

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

Lecture - 13
Implementation of Differential Evolution on MATLAB

Welcome back. In this session, we will Implement Differential Evolution on MATLAB. Before trying to implement let us just quickly go through the pseudo code of differential evolution once again.

(Refer Slide Time: 00:43)

Pseudocode of DE

Input: Fitness function, lb, ub, N_p , T, F , p_c

1. Initialize a random population (P)
2. Evaluate fitness (f) of P

```

for t = 1 to T
  for i = 1 to  $N_p$ 
    Generate the donor vector ( $V_i$ ) using mutation
    Perform crossover to generate offspring ( $U_i$ )
  end
  for i = 1 to  $N_p$ 
    Bound  $U_i$ 
    Evaluate the fitness ( $f_{U_i}$ ) of  $U_i$ 
    Perform greedy selection using  $f_{U_i}$  and  $f_i$  to update P
  end
end
end
  
```

$$V = X_{r_1} + F(X_{r_2} - X_{r_3})$$

$$u^j = \begin{cases} v^j & \text{if } r \leq p_c \text{ or } j = \delta \\ x^j & \text{if } r > p_c \text{ and } j \neq \delta \end{cases}$$

$$\left. \begin{matrix} X_i = U_i \\ f_i = f_{U_i} \end{matrix} \right\} \text{if } f_{U_i} < f_i$$

24

The input to differential evolution is the fitness function, the lower bounds, the upper bounds, the population size, the number of iterations, the scaling factor and the crossover probabilities. This crossover probability will be required to decide whether to perform crossover or not. So, the first step is to initialize a random population, then we need to evaluate the fitness of the

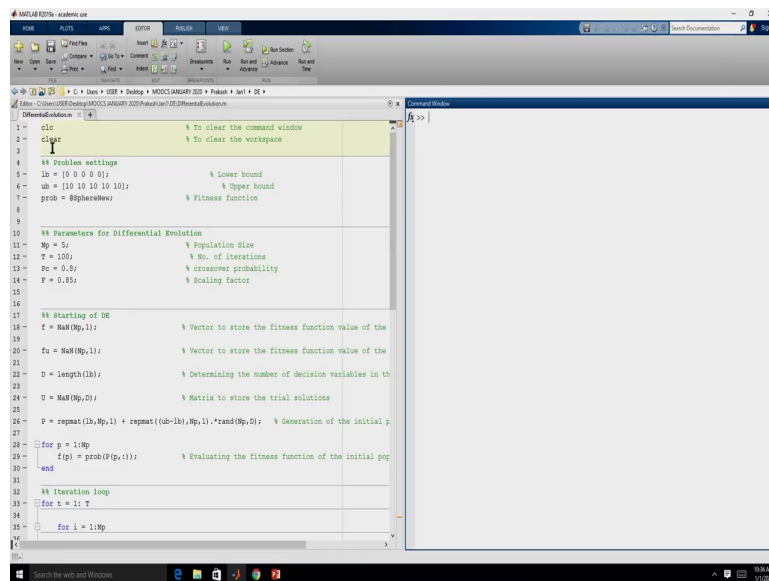
population, then we need to start the iteration loop right. So, for each population member, we are supposed to find a donor vector.

So, donor vector will be found using three random leave selector solutions X_{r1} , X_{r2} and X_{r3} right. Once we have generated the donor vector using mutation, we need to perform crossover to get the trail vector. So, the elements of the trail vector will be either from the donor vector which we generated using mutation or it will be from the target vector which is actually undergoing crossover. Once we find out this trail vector, we need to find out the trail vector for the next member right.

So, we are not supposed to update the population immediately, we need to determine all the trail vectors that is why this for loop is separately written before we bound and evaluate the fitness. So, once we have determined the trail solutions for all the target vectors, we need to bound each of them right, and we need to evaluate the fitness. Once we have evaluated the fitness we need to perform a greedy selection between the trail vector 1 and the target vector 1, trail vector 2 and target vector 2, and then update the population for the subsequent generation, so we will implement this on MATLAB right.

So, similar to what we have been doing previously, we will first walk you through the code, and then we will get into the debug mode and execute it line by line, so that it gives a better understanding of the working of differential evolution.

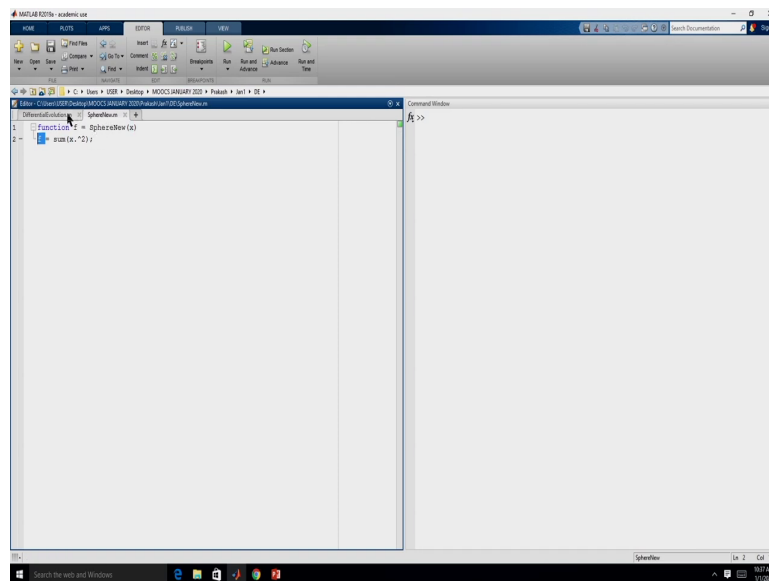
(Refer Slide Time: 02:35)



```
1 clear
2 clc
3
4 %% Problem settings
5 lb = [0 0 0 0]; % Lower bound
6 ub = [10 10 10 10]; % Upper bound
7 prob = @sphereNew; % Fitness function
8
9
10 %% Parameters for Differential Evolution
11 Np = 5; % Population Size
12 T = 100; % No. of iterations
13 Pc = 0.8; % crossover probability
14 F = 0.85; % Scaling factor
15
16
17 %% Starting of DE
18 f = NaN(Np,1); % Vector to store the fitness function value of the
19
20 fu = NaN(Np,1); % Vector to store the fitness function value of the
21
22 D = length(lb); % Determining the number of decision variables in the
23
24 D = NaN(Np,D); % Matrix to store the trial solutions
25
26 P = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D); % Generation of the initial p
27
28 for p = 1:Np
29     f(p) = prob(P(p,:)); % Evaluating the fitness function of the initial pop
30 end
31
32 %% Iteration loop
33 for t = 1:T
34     for i = 1:Np
```

So, this is the differential evolution code. So, by now you would be knowing these two lines, it will help us to clear the command window and to clear the workspace. So, line 5, 6, 7 is similar to what we have been doing in particle some optimization, teaching-learning based optimization that we define the lower bound of the problem, we define the upper bound of the problem, and we define variable prob which is actually a function handle. So, this function handle is going to contain the optimization problem which we are going to solve. So, in this case, we have given its spherenew because sphere is already a in-built function in MATLAB. So, we do not want to disturb that.

(Refer Slide Time: 03:11)



So, what we have done is we are calculating the fitness using this spherenew function. So, irrespective of the number of elements in x , each element would be squared and the sum of square of all the elements will be stored in this variable f , and this is what is being written right. So, this is the same function which we have been using for the other two metaheuristic techniques right. So, this prob will help us to determine the fitness function value. As and when we want to determine a fitness function value, we will use this variable prob which in term will access this function spherenew right.

So, and then we need to define the parameters related to differential evolutions. So, here there are four parameters; one is the population size. Right now we have taken the population has to be 5, the number of iterations to be performed, so that we have taken it to be 100, the crossover probability to be 0.8 right and the scaling factor which would be required in

mutation as 0.85. So, these are the four parameters that we need to define with respect to differential evolution.

So, this section defines the problem definition; this section defines the values of the parameters required by the algorithm. So, this variable f , we are initializing it with NaN right. So, it is supposed to contain the fitness function values of the N_p population member. Similarly, we define this vector f_u which will contain the fitness function value of the newly generated trial vector. So, f will contain the fitness function value of the target vector, and f_u will contain the fitness function value of the trial vectors right.

So, initially we assign it NaN, as and when we find the fitness function we will appropriately assign those values. So, this is just pre initialization, as of now they do not contain the fitness function value, but we just create that vector with appropriate size. So, then to determine the number of random numbers that would be required, we need to know the number of decision variables, we get the length of lower bound, so that will tell us the number of decision variables and then we employ this variable U which is supposed to contain the trial vectors right.

So, trial vectors depending upon the size of the problem, the number of columns would vary right. So, the number of column would be equal to the number of decision variable, and the number of decision variable is given in this D right. And we will get as many trial vectors as the number of target vectors and the number of target vector is given by N_p right.

So, we will get here a matrix of N_p cross D dimension N_p rows D columns all the values should be NaN to begin with right. As and when we determine the trial vector we will save it in the corresponding row. So, this step is to generate the initial population. So, what we are doing is we are replicating the lower bound N_p times and we are replicating the range ub minus lb provides us the range right.

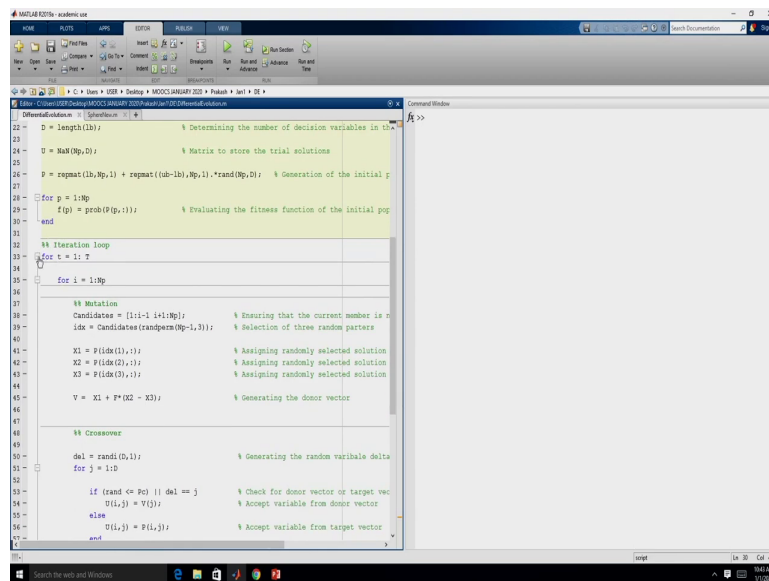
So, again that we are replicating N_p times right and then if you see this the dimension of this it will be N_p cross D because the number of rows is governed by this N_p and the number of columns is governed by the length of ub minus lb right. So, this will be N_p cross d . And we

generate N_p cross D random numbers between 0 to 1 and do an element wise multiplication with this range. So, this will help us to generate the initial population. So, the line 26 will help us to generate the initial population right, so that is similar to what we have done in teaching learning based optimization and particles from optimization right.

Line 28 to 30 is used to determine the fitness function value of each member right. So, we run this loop for p is equal to 1 to N_p and we access the solution p th solution. So, when we say p th solution, it is the p th row and all the columns that is why we give this colon operator right, so upper case p of p comma colon. So, the p th member in the population p will be send to this function spherenew through this variable prob, and it will return the fitness function value which will be stored in f of p right.

So, this loop will run N_p times, and we will be able to determine the fitness of all the individual members, so that would complete the initialization procedure. So, right now we have the initial population, the fitness function corresponding to each population member, and we have created appropriate variables for storing the trail vector and the fitness function for the trail vector right.

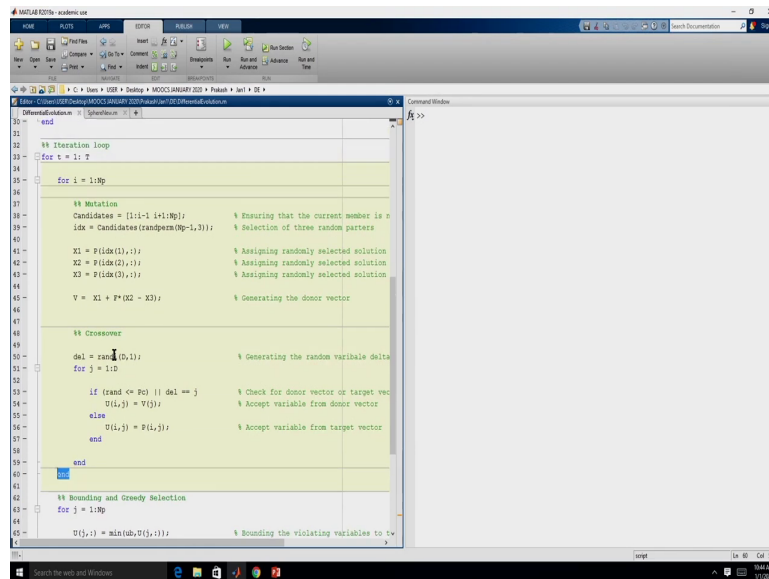
(Refer Slide Time: 07:51)



```
22 = D = length(lb); % Determining the number of decision variables in the
23
24 U = NaN(Np,D); % Matrix to store the trial solutions
25
26 P = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D); % Generation of the initial p
27
28 for p = 1:Np
29     f(p) = prob(f(p,:)); % Evaluating the fitness function of the initial pop
30 end
31
32 %% Iteration loop
33 for t = 1:T
34
35     for i = 1:Np
36
37         %% Mutation
38         Candidates = [i-1 i+1:Np]; % Ensuring that the current member is n
39         idx = Candidates(randperm(Np-1,3)); % Selection of three random partners
40
41         X1 = P(idx(1),:); % Assigning randomly selected solution
42         X2 = P(idx(2),:); % Assigning randomly selected solution
43         X3 = P(idx(3),:); % Assigning randomly selected solution
44
45         V = X1 + P*(X2 - X3); % Generating the donor vector
46
47
48         %% Crossover
49         del = randi(D,1); % Generating the random variable delta
50         for j = 1:D
51
52             if (rand <= Pc) || del == j % Check for donor vector or target vec
53                 U(i,j) = V(j); % Accept variable from donor vector
54             else
55                 U(i,j) = P(i,j); % Accept variable from target vector
56             end
57
58         end
59     end
60 end
```

So, now we start this iteration loop. So, line 33, we start the iteration loop and that will go on line 77, so that is the iteration loop. So, anything that is contained between line 33 and 77 is going to be repeated T times right because T capital T is the number of iterations that we want to perform. So, in each iteration for every member we are supposed to generate a donor vector and a trial vector right, so that is why we have this for loop anything between this line 35 and this line 60 will be repeated Np times.

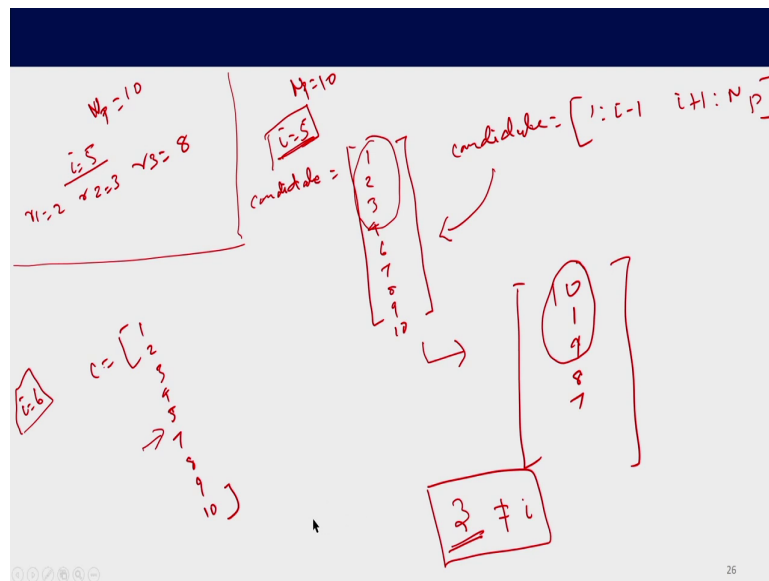
(Refer Slide Time: 08:25)



```
30 = end
31
32 %% Iteration loop
33 for t = 1: T
34     for i = 1:Np
35         %% Mutation
36         Candidates = [i-1 i+1:Np]; % Ensuring that the current member is not
37         idx = Candidates(randperm(Obj-1,3)); % Selection of three random partners
38         X1 = P(idx(1,:),:); % Assigning randomly selected solution
39         X2 = P(idx(2,:),:); % Assigning randomly selected solution
40         X3 = P(idx(3,:),:); % Assigning randomly selected solution
41         V = X1 + r*(X2 - X3); % Generating the donor vector
42
43         %% Crossover
44         del = rand(0,1); % Generating the random variable delta
45         for j = 1:D
46             if (rand <= Pc) || del == j % Check for donor vector or target vec
47                 U(i,j) = V(j); % Accept variable from donor vector
48             else
49                 U(i,j) = P(i,j); % Accept variable from target vector
50             end
51         end
52
53         %% Bounding and Greedy Selection
54         for j = 1:Np
55             U(j,:) = min(sub(0(j,:),:)); % Bounding the violating variables to tw
56         end
57     end
58 end
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
```

So, basically what we are doing is for each member, we will be performing a mutation and crossover right. So, now, we need to perform mutation. So, if you remember the equation for mutation required has to have three randomly selected solutions right. So, what we can do is generate three random numbers right between 1 and N_p , and we can check whether they are equal to i or not right.

(Refer Slide Time: 08:53)



Let us consider the population size to be let us say ten right. So and let us say we are working with the solution 5. So, we are supposed to select three numbers right which are not identical and they are not equal to 5. So, one way to do is that to randomly select three numbers, and check whether each of them are equal to one another or not, and also check whether it is equal to 5 or not. If it is equal, if it happens that lets say r_1 is we select to be 2, r_2 to be say let us say 3, r_3 to be let us say 8 right. So, we need to check whether any of this equal to 5 and all of them are unique right. If not we need to again go and generate random numbers, so that we that is one way to generate those three numbers.

But what we will be doing over here is if our population size is 10 right, and if we are working with let us say the fifth solutions. So, what we will do is we will generate something called as candidate right. So, the candidate now are all the solution except 5 right. So, it is 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. So, this we can generate using this command right candidate is equal to we can

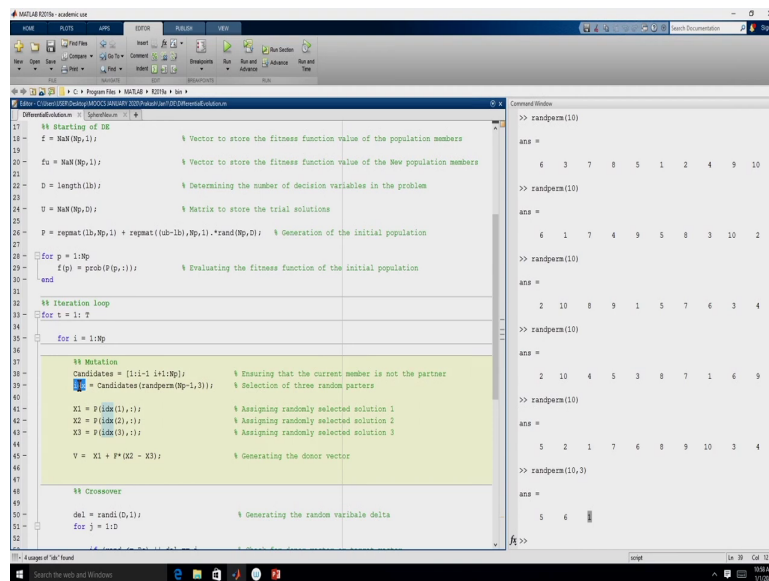
say 1 to $i - 1$ that will help us get these values and then $i + 1$ up to N_p right, so that will help us to generate this vector if i is equal to 5. If i happens to be 6, then the candidate vector would be 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 right. So, this way we can avoid the solution which we do not want. So, here we do not have 6.

So, once we have this right, so what we will do with this is we will randomly permuted this. So, when we randomly permuted let us say we get 10, 1, 9, 8, 7 and so on right. So, once we randomly permuted this vector, these first three numbers should be unique, because here if you see all the numbers are unique they do not contain the solution 5, which we need to avoid right. And all since all of them are unique I can randomly shuffle them and take the first three right; so that way we will be able to get three solutions which are unique and not equal to the i th solution right. So, this is the strategy that we will employ to generate the random solution.

So, this way we can always be sure that the three solutions which we are selecting are unique as well as they are not equal to i , and we do not need to employ if condition to check that. So, here we are selecting the candidates right. So, candidate is i to $i - 1$ right, and $i + 1$ to N_p . So, we have generated that vector which contains indexes of all the population right except for the i th member right.

So, this is that list of candidates. These are the potential candidates from which we can select the random solution right, remember these are not the entire solution vector, but just their index right. The solution vector them self are stored in p right, right now we are only identifying the random solutions. So, once we have these candidates, we can use this `randperm` function.

(Refer Slide Time: 12:13)



```

% Starting of DE
18- f = NaN(Np,1); % Vector to store the fitness function value of the population members
19- fu = NaN(Np,1); % Vector to store the fitness function value of the New population members
20-
21- D = length(lb); % Determining the number of decision variables in the problem
22-
23- U = NaN(Np,D); % Matrix to store the trial solutions
24-
25- P = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D); % Generation of the initial population
26-
27- for p = 1:Np
28-     f(p) = prob(F(p,:)); % Evaluating the fitness function of the initial population
29- end
30-
31- % Iteration loop
32- for t = 1:T
33-     for i = 1:Np
34-
35-         % Mutation
36-         Candidates = [1:i-1 i+1:Np]; % Ensuring that the current member is not the partner
37-         % Candidates(randperm(Np-1,3)); % Selection of three random partners
38-
39-         X1 = F(i&M(1,:),:); % Assigning randomly selected solution 1
40-         X2 = F(i&M(2,:),:); % Assigning randomly selected solution 2
41-         X3 = F(i&M(3,:),:); % Assigning randomly selected solution 3
42-
43-         V = X1 + P*(X2 - X3); % Generating the donor vector
44-
45-         % Crossover
46-
47-         del = randi(D,1); % Generating the random variable delta
48-         for j = 1:D
49-
50-
51-
52-

```

```

>> randperm(10)
ans =
     6     3     7     8     5     1     2     4     9     10
>> randperm(10)
ans =
     6     1     7     4     9     5     8     3     10     2
>> randperm(10)
ans =
     2     10     8     9     1     5     7     6     3     4
>> randperm(10)
ans =
     2     10     4     5     3     8     7     1     6     9
>> randperm(10)
ans =
     5     2     1     7     6     8     9     10     3     4
>> randperm(10,3)
ans =
     5     6

```

So, randperm function we can see over here. So, if I give randperm of let us say 10, it will permuted numbers 1 to 10. So, here if you see the numbers are between 1 to 10 right, and they are randomly arranged right. So, if you do this, as every time we do this we will have numbers one to ten randomly permuted right. So, this will help us to shuffle right. So, we will use this function to shuffle, and then we require only three solutions, we do not require all the 10 solutions we require only three solutions.

So, what we will do is randperm of 10 comma 3, if we do randperm of 10 comma 3, it will give us three solutions right which are unique right and they are taken from 1 to 10 right. So, this is how we can use the randperm function over here. So, here we say randperm Np minus 1 comma 3. So, this three should be clear, because we require three unique solutions right. This is Np minus 1 not Np because there are only Np minus 1 elements over here right.

So, if N_p is 10 candidates will have only 9 elements because we are eliminating the i th variable right. So, there are only 9 variables. If N_p is 10, there are only 9 variables. So, now, we are accessing those particular solution. So, when we do this candidates of `randperm(N_p - 1, 3)`, it will access candidate of 5, candidate of 6, and candidate of 1, those three values we are storing in this variable `idx` right. So, `idx` is going to contain the three randomly selected solutions, and these two lines will ensure right that these three solutions are unique, they are not equal to one and other and that they are also not equal to the i th variable.

So, this was the condition required in mutation. Remember we required three random solutions, we need to select three random solutions all the three solutions have to be unique and on top of that they should not be equal to the target solution right. So, target solution is indicated by i . So, it should not be i , we need three random solutions r_1, r_2, r_3 , these two lines will help us to implement that. When we run this in debug mode it will be much more clear.

So, now that we have identified the three random solutions right, we extract the solutions. So, we say x_1, x_2, x_3 is equal to p of index of 1 right because the fifth solution in is in position 1, if these three are the selected solutions then 5 is located at `idx` of 1, 6 is located at `idx` of 2, and 1 is located at `idx` of 3 right. So, we use that as the row, and we need to extract all the columns right we need to extract this from the population vector.

So, now we have selected the three solutions the actual three solutions `idx` contained only the index of which solution is to be selected right whereas, x_1, x_2, x_3 actually contain the solution because we are extracting it from the corresponding row from the population right. So, the dimension of $x \times [x_1, x_2, x_3]$ will be, it will be a row vector right, and the number of columns would be equal to the number of decision variables. So, once we have identified the three random solutions, we can generate the donor vector. So, this is straight forward V is equal to $x_1 + f \times (x_2 - x_3)$ right, f is already a parameter which is which is to be supplied by users. So, here we have taken f to be 0.85 right. So, this will help us to determine the donor vector.

(Refer Slide Time: 16:03)

Pseudocode of DE

Input: Fitness function, lb, ub, N_p , T, F

1. Initialize a random population (P)
2. Evaluate fitness (f) of P

```
for t = 1 to T
  for i = 1 to  $N_p$ 
    Generate the donor vector ( $V_i$ ) using mutation
    Perform crossover to generate offspring ( $U_i$ )
  end
  for i = 1 to  $N_p$ 
    Bound  $U_i$ 
    Evaluate the fitness ( $f_{U_i}$ ) of  $U_i$ 
    Perform greedy selection using  $f_{U_i}$  and  $f_i$  to update P
  end
end
```

$$V = X_{r_1} + F(X_{r_2} - X_{r_3})$$
$$u^j = \begin{cases} v^j & \text{if } r < p_c \text{ or } j = \delta \\ x^j & \text{if } r > p_c \text{ and } j \neq \delta \end{cases}$$
$$\left. \begin{matrix} X_i = U_i \\ f_i = f_{U_i} \end{matrix} \right\} \text{if } f_{U_i} < f_i$$

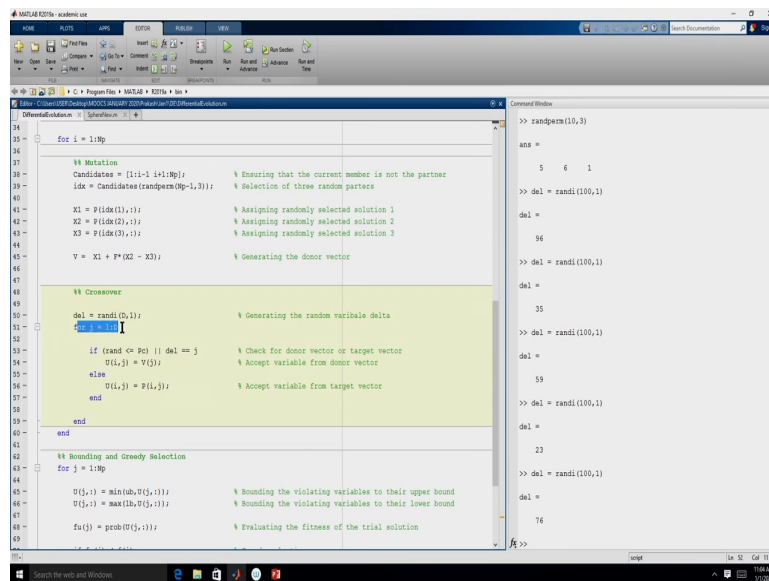
Handwritten notes: $\delta \in 1, D$

Navigation icons:

Page number: 24

So, in order to generate the trial vector right, the trial vector will contain the value of either the donor vector or the target vector right. So, to do that we will require this random number r and we also require this randomly selected number δ right. So, δ is between one and decision variable number of decision variables. So, it is a random integer which has to be between 1 and decision variable. So, we need to first generate δ right, and then we need to generate random number for each of the variable..

(Refer Slide Time: 16:35)



```
35- for i = 1:Np
36-
37-     %% Mutation
38-     Candidates = [i:-1 i+1:Np]; % Ensuring that the current member is not the partner
39-     idx = randperm(2:length(Candidates),3); % Selection of three random partners
40-
41-     X1 = P(idx(1),:); % Assigning randomly selected solution 1
42-     X2 = P(idx(2),:); % Assigning randomly selected solution 2
43-     X3 = P(idx(3),:); % Assigning randomly selected solution 3
44-
45-     V = X1 + r*(X2 - X3); % Generating the donor vector
46-
47-
48-     %% Crossover
49-
50-     del = randi(D,1); % Generating the random variable delta
51-     % del = 100;
52-
53-     if (rand <= cr) || del == j % Check for donor vector or target vector
54-         U(i,j) = V(i,j); % Accept variable from donor vector
55-     else
56-         U(i,j) = P(i,j); % Accept variable from target vector
57-     end
58-
59- end
60-
61-
62- %% Bounding and Greedy Selection
63- for j = 1:Np
64-
65-     U(j,:) = min(sub,U(j,:)); % Bounding the violating variables to their upper bound
66-     U(j,:) = max(lb,U(j,:)); % Bounding the violating variables to their lower bound
67-
68-     fu(j) = prob(U(j,:)); % Evaluating the fitness of the trial solution
69-
```

And line 50, what we are doing is del is equal to randi of D comma 1. So, randi you have seen randi the use of randi in tlbo right for generating teaching factor we had used right. So, when we say rand of D comma 1, it randomly generates one integer value from 1 to D right. And since this is 1, it returns a scalar value right. Since we are starting from 1, we do not need to give 1 to d, it is sufficient to give just D right, and we require only one value, so that is why this second one is given. So, you can quickly do this over here and see what happens, so when we said del is equal to randi of 100 comma 1. So, it will give us a random integer right one value which is from 1 to 100 right, so that is how we are generating the del value right.

So, once we have generated the del value we need to run a loop for all the decision variables. Remember that equation which we saw is for all the variables, the trail vector will have a dimension of 1 cross D right, we just need to choose whether the jth value will come from the target vector or from the donor vectors. So, we run this for loop over here for j is equal to 1

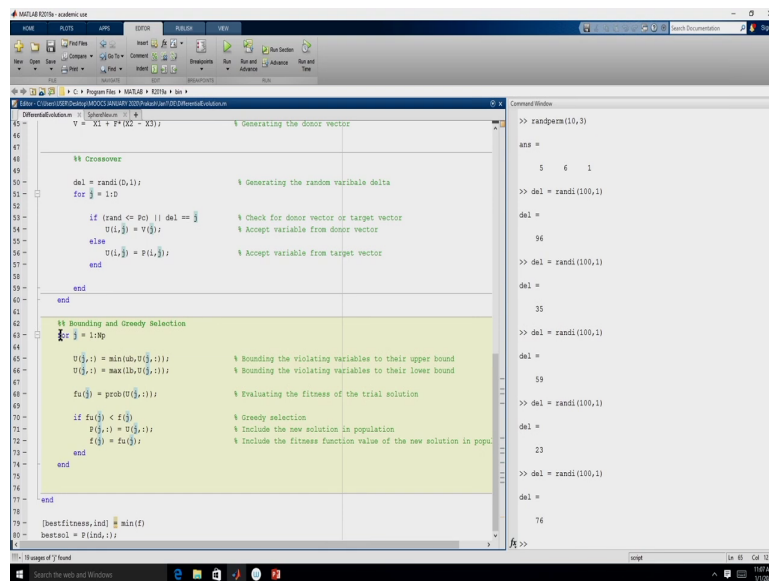
to d , we generate a random number using this function `rand` right. So, we check for this condition. If `rand` is less than or equal to `pc` right or if it is equal to `del` right, if `j` is equal to `del`, if that condition satisfied we need to take the value of the trail vector right from the donor vector right; otherwise we need to take the value of the trial vector from the target vector. So, target vector is stored in `p` trail vector is stored in `U` and our donor vector is in `V` right.

So, here we need to see that this is a double equal to sign that is a conditional check right. If either this condition is satisfied or this condition is satisfied, it will assign it from the donor vector, else it will assign it from the target vector right. So, this for loop will run for D times, because we have D decision variables right. And this for loop is inside this external for loop for every member. So, for every member that loop will run for D times because, we have D decision variable, so that would complete the crossover operation right. So, at the end of line 60, we would have completed for all the population number crossover as well as mutation, and the trial vectors are stored in `U` the target vectors are in `p`.

(Refer Slide Time: 19:15)

```
40
41 X1 = P(iDk(1,:),:); % Assigning randomly selected solution 1
42 X2 = P(iDk(2,:),:); % Assigning randomly selected solution 2
43 X3 = P(iDk(3,:),:); % Assigning randomly selected solution 3
44
45 D = X1 + r*(X2 - X3); % Generating the donor vector
46
47
48 %% Crossover
49 del = randi(Dv,1); % Generating the random variable delta
50 for j = 1:D
51
52     if (rand <= Pc) || del == j % Check for donor vector or target vector
53         U(i,j) = D(i,j); % Accept variable from donor vector
54     else
55         U(i,j) = P(i,j); % Accept variable from target vector
56     end
57 end
58 I
59 end
60 end
61
62 %% Bounding and Greedy Selection
63 for j = 1:Np
64
65     U(j,:) = min(sub,U(j,:),:); % Bounding the violating variables to their upper bound
66     U(j,:) = max(sub,U(j,:),:); % Bounding the violating variables to their lower bound
67     fu(j) = prob(U(j,:),:); % Evaluating the fitness of the trial solution
68
69     if fu(j) < f(j) % Greedy selection
70         P(j,:) = U(j,:); % Include the new solution in population
71         f(j) = fu(j); % Include the fitness function value of the new solution in pops
72     end
73 end
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
```

(Refer Slide Time: 20:09)



```
45= V = Xi + P*(X2 - X1); % Generating the donor vector
46
47
48
49 %% Crossover
50 del = randi(0,1); % Generating the random variable delta
51 for j = 1:Np
52
53     if (rand <= Pc) || del == j % Check for donor vector or target vector
54         U(i,j) = V(j); % Accept variable from donor vector
55     else
56         U(i,j) = P(i,j); % Accept variable from target vector
57     end
58 end
59 end
60
61 %% Bounding and Greedy Selection
62 for j = 1:Np
63
64     U(j,:) = min(ub,U(j,:)); % Bounding the violating variables to their upper bound
65     U(j,:) = max(lb,U(j,:)); % Bounding the violating variables to their lower bound
66     fu(j) = prob(U(j,:)); % Evaluating the fitness of the trial solution
67
68     if fu(j) < f(j) % Greedy selection
69         P(j,:) = U(j,:); % Include the new solution in population
70         f(j) = fu(j); % Include the fitness function value of the new solution in popu
71     end
72 end
73
74
75
76
77 end
78
79 [bestfitness,ind] = min(f)
80 bestsol = P(ind,:);
```

Command Window

```
>> randperm(10,3)
ans =
     5     6     1
>> del = randi(100,1)
del =
    96
>> del = randi(100,1)
del =
    35
>> del = randi(100,1)
del =
    59
>> del = randi(100,1)
del =
    23
>> del = randi(100,1)
del =
    76
fj >>
```

And then here we again run a loop for j is equal to 1 to Np, and then we bound the variables right. So, this bounding of the variables we have seen previously that if we use a min operator of ub comma the actual solution. So, the actual solution is jth row all columns. So, jth row all columns will give us the jth solution right. So, we are using the min operator. So, if in line 65 if any variable in the jth row of U violates the upper bound, it will be brought back to the upper bound right.

Similarly, line 66 will ensure that the trial vector U of j comma colon if any of those values violate the lower bound, it is brought back into the lower bound using this max operator. And if the values do not violate the lower and upper bound, it will be retained as such. So, we had seen sixty five and sixty six in detail when we learnt teaching-learning based optimization right.

So, now this trial vector is bounded right. So, what we will do is we will evaluate the fitness of it. To evaluate the fitness of it, we need the fitness function that fitness function we have stored in the variable prob. So, prob is a function handle where in we are sending the jth row of U to this prob function will be able to determine the fitness function of the jth trial vector right. Once we have the fitness function of the jth trial vector, we are ready to perform a greedy selection. So, we are checking whether the fitness of the jth trial vector is better than the fitness of the jth target vector.

So, if this condition is satisfied, we overwrite the jth member of the population with the jth member of the trail solution right. And similarly we replace the fitness function value right. So, this is how the greedy selection is performed right. So, since it is inside this particular for loop right, it will be done for N_p times where in we are comparing the first trial with first target, second trial with second target, third trail with third target and depending upon whichever is better will be taken into the population.

So, if the target is actually better, then we do not need to do anything that is why we do not have an else part to this if loop that particular solution will be retained right; but it happens the trail vector is better, then the trail vector is used to overwrite the jth member of the population right. So, this will again be done for N_p times because our population is N_p , and this entire procedure is inside this for loop which governs the iterations the number of iterations. So, this will be repeated for t times right, so that completes the implementation of differential evolution.

Just like in tlbo at the end of it we are interested in what is the best solution obtained so far right, so that can be determined by determining the minimum of the fitness function right using this min function. So, this min function will give us what is the minimum value that we are storing in this variable best fitness, and it will also give us the location as to where it is located right. So, using this ind which is the location of the best fitness function value, we extract the corresponding population member right. So, let us say if ind is 5, then we are extracting the fifth row of the population, and we are assigning it to bestsol, so that way we will be able to see what is the best fitness and what is the best solution at the end of t iterations right.

(Refer Slide Time: 23:57)

```
1 clear
2 % To clear the current script
3
4 %% Problem settings
5 lb = [0 0 0 0] % Lower bound
6 ub = [10 10 10 10] % Upper bound
7 prob = @spheresnew % Fitness function
8
9
10 %% Parameters for Differential Evolution
11 Np = 5 % Population size
12 T = 10 % No. of iterations
13 Pc = 0.8 % Crossover probability
14 F = 0.85 % Scaling factor
15
16
17 %% Starting of DE
18 f = @spheresnew % Vector to store the fitness function value of the population members
19
20 % f = @spheresnew % Vector to store the fitness function value of the new population members
21
22 D = length(lb) % Determining the number of decision variables in the problem
23
24 P = rand(Np,D) % Matrix to store the trial solutions
25
26 P = repmat(lb,Np,1) + repmat((ub-lb)/Np,1,1)*rand(Np,D) % Generation of the initial population
27
28 for p = 1:Np
29     f(p) = prob(f(p,:)) % Evaluating the fitness function of the initial population
30 end
31
32 %% Iteration loop
33 for t = 1:T
34     for i = 1:Np
35
36         %% Mutation
37         Candidate = lb + (ub-lb)*randi(1,1,1) % Ensuring that the current member is not the partner
38         C = randi(1,1,1) % Selection of three random partners
39
40         R1 = P(C(1),:) % Assigning randomly selected solution 1
41         R2 = P(C(2),:) % Assigning randomly selected solution 2
42         R3 = P(C(3),:) % Assigning randomly selected solution 3
```

Let us now run this code in the debug mode right, so let me put a breakpoint over here and if we execute this code right. So, the first step is `clc`. So, as you know that it will clear the screen, the second one is it will clear the workspace right. So, the lower bounds are defined the upper bounds are defined, and now we are defining the function handle right. So, `prob` will be defined. So, now, `prob` is a function handle. So, we are solving this objective function `spheresnew` right.

So, now we need to define the parameters right. So, population size is 5, number of iterations is 10. So, the crossover probability is being defined as 0.8, the scaling factors defined as 0.85 right. So, now, we are defining variable `f` which will have NaN values as many NaN values as the population size right. So, here the population size is 5. So, we are having 5 NaN values

right. So, the next step is to find out the length of the lower bound length of the lower bound is 5 in this case right, and then we generate the initial population as usual right.

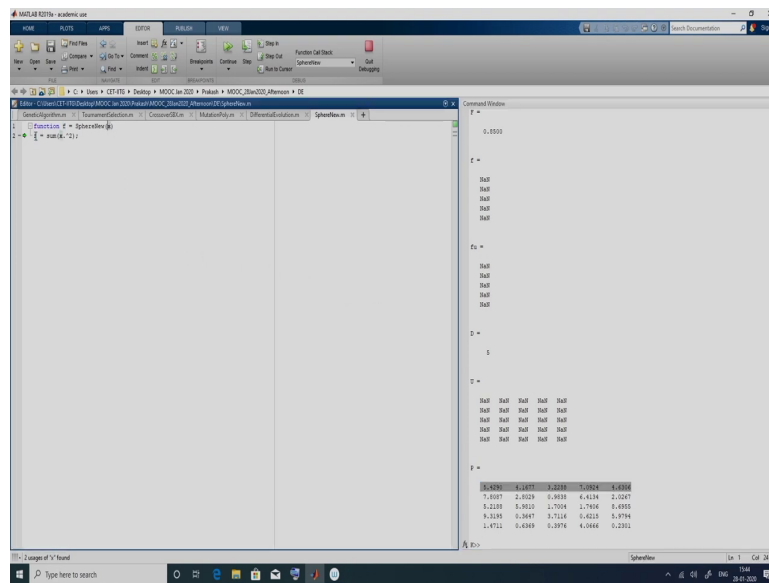
(Refer Slide Time: 24:59)

```

1 clear
2 % To clear the workspace
3
4 %% Problem settings
5 lb = [0 0 0 0]; % Lower bound
6 ub = [10 10 10 10]; % Upper bound
7 prob = @SphereEval; % Fitness function
8
9
10 %% Parameters for Differential Evolution
11 Np = 5; % Population size
12 T = 100; % No. of iterations
13 Pc = 0.5; % crossover probability
14 F = 0.35; % Scaling factor
15
16
17 %% Starting of DE
18 F = NaN(Np,1); % Vector to store the fitness function value of the population members
19 fu = NaN(Np,1); % Vector to store the fitness function value of the new population members
20
21 D = length(lb); % Determining the number of decision variables in the problem
22 U = NaN(Np,D); % Matrix to store the trial solutions
23
24 F = repmat(lb,Np,1) + repmat((ub-lb)/Np,1,1)*rand(Np,1); % Generation of the initial population
25
26 for l = 1:Np
27     F(l) = prob(F(l)); % Evaluating the fitness function of the initial population
28 end
29
30 %% Iteration loop
31 for t = 1:T
32     for l = 1:Np
33
34         %% Mutation
35         Candidate = (F(l)+1+100); % Ensuring that the crossover member is not the parent;
36         use = randi(candidatesize(Np,1),1); % Selection of three random parents
37
38         %%% = F1(x1,1); % Assigning randomly selected solution 1
39         %%% = F1(x2,1); % Assigning randomly selected solution 2
40         %%% = F1(x3,1); % Assigning randomly selected solution 3
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
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

```


(Refer Slide Time: 25:35)



So, it will determine the fitness function value and that is being returned back right.

(Refer Slide Time: 25:56)

```
clear % To clear the workspace
%% Problem settings
lb = [0 0 0 0]; % Lower bound
ub = [10 10 10 10]; % Upper bound
prob = @SphereEval; % Fitness function
%% Parameters for Differential Evolution
Np = 5; % Population size
T = 100; % No. of iterations
F = 0.5; % Crossover probability
CR = 0.35; % Mutation factor
%% Starting of DE
F = NaN(Np,1); % Vector to store the fitness function value of the population members
E = NaN(Np,1); % Vector to store the fitness function value of the new population members
D = length(lb); % Determining the number of decision variables in the problem
D = NaN(D,1); % Matrix to store the trial solutions
P = repmat(lb, Np, 1) + repmat((ub-lb)/Np, 1, Np); % Generation of the initial population
for p = 1:Np
    F(p) = prob(F(p)); % Evaluating the fitness function of the initial population
end
%% Iteration loop
for t = 1:T
    for i = 1:Np
        %% Mutation
        Candidate = lb + (ub-lb)*randi(1,1,D); % Ensuring that the current member is not the partner
        rand = randi(1,1,D); % Selection of these random partners
        r1 = randi(1,1,D); % Assigning randomly selected solution 1
        r2 = randi(1,1,D); % Assigning randomly selected solution 2
        r3 = randi(1,1,D); % Assigning randomly selected solution 3
    end
end
```

So, here if you see the first the fitness function of the first solution has been determine to be 129.0137. So, similarly if we keep doing this right, so it will determine the fitness function of all the five solutions. So, now, we have the initial population generated in this line 26, and we have evaluated the fitness function value over here right. So, now we have done that we can begin the iteration loop of differential evolution right.

(Refer Slide Time: 26:47)

```
13 = Np = 10; % crossover probability
14 = F = 0.15; % Stopping factor
15
16
17 % Starting of GA
18 = F = rand(Np,1); % Vector to store the fitness function value of the population members
19 = Fx = rand(Np,1); % Vector to store the fitness function value of the New population members
20
21 = D = length(D); % Determining the number of decision variables in the problem
22
23 = U = rand(Np,D); % Matrix to store the trial solutions
24
25 = P = repmat([Np,1] + repmat([Np,1]*rand(Np,2))); % Generation of the initial population
26
27 for p = 1:Np
28 = F(p) = prob(F(p,1)); % Evaluating the fitness function of the initial population
29
30 end
31
32 % Iteration loop
33 = iter = 1; % iter = 1; % iter = 1;
34
35 = for i = 1:Np
36
37 % Mutation
38 = Candidates = [1:i-1 i+1:Np]; % Ensuring that the current member is not the partner
39 = i = randperm(Candidates,2); % Selection of three random partners
40
41 = i1 = F(i1,1); % Assigning randomly selected solution 1
42 = i2 = F(i2,1); % Assigning randomly selected solution 2
43 = i3 = F(i3,1); % Assigning randomly selected solution 3
44
45 = V = i1 + P*(i2 - i3); % Generating the donor vector
46
47
48 % Crossover
49 = randi(D,1); % Generating the random variable index
50 = for j = 1:D
51
52 = if (rand <= P) || (i1 == j) % Check for donor vector or target vector
53 = U(i1,j) = V(j); % Accept variable from donor vector
54 =
55 = else
56 = U(i1,j) = F(i1,j); % Accept variable from target vector
57
58 end
59
60 end
61
62 % Selection
63 = Fx = rand(Np,1); % Vector to store the fitness function value of the New population members
64 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
65 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
66 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
67 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
68 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
69 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
70 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
71 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
72 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
73 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
74 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
75 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
76 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
77 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
78 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
79 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
80 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
81 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
82 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
83 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
84 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
85 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
86 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
87 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
88 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
89 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
90 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
91 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
92 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
93 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
94 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
95 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
96 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
97 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
98 = Fx = Fx + P*(F(i2,1) - F(i2,1)); % Evaluating the fitness function of the New population members
99 = Fx = Fx + P*(F(i3,1) - F(i3,1)); % Evaluating the fitness function of the New population members
100 = Fx = Fx + P*(F(i1,1) - F(i1,1)); % Evaluating the fitness function of the New population members
```

So, if you step in, so t is 1, i is also 1. So, now, we will determine who are the possible candidates, who can be partner in the mutation phase right. So, we are creating this vector 1 to i minus 1. So, in this case it will be 1 to 0, so that will not return as any value and this will be 2 to N_p . So, if we do this step, so here if you see candidates are 2, 3, 4, 5. Now, we have eliminated the i th solution from the list of candidate solution right, so because this for the first solution it cannot be partner. So, we are going to select partner from this four solutions 2, 3, 4, 5.

Now, what we are doing is a permutation randperm from 1 to 4. So, this will be 4 N_p minus 1 this value would be 4, and we are taking three randomly permuted values. So, here in this case 4, 3, 5. So, now, if you see this satisfies our purpose that we are able to select three

(Refer Slide Time: 27:38)

```
13 = % c = 0.3 % crossover probability
14 = % p = 0.15 % scaling factor
15
16
17 % Starting of DE
18 % f = fitness(Pop,1) % Vector to store the fitness function value of the population members
19 % f = fitness(Pop,1) % Vector to store the fitness function value of the New population members
20
21 D = length(x0) % Determining the number of decision variables in the problem
22
23 U = rand(10, D) % Matrix to store the trial solutions
24
25 P = repmat(x0, 10, 1) + repmat((ub-lb)/D, 10, 1).*rand(D, 1) % Generation of the initial population
26
27 for i = 1:10
28     f(i) = prob(F, P, i) % Evaluating the fitness function of the initial population
29 end
30
31 % Iteration loop
32 for i = 1:17
33     for k = 1:10
34
35         % Mutation
36         Candidates = l(i)-1 : i+1:10; % Ensuring that the current member is not the partner
37         idx = Candidates(randperm(10, 3, 1)); % Selection of three random partners
38
39         % Assigning randomly selected solution 1
40         X1 = P(idx(1), :);
41         % Assigning randomly selected solution 2
42         X2 = P(idx(2), :);
43         % Assigning randomly selected solution 3
44         X3 = P(idx(3), :);
45         V = X1 + r.*X2 - X3; % Generating the donor vector
46
47         % Crossover
48         % Generating the random variable delta
49         for j = 1:D
50             if (rand < cr || del == j) % Check for donor vector or target vector
51                 V(j, :) = P(k, :); % Accept variable from donor vector
52             else
53                 V(j, :) = P(k, :); % Accept variable from target vector
54             end
55         end
56     end
57 end
58
59 % fitness of V found
```

Command Window:

```
f =
129.0137
RMS
RMS
RMS
f =
129.0137
115.0393
144.5415
136.9019
RMS
RMS
f =
129.0137
115.0393
144.5415
136.9019
19.3189
Candidates =
```

Idx is 4 3 5. So, idx of 1 is 4, so x 1 will be the fourth solution right.

(Refer Slide Time: 27:45)

```
13 = Np = 10; % crossover probability
14 = Pc = 0.15; % Stopping factor
15
16
17 % Starting of GA
18 = f = rand(Np,1); % Vector to store the fitness function value of the population members
19 = f; % rand(Np,1) % Vector to store the fitness function value of the New population members
20
21 = D = length(x); % Determining the number of decision variables in the problem
22
23 = D; % rand(D,1) % Matrix to store the trial solutions
24 = repmat(x(Np,1) + repmat((rand(1,D))-0.5)*rand(D), Np, 1); % Generation of the initial population
25
26 = f; % prob(f,1); % Evaluating the fitness function of the initial population
27
28 = end
29
30 % Iteration loop
31 = for i = 1:T
32
33 = % Crossover
34
35 = % Mutation
36 = Candidates = [1:i-1; i+1:Np];
37 = i; % Selection of three random partners
38
39 = % Assigning randomly selected solution 1
40 = % Assigning randomly selected solution 2
41 = % Assigning randomly selected solution 3
42 = % Generating the donor vector
43
44 = % Crossover
45 = % Generating the random variable delta
46 = % Check for donor vector or target vector
47 = % Accept variable from donor vector
48 = % Accept variable from target vector
49
50 = % Stopping of GA found
```

Command Window

```
f =
129.0137
115.0393
184.9415
136.9019
15.0393

Candidates =
2 3 4 5

delta =
4 3 5

x1 =
0.1195 0.1047 3.7114 0.4210 5.9794
```

So, let us just see that. So, x 1 is this one right.

(Refer Slide Time: 27:49)

The image shows a MATLAB Editor window with a script for a Genetic Algorithm. The script includes initialization, iteration loops for selection, crossover, and mutation, and a final output window. The output window displays the fitness values of the population over iterations.

```
13: % p = 0.1 % crossover probability
14: % r = 0.15 % scaling factor
15:
16:
17: % Starting of GA
18: % f = rand(10,1) % Vector to store the fitness function value of the population members
19:
20: % f = rand(10,1) % Vector to store the fitness function value of the New population members
21:
22: % D = length(x) % Determining the number of decision variables in the problem
23:
24: % D = rand(10,D) % Matrix to store the trial solutions
25:
26: % P = repmat(10,1) + repmat((rand(10,1))*rand(10,D)) % Generation of the initial population
27:
28: % for p = 1:100
29: % f(p) = prob(f(p,1)) % Evaluating the fitness function of the initial population
30:
31:
32: % Iteration loop
33: % for i = 1:100
34:
35: % for j = 1:100
36:
37: % Mutation
38: % Candidates = 1:i-1+1:100] % Ensuring that the current member is not the partner
39: % i = randperm(100,3) % Selection of three random partners
40:
41: % i1 = f(i1,1) % Assigning randomly selected solution 1
42: % i2 = f(i2,1) % Assigning randomly selected solution 2
43: % i3 = f(i3,1) % Assigning randomly selected solution 3
44:
45: % V = i1 + P*(i2 - i3) % Generating the donor vector
46:
47:
48: % Crossover
49: % i = randi(10) % Generating the random variable index
50: % for j = 1:D
51:
52: % if (rand <= P) || mod(j,2) == 1 % Check for donor vector or target vector
53: % V(i,j) = V(i,2) % Accept variable from donor vector
54:
55: % else
56: % V(i,j) = f(i,2) % Accept variable from target vector
57:
58:
59: % end
60:
61: % end
62:
63: % end
64:
65: % end
66:
67: % end
68:
69: % end
70:
71: % end
72:
73: % end
74:
75: % end
76:
77: % end
78:
79: % end
80:
81: % end
82:
83: % end
84:
85: % end
86:
87: % end
88:
89: % end
90:
91: % end
92:
93: % end
94:
95: % end
96:
97: % end
98:
99: % end
100:
101: % end
102:
103: % end
104:
105: % end
106:
107: % end
108:
109: % end
110:
111: % end
112:
113: % end
114:
115: % end
116:
117: % end
118:
119: % end
120:
121: % end
122:
123: % end
124:
125: % end
126:
127: % end
128:
129: % end
130:
131: % end
132:
133: % end
134:
135: % end
136:
137: % end
138:
139: % end
140:
141: % end
142:
143: % end
144:
145: % end
146:
147: % end
148:
149: % end
150:
151: % end
152:
153: % end
154:
155: % end
156:
157: % end
158:
159: % end
160:
161: % end
162:
163: % end
164:
165: % end
166:
167: % end
168:
169: % end
170:
171: % end
172:
173: % end
174:
175: % end
176:
177: % end
178:
179: % end
180:
181: % end
182:
183: % end
184:
185: % end
186:
187: % end
188:
189: % end
190:
191: % end
192:
193: % end
194:
195: % end
196:
197: % end
198:
199: % end
200:
201: % end
202:
203: % end
204:
205: % end
206:
207: % end
208:
209: % end
210:
211: % end
212:
213: % end
214:
215: % end
216:
217: % end
218:
219: % end
220:
221: % end
222:
223: % end
224:
225: % end
226:
227: % end
228:
229: % end
230:
231: % end
232:
233: % end
234:
235: % end
236:
237: % end
238:
239: % end
240:
241: % end
242:
243: % end
244:
245: % end
246:
247: % end
248:
249: % end
250:
251: % end
252:
253: % end
254:
255: % end
256:
257: % end
258:
259: % end
260:
261: % end
262:
263: % end
264:
265: % end
266:
267: % end
268:
269: % end
270:
271: % end
272:
273: % end
274:
275: % end
276:
277: % end
278:
279: % end
280:
281: % end
282:
283: % end
284:
285: % end
286:
287: % end
288:
289: % end
290:
291: % end
292:
293: % end
294:
295: % end
296:
297: % end
298:
299: % end
300:
301: % end
302:
303: % end
304:
305: % end
306:
307: % end
308:
309: % end
310:
311: % end
312:
313: % end
314:
315: % end
316:
317: % end
318:
319: % end
320:
321: % end
322:
323: % end
324:
325: % end
326:
327: % end
328:
329: % end
330:
331: % end
332:
333: % end
334:
335: % end
336:
337: % end
338:
339: % end
340:
341: % end
342:
343: % end
344:
345: % end
346:
347: % end
348:
349: % end
350:
351: % end
352:
353: % end
354:
355: % end
356:
357: % end
358:
359: % end
360:
361: % end
362:
363: % end
364:
365: % end
366:
367: % end
368:
369: % end
370:
371: % end
372:
373: % end
374:
375: % end
376:
377: % end
378:
379: % end
380:
381: % end
382:
383: % end
384:
385: % end
386:
387: % end
388:
389: % end
390:
391: % end
392:
393: % end
394:
395: % end
396:
397: % end
398:
399: % end
400:
401: % end
402:
403: % end
404:
405: % end
406:
407: % end
408:
409: % end
410:
411: % end
412:
413: % end
414:
415: % end
416:
417: % end
418:
419: % end
420:
421: % end
422:
423: % end
424:
425: % end
426:
427: % end
428:
429: % end
430:
431: % end
432:
433: % end
434:
435: % end
436:
437: % end
438:
439: % end
440:
441: % end
442:
443: % end
444:
445: % end
446:
447: % end
448:
449: % end
450:
451: % end
452:
453: % end
454:
455: % end
456:
457: % end
458:
459: % end
460:
461: % end
462:
463: % end
464:
465: % end
466:
467: % end
468:
469: % end
470:
471: % end
472:
473: % end
474:
475: % end
476:
477: % end
478:
479: % end
480:
481: % end
482:
483: % end
484:
485: % end
486:
487: % end
488:
489: % end
490:
491: % end
492:
493: % end
494:
495: % end
496:
497: % end
498:
499: % end
500:
501: % end
502:
503: % end
504:
505: % end
506:
507: % end
508:
509: % end
510:
511: % end
512:
513: % end
514:
515: % end
516:
517: % end
518:
519: % end
520:
521: % end
522:
523: % end
524:
525: % end
526:
527: % end
528:
529: % end
530:
531: % end
532:
533: % end
534:
535: % end
536:
537: % end
538:
539: % end
540:
541: % end
542:
543: % end
544:
545: % end
546:
547: % end
548:
549: % end
550:
551: % end
552:
553: % end
554:
555: % end
556:
557: % end
558:
559: % end
560:
561: % end
562:
563: % end
564:
565: % end
566:
567: % end
568:
569: % end
570:
571: % end
572:
573: % end
574:
575: % end
576:
577: % end
578:
579: % end
580:
581: % end
582:
583: % end
584:
585: % end
586:
587: % end
588:
589: % end
590:
591: % end
592:
593: % end
594:
595: % end
596:
597: % end
598:
599: % end
600:
601: % end
602:
603: % end
604:
605: % end
606:
607: % end
608:
609: % end
610:
611: % end
612:
613: % end
614:
615: % end
616:
617: % end
618:
619: % end
620:
621: % end
622:
623: % end
624:
625: % end
626:
627: % end
628:
629: % end
630:
631: % end
632:
633: % end
634:
635: % end
636:
637: % end
638:
639: % end
640:
641: % end
642:
643: % end
644:
645: % end
646:
647: % end
648:
649: % end
650:
651: % end
652:
653: % end
654:
655: % end
656:
657: % end
658:
659: % end
660:
661: % end
662:
663: % end
664:
665: % end
666:
667: % end
668:
669: % end
670:
671: % end
672:
673: % end
674:
675: % end
676:
677: % end
678:
679: % end
680:
681: % end
682:
683: % end
684:
685: % end
686:
687: % end
688:
689: % end
690:
691: % end
692:
693: % end
694:
695: % end
696:
697: % end
698:
699: % end
700:
701: % end
702:
703: % end
704:
705: % end
706:
707: % end
708:
709: % end
710:
711: % end
712:
713: % end
714:
715: % end
716:
717: % end
718:
719: % end
720:
721: % end
722:
723: % end
724:
725: % end
726:
727: % end
728:
729: % end
730:
731: % end
732:
733: % end
734:
735: % end
736:
737: % end
738:
739: % end
740:
741: % end
742:
743: % end
744:
745: % end
746:
747: % end
748:
749: % end
750:
751: % end
752:
753: % end
754:
755: % end
756:
757: % end
758:
759: % end
760:
761: % end
762:
763: % end
764:
765: % end
766:
767: % end
768:
769: % end
770:
771: % end
772:
773: % end
774:
775: % end
776:
777: % end
778:
779: % end
780:
781: % end
782:
783: % end
784:
785: % end
786:
787: % end
788:
789: % end
790:
791: % end
792:
793: % end
794:
795: % end
796:
797: % end
798:
799: % end
800:
801: % end
802:
803: % end
804:
805: % end
806:
807: % end
808:
809: % end
810:
811: % end
812:
813: % end
814:
815: % end
816:
817: % end
818:
819: % end
820:
821: % end
822:
823: % end
824:
825: % end
826:
827: % end
828:
829: % end
830:
831: % end
832:
833: % end
834:
835: % end
836:
837: % end
838:
839: % end
840:
841: % end
842:
843: % end
844:
845: % end
846:
847: % end
848:
849: % end
850:
851: % end
852:
853: % end
854:
855: % end
856:
857: % end
858:
859: % end
860:
861: % end
862:
863: % end
864:
865: % end
866:
867: % end
868:
869: % end
870:
871: % end
872:
873: % end
874:
875: % end
876:
877: % end
878:
879: % end
880:
881: % end
882:
883: % end
884:
885: % end
886:
887: % end
888:
889: % end
890:
891: % end
892:
893: % end
894:
895: % end
896:
897: % end
898:
899: % end
900:
901: % end
902:
903: % end
904:
905: % end
906:
907: % end
908:
909: % end
910:
911: % end
912:
913: % end
914:
915: % end
916:
917: % end
918:
919: % end
920:
921: % end
922:
923: % end
924:
925: % end
926:
927: % end
928:
929: % end
930:
931: % end
932:
933: % end
934:
935: % end
936:
937: % end
938:
939: % end
940:
941: % end
942:
943: % end
944:
945: % end
946:
947: % end
948:
949: % end
950:
951: % end
952:
953: % end
954:
955: % end
956:
957: % end
958:
959: % end
960:
961: % end
962:
963: % end
964:
965: % end
966:
967: % end
968:
969: % end
970:
971: % end
972:
973: % end
974:
975: % end
976:
977: % end
978:
979: % end
980:
981: % end
982:
983: % end
984:
985: % end
986:
987: % end
988:
989: % end
990:
991: % end
992:
993: % end
994:
995: % end
996:
997: % end
998:
999: % end
1000:
1001: % end
1002:
1003: % end
1004:
1005: % end
1006:
1007: % end
1008:
1009: % end
1010:
1011: % end
1012:
1013: % end
1014:
1015: % end
1016:
1017: % end
1018:
1019: % end
1020:
1021: % end
1022:
1023: % end
1024:
1025: % end
1026:
1027: % end
1028:
1029: % end
1030:
1031: % end
1032:
1033: % end
1034:
1035: % end
1036:
1037: % end
1038:
1039: % end
1040:
1041: % end
1042:
1043: % end
1044:
1045: % end
1046:
1047: % end
1048:
1049: % end
1050:
1051: % end
1052:
1053: % end
1054:
1055: % end
1056:
1057: % end
1058:
1059: % end
1060:
1061: % end
1062:
1063: % end
1064:
1065: % end
1066:
1067: % end
1068:
1069: % end
1070:
1071: % end
1072:
1073: % end
1074:
1075: % end
1076:
1077: % end
1078:
1079: % end
1080:
1081: % end
1082:
1083: % end
1084:
1085: % end
1086:
1087: % end
1088:
1089: % end
1090:
1091: % end
1092:
1093: % end
1094:
1095: % end
1096:
1097: % end
1098:
1099: % end
1100:
1101: % end
1102:
1103: % end
1104:
1105: % end
1106:
1107: % end
1108:
1109: % end
1110:
1111: % end
1112:
1113: % end
1114:
1115: % end
1116:
1117: % end
1118:
1119: % end
1120:
1121: % end
1122:
1123: % end
1124:
1125: % end
1126:
1127: % end
1128:
1129: % end
1130:
1131: % end
1132:
1133: % end
1134:
1135: % end
1136:
1137: % end
1138:
1139: % end
1140:
1141: % end
1142:
1143: % end
1144:
1145: % end
1146:
1147: % end
1148:
1149: % end
1150:
1151: % end
1152:
1153: % end
1154:
1155: % end
1156:
1157: % end
1158:
1159: % end
1160:
1161: % end
1162:
1163: % end
1164:
1165: % end
1166:
1167: % end
1168:
1169: % end
1170:
1171: % end
1172:
1173: % end
1174:
1175: % end
1176:
1177: % end
1178:
1179: % end
1180:
1181: % end
1182:
1183: % end
1184:
1185: % end
1186:
1187: % end
1188:
1189: % end
1190:
1191: % end
1192:
1193: % end
1194:
1195: % end
1196:
1197: % end
1198:
1199: % end
1200:
1201: % end
1202:
1203: % end
1204:
1205: % end
1206:
1207: % end
1208:
1209: % end
1210:
1211: % end
1212:
1213: % end
1214:
1215: % end
1216:
1217: % end
1218:
1219: % end
1220:
1221: % end
1222:
1223: % end
1224:
1225: % end
1226:
1227: % end
1228:
1229: % end
1230:
1231: % end
1232:
1233: % end
1234:
1235: % end
1236:
1237: % end
1238:
1239: % end
1240:
1241: % end
1242:
1243: % end
1244:
1245: % end
1246:
1247: % end
1248:
1249: % end
1250:
1251: % end
1252:
1253: % end
1254:
1255: % end
1256:
1257: % end
1258:
1259: % end
1260:
1261: % end
1262:
1263: % end
1264:
1265: % end
1266:
1267: % end
1268:
1269: % end
1270:
1271: % end
1272:
1273: % end
1274:
1275: % end
1276:
1277: % end
1278:
1279: % end
1280:
1281: % end
1282:
1283: % end
1284:
1285: % end
1286:
1287: % end
1288:
1289: % end
1290:
1291: % end
1292:
1293: % end
1294:
1295: % end
1296:
1297: % end
1298:
1299: % end
1300:
1301: % end
1302:
1303: % end
1304:
1305: % end
1306:
1307: % end
1308:
1309: % end
1310:
1311: % end
1312:
1313: % end
1314:
1315: % end
1316:
1317: % end
1318:
1319: % end
1320:
1321: % end
1322:
1323: % end
1324:
1325: % end
1326:
1327: % end
1328:
1329: % end
1330:
1331: % end
1332:
1333: % end
1334:
1335: % end
1336:
1337: % end
1338:
1339: % end
1340:
1341: % end
1342:
1343: % end
1344:
1345: % end
1346:
1347: % end
1348:
1349: % end
1350:
1351: % end
1352:
1353: % end
1354:
1355: % end
1356:
1357: % end
1358:
1359: % end
1360:
1361: % end
1362:
1363: % end
1364:
1365: % end
1366:
1367: % end
1368:
1369: % end
1370:
1371: % end
1372:
1373: % end
1374:
1375: % end
1376:
1377: % end
1378:
1379: % end
1380:
1381: % end
1382:
1383: % end
1384:
1385: % end
1386:
1387: % end
1388:
1389: % end
1390:
1391: % end
1392:
1393: % end
1394:
1395: % end
1396:
1397: % end
1398:
1399: % end
1400:
1401: % end
1402:
1403: % end
1404:
1405: % end
1406:
1407: % end
1408:
1409: % end
1410:
1411: % end
1412:
1413: % end
1414:
1415: % end
1416:
1417: % end
1418:
1419: % end
1420:
1421: % end
1422:
1423: % end
1424:
1425: % end
1426:
1427: % end
1428:
1429: % end
1430:
1431: % end
1432:
1433: % end
1434:
1435: % end
1436:
1437: % end
1438:
1439: % end
1440:
1441: % end
1442:
1443: % end
1444:
1445: % end
1446:
1447: % end
1448:
1449: % end
1450:
1451: % end
1452:
1453: % end
1454:
1455: % end
1456:
1457: % end
1458:
1459: % end
1460:
1461: % end
1462:
1463: % end
1464:
1465: % end
1466:
1467: % end
1468:
1469: % end
1470:
1471: % end
1472:
1473: % end
1474:
1475: % end
1476:
1477: % end
1478:
1479: % end
1480:
1481: % end
1482:
1483: % end
1484:
1485: % end
1486:
1487: % end
1488:
1489: % end
1490:
1491: % end
1492:
1493: % end
1494:
1495: % end
1496:
1497: % end
1498:
1499: % end
1500:
1501: % end
1502:
1503: % end
1504:
1505: % end
1506:
1507: % end
1508:
1509: % end
1510:
1511: % end
1512:
1513: % end
1514:
1515: % end
1516:
1517: % end
1518:
1519: % end
1520:
1521: % end
1522:
1523: % end
1524:
1525: % end
1526:
1527: % end
1528:
1529: % end
1530:
1531: % end
1532:
1533: % end
1534:
1535: % end
1536:
1537: % end
1538:
1539: % end
1540:
1541: % end
1542:
1543: % end
1544:
1545: % end
1546:
1547: % end
1548:
1549: % end
1550:
1551: % end
1552:
1553: % end
1554:
1555: % end
1556:
1557: % end
1558:
1559: % end
1560:
1561: % end
1562:
1563: % end
1564:
1565: % end
1566:
1567: % end
1568:
1569: % end
1570:
1571: % end
1572:
1573: % end
1574:
1575: % end
1576:
1577: % end
1578:
1579: % end
1580:
1581: % end
1582:
1583: % end
1584:
1585: % end
1586:
1587: % end
1588:
1589: % end
1590:
1591: % end
1592:
1593: % end
1594:
1595: % end
1596:
1597: % end
1598:
1599: % end
1600:
1601: % end
1602:
1603: % end
1604:
1605: % end
1606:
1607: % end
1608:
1609: % end
1610:
1611: % end
1612:
1613: % end
1614:
1615: % end
1616:
1617: % end
1618:
1619: % end
1620:
1621: % end
1622:
1623: % end
1624:
1625: % end
1626:
1627: % end
1628:
1629: % end
1630:
1631: % end
1632:
1633: % end
1634:
1635: % end
1636:
1637: % end
1638:
1639: % end
1640:
1641: % end
1642:
1643: % end
1644:
1645: % end
1646:
1647: % end
1648:
1649: % end
1650:
1651: % end
1652:
1653: % end
1654:
1655: % end
1656:
1657: % end
1658:
1659: % end
1660:
1661: % end
1662:
1663: % end
1664:
1665: % end
1666:
1667: % end
1668:
1669: % end
1670:
1671: % end
1672:
1673: % end
1674:
1675: % end
1676:
1677: % end
1678:
1679: % end
1680:
1681: % end
1682:
1683: % end
1684:
1685: % end
1686:
1687: % end
1688:
1689: % end
1690:
1691: % end
1692:
1693: % end
1694:
1695: % end
1696:
1697: % end
1698:
1699: % end
1700:
1701: % end
1702:
1703: % end
1704:
1705: % end
1706:
1707: % end
1708:
1709: % end
1710:
1711: % end
1712:
1713: % end
1714:
1715: % end
1716:
1717: % end
1718:
1719: % end
1720:
1721: % end
1722:
1723: % end
1724:
1725: % end
1726:
1727: % end
1728:
1729: % end
1730:
1731: % end
1732:
1733: % end
1734:
1735: % end
1736:
1737: % end
1738:
1739: % end
1740:
1741: % end
1742:
1743: % end
1744:
1745: % end
1746:
1747: % end
1748:
1749: % end
1750:
1751: % end
1752:
1753: % end
1754:
1755: % end
1756:
1757: % end
1758:
1759: % end
1760:
1761: % end
1762:
1763: % end
1764:
1765: % end
1766:
1767: % end
1768:
1769: % end
1770:
1771: % end
1772:
1773: % end
1774:
1775: % end
1776:
1777: % end
1778:
1779: % end
1780:
1781: % end
1782:
1783: % end
1784:
1785: % end
1786:
1787: % end
1788:
1789: % end
1790:
1791: % end
1792:
1793: % end
1794:
1795: % end
1796:
1797: % end
1798:
1799: % end
1800:
1801: % end
1802:
1803: % end
1804:
1805: % end
1806:
1807: % end
1808:
1809: % end
1810:
1811: % end
1812:
1813: % end
1814:
1815: % end
1816:
1817: % end
1818:
1819: % end
1820:
1821: % end
1822:
1823: % end
1824:
1825: % end
1826:
1827: % end
1828:
1829: % end
1830:
1831: % end
1832:
1833: % end
1834:
1835: % end
1836:
1837: % end
1838:
1839: % end
1840:
1841: % end
1842:
1843: % end
1844:
1845: % end
1846:
1847: % end
1848:
1849: % end
1850:
1851: % end
1852:
1853: % end
1854:
1855: % end
1856:
1857: % end
1858:
1859: % end
1860:
1861: % end
1862:
1863: % end
1864
```


(Refer Slide Time: 28:47)

```
17 % Starting of DE
18 f = rand(10,1) % Vector to store the fitness function value of the population members
19 fu = rand(10,1) % Vector to store the fitness function value of the new population members
20 D = length(lb) % Determining the number of decision variables in the problem
21 lb = rand(10,D) % Matrix to store the trial solutions
22 P = repmat(lb,Np,1) + repmat(lb-ub,Np,1).*rand(Np,D) % Generation of the initial population
23 for p = 1:Np % Evaluating the fitness function of the initial population
24     f(p) = prob(f(p,1))
25 end
26 % Iteration loop
27 for t = 1:T
28     for i = 1:Np
29         % Mutation
30         randIndex = randi(1+1:D)
31         use = randi(1+1:Np)
32         X1 = P(ia(1),:)
33         X2 = P(ia(2),:)
34         X3 = P(ia(3),:)
35         Y = X1 + P*(X2 - X3) % Generating the donor vector
36
37         % Crossover
38         randi = randi(D,1)
39         for j = 1:D
40             if (rand <= Pr) || randi == 1 % Check the donor vector or target vector
41                 Y(j,1) = Y1(j)
42             else
43                 Y(j,1) = P1(j,1) % Accept variable from target vector
44             end
45         end
46     end
47 end
48
49 % Fitness evaluation function
50 function [fval] = prob(x)
51     fval = 10 + 10*(x(1) - 10)^2 + 5*(x(2) - 10)^2 + 20*(x(3) - 10)^2 + 10*(x(4) - 10)^2 + 10*(x(5) - 10)^2
52 end
```

Command Window:

```
lb =
    0     0     0     0     0
ub =
    10    10    10    10    10
prob =
    1
Function handle with value:
@probNew
fval =
     5
T =
    100
Pr =
    0.2000
f =
    0.2000
lb
ub
```

So, in this case the donor vector first variable, fourth variable and fifth variable are not in the bounds, but we will not be bounding them right. So, now, we need to perform the crossover for the first solution itself. For performing crossover, we need to first randomly select one of the decision variables right, so that is what we are doing with this randi right. So, randi of D comma 1 will give us a scalar value from 1 to capital D; D in this case is 5.

(Refer Slide Time: 29:15)

```
17 % Starting of DE
18 f = NaN(Np,1) % Vector to store the fitness function value of the population members
19 fu = NaN(Np,1) % Vector to store the fitness function value of the new population members
20 D = length(x) % Determining the number of decision variables in the problem
21 M = NaN(Np,D) % Matrix to store the trial solutions
22 P = repmat(x(Np,1) + repmat(rand(Np,1)-0.5)*rand(Np,D)) % Generation of the initial population
23 for p = 1:Np % Evaluating the fitness function of the initial population
24     f(p) = prob(Fp,1)
25 end
26 % Iteration loop
27 for t = 1:T
28     for i = 1:Np
29         % Mutation
30         randIndex = randi(1,Np-1) % Ensuring that the current member is not the partner
31         r = randi(1,randIndex,Np-1,3) % Selection of three random partners
32         X1 = P(randIndex,1) % Assigning randomly selected solution 1
33         X2 = P(randIndex(2),1) % Assigning randomly selected solution 2
34         X3 = P(randIndex(3),1) % Assigning randomly selected solution 3
35         V = X1 + P*(X2 - X3) % Generating the donor vector
36         % Crossover
37         R = rand(1,1) % Generating the random variable delta
38         for j = 1:D
39             if (R <= delta)
40                 V(j,1) = P(i,1) % Check for donor vector or target vector
41             else
42                 V(j,1) = P(i,1) % Accept variable from donor vector
43             end
44         end
45     end
46 end
```

Command Window Output:

```
f =
129.0137
NaN
NaN
NaN
NaN
f =
129.0137
115.0393
144.5415
136.9129
NaN
f =
129.0137
115.0393
144.5415
136.9129
NaN
f =
129.0137
115.0393
144.5415
136.9129
15.0107
```

So, 1 to 5, it will give us one random value.

(Refer Slide Time: 29:05)

```
%% Genetic Algorithm
% Parameters
P = repmat(10, Np, 1) + repmat(100-10, Np, 1); % Generation of the initial population
% Evaluating the fitness function of the initial population
for i = 1:Np
    F(i) = prob(F(i));
end
% Iteration loop
for i = 1:T
    % Mutation
    Candidates = 1:i-1:i+1:Np; % Ensuring that the current member is not the partner
    [del, Candidates] = randperm(Np-1, Np); % Selection of three random partners
    X1 = F(Candidates(1)); % Assigning randomly selected solution 1
    X2 = F(Candidates(2)); % Assigning randomly selected solution 2
    X3 = F(Candidates(3)); % Assigning randomly selected solution 3
    Y = X1 + P*(X2 - X3); % Generating the donor vector
    % Crossover
    del = randi(Np, 1); % Generating the random variable delta
    for j = 1:del
        if (rand == F(j) || del == j) % Check for donor vector or target vector
            U(j) = Y(j); % Adopt variable from donor vector
        else
            U(j) = F(j); % Adopt variable from target vector
        end
    end
    % Bounding and Greedy Selection
    for j = 1:Np
        F(j) = max(min(U(j), 1)); % Bounding the violating variables to their upper bound
        F(j) = max(min(F(j), 1)); % Bounding the violating variables to their lower bound
    end
    F(i) = prob(F(i));
end
% Fitness of the final solution
```

Command Window

```
T =
    200
F =
    2.29137
    1.10393
    1.44545
    1.86219
    1.83295
Candidates =
    2     3     4     5
del =
    4     3     5
X1 =
    5.3195    0.3647    3.7116    0.4210    5.9794
X2 =
    5.2289    1.3010    1.7094    1.7404    5.4950
X3 =
    1.4711    0.4269    0.3976    4.0466    0.2301
Y =
    12.0500    4.3073    4.1190   -1.3956   13.1760
U =
    4
```

So, in this case it happens to be 4. So, this value is randomly selected right. Now, we need to perform the crossover operation with these two conditions. So, if we need to select a random number if it is less than the crossover probability or if del is equal to j, we need to copy from the donor vector, else we need to copy from the trial vector. So, the trial vector is going to be stored in U for j is equal to 1 the value of j is 1 now right.

So, it goes into this condition is satisfied. So, the random number that it generated was less than crossover probability because del is 4 and j is 1. So, this condition was not satisfied this conditions was satisfied, because as soon as we say rand we get a random number. So, here we did not save it in a particular variable otherwise you could have seen what is the actual value. So, now, U if we see it is copied from the donor vector.

(Refer Slide Time: 32:05)

```
21 % Mutation
22 Candidates = 1:1:1+1:10; % Ensuring that the current number is not the partner
23 size = Candidates(randperm(10,3)); % Selection of three random partners
24
25 X1 = F1(size(1,:)); % Assigning randomly selected solution 1
26 X2 = F1(size(2,:)); % Assigning randomly selected solution 2
27 X3 = F1(size(3,:)); % Assigning randomly selected solution 3
28
29 V = X1 + P*(X2 - X3); % Generating the donor vector
30
31
32 % Crossover
33 rand = randi(10); % Generating the random variable delta
34 for j = 1:10
35
36     if (rand <= P) || delta == 1 % Check for donor vector or target vector
37         % Accept variable from donor vector
38         V(j,:) = F1(j,:);
39     else
40         % Accept variable from target vector
41         V(j,:) = F1(j,:);
42     end
43 end
44
45 % Stochastic and Greedy Selection
46 for i = 1:10
47
48     F1(i,:) = min(F1(i,:), V(i,:)); % Bounding the violating variables to their upper bound
49     F1(i,:) = max(F1(i,:), V(i,:)); % Bounding the violating variables to their lower bound
50
51     C1(i) = popz(F1(i,:)); % Evaluating the fitness of the trial solution
52
53     if C1(i) < C1(i)
54         % Greedy selection
55         F1(i,:) = V(i,:);
56     end
57     % Include the fitness function value of the new solution in population
58     F1(i) = C1(i);
59 end
60
61 % Best fitness and its location
62 [bestFitness, ind] = min(F1);
63 [bestParam, indParam] = min(C1);
```

Candidates =	1	3	4	5	
size =	4	1	3		
X1 =	9.3195	0.3647	3.7114	0.4215	5.9794
X2 =	5.4290	4.1477	3.2289	7.0924	4.4304
X3 =	5.2189	5.9810	1.7004	1.7434	5.4955
V =	9.4991	-1.1766	5.0109	5.1704	2.5243
delta =	3				
P =	12.0000	4.9079	4.4139	-1.3534	13.1700
F1 =	9.4991	best	best	best	best
bestParam =	best	best	best	best	best

So, similarly the candidates are expect 2, 1 3 4 5, we need to select three candidates from here 1 3 4 5. So, we randomly permuted values 1 to 4, and select three values and then extract three values from here right. So, in this case the solution 4 1 3 are the partner for the ith solution, ith solution is currently the second solution. So, the partner for the second solution are fourth solution first greedy solution and third solution. So, line 41, 42, 43 again will be just assigning those solutions from the population to x 1, x 2, x 3 right. So, these are the three solutions x 1, x 2 and x 3.

Again we need to calculate the donor vector right. So, this is the donor vector. Again donor vector is not within the bounds. So, we need not worry about the bounding of the solution over here. So, in crossover we require a random decision variable. So, we select that right. So, in this case the decision variable is 3 right. So, here if you see even in this case the random number generated was less than crossover probability right. So, again in this second case 2,

the third case 2, the fourth case, so this is the first time if you see it is coming to this else part right.

(Refer Slide Time: 33:35)

```

% Genetic Algorithm
% Parameters
N = 100; % Number of iterations
M = 10; % Number of individuals in the population
Pc = 0.8; % Crossover probability
Pm = 0.05; % Mutation probability
% Initial population
P = randi(100, M, 4);
% Fitness function
function f = fitness(x)
    f = sum(x.^2);
end
% Main loop
for i = 1:N
    % Mutation
    for j = 1:M
        % Generating the random number in the range [0,1]
        r = rand;
        % Selection of three random partners
        [i1, i2, i3] = randperm(M, 3);
        % Assigning randomly selected solution 1
        x1 = P(i1, :);
        % Assigning randomly selected solution 2
        x2 = P(i2, :);
        % Assigning randomly selected solution 3
        x3 = P(i3, :);
        % Generating the donor vector
        xD = P(i3, :) - x1;
    end
    % Crossover
    for j = 1:M
        % Generating the random variable delta
        delta = randi(10);
        for k = 1:delta
            % Check for donor vector of target vector
            if (rand < Pc) || delta == 1
                % Accept variable from donor vector
                P(i1, k) = xD(k);
            else
                % Accept variable from target vector
                P(i1, k) = P(i1, k);
            end
        end
    end
    % Bounding and Greedy Selection
    for j = 1:M
        % Bounding the violating variables to their upper bound
        P(i1, :) = min(P(i1, :), 100);
        % Bounding the violating variables to their lower bound
        P(i1, :) = max(P(i1, :), 0);
        % Evaluating the fitness of the trial solution
        f1 = fitness(P(i1, :));
        % Greedy selection
        if f1 < f(i1)
            % Include the new solution in population
            P(i1, :) = P(i1, :);
            % Include the fitness function value of the new solution in population
            f(i1) = f1;
        end
    end
end

```

So, now the donor vector was over here 9.49, minus 1.17, 5.01, 5.17 and 2.52. So, the first three values were copied. For the fourth value since neither of this condition is satisfied, we will have to take the value from the population right. So, population if you see we are currently in the second solution and j is 4 right. So, second, second row fourth column if you see it is 6.4134 that will be coming over here right.

To step in right, so as expected 6.4134 comes over here right, continuing that for the next variable right, so it comes from the donor vector. So, now, we have generated two trail vectors right. So, similarly you can debug and look for all the other trial vectors. What we will do is we will have a breakpoint over here, and we will click on this continue right.

(Refer Slide Time: 34:35)

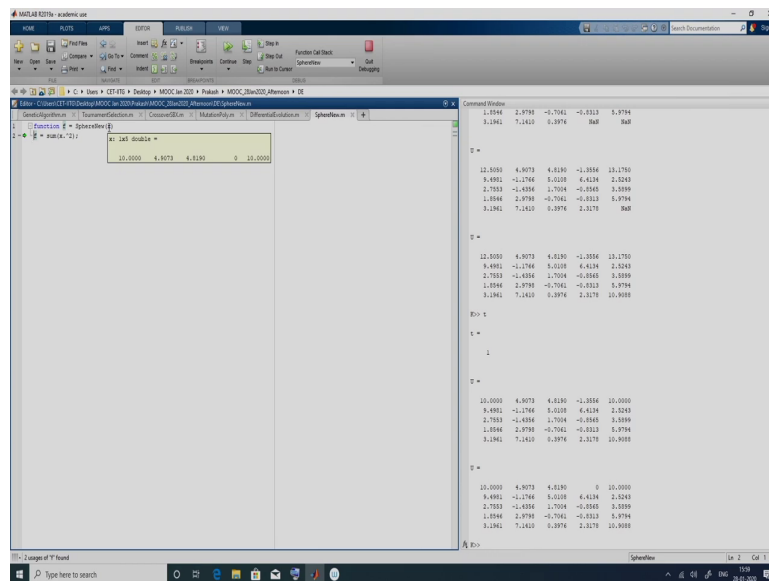
The image shows a MATLAB script for Differential Evolution optimization. The script is divided into several sections: initialization, crossover, mutation, and selection. The initialization section sets up the population size (Np), number of iterations (Nmax), and the fitness function. The crossover section generates random vectors. The mutation section generates mutant vectors. The selection section evaluates the fitness of the trial vectors and selects the best ones. The script is executed in the MATLAB environment, and the results are displayed in the Command Window.

```
27 % Initialization
28 Candidates = randi(100, Np, 1); % Ensuring that the current number is not the partner
29 % Candidates = randperm(Np, Np); % Selection of three random partners
30
31 % Assigning randomly selected solution 1
32 % Assigning randomly selected solution 2
33 % Assigning randomly selected solution 3
34
35 % Generating the donor vector
36
37
38 % Crossover
39 % Generating the random variable r1
40 for j = 1:Np
41
42 % Check for donor vector or target vector
43 % Accept variable from donor vector
44
45 % Accept variable from target vector
46
47 end
48
49 % Storing and Greedy Selection
50 for i = 1:Np
51
52 % Storing the violating variables to their upper bound
53 % Storing the violating variables to their lower bound
54
55 % Evaluating the fitness of the trial solution
56
57 % Greedy selection
58 % Include the new solution in population
59 % Include the fitness function value of the new solution in population
60
61 end
62
63
64 % [bestFitness, ind] = min(F);
65 % bestInd = F(ind);
```

F	X1	X2	X3	X4
12.5000	4.9073	4.6190	-1.3556	13.1700
9.4992	-1.2766	5.0209	4.6194	2.5243
2.7533	-1.4396	1.7004	-0.8565	3.5899
1.8546	2.9798	-0.7042	-0.8313	5.9794
3.1941	7.1420	0.8978	2.1178	8.68

So, now all the five trial vectors are generated right. So, since it is similar we did not go through each and every solution. So, we had seen both the cases where it goes into this if condition. So, now that we have generated all the trial vectors. We will be checking whether they are in the bounds; if whichever value is not in the bounds, we will be bringing it either to its lower bound or the upper bound right, and then evaluate its fitness and perform a greedy selection strategy right. So, this loop also runs from for j is equal to 1 to Np. Remember we are still in the first iterations.

(Refer Slide Time: 36:23)



So, if you do step in, it is going to this function right. And again the fitness function will be evaluated. So, as we can see the first row of `U` has been sent into the function as input right.

(Refer Slide Time: 36:37)

The screenshot shows a MATLAB script for a Genetic Algorithm. The script includes functions for mutation, crossover, and fitness evaluation. The output console displays the results of the algorithm, including the fitness function value and the best solution found.

```

27 % Mutation
28 Candidates(i+1:i+100) % Ensuring that the current number is not the partner
29 size Candidates(1:100,1:3) % Selection of three random partners
30
31 X1 = F104(i+1,1) % Assigning randomly selected solution 1
32 X2 = F104(i+1,2) % Assigning randomly selected solution 2
33 X3 = F104(i+1,3) % Assigning randomly selected solution 3
34
35 Y = X1 + F104 - X3 % Generating the donor vector
36
37
38 % Crossover
39
40 for i = randi(100) % Generating the random variable index
41 for j = 1:100
42
43 if rand <= Pc || dci == 1 % Check for donor vector or target vector
44     % Accept variable from donor vector
45     % Accept variable from target vector
46     % Accept variable from target vector
47 end
48 end
49
50 % Stopping and Greedy Selection
51 for i = 1:100
52
53     % [1,1] % minmax(F(i,1))
54     % [1,1] % maxmin(F(i,1))
55     % [1,1] % maxmin(F(i,1))
56     % [1,1] % maxmin(F(i,1))
57     % [1,1] % maxmin(F(i,1))
58     % [1,1] % maxmin(F(i,1))
59     % [1,1] % maxmin(F(i,1))
60     % [1,1] % maxmin(F(i,1))
61     % [1,1] % maxmin(F(i,1))
62     % [1,1] % maxmin(F(i,1))
63     % [1,1] % maxmin(F(i,1))
64     % [1,1] % maxmin(F(i,1))
65     % [1,1] % maxmin(F(i,1))
66     % [1,1] % maxmin(F(i,1))
67     % [1,1] % maxmin(F(i,1))
68     % [1,1] % maxmin(F(i,1))
69     % [1,1] % maxmin(F(i,1))
70     % [1,1] % maxmin(F(i,1))
71     % [1,1] % maxmin(F(i,1))
72     % [1,1] % maxmin(F(i,1))
73     % [1,1] % maxmin(F(i,1))
74     % [1,1] % maxmin(F(i,1))
75     % [1,1] % maxmin(F(i,1))
76     % [1,1] % maxmin(F(i,1))
77     % [1,1] % maxmin(F(i,1))
78     % [1,1] % maxmin(F(i,1))
79     % [1,1] % maxmin(F(i,1))
80     % [1,1] % maxmin(F(i,1))
81     % [1,1] % maxmin(F(i,1))
82     % [1,1] % maxmin(F(i,1))
83     % [1,1] % maxmin(F(i,1))
84     % [1,1] % maxmin(F(i,1))
85     % [1,1] % maxmin(F(i,1))
86     % [1,1] % maxmin(F(i,1))
87     % [1,1] % maxmin(F(i,1))
88     % [1,1] % maxmin(F(i,1))
89     % [1,1] % maxmin(F(i,1))
90     % [1,1] % maxmin(F(i,1))
91     % [1,1] % maxmin(F(i,1))
92     % [1,1] % maxmin(F(i,1))
93     % [1,1] % maxmin(F(i,1))
94     % [1,1] % maxmin(F(i,1))
95     % [1,1] % maxmin(F(i,1))
96     % [1,1] % maxmin(F(i,1))
97     % [1,1] % maxmin(F(i,1))
98     % [1,1] % maxmin(F(i,1))
99     % [1,1] % maxmin(F(i,1))
100    % [1,1] % maxmin(F(i,1))

```

The output console shows the following results:

```

1.194e+002 2.377e+001 -7.174e+000 -0.0313 0.9774
3.194e+002 7.141e+000 0.3974 2.317e+000 NaN
...
247.0048
NaN
NaN
NaN
NaN

```

So, the fitness function is calculated. So, now, the fitness function value is 247 right. So, if we look at f_u the first value is 247, and the solution which are already there right that fitness is 129. So, this condition is not satisfied right now we are looking at j is equal to 1.. So, f_u of 1 is not less than f of 1. So, this part will not be executed right and the population remains the same right. So, this solution is now discarded. So, if we continue this for the second solution, again this is the only variable which is violating the bounds right. So, this will become 0, once line 65, 66 are executed right.

(Refer Slide Time: 37:23)

The image shows a MATLAB Editor window with a script file named 'DifferentialEvolution.m'. The script implements a genetic algorithm with the following sections:

- Initialization:** Defines population size (100), number of generations (100), and fitness function handle. It generates a random initial population $P(1:100, 1:3)$.
- Crossover:** Iterates over the population. For each individual, it generates a random vector r and a random number $rand$. It then generates a trial vector $F_{i,1}$ based on the crossover probability.
- Selection and Greedy Selection:** Evaluates the fitness of the trial vector $F_{i,1}$ and compares it with the fitness of the current individual $F(i,1)$. If the trial vector has a better fitness, it is accepted and the current individual is updated.

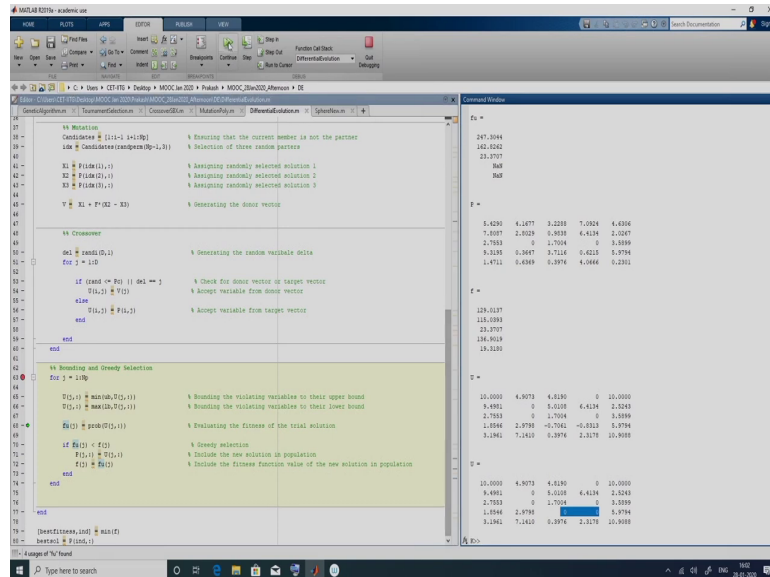
The Command Window on the right displays the output of the script, showing the fitness values for the population over 100 generations. The fitness values start at approximately 10,000 and decrease to around 23.37 by the end of the process.

So, this is 0 right. Again the fitness function will be evaluated with this f_u right. So, in this case, the second value is 162.8262, and over here it is 115 right. So, even in this case the newly generated trail vector is not better than the second solution. So, in the third case over here if you see this variable is violating, this variable is violating, both of them are violating the lower bounds right. So, both of them have been reset to 0 right. Again calculating its fitness function value, so the fitness function value would be 23.3707; so the third fitness function is 144. The current third solution in the population has a fitness of 144 whereas the newly generated trail vector has a fitness of 23.37. So, the newly generated solution will be taken into the population right.

So, we are overwriting the j th row with a third row of U right. So, U contains the trail vectors. So, both the population as well as the fitness function have to be updated right, so

that is what we are doing in line 71 and 72 right. And then we need to continue this for the fourth solution.

(Refer Slide Time: 38:45)



So, for the fourth solution right, so fourth solution if you see the third and the fourth variable are violating the lower bounds right. So, those are set to 0s. Again we evaluate the fitness function right, even in this case the newly generated solution fourth solution has a fitness function of 48, whereas the original one has 136 right, so that is why we overwrite this.

(Refer Slide Time: 39:13)

```

% Initialization
P1 = randi(100, 1, 1); % Ensuring that the current number is not the partner
P2 = randi(100, 1, 1); % Selection of three random partners
P3 = randi(100, 1, 1); % Assigning randomly selected solution 1
P4 = randi(100, 1, 1); % Assigning randomly selected solution 2
P5 = randi(100, 1, 1); % Assigning randomly selected solution 3
X = P1 + P2*(30 - X); % Generating the donor vector

% Crossover
for i = 1:100
    randi(100, 1, 1); % Generating the random variable delta
    for j = 1:100
        if (rand <= P1 || delta == 1) % Check the donor vector vs target vector
            X(i,j) = P2(j); % Accept variable from donor vector
        else
            X(i,j) = P1(j); % Accept variable from target vector
        end
    end
end

% Stopping and Greedy Selection
for i = 1:100
    F1(i) = min(F1(1:i)); % Stopping the violating variables to their upper bound
    F2(i) = max(F2(1:i)); % Stopping the violating variables to their lower bound
    C(i) = cost(F1(i)); % Evaluating the fitness of the trial solution
    if C(i) <= C(i-1)
        % Greedy selection
        F1(i) = C(i); % Include the new solution in population
        % Include the fitness function value of the new solution in population
    end
end
end

[bestFitness, ind] = min(F1);
bestCost = F1(ind);

```

Command Window Output:

F =	5.4290	4.1477	3.2289	7.5924	4.4304
	7.6087	2.6029	6.8838	6.4134	2.0247
	2.7583	0	1.7004	0	3.8589
	1.8844	2.8798	0	0	5.8794
	1.4711	0.4369	0.3974	4.0466	0.2302
E =	129.0137				
	110.0393				
	23.3707				
	48.0723				
	19.3180				
D =	10.0000	4.9073	4.8190	0	10.0000
	9.4892	0	5.0209	6.4134	2.0243
	2.7583	0	1.7004	0	2.9599
	1.8844	2.8798	0	0	5.8794
	3.1562	7.1420	0.3974	2.1178	10.0000
D =	10.0000	4.9073	4.8190	0	10.0000
	9.4892	0	5.0209	6.4134	2.0243
	2.7583	0	1.7004	0	3.8589
	1.8844	2.8798	0	0	5.8794
	3.1461	7.1420	0.3974	2.1178	10.0000
F0 =	247.0046				
	140.0042				
	23.3707				
	48.0723				
	146.7393				

And finally, the fifth solution similarly right. So, for the fifth solution the fitness function of the trail vector is 166, whereas the original solution which we have is 19.3. So, we do not need to overwrite. So, if we step this right, so this completes one iteration of differential evolution right. So, similarly we will have to perform all the iterations the second iteration the 3 iteration all the way up to 100 iterations, because in this case we had set the value of t to be 100, right.

(Refer Slide Time: 39:43)

```
clear
%% Problem settings
L = [0 0 0 0 0]; % Lower bound
ub = [10 10 10 10 10]; % Upper bound
prob = @SphereEval; % Fitness function

%% Parameters for Differential Evolution
Np = 5; % Population size
T = 100; % No. of iterations
Pc = 0.4; % Crossover probability
Pm = 0.05; % Mutation rates

%% Starting of DE
f = NaN(Np,1); % Vector to store the fitness function value of the population members
Ev = NaN(Np,1); % Vector to store the fitness function value of the best population members

D = length(L); % Determining the number of decision variables in the problem
D = NaN(D,1); % Matrix to store the trial solutions
P = repmat(L,Np,1) + repmat(ub-L,Np,1).*rand(Np,D); % Generation of the initial population

for p = 1:Np
    f(p) = prob(P(p,:)); % Evaluating the fitness function of the initial population
end

%% Iteration loop
for t = 1:T
    for i = 1:Np
        %% Mutation
        randIndex = randi(1,D); % Ensuring that the current member is not the partner
        r = rand(1,D); % Selection of three random partners
        r1 = randi(1,3); % Assigning randomly selected solution 1
        r2 = randi(1,3); % Assigning randomly selected solution 2
        r3 = randi(1,3); % Assigning randomly selected solution 3
    end
end
```

So, as you can see the implementation of differential evolution is very easy right. So, let me just execute this right. Let me put a breakpoint over here as soon as it completes all the iterations, I want it to pause. So, now I can give this continue right, let me remove this breakpoint right. So, now, I am not interested in doing every iteration right, because we have seen what is happening in every iteration. So, let me just do this continue. Now, let me just clear this screen right.

right. So, at the end of it, it gives what is the best fitness function value and the best solution right.

(Refer Slide Time: 41:41)

```

clear
% To clear the current workspace
% To clear the workspace

%% Problem settings
lb = -100*ones(1,50); % Lower bound
ub = 100*ones(1,50); % Upper bound
prob = @SphereEval; % Fitness function

%% Parameters for Differential Evolution
Np = 50; % Population Size
T = 100; % No. of iterations
Pc = 0.8; % crossover probability
Pm = 0.05; % Mutation chance

%% Starting of DE
f = NaN(50,1); % Vector to store the fitness function value of the population member
fu = NaN(50,1); % Vector to store the fitness function value of the new population
D = length(lb); % Determining the number of decision variables in the problem
U = NaN(50,D); % Matrix to store the trial solutions
P = repmat(lb,Np,1) + repmat((ub-lb)/Np,1,rand(Np,D)); % Generation of the initial population

for p = 1:100
    F(p,:) = prob(P(p,:)); % Evaluating the fitness function of the initial population
end

%% Iteration loop
for k = 1:T
    for i = 1:Np
        % Mutation
        randIndex = [1:i+1:50];
        l = randi(3); % Selection of three random parents
        z1 = P(i,randIndex,l);
        z2 = P(i,randIndex,l);
        z3 = P(i,randIndex,l);
    end
end

```

Command Window Output:

```

prob =
function_handle with value:
@SphereEval
bestfitness =
9.2608e-04
bestsol =
Columns 1 through 5
64.2461 35.8424 -10.2992 -40.4628 -43.7159 0.8171 65.0617 74.5643
Columns 10 through 18
55.8529 17.0502 55.6161 7.3944 -8.2120 8.2818 27.1318 -5.0038 -66.6032
Columns 19 through 27
34.5124 -43.5532 -33.6947 -39.5148 31.0392 33.1355 -37.7070 15.4994 -79.4733
Columns 28 through 34
12.4235 30.9535 -37.3823 17.5294 14.2377 64.4638 -61.4949 -13.3233 -30.7054
Columns 37 through 45
82.3232 -36.7547 27.2294 18.2223 61.8742 29.7720 -49.3774 10.2229 18.4675
Columns 46 through 50
-43.1130 -47.9327 9.1438 82.7944 -60.8118

```

So, so now let me remove all the breakpoints right, and put a semicolon. So, now, we can change this lower and the upper bound and see how it is performing. So, what we will do is we will say that the lower bound is minus 100 right, once of 1 comma let us say I have 50 variables right. So, one row 50 columns lower bound will give 50 times minus 100 right.

So, same thing I can copy it over here, and let us say the upper bound is 100 right, we are not changing the objective function just we are changing the lower and the upper bound right. As you know these problems are scalable, so we just want to test that if the problem size is really big like instead of five variables let us say if it is 50 variables and we are working with the population size of 50 and 100 iterations.

So, let us see whether it is able to find the globally optimal solution. In this case, it is able to find a solution which has a fitness function value of 9.07×10^4 , and all the variables if you see it is between minus 100 and plus 100 right. So, the bounds are no longer 0 to 10 right. So, the bounds are between minus 100 to 100. So, this minus 67.9827, minus 63.1130 all those are within bounds right. So, in this case we see that it is not able to find the optimal solutions.

So, the optimal solution was again 0s right. So, if you take all the 50 variable to be 0, we will get a fitness function value of 0 that is the end of this session. In the next session, we will look at genetic algorithms first we will look into binary coded genetic algorithm, then we look into real coded genetic algorithm, and then will only implemented real coded genetic algorithm on MATLAB.

Thank you.