 -     U(j,:) = max(lb,U(j,:)); % Bounding the violating variables
58 -
59 -     [fu(j),U(j,:)] = prob(U(j,:)); % Evaluating the fitness
60 -
61 -     if fu(j) < f(j) % Greedy selection
62 -         P(j,:) = U(j,:); % Include the new solution in population
63 -         f(j) = fu(j); % Include the fitness function value
64 -     end
65 - end
66 -
67 - BestFitIter(t+1) = min(f); % Storing the best value of each iteration
68 - end
69 -
70 - [bestfitness,ind] = min(f);
71 - bestsol = P(ind,:);
72 -
73 - %line 17, 59
```

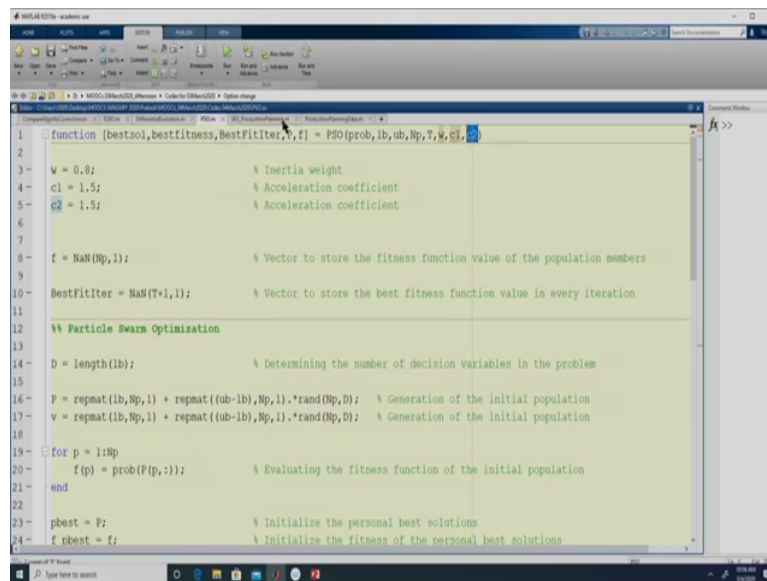
```
>> Stat
Stat =
-400.59 -322.08 -372.10
116906494035348127744.00 9101731564140766.00
-546.20 -285.63 -420.77

>> bestfitness
bestfitness =
-364.34 50000000000000000000.00
-400.59 172377968659476938752.00
-380.20 533635303080060256256.00
-322.08 910173156414076616704.00
-393.31 116906494035348127744.00

f >>
```

So, the solution which we are sending is the j th row of the variable u right. So, what we will get back is another solution. So, the newly obtain solution is plugged in the same position right. So, this we have done for differential evolution. So, differential evolution; we call the objective function only twice once is for the initial population and then once when we have calculated all the solutions right. So, this is inside a loop, so that way we are calling it np times.

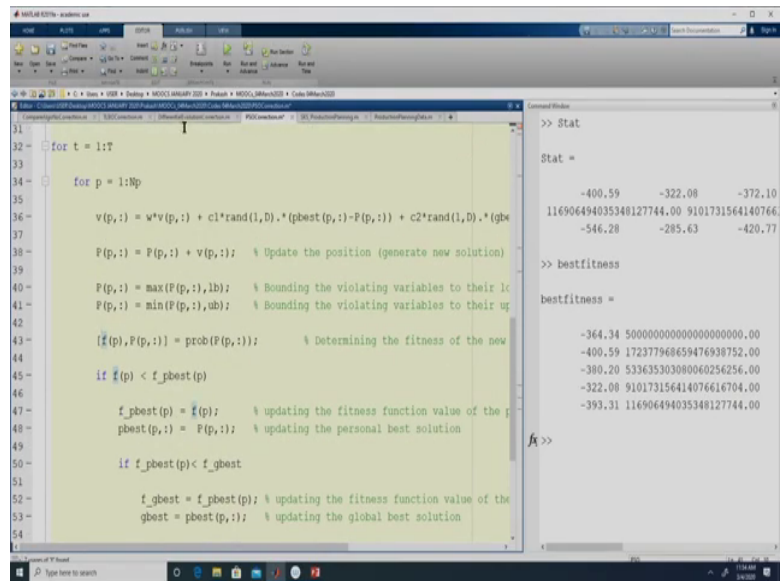
(Refer Slide Time: 14:01)



```
1 function [bestsol,bestfitness,BestFitter,T,f] = PSO(prob,lb,ub,Np,T,W,cl)
2
3 w = 0.8; % Inertia weight
4 cl = 1.5; % Acceleration coefficient
5 c2 = 1.5; % Acceleration coefficient
6
7
8 f = NaN(Np,1); % Vector to store the fitness function value of the population members
9
10 BestFitter = NaN(T+1,1); % Vector to store the best fitness function value in every iteration
11
12 %% Particle Swarm Optimization
13
14 D = length(lb); % Determining the number of decision variables in the problem
15
16 P = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D); % Generation of the initial population
17 v = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D); % Generation of the initial population
18
19 for p = 1:Np
20     f(p) = prob(P(p,:)); % Evaluating the fitness function of the initial population
21 end
22
23 pbest = P; % Initialize the personal best solutions
24 f_rbest = f; % Initialize the fitness of the personal best solutions
```

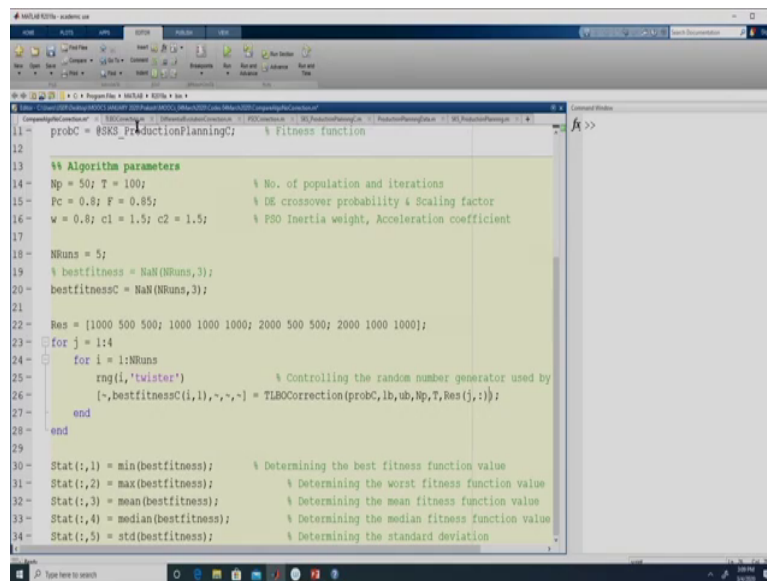
For particle swarm optimization also, we will be calling it only twice. One is during the initial phase. So, we need to make a copy of this right. So, now here we will have the new solution similarly, we will be using the objective function over here right.

(Refer Slide Time: 14:24)



```
31
32 for t = 1:T
33
34     for p = 1:Np
35
36         v(p,:) = w*v(p,:) + c1*rand(1,D).*(pbest(p,:)-P(p,:)) + c2*rand(1,D).*(gbe
37
38         P(p,:) = P(p,:) + v(p,:); % Update the position (generate new solution)
39
40         P(p,:) = max(P(p,:),lb); % Bounding the violating variables to their lo
41         P(p,:) = min(P(p,:),ub); % Bounding the violating variables to their up
42
43         [f(p),P(p,:)] = prob(P(p,:)); % Determining the fitness of the new
44
45         if f(p) < f_pbest(p)
46
47             f_pbest(p) = f(p); % updating the fitness function value of the
48             pbest(p,:) = P(p,:); % updating the personal best solution
49
50             if f_pbest(p) < f_gbest
51
52                 f_gbest = f_pbest(p); % updating the fitness function value of the
53                 gbest = pbest(p,:); % updating the global best solution
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
```

(Refer Slide Time: 14:50)



The image shows a MATLAB script for the TLBO (Teaching-Learning Based Optimization) algorithm. The script is written in a MATLAB editor window. The code defines the fitness function, algorithm parameters, and the main optimization loop. The fitness function is defined as `probC = @SKS_ProductionPlanningC;`. The algorithm parameters are: `Np = 50; T = 100;` (No. of population and iterations), `Pc = 0.8; F = 0.85;` (DE crossover probability & Scaling factor), `w = 0.8; c1 = 1.5; c2 = 1.5;` (PSO Inertia weight, Acceleration coefficient), and `NRuns = 5;`. The script then initializes the best fitness values: `bestfitness = NaN(NRuns,3);` and `bestfitnessC = NaN(NRuns,3);`. The results are stored in a matrix: `Res = [1000 500 500; 1000 1000 1000; 2000 500 500; 2000 1000 1000];`. The main loop is a `for` loop over `i = 1:NRuns`. Inside the loop, a random number generator is used: `rng(i,'twister')`. The TLBO correction is applied: `[~,bestfitnessC(i,1),~,~,~] = TLBOCorrection(probC,lb,ub,Np,T,Res(j,:));`. The loop ends with `end`. Finally, the statistics are calculated: `Stat(:,1) = min(bestfitness);` (Determining the best fitness function value), `Stat(:,2) = max(bestfitness);` (Determining the worst fitness function value), `Stat(:,3) = mean(bestfitness);` (Determining the mean fitness function value), `Stat(:,4) = median(bestfitness);` (Determining the median fitness function value), and `Stat(:,5) = std(bestfitness);` (Determining the standard deviation).

(Refer Slide Time: 14:55)

```

24 = for i = 1:NRuns
25 =     rng(i,'twister')           % Controlling the random number generator used by rand, randi
26 =     [~,bestfitness(i,1),-,,-] = TLBO(prob,lb,ub,Np,T);
27
28 =     rng(i,'twister')           % Controlling the random number generator used by rand, randi
29 =     [~,bestfitness(i,2),-,,-] = DifferentialEvolution(prob,lb,ub,Np,2*T,Pc,F);
30
31 =     rng(i,'twister')           % Controlling the random number generator used by rand, randi
32 =     [~,bestfitness(i,3),-,,-] = PSO(prob,lb,ub,Np,2*T,w,c1,c2);
33
34 =     rng(i,'twister')           % Controlling the random number generator used by rand, randi
35 =     [~,bestfitnessC(i,1),-,,-] = TLBOCorrection(prob,lb,ub,Np,T);
36
37 =     rng(i,'twister')           % Controlling the random number generator used by rand, randi
38 =     [~,bestfitnessC(i,2),-,,-] = DifferentialEvolutionCorrection(prob,lb,ub,Np,2*T,Pc,F);
39
40 =     rng(i,'twister')           % Controlling the random number generator used by rand, randi
41 =     [~,bestfitnessC(i,3),-,,-] = PSOCorrection(prob,lb,ub,Np,2*T,w,c1,c2);
42
43 = end
44
45 = Stat(:,1) = min(bestfitness);   % Determining the best fitness function value
46 = Stat(:,2) = max(bestfitness);  % Determining the worst fitness function value
47 = Stat(:,3) = mean(bestfitness); % Determining the mean fitness function value

```

```

>> Stat
Stat =
    -400.59   -322
116906494035348127744.00
    -546.28   -285

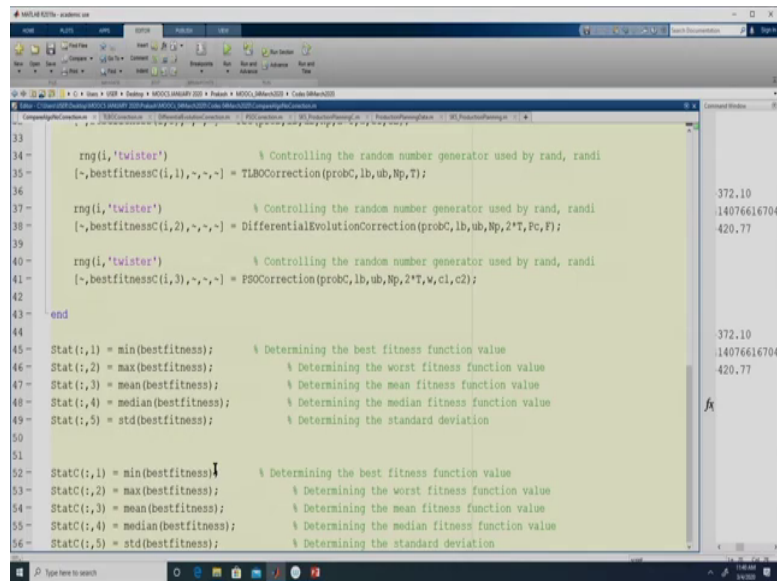
>> bestfitness
bestfitness =
    -364.34 5000000000
    -400.59 1723779686
    -380.20 5336353030
    -322.08 9101731564
    -393.31 1169064940

```

So, that we can compare the correction approach and the original approach right. So, here we need to call this correction. So, this correction has to be given over here right and again over here right.

So, let me use another variable for this thing, best fitness c right and similarly, let me define this over here right. So, this best fitness will give us the performance of the algorithms without correction whereas, best fitness c would give us the performance of the algorithm with correction. We can also repeat these lines right.

(Refer Slide Time: 15:30)



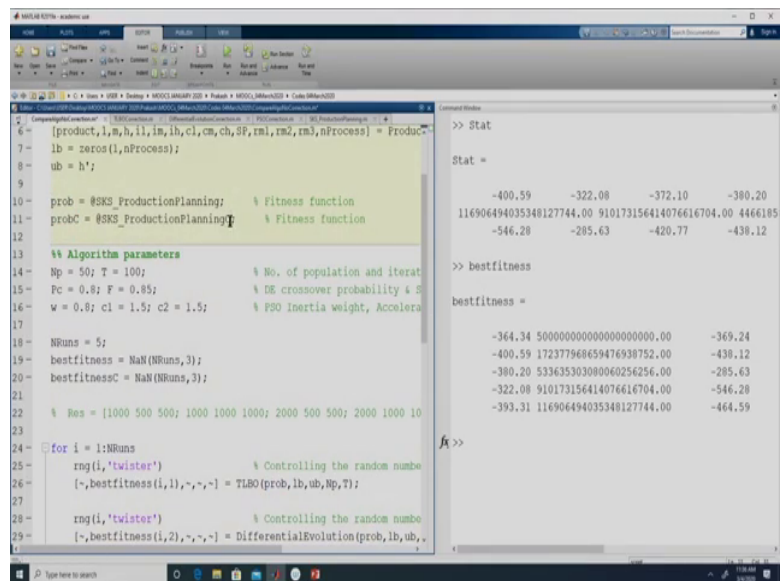
```
33
34 ~ rng(i,'twister')           % Controlling the random number generator used by rand, randi
35 ~ [-,bestfitness(i,1),-, -,] = TLBOCorrection(probc,lb,ub,Np,T);
36
37 ~ rng(i,'twister')           % Controlling the random number generator used by rand, randi
38 ~ [-,bestfitness(i,2),-, -,] = DifferentialEvolutionCorrection(probc,lb,ub,Np,2*T,Fc,F);
39
40 ~ rng(i,'twister')           % Controlling the random number generator used by rand, randi
41 ~ [-,bestfitness(i,3),-, -,] = PSOCorrection(probc,lb,ub,Np,2*T,W,c1,c2);
42
43 ~ end
44
45 ~ Stat(:,1) = min(bestfitness); % Determining the best fitness function value
46 ~ Stat(:,2) = max(bestfitness); % Determining the worst fitness function value
47 ~ Stat(:,3) = mean(bestfitness); % Determining the mean fitness function value
48 ~ Stat(:,4) = median(bestfitness); % Determining the median fitness function value
49 ~ Stat(:,5) = std(bestfitness); % Determining the standard deviation
50
51
52 ~ StatC(:,1) = min(bestfitness); % Determining the best fitness function value
53 ~ StatC(:,2) = max(bestfitness); % Determining the worst fitness function value
54 ~ StatC(:,3) = mean(bestfitness); % Determining the mean fitness function value
55 ~ StatC(:,4) = median(bestfitness); % Determining the median fitness function value
56 ~ StatC(:,5) = std(bestfitness); % Determining the standard deviation
```

372.10
14076616704
420.77

372.10
14076616704
420.77

So, let me use this variable. So, again this is a bit crude way of doing it. If you have sufficient coding skills, you can do this same thing in a much better way right. So, now, what we are doing is we are solving the same production planning problem with three different algorithms. TLBO differential evolution and particle swarm optimization with correction and without correction right. So, these are without correction. So, the first three are without correction the last three are with correction right.

(Refer Slide Time: 16:06)



```
6 = [product, l, m, h, ll, lb, cl, cm, ch, SF, rm1, rm2, rm3, nProcess] = Product...
7 - lb = zeros(1, nProcess);
8 - ub = h';
9
10 - prob = @SKS_ProductionPlanning; % Fitness function
11 - probC = @SKS_ProductionPlanningT; % Fitness function
12
13 %% Algorithm parameters
14 - Np = 50; T = 100; % No. of population and iterations
15 - Pc = 0.9; F = 0.85; % DE crossover probability & scaling factor
16 - w = 0.8; c1 = 1.5; c2 = 1.5; % PSO Inertia weight, Acceleration coefficients
17
18 - NRuns = 5;
19 - bestfitness = NaN(NRuns, 3);
20 - bestfitnessC = NaN(NRuns, 3);
21
22 % Res = [1000 500 500; 1000 1000 1000; 2000 500 500; 2000 1000 1000]
23
24 - for i = 1:NRuns
25 -     rng(i, 'twister') % Controlling the random number generator
26 -     [-, bestfitness(i, 1), -, -, -] = TLBO(prob, lb, ub, Np, T);
27
28 -     rng(i, 'twister') % Controlling the random number generator
29 -     [-, bestfitness(i, 2), -, -, -] = DifferentialEvolution(prob, lb, ub, Np, T);
30
31 - end
```

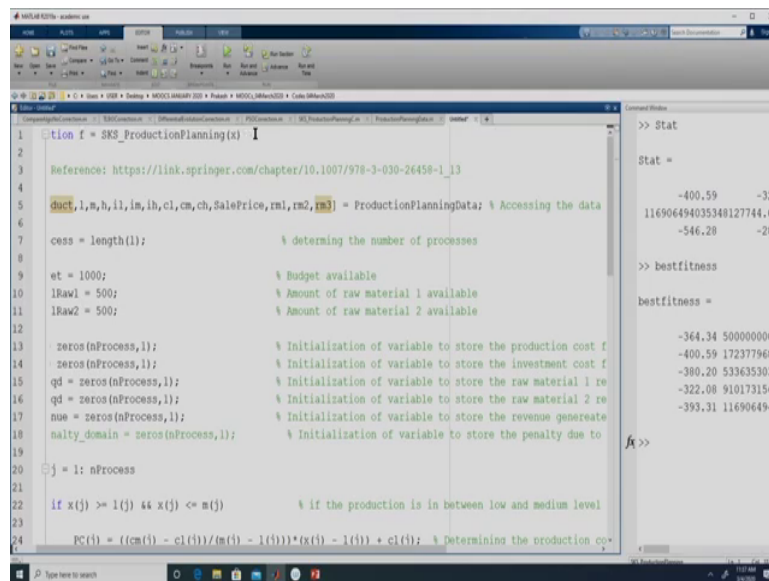
```
>> Stat
Stat =
    -400.59    -322.08    -372.10    -380.20
116906494035348127744.00  910173156414076616704.00  4466185
    -546.28    -285.63    -420.77    -438.12

>> bestfitness
bestfitness =
    -364.34  5000000000000000000.00    -369.24
    -400.59  172377968659476938752.00    -438.12
    -380.20  533635303080060256256.00    -285.63
    -322.08  910173156414076616704.00    -546.28
    -393.31  116906494035348127744.00    -464.59

fx >>
```

So, the fitness function also we need to have it in two forms right. So, one is prob another one is prob C. So, one employs correction another one does not employ correction right. So, for those functions which do not employ corrections, we do not need to make any changes wherever we are employing correction we need to pass the appropriate file right. And then this production planning we need to have us production planning c. So, this is production planning correction approach right.

(Refer Slide Time: 16:47)



```
function f = SKS_ProductionPlanning(x)
%
Reference: https://link.springer.com/chapter/10.1007/978-3-030-26458-1_13
%
[duct, l, m, h, li, im, ih, cl, cm, ch, SalePrice, rm1, rm2, rm3] = ProductionPlanningData; % Accessing the data
%
cess = length(l); % determining the number of processes
%
et = 1000; % Budget available
lRaw1 = 500; % Amount of raw material 1 available
lRaw2 = 500; % Amount of raw material 2 available
%
zeros(nProcess, 1); % Initialization of variable to store the production cost f
zeros(nProcess, 1); % Initialization of variable to store the investment cost f
qd = zeros(nProcess, 1); % Initialization of variable to store the raw material 1 re
qd = zeros(nProcess, 1); % Initialization of variable to store the raw material 2 re
nue = zeros(nProcess, 1); % Initialization of variable to store the revenue generate
nalty_domain = zeros(nProcess, 1); % Initialization of variable to store the penalty due to
%
j = 1: nProcess
%
if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low and medium level
%
PC(i) = ((cm(i) - cl(i))/(m(i) - l(i))) * x(i) - l(i)) + cl(i); % Determining the production co
```

>> Stat

Stat =	
	-400.59 -322
	116906494035348127744.00
	-546.28 -285

>> bestfitness

bestfitness =	
	-364.34 5000000000
	-400.59 1723779686
	-380.20 5336353030
	-322.08 9101731564
	-393.31 1169064940

So, for without correction we will have this only f is written right without correction right and so this line would be active right.

(Refer Slide Time: 17:01)

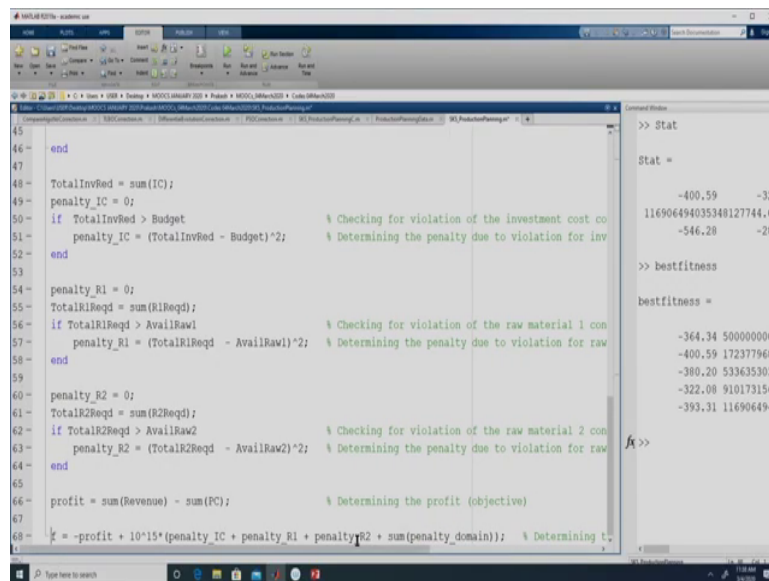
```
31 - elseif x(j) > m(j) || x(j) <= b(j) % If the production is in between medium and high level
32 -
33 - FC(j) = ((ch(j) - cm(j))/(h(j) - m(j)))*(x(j) - m(j)) + cm(j); % Determining the production co
34 - IC(j) = ((lh(j) - lm(j))/(h(j) - m(j)))*(x(j) - m(j)) + lm(j); % Determining the investment co
35 - R1Req(j) = x(j)*rml(j); % Determining the raw material 1 required for process
36 - R2Req(j) = x(j)*rm2(j); % Determining the raw material 2 required for process
37 - Revenue(j) = SalePrice(j)*x(j); % Determining the revenue generated by selling the pr
38 -
39 -
40 - elseif 0 < x(j) && x(j) < l(j) % if the production is greater than zero but less than
41 -
42 - penalty_domain(j) = 10^5; % Assigning penalty for violating the domain hole const
43 - x(j) = 0;
44 - end
45 -
46 - I
47 -
48 - lInvReq = sum(IC);
49 - lty_IC = 0;
50 - TotalInvReq > Budget % Checking for violation of the investment cost constr
51 - penalty_IC = (TotalInvReq - Budget)^2; % Determining the penalty due to violation for investm
52 -
53 -
54 - ltv_R1 = 0;
```

```
>> Stat
Stat =
    -400.59    -322
 116906494035348127744.00
    -546.28    -285

>> bestfitness
bestfitness =
    -364.34 5000000000
    -400.59 1723779686
    -380.20 5336353030
    -322.08 9101731564
    -393.31 1169064940

f >>
```

(Refer Slide Time: 17:04)



```
45
46 - end
47
48 - TotalInvReqd = sum(IC);
49 - penalty_IC = 0;
50 - if TotalInvReqd > Budget % Checking for violation of the investment cost co
51 -     penalty_IC = (TotalInvReqd - Budget)*2; % Determining the penalty due to violation for inv
52 - end
53
54 - penalty_R1 = 0;
55 - TotalR1Reqd = sum(R1Reqd);
56 - if TotalR1Reqd > AvailRaw1 % Checking for violation of the raw material 1 con
57 -     penalty_R1 = (TotalR1Reqd - AvailRaw1)*2; % Determining the penalty due to violation for raw
58 - end
59
60 - penalty_R2 = 0;
61 - TotalR2Reqd = sum(R2Reqd);
62 - if TotalR2Reqd > AvailRaw2 % Checking for violation of the raw material 2 con
63 -     penalty_R2 = (TotalR2Reqd - AvailRaw2)*2; % Determining the penalty due to violation for raw
64 - end
65
66 - profit = sum(Revenue) - sum(PC); % Determining the profit (objective)
67
68 - f = -profit + 10^15*(penalty_IC + penalty_R1 + penalty_R2 + sum(penalty_domain)); % Determining t
69
```

```
>> Stat
Stat =
-400.59 -322
116906494035348127744.00
-546.28 -285

>> bestfitness
bestfitness =
-364.34 5000000000
-400.59 1723779686
-380.20 5336353030
-322.08 9101731564
-393.31 1169064940
f >>
```

So, this is to be converted plus sum of penalty underscore domain and we can also uncomment this line right. So, what we have now is two objective function file; one employs the correction approach right f comma X right. The other one returns only the fitness function value f right. Over here in this SKS underscore production planning c, we employ the correction approach right. No penalty is being assigned and the variable value is converted to 0 and that set of modified decision variable is passed back to the algorithm and we do not have any penalty over here right.

Whereas this production planning is the same file which we have been using in the previous session right. So, we assign a penalty over here and we add the penalty right. So, this is the violation with respect to every decision variable. We sum it up and add it to total fitness function right.

(Refer Slide Time: 18:15)

```

33
34-   rng(i,'twister'
35-   [~,bestfitnessC
36
37-   rng(i,'twister'
38-   [~,bestfitnessC
39-   rng(i,'twister'
40-   [~,bestfitnessC
41-   end
42
43- end
44
45- Stat(:,1) = min(best
46- Stat(:,2) = max(best
47- Stat(:,3) = mean(best
48- Stat(:,4) = median(best
49- Stat(:,5) = std(best
50
51
52- StatC(:,1) = min(best
53- StatC(:,2) = max(best
54- StatC(:,3) = mean(best
55- StatC(:,4) = median(best
56- StatC(:,5) = std(best

```

```

>> Stat
Stat =
-400.59    -322.08    -372.10    -380.20    31.18
116906494035348127744.00  910173156414076616704.00  446618584437792374784.00  5000000000000000000.00  3198
-546.28    -285.63    -420.77    -438.12    98.61

>> StatC
StatC =
-699.38    -655.02    -680.26    -684.56    20.45
-690.82    -656.39    -673.27    -670.60    13.65
-710.04    -557.39    -605.51    -579.70    25.11

```

So, now if we execute this, let us see if we are able to execute it without any error. So, now, let us look at both the variables stat and stat C right. So, remember stat and statC are identical in nature right. Only difference is for stat we did not employ a correction approach for statC we employed a correction approach.

So, this has to be bestfitness C. So, now, if we execute this, so we can have a look at the variable stat right statC remember the nature of stat and statC is the same right. So, each row indicates an algorithm, the first column indicates the best value determined by the algorithm, the second column indicates the worst value, third one mean the fourth one median and the fifth one is standard deviation.

The first row is for TLBO, second row is for differential evolution and the third row is for particle swarm optimization. So, stat is the same set of results which we would have obtained

previous right. Whereas, statC, the columns are the same the rows are the same only in this case we employed a correction approach right.

Now, if you see the best value reported by TLBO right was minus 400.59 without correction approach whereas, with correction approach it gives a value minus 699.38 which is significantly better right. Even the best solution was not feasible right. Here now it is able to find a solution which is minus 690.82, particle swarm optimization was previously able to determine a solution of minus 546.28. Now, is able to determine a solution of minus 710 right. So, if you think about it, we employed a very simple procedure right, but that simple procedure has helped us to get significantly better result right.

So, even if the standard deviation also has considerably dropped right. So, from 31.18 and 98.61, it has dropped to 20.45 and 62.19 right. That is the benefit of correction approach right. So, again we need to remember that the correction approach is very problem specific right. So, for this problem, pushing the variable to a value of 0 helped us to satisfy the domain constraint right. So, it is not necessary that the same correction procedure will work in all problems. It is very problem specific that is why we chose to include that correction mechanism in the fitness function evaluation file and not in the algorithm as such right.

We could have even implemented it over there, but then that may or may not work for all problems right, but the correction approach, usually is expected to give better results right than an approach without correction. So, in this case we did not assign any penalty, but we helped algorithm by correcting the solution. So, we need to remember that when we correct a solution; we not only should pass the objective function value, but should also pass the corrected solution, that is one of the correction approach right. So, you can also think of other approaches right.

(Refer Slide Time: 21:17)

Different correction approaches

Processes		P1	P2	P3	P4	P5
Low level capacity (l)		5	9	1	3	4
Decision variables	X	12	6	2	19	2
Approach 1 (Fix it to zero)	X_c	12	0	2	19	0
Approach 2 (Fix it to low level)	X_c	12	9	2	19	4
Approach 3 (Fix randomly)	X_c	12	0	2	19	4

$r = 0.3$

$r = 0.8$

12 6 2 19 2

Approach 1

$$x_i = \begin{cases} 0 & \text{if } x_i < l_i \text{ and } x_i \neq 0 \\ x_i & \text{else} \end{cases}$$

$\forall i = \{1, 2, \dots, D\}$
where D is the problem dimension

Approach 2

$$x_i = \begin{cases} l_i & \text{if } x_i < l_i \text{ and } x_i \neq 0 \\ x_i & \text{else} \end{cases}$$

$\forall i = \{1, 2, \dots, D\}$
where D is the problem dimension

Approach 3

$$x_i = \begin{cases} 0 & \text{if } x_i < l_i \text{ and } x_i \neq 0 \text{ and } r \leq 0.5 \\ l_i & \text{if } x_i < l_i \text{ and } x_i \neq 0 \text{ and } r > 0.5 \\ x_i & \text{else} \end{cases}$$

$\forall i = \{1, 2, \dots, D\}$
where D is the problem dimension

So, for example, let us assume this is the low level capacity, this is coming from the data right. So, it is more like we cannot produce anything less than 5, we cannot produce anything less than 9 for second process. We cannot produce anything less than 1 right. We cannot produce anything less than 3 for process 4 and we cannot produce less than 4 for process 5, again 0 is allowed right.

So, if there is a 0 that is not an issue right. So, let us assume that this is the decision variable that we are getting from a metaheuristic technique right. Let us say we get 12, 6, 2, 19, 2 right. These two values are non-zero, but they are also lower than the respective l value right.

So, this is the approach which we discuss right. So, for all those variables which do not satisfy the domain constraint right. We will assign it a value of 0. So, this is the approach which we discussed earlier. So, here what we are doing is if it is not equal to 0 and if it is less than l , we

are assigning a value of 0 else we retain the value of X_i as given by the metaheuristic technique. So, for example, here the value given by metaheuristic technique is 12, 2 and 19. So, we are not modifying those values right, those values satisfy the domain constraint. So, we do not change them only those things which violate the domain constraint we change it to 0 right.

So, the other approach could be fixing it to low level right. So, in this case what we are doing is; if the value is within the domain, then it is fine right. Otherwise, if it is less than 1 and if it is non zero, we make it as 1 right. So, for example, this variable was violating the 6 and 2 are violating. So, right now what we do is; this 6 will be converted to 9. So, this is nine right because the low level value is 9. Previously, in this first approach we converted it to 0, here we will convert it into nine right whereas, for the fifth variable the value that we get from the metaheuristic technique is 2 right, but the 1 value is 4.

So, here we fix it to 4 right. So, this is another approach. So, we can either employ this approach or we can either employ this approach. So, there is also another approach wherein we fix it randomly right. So, here we generate a random number for every decision variable that is violating the domain constraint. If the random number generated is less than or equal to 0.5, we fix it to 0. If the random number is greater than 0.5, we fix it to corresponding 1 values right. Otherwise if it is in the proper domain, we do not need to change it. So, for example, consider the second process right.

So, the low level value is 9, the value that we get is 6 right. So, if we generate a random number and let us say the random number happens to be 0.3 right. So, if r is less than or equal to 0.5, so 0.3 is less than or equal to 0.5 so, we fix it to 0 right. Similarly if we see the fifth variable r is 0.8 right. So, 0.8 is greater than 0.5. So, we fix it to its low level value. So, this two gets converted into. So, what we got from the algorithm is 12, 6, 2, 19 and 2, what we will be returning back is one of these threes right. Either 12, 0, 2, 19, 0 or 12, 9, 2, 19, 4 or 12, 0, 2, 19, 4. Depending upon which approach we take.

So, over here we will demonstrate the approach 1 right. We will implement it on the course that we have you can evaluate both of these approaches right or design your own approach to see if you are able to get better results than what we are discussing over here.

Now the question can be why did we correct, only for the domain constraint, why not for raw material constraint or the investment cost constraint? If you can design a correction mechanism for investment cost as well as raw material constraint you can implement it. So, if you are able to design a correction mechanism to handle investment cost and raw material constraint, you can even choose to implement it and check if it is actually benefiting the algorithm.

In the previous session we had seen solution of the production planning problem using teaching learning based optimization. We did not compare it with other algorithms right. So, first what we will do is; we will solve the same problem right with particle swarm optimization and differential evolution along with TLBO right. So, out of the 5 techniques which we have discussed in this course. Out of the 5 techniques which we discussed in this course we are selecting only 3 right. So, particle swarm optimization, teaching learning based optimization and differential evolution.

The reason for doing this is that for these three algorithms the maximum number of functional evaluation is a deterministic expression right. So, for example, for teaching learning based optimization, it was $N_p + 2 N_p T$ right where N_p was the population size and T was number of iterations. For particle swarm optimization and differential evolution the expression if you remember it is $N_p + N_p T$ this is because in teaching learning based optimization in every iteration for every member we evaluate the fitness function twice whereas, for the other two algorithms particle swarm optimization and differential evolution it was only once.

We want to compare these algorithms with respect to maximum number of fitness function evaluation. Whereas, the other two algorithms; genetic algorithm and artificial bee colony optimization did not have a deterministic expression for the number of maximum functional evaluation right. So, for example, in artificial bee colony optimization depending upon when

we encounter the scout phase. The number of fitness function evaluation would change right. So, it can vary from Np plus $2 Np T$ right. So, that is the minimum number of functional evaluation and the maximum number of functional evaluation is np plus $2 Np T$ plus t .

Assuming that the scout phase is encountered in every iteration. Similarly, for genetic algorithm also the number of maximum functional evaluation is not a deterministic expression. It depends upon whether an offspring undergoes mutation or not. So, for this particular comparison, we are only taking three algorithms because for all the three algorithms we have coded with respect to maximum number of iterations right. So, if you remember that loop for t is equal to one to t that was there in all three of them right. We did not write it with respect to number of fitness function evaluation right.

So, we can do that right. So, for example, all the 5 codes we can convert it for maximum number of fitness function evaluation that only requires a little bit of coding skill you should be able to do it on your own. We will also upload those codes with respect to the number of fitness function evaluation on the course base. So, for this session we will only restrict with particle swarm optimization, differential evolution and teaching learning based optimization right. So, we are not going to change anything in this function file. So, this is the SKS underscore production planning. So, this function file is what gives us the fitness function value right.

So, we are not going to do anything to this function file. Production planning data is just passing the data to this file which evaluates the fitness function right and these are the three algorithms; teaching learning based optimization, differential evolution and particle swarm optimization. So, all these three are now function files right. So, the TLBO returns the best solution at the end of specified number of iteration. The value of the fitness function corresponding to this best solution. So, `bestsol` is actually the set of decision variable best fitness is the fitness function value of `bestsol` `BestFitIter` gives the convergence curve.

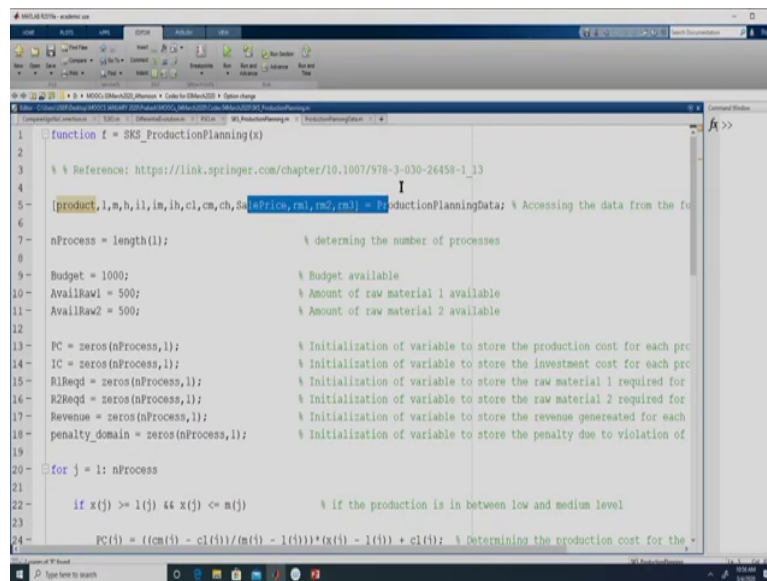
So, it basically tells what is the best fitness function value obtained in every iteration and that includes the first initial population. This p is the final population. So, the dimension of p would be $N p$ cross D where Np is the population size and D is the number of decision variable.

Whereas, f would be N_p cross 1. So, for each solution in p , we will have a fitness function value. So, that is what is given in f right. So, the output is same for the three algorithm; bestsol, best fitness, BestFitIter p and f . For TLBO, the input is the fitness function file, the lower bound, the upper bound, population size and the number of iteration right.

For differential evolution it is the same thing right. In addition we need to provide crossover probability as well as the scaling factor. So, this crossover probability is required in crossover operation and the scaling factor is required in the mutation right.

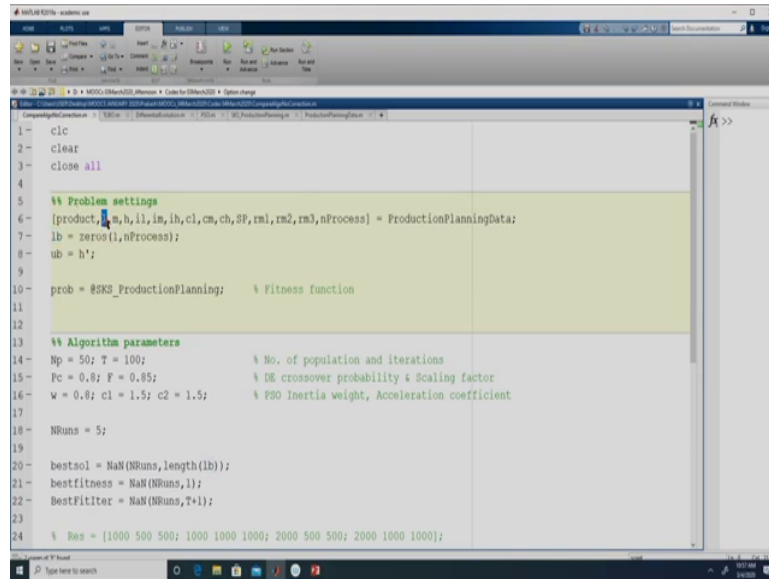
So, similarly for particle swarm optimization, the output from the algorithm is same right, input is the problem the lower and upper bounds, the population size number of iteration. In addition to that we need to give the inertia weight and two acceleration coefficients c_1 and c_2 right. So, now, we have these three metaheuristic techniques right and we have our problem over here the production planning problem.

(Refer Slide Time: 29:54)



```
1 function f = SKS_ProductionPlanning(x)
2
3 %% Reference: https://link.springer.com/chapter/10.1007/978-3-030-26458-1_13
4
5 [product, l, m, h, il, im, ih, cl, cm, ch, Re, c] = ProductionPlanningData; % Accessing the data from the fo
6
7 nProcess = length(l); % determining the number of processes
8
9 Budget = 1000; % Budget available
10 AvailRaw1 = 500; % Amount of raw material 1 available
11 AvailRaw2 = 500; % Amount of raw material 2 available
12
13 PC = zeros(nProcess,1); % Initialization of variable to store the production cost for each pro
14 IC = zeros(nProcess,1); % Initialization of variable to store the investment cost for each pro
15 R1Reqd = zeros(nProcess,1); % Initialization of variable to store the raw material 1 required for
16 R2Reqd = zeros(nProcess,1); % Initialization of variable to store the raw material 2 required for
17 Revenue = zeros(nProcess,1); % Initialization of variable to store the revenue generated for each
18 penalty_domain = zeros(nProcess,1); % Initialization of variable to store the penalty due to violation of
19
20 for j = 1: nProcess
21
22     if x(j) >= l(j) && x(j) <= m(j) % if the production is in between low and medium level
23
24         PC(j) = ((cm(j) - cl(j))/(m(j) - l(j)) * (x(j) - l(j)) + cl(j); % Determining the production cost for the
```

(Refer Slide Time: 30:01)



```
1- clc
2- clear
3- close all
4
5
6- %% Problem settings
7- [product, m, h, l, lb, lb, cl, cm, ch, SP, rm1, rm2, rm3, nProcess] = ProductionPlanningData;
8- lb = zeros(1, nProcess);
9- ub = h';
10
11- prob = @SKS_ProductionPlanning; % Fitness function
12
13
14- %% Algorithm parameters
15- Np = 50; T = 100; % No. of population and iterations
16- Pc = 0.9; F = 0.85; % DE crossover probability & Scaling factor
17- w = 0.9; c1 = 1.5; c2 = 1.5; % PSO Inertia weight, Acceleration coefficient
18
19- NRuns = 5;
20- bestsol = NaN(NRuns, length(lb));
21- bestfitness = NaN(NRuns, 1);
22- BestFitter = NaN(NRuns, T+1);
23
24- % Res = [1000 500 500; 1000 1000 1000; 2000 500 500; 2000 1000 1000];
```

And this is the data for the production planning problem right. So, this is a script wherein we are going to compare the three algorithms right. So, the first three lines uses to clear the command window the workspace and close any figure if it is open right.

So, here in line 6 we are accessing this function production planning data which contains the data right. So, because we need to provide the upper bound right. Upper bound if you remember it is the h value right the maximum production that is possible right and lower bound is 0. Remember it is not the low level value because if we put it as low level value, then we are enforcing an artificial constraint that each process has to be use right.

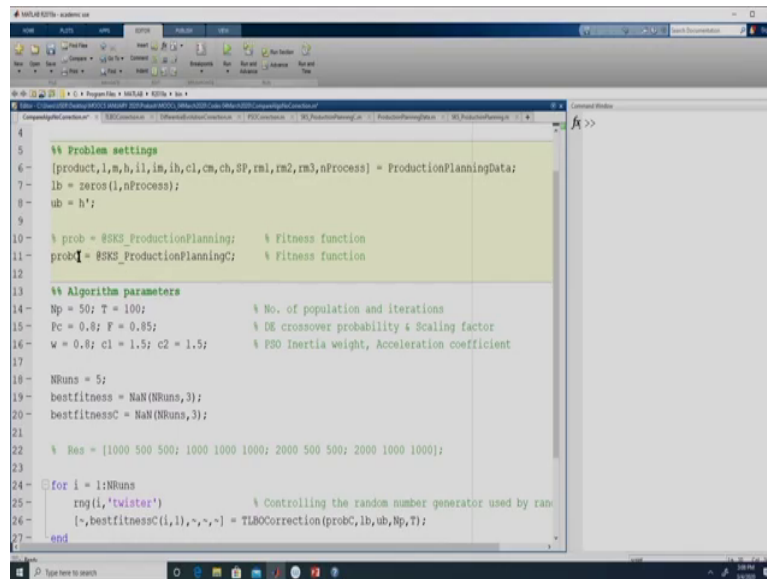
So, that constraint is not part of the problem. The problem specifies that either we can choose not to produce or if you decide to produce it has to be l or greater than l and it has to be less than h. So, there is a domain hold. So, that is why we are taking the lower bound as zeros. So, this n process will also tell us the number of processes or in this case the number of decision

variable right. So, this is lower bound this is the upper bound. So, prob is a function handle right. So, we are assigning the name of this file SKS underscore production planning right.

To prob right, and then since we are using three algorithms. We need to specify the parameters that we are going to use. So, we are fixing the population size to be 50 and the number of iterations to be 100 right. So, these two values remain constant for all the three algorithms right. In addition to this for TLBO we do not require any other parameter. Whereas, for D E, we require the crossover probability and scaling factor. So, we have fix the crossover probability to 0.8 and the scaling factor to 0.85 right and for particle swarm optimization, we need to provide the inertia weight. We have taken it as 0.8.

The acceleration coefficients c_1 and c_2 as 1.5, 1.5. So, that helps in defining the parameters which are required for the algorithm. So, NRuns is the number of runs. So, so for the purpose of demonstration we are just taking 5 runs right.

(Refer Slide Time: 32:04)



```
4
5 %% Problem settings
6 [product,l,m,h,lb,lm,lb,cl,cm,ch,SP,rm1,rm2,rm3,nProcess] = ProductionPlanningData;
7 lb = zeros(1,nProcess);
8 ub = h';
9
10 % prob = @SKS_ProductionPlanning; % Fitness function
11 probf = @SKS_ProductionPlanningC; % Fitness function
12
13 %% Algorithm parameters
14 Np = 50; T = 100; % No. of population and iterations
15 Pc = 0.8; F = 0.85; % DE crossover probability & Scaling factor
16 w = 0.8; c1 = 1.5; c2 = 1.5; % PSO Inertia weight, Acceleration coefficient
17
18 NRuns = 5;
19 bestfitness = NaN(NRuns,3);
20 bestfitnessC = NaN(NRuns,3);
21
22 % Res = [1000 500 500; 1000 1000 1000; 2000 500 500; 2000 1000 1000];
23
24 for i = 1:NRuns
25     rng(i,'twister') % Controlling the random number generator used by ran
26     [~,bestfitness(i,1),-,,-] = TLBOCorrection(probC,lb,ub,Np,T);
27 end
```

Let me just remove this. We are not going to use these two, right. We are going to use this variable best fitness to store the fitness function value reported by every algorithm right and each run right. So, we are going to have NRuns. So, in this case we are going to have 5 runs right and we have 3 algorithms. So, that is why we have taken 3 columns right.

So, what we are basically trying to do is; the first column we will use it for TLBO, the second column we will use it for differential evolution and third column we will use it for particle swarm optimization of this variable fitness and for each algorithm, we are going to run 5 times right. For each run that we complete, we will populate best fitness with the solution obtain from that algorithm right. So, this is the for loop we are going to compare these three algorithms right based on 5 runs right. So, that is why we have this loop for i is equal to 1 to NRuns right and again we are fixing the seed right.

So, rng of i comma twister. So, that we can reproduce the result right and in line 24, we are solving the problem with TLBO right. So, we are giving the necessary input prob lb, ub, Np and T. So, the solution that we will be returned by TLBO is all of this thing the set of decision variable, the fitness function the convergence curve for that run, the final population and the final fitness function right. So, right now we are interested only in the fitness function value. So, we will do statistical analysis based on that fitness function value. So, that is why we are not receiving any of this four values right

We are not receiving what is the set of decision variable, we are not receiving the values for plotting the convergence curve, we are not receiving the final population and we are also not receiving the fitness function values of the final population right. So, if you are interested you can just specify a variable name and then you can appropriately analyze whatever you wish to. So, this will help us to solve with TLBO right. So, this line again specifies that. So, for differential evolution also we want to control the random numbers right. So, that we can reproduce it.

So, that is why we are including this line right. So, over here as well as in over here for particle swarm optimization. Even if we remove line 26 and line 29, it is correct right. Only thing is that we will no longer be able to reproduce the results of a particular run right. So, for example, let us say we do not have this, let us say we comment these lines right.

And if you want to reproduce the third run of PSO. Only the third run of PSO it is not possible right, but if we have these lines over here right, then we do not need to execute TLBO differential evolution. We can merely say rng of whichever run was the best run comma twister and we can get those results.

So, that is why we are fixing the random number for each of the algorithm right. So, similar to TLBO for differential evolution and particle swarm optimization. We are only interested in the fitness function value of the best solution right. So, here we stored in the first column of best fitness, here we are storing it in the second column, here we are storing it in the third column right.

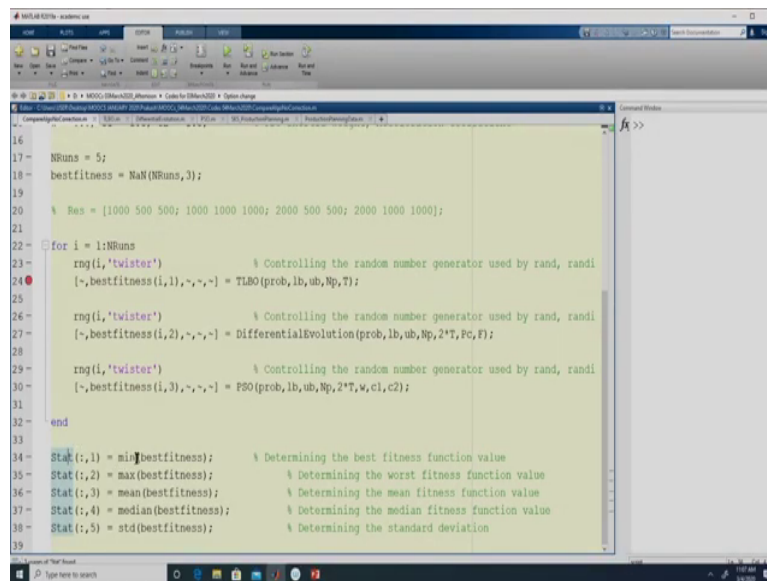
So, the input for differential evolution is problem its lower bound and its upper bound the population size right, the crossover probability and the fitness function. Similarly, for particle swarm optimization these three are with respect to the problem population size, inertia weight and the two acceleration coefficient.

For TLBO, we had given only T iterations right. So, here we have specify T, but for differential evolution and particle swarm optimization, we are providing the number of iterations as 2 into T. So, in this case T we have set as 100 iterations. So, we will run TLBO for 100 iteration, but we will execute TLBO for 100 iteration, but for differential evolution and particle swarm optimization. We will execute it for 200 iterations right.

So, only then the number of fitness function used by TLBO, differential evolution and particle swarm optimization would be identical right. So, remember we want to compare the results of these three algorithms, but the termination criteria cannot be the number of iterations, then TLBO will be using more number of fitness function evaluation compared to particle swarm optimization and differential evolution right.

In order to avoid that we are executing differential evolution as well as particle swarm optimization for twice the number of iterations as compared to teaching learning based optimization right. So, this will complete execution of all the runs right.

(Refer Slide Time: 36:44)



```
16
17 - NRuns = 5;
18 - bestfitness = NaN(NRuns,3);
19
20 % Bas = [1000 500 500; 1000 1000 1000; 2000 500 500; 2000 1000 1000];
21
22 - for i = 1:NRuns
23 -     rng(i,'twister')           % Controlling the random number generator used by rand, randi
24 -     [~,bestfitness(i,1),-,-] = TLBO(prob,lb,ub,Np,T);
25
26 -     rng(i,'twister')           % Controlling the random number generator used by rand, randi
27 -     [~,bestfitness(i,2),-,-] = DifferentialEvolution(prob,lb,ub,Np,2*T,Pc,F);
28
29 -     rng(i,'twister')           % Controlling the random number generator used by rand, randi
30 -     [~,bestfitness(i,3),-,-] = PSO(prob,lb,ub,Np,2*T,w,c1,c2);
31
32 - end
33
34 - Stat(:,1) = min(bestfitness);   % Determining the best fitness function value
35 - Stat(:,2) = max(bestfitness);   % Determining the worst fitness function value
36 - Stat(:,3) = mean(bestfitness);  % Determining the mean fitness function value
37 - Stat(:,4) = median(bestfitness); % Determining the median fitness function value
38 - Stat(:,5) = std(bestfitness);   % Determining the standard deviation
39
```

So, here we have the statistical analysis right. We are finding the best fitness of all the three algorithms right. So, when we do min of best fitness it will give us a column vector right. So, the first column of the variable stat right. So, this is just a variable name, we are saying that the first column of the variable stat is the best value obtained by each of the algorithm right.

So, that will have three rows right and the first column will contain the best objective function value. The second column similarly for all the three algorithms we will contain the worst fitness function value. Remember all these three algorithms have been written for minimization right. So, this is max of best fitness and then similar to what we did previously mean for each algorithm, median for each algorithm, standard deviation for each algorithm considering the 5 runs which we are executing right. So, let us see what happens if we execute this right. So,

here I had kept a break point right let me remove that breakpoint. So, it takes a little while to solve.

So, now, we have the result correct. So, the first toe is for teaching learning based optimization right. So, the best value obtained by TLBO is minus 400, because it is in the first column right. So, the first column gives the best solution with respect to each of the three algorithm, the second column gives us the worst solution with respect to each of the three algorithm, the third column gives us mean fourth median and the fifth one gives us standard deviation right. So, remember our previous discussion that for the current problem the fitness function will have to have a negative value for the solution to be even feasible.

So, in this case we can see that differential evolution is not able to obtain any feasible solution right. Because, the best solution obtained by differential evolution in all of the 5 runs right is positive value; that means, there is some penalty which indicates that the solution is not feasible with respect to the best solution obtained by the three algorithm. The best algorithm is particle swarm optimization because it discovers a solution of minus 546.28. Whereas, TLBO is able to determine solution which has a fitness of minus 400.59 right.

So, if one way to implement a production plan right, they would prefer to choose the solution given by particle swarm optimization right. So, similarly if we compare the worst value, the worst which is determined by TLBO is better than what is determined by PSO right.

(Refer Slide Time: 39:21)

```

16
17 - NRuns = 5;
18 - bestfitness = NaN(NRuns,3);
19
20 % Res = [1000 500 500; 100
21
22 - for i = 1:NRuns
23 -     rng(i,'twister')
24 -     [~,bestfitness(i,1)],-,-
25
26 -     rng(i,'twister')
27 -     [~,bestfitness(i,2)],-,-
28
29 -     rng(i,'twister')
30 -     [~,bestfitness(i,3)],-,-
31
32 - end
33
34 - Stat(:,1) = min(bestfitness
35 - Stat(:,2) = max(bestfitness
36 - Stat(:,3) = mean(bestfitness
37 - Stat(:,4) = median(bestfitness
38 - Stat(:,5) = std(bestfitness
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Stat =

-400.59	-322.08	-372.10	-380.20	31.18
116906494035348127744.00	910173156414076616704.00	446618584437792374784.00	50000000000000000000.00	
-546.28	-285.63	-420.77	-438.12	10.61

bestfitness =

-364.34	50000000000000000000.00	-369.24
-400.59	172377968659476938752.00	-438.12
-380.20	533635303080060256256.00	-285.63
-322.08	910173156414076616704.00	-546.28
-393.31	116906494035348127744.00	-464.59

So, we can actually look at this best fitness. So, these are the results of the 3 run. The first column indicates TLBO, the second column indicates D, the third column indicates particle swarm optimization right. So, in this case the best value was minus 400.59 and the worst is minus 322.08. Whereas the worst in this is minus 285.63 right though particle swarm optimization discovers a better solution than teaching learning based optimization for this set of settings.

But the standard deviation across the 5 runs of TLBO is less than particle swarm optimization, but the standard deviation of TLBO is less than that of particle swarm optimization right. Again, remember this set of result is only for the set of parameters right. So, if we change this parameters of differential evolution, the solution might even improve right.

So, for an arbitrary problem it is not possible to say what is the best value of P_c , F , w , c_1 and c_2 right. So, when we execute an algorithm we need to try with multiple values of this tuning parameter right. Though we have shown your statistical analysis right it is very preliminary right because we are also supposed to change this parameters and then we are also supposed to analyze the impact of these user defined parameters.

In this case we showed you preliminary comparison of these three algorithms right. So, if we have code of ga and abc in which we can specify the number of functional evaluation. The algorithm would stop after utilizing that many number of functional evaluation. We can also include them over here.

Right now, we know how many functional evaluation we took for TLBO all right. So, that is N_p plus $2 N_p T$ right. So, that value can be calculated and it can be given to that specific code of genetic algorithm and artificial bee colony optimization. So, in that case all the five algorithms utilize the same number of fitness function evaluation and similar statistical analysis can be performed right. So, that concludes the comparison of algorithms for this production planning problem.

Many times what happens is we want to solve the same problem with different set of data. For example, in the case study which we were discussing so far, the budget value was 1000 and the raw material 1 that is available was 500, the raw material 2 that was available is 500 right.

(Refer Slide Time: 41:48)

	B	R1	R2	Profit
Case 1	1000	500	500	x_1
Case 2	1000	1000	1000	x_2
Case 3	2000	500	500	x_3
Case 4	2000	1000	1000	x_4

So, this is what we were having, let me say this as case 1 right. So, for case 1; the budget that is available is 1000 right and the raw material 1 that is available is 500 and the raw material 2 that is available is 500 right. So, for this let us say we get some profit right. So, depending upon which algorithm we are working with, we will get some value of profit over here right.

So, same problem we want to solve, let us say with budget as 1000 right. And the amount of raw material 1 that is available is not 500, but it is 1000, the amount of raw material 2 that is available is also 1000. So, we will get some profit over here let us term this profit that we get over here as x_1 and the profit that we get over here in case 2 as x_2 right. So, if you think about it, here we have only increase the resources right. Some 500 to 1000, we have only increase the resources. So, any solution which is feasible over here is also feasible for this one right. So, because we are only relaxing the problem right.

So, we should at least get a solution which has a profit of x_1 . We can get a solution which may have a better profit than x_1 , but it should not happen that x_2 is inferior to x_1 . Because x_2 is the solution for a relaxed problem right. So, similarly for this study we have two more cases right. Where in the budget is increased to 2000, the raw material is at 500 right and the 4th case is the budget available is 2000 and the raw material that is available is 1000. So, let us call this as profit as x_3 and x_4 right. So, over here if we see x_4 cannot be inferior to any other value right. Because x_4 is a relaxed version of any of the other three problem.

So, any solution which satisfies these three cases would also satisfy this. We should not have a solution whose profit is inferior to any of this three. Now, the question is how do we implement this? Right. So, remember for each of this case, we need to run multiple times right because case 1 is a individual optimization problem, case 2 is an individual optimization problem similarly, case 3 and case 4.

So, now, we will see how to execute such a problem. Over here if we see the first figure shows the approach without correction. The second figure shows the approach with correction right. So, whatever we are discussing holds true for both of them right. If you look at our implementation we had those metaheuristic technique that was sending the value of X and it was receiving the value of f right.

And we had the script file right. So, through the script file we were passing the algorithm parameter and problem details right. So, every time what we were getting from the solution of metaheuristic technique was the best solution its corresponding objective function, the final population, the fitness function corresponding to the final population and the best fitness function value obtained in every iteration, that was what we use for plotting convergence curve. So, this is what we were receiving from the metaheuristic technique right. So, now, what we can do is; when we pass this problem details, we can also pass the values of budget, the amount of raw material 1, the amount of raw material 2, that is required.

So, previously we are only passing the upper bound, lower bound and name of the fitness function file right. So, this B , R_1 , R_2 can be pass to the metaheuristic technique right, but

this B, R 1, R 2 is actually required in this fitness function right. So, we will have to now change our metaheuristic technique right to receive this input from the script file right and as well as transfer that data to the optimization problem. So, now, we will say that what we get from the metaheuristic technique is x as well as we will say what are the resources right. So, this three put together let me call it as variable Res. So, this Res can be pass to the optimization problem.

So, the fitness function which could previously receive only X should now also be capable of receiving Res right. So, what we are discussing here is more with respect to the implementation right and not necessarily with respect to optimization. But, since implementation becomes a part of the optimization study that is why we are discussing it over here.

So, this is also true for even if we have correction right. So, we will have to pass B, R 1, R 2 every time right. So, we have 4 sets; so first time we will pass the first set, second time we will pass the second set. So, that will pass 4 times and this metaheuristic technique along with the decision variable will pass the parameters and those parameters will be utilize in fitness function and the corrected fitness value and corrected solution will be returned back to the metaheuristic technique right.

So, this is one way of doing it. Otherwise in MATLAB a direct communication can be establish between these two files right using a global variable. So, we will not use that approach. So, this is the file that we were working with right. So, here let us get rid of things that we do not want. So, we do not want this. Now, we are working only with the correction approach right. Let me uncomment this right. So, these are the 4 cases that for the first time we need to pass 1000, 500, 500, for the second time we need to pass 1000, 1000, 1000, for the third time we need to pass this value and the for the fourth time we need to pass this value. So, every case is a row now right.

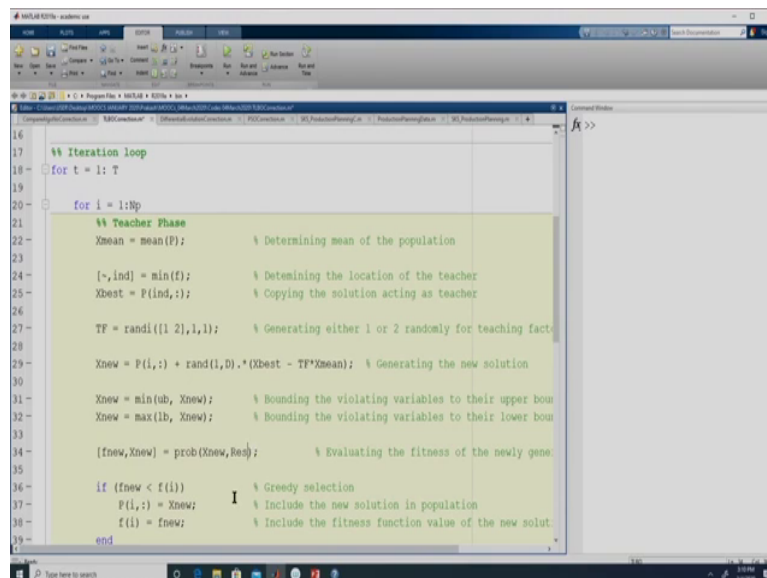
And we have four such problems. So, we will put another loop over here and now in addition to this we need to pass Res, that is the name of the variable in which we have defined the 4 cases right. The j th row we need to pass and all the columns right. So, these 3 are details with

respect to the problem, these 2 are details with respect to the algorithm right and again this is a detail with respect to the problem.

So, so far we were saying the input to the algorithm is just fitness function lower bound and upper bound, but we can also pass other parameters. But this will not be used in the algorithm. Since algorithm happens to be in between our script file and the fitness function file, we are passing it to the algorithm and subsequently the algorithm will pass it on to the fitness function right.

So, over here we have change the algorithm right. So, our algorithm is over here. So, this should be capable of receiving that variable right and wherever we call the objective function, we need to pass this value.

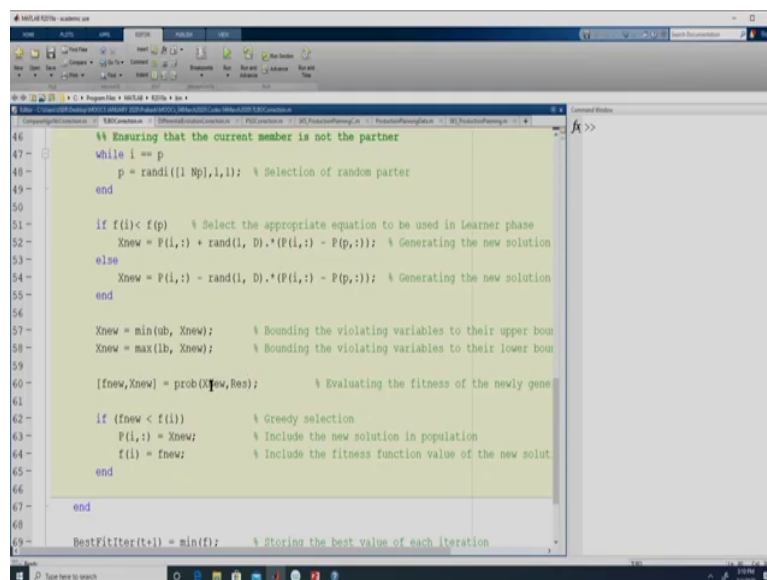
(Refer Slide Time: 48:01)



```
16
17 %% Iteration loop
18 for t = 1: T
19
20     for i = 1:Np
21         %% Teacher Phase
22         Xmean = mean(F);           % Determining mean of the population
23
24         [~,ind] = min(f);         % Determining the location of the teacher
25         Xbest = P(ind,:);        % Copying the solution acting as teacher
26
27         TF = randi([1 2],1,1);   % Generating either 1 or 2 randomly for teaching fact
28
29         Xnew = P(i,:) + rand(1,D).*(Xbest - TF*Xmean); % Generating the new solution
30
31         Xnew = min(ub, Xnew);     % bounding the violating variables to their upper bou
32         Xnew = max(lb, Xnew);    % bounding the violating variables to their lower bou
33
34         [fnew,Xnew] = prob(Xnew,Res); % Evaluating the fitness of the newly gene
35
36         if (fnew < f(i))         % Greedy selection
37             P(i,:) = Xnew;      % Include the new solution in population
38             f(i) = fnew;       % Include the fitness function value of the new solut
39         end
40     end
41 end
```


So, we call the objective function here and then over here in the teacher phase and in the student phase right.

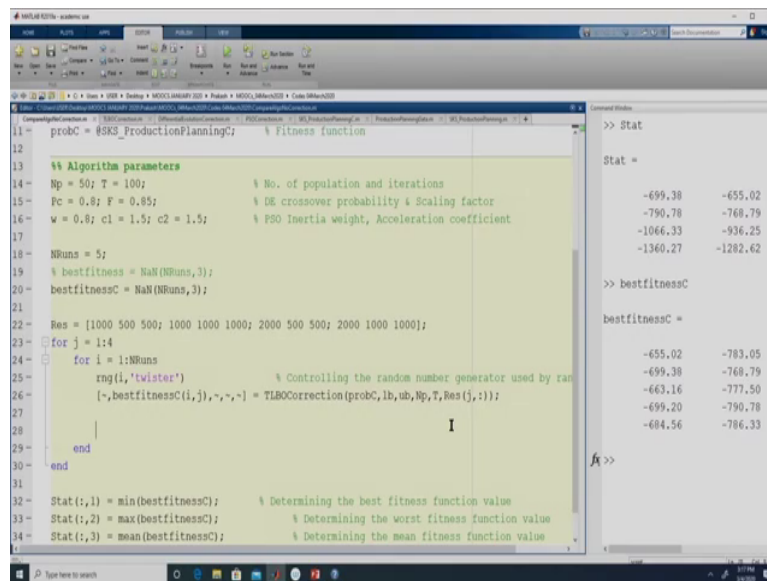
(Refer Slide Time: 48:13)



```
46 % Ensuring that the current member is not the partner
47 while i == p
48     p = randi([1 Np],1,1); % Selection of random partner
49 end
50
51 if f(i) < f(p) % Select the appropriate equation to be used in learner phase
52     Xnew = P(i,:) + rand(1, D) .* (P(i,:) - P(p,:)); % Generating the new solution
53 else
54     Xnew = P(i,:) - rand(1, D) .* (P(i,:) - P(p,:)); % Generating the new solution
55 end
56
57 Xnew = min(ub, Xnew); % Bounding the violating variables to their upper bound
58 Xnew = max(lb, Xnew); % Bounding the violating variables to their lower bound
59
60 [fnew, Xnew] = probDFnew, Res); % Evaluating the fitness of the newly genes
61
62 if (fnew < f(i)) % Greedy selection
63     P(i,:) = Xnew; % Include the new solution in population
64     f(i) = fnew; % Include the fitness function value of the new solution
65 end
66
67 end
68
69 BestFitIter(t+1) = min(f); % Storing the best value of each iteration
```

So, Res, we are not actually using it right its Res; if you see we are not actually using it in the algorithm. So, going back to this problem right. So, over here we will get Res right.

(Refer Slide Time: 48:26)



```
11 = probC = @SKS_ProductionPlanning; % Fitness function
12
13 %% Algorithm parameters
14 Np = 50; T = 100; % No. of population and iterations
15 Pc = 0.8; F = 0.85; % DE crossover probability & Scaling factor
16 w = 0.8; c1 = 1.5; c2 = 1.5; % PSO Inertia weight, Acceleration coefficient
17
18 NRuns = 5;
19 % bestfitness = NaN(NRuns,3);
20 bestfitnessC = NaN(NRuns,3);
21
22 Res = [1000 500 500; 1000 1000 1000; 2000 500 500; 2000 1000 1000];
23 for j = 1:4
24     for i = 1:NRuns
25         rng(i,'twister') % Controlling the random number generator used by rand
26         [~,bestfitness(i,j),~,~,~] = TLBOCorrection(probC,lb,ub,Np,T,Res(j,:));
27
28         |
29     end
30 end
31
32 Stat(:,1) = min(bestfitnessC); % Determining the best fitness function value
33 Stat(:,2) = max(bestfitnessC); % Determining the worst fitness function value
34 Stat(:,3) = mean(bestfitnessC); % Determining the mean fitness function value
35
```

Command Window Output:

```
>> Stat
Stat =
    -699.38    -655.02
    -790.78    -768.79
   -1066.33    -936.25
   -1360.27   -1282.62

>> bestfitnessC
bestfitnessC =
    -655.02    -783.05
    -699.38    -768.79
    -663.16    -777.50
    -699.20    -790.78
    -684.56    -786.33
```

So, these values are to come from the variables. So, we have written in such a way that the first value contains the budget right. The second value contains the amount of raw material 1, that is available and the third value contains the amount of raw material 2 that is available to us.

So, now from the script file we can run all this 4 cases. Now, we have defined this outer loop right. So, this outer loop we will take care of the four problems that we have and for each of the problem we need to implement 10 runs right and this is the statistical analysis part which we are doing over here right. So, right now we are not saving the best solution or the values for the convergence curve or the final population. We are only storing the fitness function value right; so best fitness C. So, the first time it will be the first column right. So, instead of this 1, we need to give j. So, that we will get 4 columns right and we will have 5 rows.

So, each row corresponds to a particular run and each column will correspond to these cases right. So, for the first case, second case, third case and the fourth case right. So, if we execute this it will take a little while right. So, now, we have the command prompt. So, we can look at this variable stat right.

(Refer Slide Time: 49:43)

```

11- probC = BSKS_ProductionPlanningC;
12-
13- %% Algorithm parameters
14- Np = 50; T = 100; % No. C
15- Pc = 0.8; F = 0.85; % DE ct
16- w = 0.8; c1 = 1.5; c2 = 1.5; % PSO I
17-
18- NRuns = 5;
19- % bestfitness = NaN(NRuns,3);
20- bestfitnessC = NaN(NRuns,3);
21-
22- Res = [1000 500 500; 1000 1000 1000; 2000 1000 1000];
23- for j = 1:4
24-     for i = 1:NRuns
25-         rng(i,'twister') % C
26-         [~,bestfitnessC(i,j),-,,-] = TLBOC(Res,probC);
27-     end
28- end
29-
30- Stat(i,j) = min(bestfitnessC); % Det
31- Stat(i,2) = max(bestfitnessC);
32- Stat(i,3) = mean(bestfitnessC);
33- Stat(i,4) = median(bestfitnessC);
34- Stat(i,5) = std(bestfitnessC);

```

```

>> Stat
Stat =
    -699.38    -655.02    -680.26    -684.56    20.45
    -790.78    -768.79    -781.29    -783.05    8.50
   -1066.33    -936.25    -990.01    -952.27    64.72
   -1360.27   -1282.62   -1318.01   -1304.77    36.82

>> bestfitnessC
bestfitnessC =
    -655.02    -783.05   -1054.61   -1360.27
    -699.38    -768.79   -1066.33   -1354.30
    -663.16    -777.50    -940.58   -1288.12
    -699.20    -790.78    -936.25   -1282.62
    -684.56    -786.33    -952.27   -1304.77

```

So, each row indicates a particular run. So, if we look at best fitness right. So, these are the 4 cases which we have right, case 1, case 2, case 3, case 4. So, the first column of best fitness c is case 1. So, for the case 1; we see that the values in 5 run are the 5 rows of the first column. Similarly, for the second problem right. Second problem in the sense with the budget of 1000, raw material 1 available being 1000 and the amount of raw material 2 available being 1000.

So, these are the results of 5 run. Similarly, the third case and the fourth case right. Now, if we implement a statistical analysis of this. Let us say the best value that we obtain is for case 1 is

minus 699.38 right. So, that is what is over here right. So, the first column indicates the best values, because over here if you see stats the first column indicates the best value right.

So, the third value this minus 1066.33 corresponds to the best value of the 5 runs for case 3 and similarly for case 4 right. The second column of stat indicates the worst value, the third column indicates the mean, the fourth one indicates the median and the fifth one indicates the standard deviation right. So, here we have shown you only for TLBO right. So, even in that case we get a table right.

(Refer Slide Time: 51:10)

The screenshot shows a MATLAB script for a TLBO optimization problem. The script defines a fitness function, sets algorithm parameters (population size, iterations, crossover probability, scaling factor, inertia weight, acceleration coefficient), and runs the optimization for 5 iterations. The results are displayed in a table format.

```

11- probC = $KES_ProductionPlanningC; % Fitness function
12-
13- %% Algorithm parameters
14- Np = 50; T = 100; % No. of population and iterations
15- Pc = 0.8; F = 0.85; % DE crossover probability & Scaling factor
16- w = 0.8; c1 = 1.5; c2 = 1.5; % PSO Inertia weight, Acceleration coefficient
17-
18- NRuns = 5;
19- % bestfitness = NaN(NRuns,3);
20- bestfitnessC = NaN(NRuns,3);
21-
22- Res = [1000 500 500; 1000 1000 1000; 2000 500 500; 2000 1000 1000];
23- for j = 1:4
24-     for i = 1:NRuns
25-         rng(i, 'twister') % Controlling the random number generator used by rand
26-         [-,bestfitnessC(i,j),-,,-] = TLBOCorrection(probC,lb,ub,Np,T,Res(j,:));
27-
28-         |
29-     end
30- end
31-
32- Stat(:,1) = min(bestfitnessC); % Determining the best fitness function value
33- Stat(:,2) = max(bestfitnessC); % Determining the worst fitness function value
34- Stat(:,3) = mean(bestfitnessC); % Determining the mean fitness function value

```

Command Window Output:

```

>> Stat
Stat =
    -699.38    -655.02
    -790.78    -768.79
   -1066.33    -936.25
   -1360.27   -1282.62

>> bestfitnessC
bestfitnessC =
    -655.02    -783.05
    -699.38    -768.79
    -663.16    -777.50
    -699.20    -790.78
    -684.56    -786.33

```

So, what you can do is in addition to TLBO you can also run the problem with particle swarm optimization, differential evolution, genetic algorithm and artificial bee colony optimization. This is how we can run the same problem with different set of data for any number of specified runs with a particular algorithm. With that we will conclude the session.

Thank you.