

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

Lecture – 22
Constrained Optimization Problems

Welcome back. So far we have seen the five metaheuristic techniques right – artificial B colony optimization, genetic algorithm, differential evolution, particle swarm optimization and teaching learning based optimization right. In all the five cases when we were discussing these metaheuristic techniques, we took only the objective function ah. There were no constraints apart from the bound constraints and bound constraints we handled by corner bounding the solution. Whenever a solution violated the bounds, we brought it back to the bound right.

So, first we will look into three examples right; subsequently, we will discuss how to convert those constraint optimization problem into unconstrained optimization problem and continue using those five algorithms which we have discussed right.

(Refer Slide Time: 01:14)

Example 1

A farm uses at least 800 kg of special feed daily. The special feed is a mixture of corn and soybean meal with the following compositions.

Feedstuff	kg per kg of feedstuff		Cost (\$ per kg)
	Protein	Fiber	
Corn ✓	0.09	0.02	0.3
Soyabean meal	0.6	0.06	0.9

The dietary requirements of the special feed are at least 30% protein and at most 5% fiber. The goal is to determine the daily minimum-cost of feed mix.

Operations research: An introduction, H. A. Taha

The first example is a farm example right. So, wherein 800 kg of special feed is required daily right. So, the special feed can be prepared by a mixture of corn and soya bean meal. So, we have corn and we have soya bean meal right; using these two. We need to combine these two to come up with at least 800 kg ah. Remember, this word at least is important right. So, at least 800 kg we can come up with more than 800 kg that is fine but, at least 800 is required right.

Every kg of corn that we take we will have 0.09kg of protein and 0.02kgs of fiber. Similarly for every kg of soya bean meal we take we will have 0.6kgs of protein and 0.06kgs of fiber. The cost of corn is 0.3 dollars per kg and the cost of soya bean meal is 0.9 dollars per kg right. So, we need to come up with at least 800 kg. So, for example, now we can say since corn is cheaper, we will say all the 800 kg is nothing, but corn right, but there are constraints right.

The dietary requirements of the special feed; so, special feed is what we are preparing by mixing corn and soya bean should have at least 30 percent protein and at most 5 percent fiber. So, if the protein content is more than 30 it is fine, but the fiber content should not be more than 5 percent right. So, this is at most and over here it is at least ok.

(Refer Slide Time: 02:45)

Example 1

Let x_1 = kg of corn in the daily mix
 x_2 = kg of soybean meal in the daily mix

Handwritten notes: $x_1 \geq 0$, $x_2 \geq 0$

Objective: determine minimum-cost feed mix.

Minimize $z = 0.3x_1 + 0.9x_2$

Handwritten note: $x_1 = 2-3$

Subject to,

Feedstuff	kg per kg of feedstuff		Cost (\$ per kg)
	Protein	fiber	
Corn	0.09	0.02	0.3
Soyabean meal	0.6	0.06	0.9
Dietary requirements	At least 30%	At most 5%	

Handwritten notes: $x_1 + x_2 \geq 800$ Daily demand

Handwritten note: farm uses at least 800 lb of special feed daily

Handwritten note: $0.09x_1 + 0.6x_2 \geq 0.3(x_1 + x_2)$ Protein requirement in total mix

Handwritten note: $-0.21x_1 + 0.3x_2 \geq 0$

Handwritten note: $0.02x_1 + 0.06x_2 \leq 0.05(x_1 + x_2)$ Fiber requirement in total mix

Handwritten note: $-0.03x_1 + 0.01x_2 \leq 0$

Handwritten note: $x_1, x_2 \geq 0$ Lower bounds of the decision variables

Handwritten notes: 2 DV, 1 obj fn, 3 const, LP

Operations research: An introduction, H. A. Taha

So, first let us see how to model this problem. So, here the decision variables are we do not know how many kgs of corn and how many kgs of soya bean are to be mixed right. So, that is what is our decision variable; x_1 and x_2 are our decision variable right. So, our objective function is to minimize the cost ah. So, the cost of 1 kg of corn is 0.3. So, the total cost of corn would be 0.3 into x_1 . So, x_1 is what is the decision variable which is what we want from the optimization procedure and the cost of soya bean is 0.9 right.

So, the total cost of soya bean is $0.9x_2$. So, this is the cost of corn this is the cost of soya bean. So, the total cost is $0.3x_1$ plus $0.9x_2$ right. So, that is the total cost and we want to minimize this right. So, if we directly solve this problem the answer is x_1 is minus infinity x_2 is minus infinity right since minus infinity does not make sense for this mass kg of corn. So, the minimum quantity is 0 right. So, the lower bound for x_1 and x_2 is 0. So, x_1 and x_2 have to be greater than 0 right.

So, if we just solve this problem then we will get a 0 is the cost right, but we have constraints on this right. The first constraint is that the total amount of material that we have like whatever kgs of corn we take and whatever kgs of soya bean we take the sum total of that has to be at least 800; remember, this is at least. So, that is why it is greater than or equal to this is kg 800 right.

The second constraint which we had was that at least 30 percentage of whatever mixture we have should contain a protein. So, the mixture that we have is x_1 plus x_2 . So, the 30 percentage of that; so, $0.3(x_1 + x_2)$, this should be the minimum amount of protein which is to be there right. So, we put this greater than equal to constraint over here right. So, in the mixture that we take; so, this is what is required right $0.3(x_1 + x_2)$ what we actually would have is a this $1.09x_1$ because corn contains only 0.09 protein right so, $0.09x_1$.

Similarly, soya bean contains only 0.6 right. So, $0.6x_2$ so, $0.091x_1 + 0.6x_2$ will be the amount of protein in x_1 kg of corn and x_2 kg of soya bean. We want it to be greater than or equal to 30 percent of the mixture itself. So, the mixture the total amount of mixture is $x_1 + x_2$. So, the 30 percentage of that is on this right hand side right.

So, now, if we simplify this equation right so, $0.09x_1$ minus $0.3x_1$. So, this $0.3x_1$ can be taken to this left hand side right. So, $0.09x_1$ minus $0.3x_1$ that will be minus $0.2x_1$ similarly this $0.6x_2$ minus $0.3x_2$ we will give $0.3x_2$. So, this is our constraint. This is second constraint, this is our first constraint, this is the objective function.

So, we have one more constraint right which says that the total amount of fiber should be at the maximum 5 percent right. So, same thing the amount of fiber content would be $0.02x_1$ because corn has 0.02 and x_1 is the amount of corn. So, $0.02x_1$ plus $0.06x_2$ per kg of soya bean we have 0.06 kg of soya bean and now, we have taken x_2 kg of soya bean. So, the total amount of fiber would be $0.06x_2$. So, this has to be less than because here it is at most right. So, that is why it is less than equal to 5 percentage of the mixture.

So, 5 percent is 0.05 of the mixture. So, just like we had x_1 plus x_2 here also we have x_1 plus x_2 right. Similarly, this can be simplified. So, the x_1 term can be taken to the left hand side; the x_2 term can be taken to the left hand side and we will have this equation right. So, as you can see even for a simple problem we have a constraints right. So, this is called the modeling phase.

So, given a problem description so, in this slide what we had was just the problem description first this has to be modeled right. So, once we model we have the problem description in a mathematical form right. So, this is the objective function and three constraints. So, here if you see the type of problem so, we have x_1 and x_2 as the decision variable right and both of them are continuous, we do not have integer variables here right.

Let us say x_1 is equal to 2.3 kg could be permissible. Let us see the objective function. This is a linear objective function x_1 and x_2 say this constraint is also linear x_1 plus x_2 greater than or equal to 800, this constraint is also linear right to the power one and this constraint is also linear. So, in this problem we have two decision variables right one objective function right and the three constraints both of these are linear. So, the objective function and the constraints are linear right and the decision variables are continuous.

So, if you recollect our first introduction lecture right. So, this falls under the problem category of linear programming, wherein the constraints and the objective function are linear and the decision variable are continuous.

(Refer Slide Time: 08:17)

Example 2

- Reddy Mikks company produces interior and exterior paints from raw materials, M1 and M2.
- Daily demand for interior paint cannot exceed that for exterior paint by more than 1 unit.
- Maximum daily demand for the interior paint is 2 units.
- Determine optimum quantity of interior and exterior paints that maximizes total daily profit.

	Exterior paint	Interior paint	Availability
M1	6	4	24
M2	1	2	6
Profit	5	4	

Operations research an introduction, H.A. Taha

So, we look at the second example right. So, in this example we have a paint company. This example is also taken from this book Operation research an introduction by Taha. There is a paint company Reddy Mikks company. It produces two types of paint; interior paint and exterior paint right. So, to produce this paint it requires raw material 1 M1 and raw material M2. So, M1 and M2. So, for producing one unit of exterior paint it requires 6 units of M1 and 1 unit of M2 for producing this interior paint it requires 4 units of M1 and 2 units of M2.

So, the total amount of M1 that is available is 24 and the total amount of M2 that is available is 6. 5 dollars of profit is made by selling 1 unit of exterior paint and let us say 4 dollars of profit are made by selling 1 unit of interior paint right. So, the question now is how much of interior paint and exterior paint has to be produced? Right. Determined optimum quantity of

interior and exterior paints that has to be produced, so as to maximize the total daily profit right.

So, here there are two conditions, two additional conditions that the maximum daily demand for the interior paint is only 2 units right. So, there is no point of producing interior paint more than 2 units because there is no demand you cannot sell that in market right. So, that is one constraint. The other constraint is daily demand for interior paint right cannot exceed that of exterior paint by more than 1 unit right. So, we will see how to model that now.

(Refer Slide Time: 09:59)

Example 2

Let x_1 = Units of exterior paint produced daily
 x_2 = Units of interior paints produced daily

Maximize Profit, $Z = 5x_1 + 4x_2$

Subject to

$6x_1 + 4x_2 \leq 24$
 $x_1 + 2x_2 \leq 6$
 $x_2 \leq x_1 + 1$
 $x_1, x_2 \geq 0$
 $x_2 \leq 2$

Decision variables: x_1, x_2 (integers)

Objective function: Maximize Profit, $Z = 5x_1 + 4x_2$

	Ext. paint	Int. paint	Availability
M1	6	4	24
M2	1	2	6
Profit	5	4	

Constraints:

- Raw material constraints: $6x_1 + 4x_2 \leq 24$, $x_1 + 2x_2 \leq 6$
- Demand constraint: $x_2 \leq x_1 + 1$
- Bound constraints: $x_1, x_2 \geq 0$, $x_2 \leq 2$

Daily demand for interior paint cannot exceed that for exterior paint by more than 1 Unit

$x_1 \leq \dots$
 $x_2 \leq 2$

ILP

Now, the question is how much of paint 1 and paint 2 has to be produced right. So, that is the decision variable. So, we are defining the two decision variable x_1 and x_2 ; x_1 indicates units of exterior paint produce daily, x_2 indicates units of interior paints that is produced daily. So, those are our decision variables right. So, here in this case since these are units these are

integer variables there is no meaning of 2.5 units right. So, the unit has to be integer. So, it is like number of pens. So, the number of pens can only be an integer value right.

So, first we have decided the decision variables. So, in this case the decision variables are integers. So, if we produce x_1 units of exterior paint and x_2 units of interior paint then the total profit is $5x_1$ plus $4x_2$, this has to be maximized right. So, maximize profit we are indicating that by the variable Z right is $5x_1$ plus $4x_2$. So, right now this objective function is also linear. Then, we have the raw material constraint right if you are to produce 1 unit of exterior paint we require 6 units of M1 right.

So, if you have to produce x_1 units then we required $6x_1$ units. Similarly, if we produce one unit of interior paint we require 4 units of M1 right. So, if we produce x_2 units we require $4x_2$. So, the total amount of raw material M1 that is required is $6x_1$ plus $4x_2$. So, this has to be less than equal to 24 because that is the maximum amount that is available. It is not equal to 24 because it is not necessary that we use all the 24 we can use less than that also right. So, that is why we have this constraint less than equal to right.

Then similarly we can develop this constraint for a raw material 2 right. So, if we produce x_1 unit then the total amount of M2 that is required is $1x_1$ plus because of this two because it requires 2 unit per unit of interior paint. So, $2x_2$ this has to be less than equal to 6 because that is what is available right. Right now, we have one objective function which is linear; we have two decision variables which are integers and we have two constraints which are also linear ah.

So, the next constraint is this one we need to model right. Daily demand for interior paint cannot exceed that for exterior paint by more than 1 unit. So, interior paint is x_2 right. So, it cannot exceed. So, when we say it cannot exceed it has to be less than equal to right, it cannot exceed that for exterior paint. So, exterior paint is x_1 right plus 1 because by more than 1 unit is problem specification right. So, this is the other constraint x_2 is less than equal to x_1 plus 1 right.

And, then we have this constraint that x_1 and x_2 have to be greater than or equal to 0. We had this additional constraint that maximum daily demand for the interior paint is 2 units right. So, that is why we have x_2 is less than equal to 2 because x_2 indicates the units of interior paint right.

So, now we have objective function which is linear, these three constraints which are linear the lower bound of both the decision variables are 0. The upper bound of decision variable 1 is not explicitly given. The upper bound of variable x_2 is given x_2 is less than equal to 2; this x_1 may not have upper bound right, but the other set of constraints will be bounding this x_1 right.

So, in this case we have a integer linear programming, all the constraints and the objective function are linear so, linear programming and both the variables are integer. So, we have integer linear programming.

(Refer Slide Time: 13:45)

Example 3

A farmer has 2400 m of fencing and wants to fence off a rectangular field that borders a straight river. No fence is required along the river. What are the dimensions of the field that has the largest area?

Area of a rectangular region = lb

Objective: Maximize the area $\max f = lb$

Constraint: Perimeter should not exceed 2400m

$l + 2b \leq 2400$

Bounds: $l, b > 0$

$l + 2b \leq 2400$

l, b

NLP

QP

<https://studontsuccess.com/sites/default/files/optimization.pdf>

So, the next example that we will see is non-linear programming example. So, in this case there is a river right and a farmer has fencing of 2400 meter. So, this is the amount of fencing that the farmer has right and he wants to have a rectangular area right. So, the question is what should be the length of the rectangular area and what should be the breadth of the rectangular area. So, what is this l , what is this b ?

The farmer does not know what has to be l and the farmer does not know what should be the breadth right. But, the total amount of fencing material that he has is 2400 meter right and fencing is not required in the side right. So, the fencing is required only on the three sides over here, over here and over here. So, the total amount of fencing is l plus $2b$ right.

So, this should be less than equal to 2400 right. So, here also remember he does not need to use all the 2400 meters of fencing, but at the maximum he has 2400 meters of fencing right.

So, that is our constraint right and these two are logical constraints that l and b has to be greater than 0. The farmer wants to maximize the area; area is given by l into b .

So, the objective function is now non-linear right. In fact, it is quadratic; l is for length; b is for breath. So, there are no explicit lower and upper bounds for l and b . So, the problem is to find out the value of l comma b ; l and b such that l into b is maximum and it also satisfies this constraint. So, over here if we see this is a linear constraint l plus $2b$ is less than equal to 2400. This is a linear constraint so, but this objective function is quadratic right. So, we will call it as non-linear programming problem or a QP constraint QP because there is a constraint involved.

Now, we have seen three problems right. The first problem was a linear programming problem, the second was an integer linear programming problem the third one was a non-linear programming problem. In all these three problems the problems were extremely simple, but even then they were multiple constraints involved, but the techniques we have seen so far are for unconstrained optimization problem. So, now, the question is how do we accommodate the constraints right into this metaheuristic techniques

(Refer Slide Time: 16:09)

Constraints

- Inequality constraints (usually resource/requirement constraints)
 - In general denoted by $g(x) \leq 0$
 - Conversion from one form to the other by multiplying by -1
- Equality Constraints (usually material and energy balances)
 - In general denoted by $g(x) = 0$
- Feasible solution: Satisfies all the constraints ✓
- Infeasible solution: Does not satisfy at least one constraint

Multi-objective optimization using evolutionary algorithms, K. Deb

So, let us look at the types of constraints that we can generally have. So, we can have inequality constraints as well as equality constraints right. So, inequality constraints are usually denoted by g of x is less than equal to 0. So, if we have a constraint which has greater than or equal to 0 let us say $x_1 + x_2$ has to be greater than or equal to 0 we have seen that this can be converted into this less than equal to form by multiplying with the minus 1 on both sides right. In equality constraints we have g of x is equal to 0 right.

So, solutions we classify in to two types – feasible solution and infeasible solution. So, feasible solution is one which satisfies all the constraints and we are interested in feasible solutions; infeasible solutions are those which violate at least one constraint. So, if there is a solution which has a very good objective function value, but even if it is not satisfying one constraint

that solution is not useful for us because we are interested in feasible solutions. And, among feasible solution the one that is the best is what we are looking for.

(Refer Slide Time: 17:06)

Constraints

- Any feasible solution is preferred to any infeasible solution
- Among two feasible solutions, the one having a better objective function value is preferred
- Among two infeasible solutions, the one having a smaller constraint violation is preferred
- A solution $x^{(i)}$ is said to be *constrain - dominate* a solution $x^{(j)}$:
 - Solution $x^{(i)}$ is feasible and solution $x^{(j)}$ is not.
 - (S1 constrain-dominate S2)
 - Solutions $x^{(i)}$ and $x^{(j)}$ are both infeasible, but solution $x^{(i)}$ has a smaller constraint violation.
 - (S3 constrain-dominate S2)
 - Solutions $x^{(i)}$ and $x^{(j)}$ are feasible and solution $x^{(i)}$ dominates solution $x^{(j)}$ with respect to objective function.
 - (S4 constrain-dominate S4)

	S1	S2	S3	S4
x	0.7	0.4	0.65	0.8
f	0.35	0.2	0.325	0.4
violation	0 ✓	0.29	0.0275	0 ✓

Min $f = \frac{x}{2}$
 s.t. $x^2 \geq 0.45$
 $0 < x < 1$

Multi-objective optimization using evolutionary algorithms, K. Deb

These are the three conditions that we need to satisfy that given any two solution right a feasible solution is preferred to any infeasible solutions. So, irrespective of the objective function value and if we have two feasible solutions right so, both the solutions satisfy the all the constraints, then the one having a better objective function value is preferred right. And, in case if we have both solutions which are infeasible right the one which has a smaller constraint violation is preferred right.

So, here if we see in these three rules we are actually giving more weightage to feasibility then the objective function right. So, objective function plays a role only if both the solutions are feasible right and if you have a feasible solution and in feasible solution we do not worry about

what is the objective function value, we select the feasible solution. And, in event wherein we have two infeasible solutions we will be selecting the one which has lower constraint violation.

So, we are taking two solutions x_i and x_j ; x_i is said to be constrained dominate x_j if x_i is feasible and x_j is not. So, that corresponds to this condition. We have a feasible solution and an infeasible solution; so, the feasible solution is to be preferred. So, x_i is set to constraint dominate x_j right and if it happens that both x_i and x_j are infeasible right.

So, we are looking at the third case now. So, if both of them are infeasible then x_i is said to constraint dominate x_j if x_i has a smaller constraint violation than x_j right, similarly this corresponds to the second case where in both x_i and x_j are feasible solutions. x_i is set to dominate x_j if it has a better objective function value than x_j right. So, it is the same three rules we are explaining again with taking two solutions right.

So, now, let us look at an example. Let us say we have these four solutions S_1, S_2, S_3, S_4 right. This is the decision variable x . So, let us say it is a single variable optimization problem given over here that we want to minimize f which is nothing, but x by 2 and we want to satisfy this constraint x square should be greater than or equal to 0.45 and x is between 0 and 1. So, x has to be greater than 0 and it has to be less than 1 right. So, these are the four values of the decision variable x corresponding to solution 1, solution 2, solution 3, solution 4.

These are the here corresponding objective function values right and since we know this is 0.7 we can check whether 0.7 square is greater than or equal to 0.45. In this case it satisfies this constraint so, the violation is 0. Even in this case the you can check that 0.8 square would be greater than or equal to 0.45 right. So, this also satisfies whereas, in these two cases x is equal to 0.4 and x is equal to 0.65, if you take x square it will not be greater than or equal to 0.45 right. So, it will be violating that constraint by this much amount by 0.29 and 0.0275 right.

So, now let us say this is our population right and if we are to select from this four solutions, let us say we are working with teaching learning based optimization this is our population right and we are now supposed to select the teacher and we are working with the minimization or problem right. So, we are not supposed to go by this point to right because it has a better

objective function value, but it also violates the constraint right. So, in that case we need to first find out what are the feasible solutions.

The feasible solution here is S1 and S4 between S1 and S4 we would select S1 because it has a better objective function value right. So, here if we see S1 constraint dominates S2. So, let us take S1 and S2 right. So, S1 is feasible S2 is infeasible; so, obviously, S1 constraint dominates S2. In the second case, S3 and S2 we are comparing. So, in this case if you see both of them are in feasible solution right. So, that is why it is listed over here, but the violation over here in S3 is less compared to in S2 right. So, that is why we say S3 constraint dominates S2 right.

And, the last cases between S4 and S1 right. So, in this case if we see that S1 actually constraint dominates S4 because S1 has a better objective function value. So, this has to be S1 constraint dominates S4 right so, because we need to go by the objective function value.

So, these are the three rules that any constraint handling mechanism that we will study should ideally follow these three rules that between two feasible solution we will select the one which has better objective function value between two infeasible solutions we will select the one which has lower violation of the constraints; between two feasible solutions we will select the one which has better objective function value. And, between a feasible and infeasible solution the choice is very clear that we will not worry about the objective function value, we will go with the feasible solution.

(Refer Slide Time: 22:08)

Preserving feasibility of solutions

Minimize $3x_1 + 2x_2 + x_3$
s.t. $2x_1 + x_2 + x_3 \geq 5$
 $x_1 + x_2 + 0.5x_3 \leq 4$
 $x_1 - x_2 = 2$
 $x_1, x_2, x_3 \in \mathbb{R}$

(Handwritten: m_1, m_2, m_3 next to constraints; $3, 3$ below)

$x_2 = x_1 - 2$

(Handwritten: $\eta, k, \eta - k$ around the equation)

Minimize $5x_1 + x_3 - 4$
s.t. $3x_1 + x_3 \geq 7$
 $2x_1 + 0.5x_3 \leq 6$
 $x_1, x_3 \in \mathbb{R}$

(Handwritten: $2, 2$ below)

(Handwritten: $5x_1 + 2(4 - x_1) + x_3$ above the arrow)

(Handwritten: m_1, m_2, m_3 and $= 5$ in a box to the right)

Multi-objective optimization using evolutionary algorithms, K. Deb

Sometimes we can reduce the number of constraints that we have. So, for example, consider this problem this is a three variable problem we are supposed to find out the values of x_1, x_2, x_3 right such that this function $3x_1 + 2x_2 + x_3$ has the minimum possible values and we need to satisfy these three constraints right. So, we have two inequality constraints and one equality constraint. So, the value of x_1 and x_2 should be such that $x_1 - x_2$ should be equal to 2 and $x_1 + x_2 + 0.5x_3$ should be less than equal to 4 and $2x_1 + x_2 + x_3$ should be greater than or equal to 5.

So, in this case we can directly take this problem right and say that we are having three decision variables or what we can do is we can exploit this constraint right and reduce the number of decision variables. So, for example, this constraint we can rewrite it as x_2 is equal

to $x_1 - 2$ or we could have chosen as $x_1 = 2 + x_2$ right either we could have used this or we could have used this.

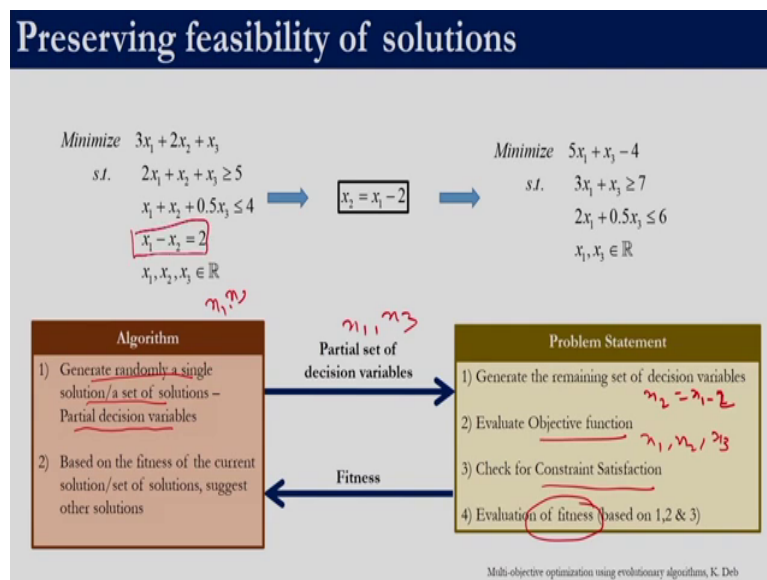
Now, I can use this relation to completely eliminate x_2 from all other equations right instead of writing $3x_1 + 2x_2 + x_3$ we can say $3x_1 + 2(x_1 - 2) + x_3$ right. So, that should work out to $5x_1 + x_3 - 4$ similarly we can get rid of x_2 from this equation we can get rid of x_2 from this equation right.

So, this problem which had three decision variable three constraints has been reduced to two decision variable with two constraints right. So, we exploited this equality constraint. So, we can choose to solve this problem instead of this problem right. So, here the number of decision variable is less so, our search spaces also less right. So, the algorithm is typically expected to perform better on this problem than on this problem and once we solve this problem we will get the values of x_1 and x_3 right. Once we have the optimal value of x_1 and x_3 from any of the five algorithms which we have studied right then we can actually calculate x_2 from this relation right. So, x_2 can be obtained at the end.

So, in this case we have implicitly handled this constraint right we use this constraint to completely eliminate one variable right. So, that way we have also taken care of that constraint and while taking care of that constraint we have also managed to eliminate one decision variable if there is an optimization problem with n decision variables and if there are k linear equations right. So, this is a linear equation if there are k linear equation then we can derive k relations right and use them in the optimization problem to convert this n variable optimization problem into a problem with $n - k$ decision variables right.

So, this is one way of handling the constraints. So, if it is an explicit equation we can do it right let us say if we have an implicit equation which says let us say $x_1^2 + x_2^2 + x_3 = 5$ right. In this case what we can do is we can completely eliminate one variable right we can have just two decision variable x_1 and x_2 right and once we have these two variables x_1 and x_2 we can actually solve this equation that $x_1^2 + x_2^2 + x_3 - 5 = 0$ and get the appropriate value of x_3 right.

(Refer Slide Time: 25:38)



So, either we can choose to eliminate variables like this or if we are not interested in eliminating variables we can still have those equations as such. So, this figure you might remember right. So, the algorithm generates randomly one solution or a set of solutions right. So, previously whatever we had seen in that for unconstrained optimization problems it was generating all the decision variables right. So, here we can say that it would generate only partial decision variables.

So, for example, in this case it will generate only x_1 comma x_3 right. So, this partial set of decision variable is sent through the problem statement. Here we would have this equation that x_2 is equal to x_1 minus 2 right and calculate x_2 right. Once we know x_2 x_1 x_3 is given by the algorithm x_2 is generated over here. So, we can calculate the objective function. We can check for the constraint satisfaction right whether it is satisfying these two constraints

or not because now we know all the values x_1, x_2, x_3 are known right. So, we can check for constraint satisfaction.

Depending upon whether it is satisfying constraints or not or how many of the constraint it is satisfying or how much is the violation we will see procedure to come up with this fitness function value which is what will be returned back to the algorithm right. So, this way we can make use of equality constraints to reduce the number of decision variables instead of explicitly handling them as constraints.

(Refer Slide Time: 27:01)

Constraint Handling

➤ Case 1: Unconstrained optimization problem

- Fitness function = Objective function (ABC)

Solution	Objective function	Fitness function
S_1	5	5
S_2	7	7

➤ Case 2: Constrained optimization problem (Minimization case)

- Fitness function = Objective function + λ (Penalty) - λ Penalty

Solution	Objective function	Feasibility
✓ S_1	5	Yes
✓ S_2	7	No

$$\text{Min } f(x)$$

$$\text{s.t. } g(x) \geq b$$

Multi-objective optimization using evolutionary algorithms, K. Deb

So, in unconstrained optimization problem right whatever the objective function value was that was the same as fitness function right. So, for all the problems that is what we did except for ABC right because in ABC we had that fitness function was inversely related to the objective function right. Otherwise in all the four algorithms the fitness function was nothing,

but the objective function value that is true only in the case of unconstrained optimization problem.

When we have constrained optimization problem what we will do is we will add a penalty right. So, what we are doing is for those solutions which does not satisfy the constraint, we will calculate something called us penalty right and we will add it to the objective function, this is true for minimization problem right. For maximization problem it has to be objective function minus lambda times penalty right.

So, what we are basically doing is we are deteriorating the objective function value using the penalty right. We are adding the penalty because the solution is not satisfying one or more constraints right. So, for example, let us look at these two solutions solution 1 and solution 2. Let us say for some arbitrary objective function the objective function value of solution 1 is 5; the objective function value of solution 2 is 7 and let us say this is a feasible solution this satisfies whatever the constraint and this does not satisfy the constraint right.

(Refer Slide Time: 28:23)

Constraint Handling

➤ Case 1: Unconstrained optimization problem

- Fitness function = Objective function

Solution	Objective function	Fitness function
S ₁	5	5
S ₂	7	7

7 + 20(2)
= 47

➤ Case 2: Constrained optimization problem (Minimization case)

- Fitness function = Objective function + λ (Penalty)

$\text{Min } f(x)$
 $\text{s.t. } g(x) \geq b$

Solution	Objective function	Feasibility	Violation	Fitness function
S ₁	5	Yes	0	5
S ₂	7	No	2	47

$\lambda = 20$

Multi-objective optimization using evolutionary algorithms, K. Deb

And, the violation is 0 because the solution is feasible. So, it is not violating any constraint so, the violation is 0. Let us say for a particular constraint the violation is 2 right. So, in this case if we take lambda is equal to 20 right and we apply this equation objective function plus lambda times penalty right. So, here since the violation is 0 right so, this is violation or penalty right. So, since this is 0, if we apply this equation this 5 will continue to remain as 5 right whereas, this 7 right will become 47 because now the fitness function is 7 plus 20 times 2. So, this is 40 plus 7, 47; so, that is this particular value.

So, once we calculate the fitness function, we do not need to worry about whether the solution is feasible or not right because the fitness function actually captures whether the solution is feasible or not.

(Refer Slide Time: 29:21).

Constraint Handling

➤ Case 1: Unconstrained optimization problem

- Fitness function = Objective function

Solution	Objective function	Fitness function
S ₁	5	5
S ₂	7	7

2 + 20(0.1)
4

➤ Case 2: Constrained optimization problem (Minimization case)

- Fitness function = Objective function + λ (Penalty)

Solution	Objective function	Feasibility	Violation	Fitness function
S ₁	5	Yes	0	5
S ₂	7	No	2	47
S ₃	2	No	0.1	4

λ = 20

Min f(x)
s.t. g(x) ≥ b

Multi-objective optimization using evolutionary algorithms, K. Deb

So, let us consider this case. We have one more solution S3 and let us say it is also not a feasible solution right and the violation of some constraint is to the magnitude of 0.1 right. So, now if we go just by the objective function this solution would be picked up. So, for example, remember we were identifying the teacher by looking at the least value right. So, in that case this will be picked up. So, that is why we do not want to work with objective function when we have constrained optimization problems, we want to work with fitness function right.

If you have taken lambda value as 20 in this case what will happen is it will be 2 plus lambda is 20 into violation is 0.1 right so, this will become 4. Actually, between these three solutions which one do we want to select according to the three rules which we have seen? Both of these are infeasible right. So, this is the one which has to be selected, but now within TLBO if

you look at what is the minimum value among these three solutions right, this one would be picked up S3 would be picked up.

So, S3 is getting picked up, but what we want is S1 to be picked up right. So, this happens because the lambda value is not appropriately chosen over here.

(Refer Slide Time: 30:30)

Constraint Handling

➤ Case 1: Unconstrained optimization problem

- Fitness function = Objective function

Solution	Objective function	Fitness function
S ₁	5	5
S ₂	7	7

$7 + 200(2) = 22$
 $2 + 200(0.1) = 22$

➤ Case 2: Constrained optimization problem (Minimization case)

- Fitness function = Objective function + λ (Penalty)

$$\text{Min } f(x)$$

$$\text{s.t. } g(x) \geq b$$

Solution	Objective function	Feasibility	Violation	Fitness function $\lambda = 20$	Fitness function $\lambda = 200$
S ₁	5	Yes	0	5	5
S ₂	7	No	2	47	407
S ₃	2	No	0.1	4	22

Multi-objective optimization using evolutionary algorithms, K. Deb

So, if we choose a lambda value of let us say 200 instead of 20 which we discussed previously if we take 200 then this solution will continue to be 5 because it is a feasible solution. No matter what is the value of lambda the penalty or the violation is 0 right. So, the objective function is the same as fitness function value whereas, this 7 will now become 7 plus 200 into 2 right. So, this becomes 407 and this 2 becomes 2 plus lambda is 200 into 0.1 right.

So, this will become 22 right. So, now, if we pick up the best solution from this fitness function value right so, the best value is 5 right. So, S1 would get picked up which is what we wanted in the first place right. So, this lambda helps us to handle the constraints right, but the value of lambda has to be appropriately chosen. If a poor value of lambda is chosen, then that rule that between an infeasible solution and a feasible solution we want to pick up the feasible solution would get violated right. So, this lambda has to be appropriately selected.

So far what we were doing is we just told you that take the violation as 0 because here you can understand it is 0 because the solution is feasible. In these two cases we just said that take the violation is 2 and 0.1, so, now, let us take an example and see how to actually come up with this violation value.

(Refer Slide Time: 31:57)

Constraint Handling

➤ Given Problem	$\begin{aligned} \text{Min } & f(x) \\ \text{s.t. } & g_j(x) \geq b \quad j=1,2,\dots,J \\ & x_l \leq x_i \leq x_u \quad i=1,2,\dots,I \end{aligned}$	
➤ Normalize Constraints	$\underline{g}_j(x) \geq 0 \quad j=1,2,\dots,J$	
➤ Estimate the extent of violation	$\omega_j(x) = \begin{cases} \underline{g}_j(x), & \text{if } \underline{g}_j(x) < 0 \\ 0, & \text{otherwise} \end{cases}$	

Multi-objective optimization using evolutionary algorithms, K. Deb

So, let us assume that this is the problem that we are given minimize f of x right and there are j constraints right. So, this g of j of x is greater than or equal to b_j . So, j constraints are given and x_i we have i variables right and all of them have a lower and upper bound right.

So, first thing is to normalize this constraint right. So, normalize this constraint in the sense like divide the coefficients in this constraint by b_j . So, we will have b_j by b_j , so, this will become 1. We want all the constraints to be of this form ah . So, the first step is to normalize the constraints.

Once we have normalized the constraint, we need to find out the extent of violation. So, in this case remember when we are working with metaheuristic algorithms, we would have the decision variables so, the value of the decision variables right. That is what the algorithm gives, no matter which algorithm you are working with it will give you the value of the decision variables. So, once we know the value of the decision variable we can actually check whether the constraint is satisfied or not. So, this is something that we can check.

So, for example, let us say the constraint is x_2 square plus x_3 should be greater than or equal to 0. So, now we will have the value of x_2 and x_3 so, we can actually check for this condition right. So, let us say if this is 2, this is 8 we can actually check for this condition. So, we will calculate this and if it happens to be greater than or equal to 0, then this ω_j is taken to be 0, else if it is less than 0 so, for example, let us say this happens to be minus 5 for some value of x_2 and x_3 , then this ω_j is equal to absolute of this violation.

So, what we are calculating is the violation. So, the absolute of that right because this is going to be a negative value right it comes into this part only if g of x is less than 0 right. So, we take the absolute value of that that is how ω_j is calculated right. So, if we have ten constraint we will have ten such ω_j values.

(Refer Slide Time: 33:52)

Constraint Handling

➤ Given Problem $Min f(x)$
 $s.t. g_j(x) \geq b \quad j=1,2,\dots,J$
 $x_i \leq x_i \leq x_u \quad i=1,2,\dots,I$

➤ Normalize Constraints $\underline{g}_j(x) \geq 0 \quad j=1,2,\dots,J$

➤ Estimate the extent of violation

➤ Estimate the total penalty

➤ Estimate the fitness function

Penalty factor (R_m): Usually an order of magnitude than the objective function value

1	5
2	3
3	15
	23

$$\omega_j(x) = \begin{cases} |g_j(x)|, & \text{if } \underline{g}_j(x) < 0 \\ 0, & \text{otherwise } \geq 0 \end{cases}$$

$$\Omega(x) = \sum_{j=1}^J \omega_j(x) \quad 23$$

$$F(x) = f(x) + R_m \Omega(x)$$

Multi-objective optimization using evolutionary algorithms, K. Deb

The next thing is to estimate the total penalty. So, the total penalty is nothing, but taking the summation of all the individual penalties right. So, let us say first constraint we are violating by 5; second constraint we are violating by 3; third constraint we are violating by let us say 15 right. So, this capital omega is nothing, but a scalar value which will be 23 which is nothing, but the summation of this right.

So, once we have calculated this omega, the fitness function F is the objective function right plus the violation right multiplied by R_m . So, R_m can be a constraint value. So, this is what we saw as lambda in the previous slide right. So, usually this penalty factor is one order of magnitude higher than the objective function value, so that when we are comparing a feasible solution and an infeasible solution, the feasible solution actually wins over the infeasible

solution right. To simplify you can just consider this as x right you do not need to look at x of i.

So, the R m has to be a magnitude higher than the objective function, since we are working with minimization problems it has to be a magnitude higher than the objective function.

(Refer Slide Time: 35:01)

Evaluation of Fitness function

Minimize $f(x) = \frac{1+x_2}{x_1}$

s.t. $g_1(x) = 9x_1 + x_2 \geq 6$
 $g_2(x) = 9x_1 - x_2 \geq 1$
 $0.1 \leq x_1 \leq 1$ $0 \leq x_2 \leq 5$

Normalized Constraints

$$\underline{g}_1(x) = \frac{9x_1 + x_2}{6} - 1 \geq 0$$

$$\underline{g}_2(x) = 9x_1 - x_2 - 1 \geq 0$$

$R_m = 20$

$$\omega_j(x) = \begin{cases} \frac{|\underline{g}_j(x)|}{\underline{g}_j(x)}, & \text{if } \underline{g}_j(x) < 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\Omega(x) = \sum_{j=1}^J \omega_j(x)$$

$$F(x) = f(x) + R_m \Omega(x)$$

Solution	Decision Variables		Objective function (f)
	x_1	x_2	
S1	0.31	0.89	6.10
S2	0.38	2.73	9.82
S3	0.22	0.56	7.09
S4	0.59	3.63	7.85
S5	0.46	2.9	8.48
S6	0.66	4.11	7.74

Multi-objective optimization using evolutionary algorithms, K. Deb

So, now let us consider an example right. So, let us say this is our objective function. We have two decision variables right x 1 and x 2 right and there are two constraints which we have is 9x 1 plus x 2 should be greater than or equal to 6 and 9x 1 minus x 2 should be greater than or equal to 1 right.

So, we have an objective function involving two decision variables, we have two constraints right. These are not equally constraint. If it had been an equality constraint and since this is

linear, we could have used it to eliminate a variable right. So, right now we cannot do that right and the two decision variables x_1 and x_2 have their own bounds. So, x_1 has to be between 0.1 and 1 and x_2 has to be between 0 and 5 right.

So, first step is to normalize the constraint right. So, this is what we want to do. So, over here this is normalized constraint right. So, how do we normalize is $9x_1$ plus x_2 divided by 6. So, we will have a 1 over here right if we take 6, then we bring the one to the left hand side so that we get the constraint in this form. So, $9x_1$ plus x_2 by 6 minus one should be greater than or equal to 0 and similarly we can transform the other constraint right. So, these two constraints are nothing, but the normalized form of these two constraints right.

So, now, let us consider these six solutions. Since it is a 2 variable problem let us consider the values of x_1 and x_2 as given over here right. So, this may be our initial population right. So, for initial population the first thing that we need to do is calculate the objective function right. So, this is the objective function which is nothing, but plugging in these values x_1 and x_2 in this equation f of x is equal to 1 plus x_2 by x_1 . So, we will be able to calculate these six values right.

(Refer Slide Time: 36:48)

Evaluation of Fitness function

Minimize $f(x) = \frac{1+x_2}{x_1}$

s.t. $g_1(x) = 9x_1 + x_2 \geq 6$
 $g_2(x) = 9x_1 - x_2 \geq 1$
 $0.1 \leq x_1 \leq 1, 0 \leq x_2 \leq 5$

Normalized Constraints

$$\underline{g}_1(x) = \frac{9x_1 + x_2}{6} - 1 \geq 0$$

$$\underline{g}_2(x) = \frac{9x_1 - x_2}{1} - 1 \geq 0$$

$R_m = 20$

$$\omega_j(x) = \begin{cases} \underline{g}_j(x) & \text{if } \underline{g}_j(x) < 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\Omega(x) = \sum_{j=1}^m \omega_j(x)$$

$$F(x) = f(x) + R_m \Omega(x)$$

6.10 + 20(0.39)

Solution	Decision Variables		Objective function (f)	Feasibility		Violations		Total penalty Ω	Fitness (F)
	x_1	x_2		g_1	g_2	ω_1	ω_2		
S1	0.31	0.89	6.10	✗	✓	0.39	0.39	13.90	
S2	0.38	2.73	9.82	✓	✗	0	0.31	16.02	
S3	0.22	0.56	7.09	✗	✓	0.58	0.58	18.69	
S4	0.59	3.63	7.85	✓	✓	0	0	7.85	
S5	0.46	2.9	8.48	✓	✓	0	0	8.48	
S6	0.66	4.11	7.74	✓	✓	0	0	7.74	

Multi-objective optimization using evolutionary algorithms, K. Deb

So, on the basis of this if we were to pick up the best solution right, we might pick up solution 6.10 right. So, but if you plug in these values of x_1 and x_2 and plug into these two normalized constraint, you will see that these two values do not satisfy g_1 right. So, this is g_1 bar, g_2 bar they do not satisfy this constraint. So, the first solution does not satisfy g_1 , but it satisfies g_2 right. Similarly, the second solution satisfies g_1 , but does not satisfy g_2 ; the third solution does not satisfy g_1 , but satisfies g_2 whereas, the rest of the solutions right satisfy both the constraints right.

So, the first three solutions are infeasible the last three solutions are feasible. So, if you were to select what is the best solution among this? Right We would actually go with 7.74 right though it has a poor objective function value than S1, it is a feasible solution whereas, S1 is an infeasible solution right. So, this is what we want to happen right, but what we can actually supply to the algorithm is just a fitness function value corresponding to each of this solution

and then the algorithm is going to select the best solution. And, we want the algorithm to actually select S6 and not S1 right. So, the only control that we have is over the fitness function which we are passing to the algorithm right.

So, what we can do is calculate this omega 1 and omega 2 right. So, for these solutions wherever it is infeasible so, this is 0, this is also 0, this is 0, 0, 0, 0, 0, 0. So, wherever it is feasible the omega value would be 0, because it is satisfying the constraint right. So, the constraint is greater than or equal to 0, so, that is why we take omega as 0 right. But, for the other three cases we would be able to calculate the amount by which it is violating right.

So, once we have both of these omega 1 and omega 2 right that is calculated using this equation right subsequently we can just sum both the penalties 0.39, 0.31, 0, 0, 0 by adding it is respective omega 1 and omega 2 right. So, this is the total penalty value. So, we have now estimated this right and for this case we are taking R m as 20. So, this R m has to be selected for a particular problem right. So, it has to be selected in a way such that all the three rules which we have discussed right should be satisfied.

So, now, we have the objective function here and we have the total penalty here and the equation for calculating fitness is here right. So, it is 6.10 plus 20 into 0.39. So, this will work out to be 13.90 right. So, similarly the fitness function for all of these solutions can be calculated right.

So, since these three solutions are feasible, it satisfies both the constraints. So, both the omegas would be 0 for the feasible solutions. So, the summation of them is also going to be 0 right. So, the objective function is nothing, but the fitness function for a feasible solution right. So, that is why for these three solutions you can directly tell the fitness function without necessarily calculating w 1, w 2 or omega.

Now, among these six solutions right, so, if our algorithm is to pick it will pick 7.74 because that is the least value in this so, which is what we actually wanted the algorithm to do. We wanted the algorithm to pick S6 because it is a feasible solution and it has the best objective

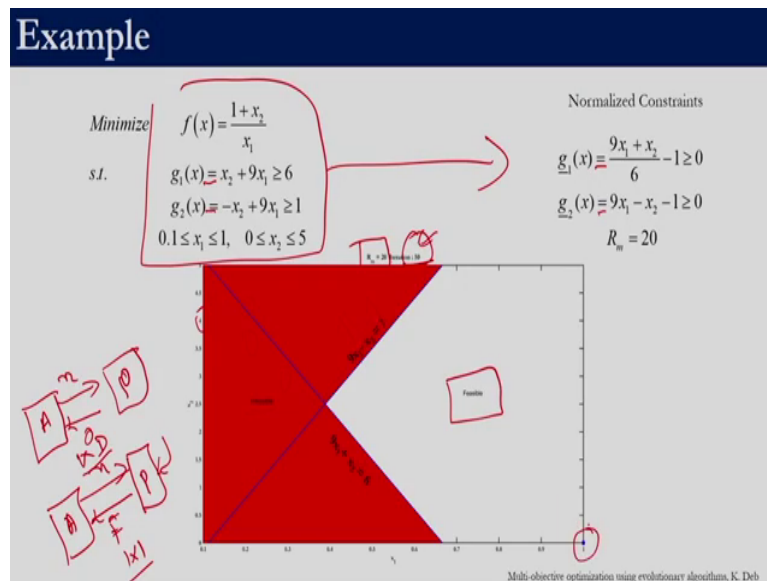
function value among the feasible solutions right. So, this is what we will return in constraint optimization problem.

So, in unconstrained optimization problem we do not need to worry about constraint because there are no constraint there may only be bound constraints which can be taken care by the corner bounding strategy which we discussed right. For unconstrained optimization problem there are no constraints right. So, objective function is the same as fitness function whereas, for constrained optimization problem right we calculate the amount of violation and we add it to the objective function. For a minimization problem we add it to the objective function with a factor lambda or R_m .

The value of lambda or R_m should be such that a feasible solution should be preferred over an infeasible solution. If you remember we had one example in the previous slide where in we had chosen a lambda value of 20 and that did not work out for the solution S3 right.

So, we should ensure that a feasible solution always wins over an infeasible solution right. So, in all these cases we knew the solutions right. So, it was easy to choose a lambda value which actually worked out for an arbitrary problem the selection of lambda should be carefully done.

(Refer Slide Time: 41:32)



So, the same problem right so, we solved using TLBO right. So, it is the same problem over here, this is the normalization of those constraints and then we applied TLBO. Remember, we do not need to do any change in the algorithm right. So, previously what we were doing in unconstrained problem is we were receiving x and we were sending the objective function value o right. Now, communication is the same thing right we will receive x and what we are sending back is the fitness function value.

So, this is the algorithm, this is the objective function or the problem, this is all algorithm the problem. So, whatever the constraints are there are taken care over here right. So, still we are just passing a scalar value for a particular solution. So, if you get one solution we would still be passing a just a 1 cross 1 right. So, if this is 1 cross D, we will just pass a 1 cross 1. So, for this problem R_m is equal to 20 right. So, here is the iteration counters. So, this is a video. So,

once we play you will see this iteration counter changing, these are our initial five solutions right.

So, this is the infeasible region right. So, those both the constraints we have plotted right. So, $9x_1 - x_2 = 1$ is this line. So, we have plotted both those constraints right and this is the infeasible area right. So, initially all the five solutions which we randomly generated are in the infeasible zone whereas, this is our feasible zone. So, what we are expecting is as the algorithm proceeds, we expect these solutions to move right to a feasible region and once they have moved to a feasible region they would start finding better solutions right.

So, as you can see all the five solutions are moving right so, they have moved now into the feasible region right. So, along the feasible region they are still searching for a better fitness function value. So, we run it for 50 iterations, you can see the iteration counter over here right. So, at the end of 50 iterations all of the solutions seem to have converged to a particular point right.

(Refer Slide Time: 43:33)

Constraint Handling

➤ Using hard penalty

$$F(x) = f(x) + \sum_{j=1}^J M_j$$

$$M_j = \begin{cases} \lambda_j & \text{if } g_j(x) < 0; \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} & \text{Min } f(x) \\ \text{s.t. } & g_j(x) \geq 0 \\ & h_k(x) = 0 \end{aligned} \quad \forall j=1,2,\dots,J$$

$f = f + \gamma$

$\gamma_1 + \gamma_2 \geq 0$

$(x_1) \geq (x_2)$

If $x_1 = x_2$

$\gamma = 10^5$

and

Multi-objective optimization using evolutionary algorithms, K. Deb

In the previous case what we did is, we added a penalty which was in proportion to the violation right. But, if some condition is not satisfied we can add a constraint value right no matter how much the violation is right we can add a constraint value. So, we have J conditions; for each condition which is not satisfied we can assign a penalty which cannot be added to the objective function to determine the fitness function. So, for example, it is not necessary in metaheuristic techniques that all the constraints have to be this form. Let us say x 1 plus x 2 is greater than or equal to 0.

So, this constraint can be handled by metaheuristic techniques right, but metaheuristic techniques can also handle these type of constraints. So, I want let us say we have two variables x 1 and x 2 and this is the condition that we have right. So, basically this is a

constraint that the value of x_1 and x_2 should not be the same. So, if the value of x_1 and x_2 is same right, we want to assign a penalty right.

So, in that case we can merely write this condition that if x_1 is equal to x_2 we will say λ is equal to let us say 10^5 and end. If we have M such conditions right x_1 naught equal to x_2 , x_1 naught equal to x_3 , x_1 naught equal to x_4 , we will have that many values of λ right.

And, here if we see we are directly assigning a value. So, it does not matter what is the value of x_1 and x_2 . If they are equal we add a constraint value so, that we will be calling it as hard penalty right. So, irrespective of the amount of violation we add a penalty. So, this λ value itself is selected in such a way that we do not need to include λ over here right.

So, if this is sufficiently big such that whatever the role this λ was playing here, right in the previous slide we had seen F is equal to f plus λ times the penalty the total penalty right. If it is calculated in such a way that it is able to capture the feature that a feasible solution will always win over an infeasible solution, irrespective of the objective function value it is fine.

(Refer Slide Time: 45:36)

Constraint Handling

➤ Using hard penalty

$$F(x) = f(x) + \sum_{j=1}^J M_j$$

$$M_j = \begin{cases} \lambda_j, & \text{if } g_j(x) < 0; \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{array}{l} \text{Min } f(x) \\ \text{s.t. } g_j(x) \geq 0 \\ h_k(x) = 0 \end{array} \quad \forall j=1,2,\dots,J$$

*k=1,2,...,K

< g(x) >

➤ Using penalty function

$$F(x) = f(x) + \lambda \sum_{j=1}^J \langle g_j(x) \rangle + \sum_{k=1}^K h_k(x)$$

$$\langle g_j(x) \rangle = \begin{cases} |g_j(x)|, & \text{if } g_j(x) < 0; \\ 0, & \text{otherwise} \end{cases}$$

$x_1 + x_2^2 + x_3^3 = 5$

$\frac{x_1 + x_2^2 + x_3^3 - 5}{5} = 0$

$g(x) = -2$
 $\langle g(x) \rangle = |-2| = 2$

$g(x) = 2$
 $\langle g(x) \rangle = 0$

Multi-objective optimization using evolutionary algorithms, K. Deb

So, this is the one that we have discussed in the previous slide right using a penalty function. In the previous slide we had considered only the case of inequality constraints. So, similarly we can also consider for equality constraints. Let us say we have a constraint x_1 plus x_2 square plus x_3 cube is equal to let us say 5. So, we will convert this constraint into x_1 plus x_2 square plus x_3 cube by 5 minus 1 equal to 0 right.

So, once we have the values of x_1 , x_2 , x_3 from the algorithm we will calculate this, this left hand side right. If it is equal to 0, then we do not assign penalty; if it is not equal to 0, whatever that value is that is considered to be the amount of violation. Here if you see we have this operator over here, so, this basically does what we have discussed previously right. So, this operator is nothing, but it will assign a value of 0, if the constraint is satisfied; if the constraint is not satisfied, so, if it is a negative value right, then it will take the absolute of this.

So, for example, consider this case that g of x is equal to minus 2, since it is less than 0 it actually violates the constraint. So, this bracket operator will take just the absolute of this. So, absolute of minus 2 is equal to 2, but if g of x was actually 2 right so, then this bracket operator will return 0. So, this bracket operator is doing exactly the same as we have done in the previous slide for calculating w right. So, w was absolute of g_j of x if it is violating the constraint else it was 0 right. So, compactly instead of writing this big expression right we can directly use the bracket operator over here and some those penalty.

So, there are capital J number of inequality constraints and capital K equality constraints. So, the violation in each is to be added to the objective function with an appropriate lambda value right this is either lambda or R_m and that is how we determine the fitness function value right. So, here it has to be for j is equal to 1 to J and for k is equal to 1 to K . So, the number of inequality constraints and equality constraints need not be same right. So, for equality constraint we are using this index k ; for inequality constraints we are using this index j .

(Refer Slide Time: 47:57)

Constraint Handling

➤ Using dynamic penalty function

$$F(x) = f(x) + (C(t))^\alpha \left[\sum_{j=1}^J (g_j(x))^\beta + \sum_{k=1}^K |h_k(x)|^\beta \right]$$

C, α, β : user defined parameters
 t : iteration counter

$\beta = 2$

lb
ub
f
 α
 β
C

Multi-objective optimization using evolutionary algorithms, K. Deb

There are couple of other procedures right. So, one is dynamic penalty function wherein this we have a power beta over here and over here we have a power beta. So, beta is user defined parameter. So, this C and alpha again are user defined parameters. So, in addition to the algorithmic parameters for a problem we need to specify lower bound, upper bound the objective function value. So, if we choose to use this method to calculate the fitness function, we also need to specify what is alpha beta and C and here t is the iteration counter right. So, over here if you see this term t appears over here.

So, in this case what is happening is this penalty value is changing this is no longer a constraint right it is changing as the iteration progresses. So, the usual value of beta that is suggested is to right additional details about this dynamic penalty function can be looked into this book which again refers to a article. So, this is a dynamic penalty function right.

(Refer Slide Time: 48:58)

Constraint Handling

➤ Using dynamic penalty function

$$F(x) = f(x) + (C \cdot t)^\alpha \left[\sum_{j=1}^J (g_j(x))^\beta + \sum_{k=1}^K |h_k(x)|^\beta \right]$$

C, α, β : user defined parameters
 t : iteration counter

100

} 25

} 25

➤ Method based on feasible over infeasible solutions.

$$F(x) = f(x) + \lambda \left[\sum_{j=1}^J (g_j(x)) + \sum_{k=1}^K |h_k(x)| \right] + R(t, x)$$

$R(t, x)$: difference between the best static penalized function value among all infeasible solutions and the worst feasible solution

Multi-objective optimization using evolutionary algorithms, K. Deb

Other method that we have is based on feasible over infeasible solution. So, we want to prioritize feasible solution over infeasible solution right. So, this term F of x is the objective function right, this term we have discussed. So, this $R(t, x)$ is difference between the best static penalized function value among all of infeasible solution.

So, let us say we have 100 solutions, right at the bottom we have place those infeasible solution. Let us say 25 infeasible solutions are there. So, among the infeasible solution which is having the best static penalized function value? Right; best static penalized function value is this term right. So, that term and the worst visible value right so, we will have a particular scalar value corresponding to this and then the worst feasible solution. So, we have let us say 75 other feasible solution; among this there will be a solution which has the worst objective function value. It is feasible, but it is the worst objective function value.

So, since all of the 75 solutions are feasible solution, we can directly compare the objective function value. The one which has the worst objective function value that value and this best static penalized function value among all of invisible solution. So, the difference between that can be added for all these 25 solutions.

So, if you actually work this out you will see that the best infeasible solution and the worst feasible solution will have the same fitness function value and, similarly this R t of x is getting added to all the other 24 solutions also right. So, this is another way to determine fitness function which is how we handle the constraints.

(Refer Slide Time: 50:43)

Constraint Handling

➤ Using dynamic penalty function

$$F(x) = f(x) + (C \cdot t)^\alpha \left[\sum_{j=1}^J (g_j(x))^\beta + \sum_{k=1}^K |h_k(x)|^\beta \right]$$

C, α, β : user defined parameters
 t : iteration counter

➤ Method based on feasible over infeasible solutions.

$$F(x) = f(x) + \lambda \left[\sum_{j=1}^J (g_j(x)) + \sum_{k=1}^K |h_k(x)| \right] + R(t, x)$$

$R(t, x)$: difference between the best static penalized function value among all infeasible solutions and the worst feasible solution

Handwritten notes:
 - A bracket on the right side of the slide groups the two methods, with a circled 'F' next to it.
 - A bracket on the left side of the slide groups the first method, with a circled '1/2' and '3/4' next to it.
 - A bracket on the right side of the slide groups the second method, with '75' and '25' written next to it, and '100' written to the right.
 - A circled 'F' is written next to the second method's equation.
 - A circled 'F' is written next to the definition of $R(t, x)$.
 - A circled 'F' is written next to the piecewise definition of $F(x)$.
 - A circled 'F' is written next to the definition of f_{\max} .

Handwritten notes for the piecewise function:
 - "minimization" written above the equation.
 - "F = f + (penalty)" written below the equation.
 - "F = f + penalty" written below the equation.

$F(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible} \\ f_{\max} + \sum_{j=1}^J (g_j(x)) + \sum_{k=1}^K |h_k(x)|, & \text{otherwise} \end{cases}$

f_{\max} : objective function value of worst feasible solution

Multi objective optimization using evolutionary algorithms, K Deb

Another way to handle constraint is using this f max right. So, if a solution is feasible, then the fitness function is nothing, but the objective function value which is clear right. But, if a solution is infeasible, so so far whatever strategies we have seen even if the solution is

infeasible we were determining the objective function value right which is kind of not necessarily correct right. So, for mathematical problems you plug in some value so, you will get some value for the fitness function, but let us assume that we have to perform an instrument and the instrument can be performed only for feasible values right.

Then for infeasible values calculation of this objective function itself is not possible. In those cases particularly this strategy helps right. So, here it is the objective function is the same as the fitness function if the solution is feasible, if the solution is not feasible right. So, in this case so, otherwise is infeasible solution we calculate the penalty since the solution is known we can plug into the equations and we can check whether it is satisfying the constraints right. So, since we are saying that it is an infeasible solution we will be able to calculate these values.

So, so far we had f all these places right what we are doing is f_{\max} right. So, f_{\max} is the objective function value of worst feasible solution. So, in this case let us say we have this 100 solutions, let us say 75 are feasible and the remaining 25 are infeasible, then for these 25 solutions we do not calculate the objective function value right. What we will do is the worst objective function value whatever is over here that will be taken and the penalty for all this 25 solutions would be added.

So, in this way if you see it will satisfy our requirement that none of these 25 solutions will defeat any of this 75 solutions because what we are adding over here is the worst possible solution. Let us say we have 1, 2, 3 like this let us say till 75 here what we are doing is 75 plus the penalty 1 right for the 76 solution 75 plus penalty of 2 right so, we are doing this. If any of the infeasible solution and feasible solution are going to have a competition the infeasible solution will always lose out right because this is the worst and we are adding something to it right.

So, this is a very nice strategy wherein we do not even need to depend computational effort to find out the fitness function of a infeasible solution. So, this strategy can also be adopted right. Again remember all of this is true only for minimization problems right. So, for maximization problems we will have to appropriately change it. So, our minimization problem we say the fitness function is F is equal to the objective function plus whatever that penalty is right.

Whereas, for maximization problem it has to be fitness is equal to f minus penalty right or you do not need to worry about this convert whatever maximization problem you have to a minimization problem and that will take care of this penalty.

So, the penalty can be either hard penalty or it can be a constraint penalty depending upon the amount of violation or it can be an adaptive penalty wherein as iteration progresses we change the penalty factor right. And, the final one that we looked at is that we do not even calculate the objective function value of the solution which is infeasible right. We just work with the worst feasible solution and we add that to the violation of the infeasible solutions this is how we can handle constraints in metaheuristic techniques.

(Refer Slide Time: 54:13)

```
function [bestfit,bestfitIter,bestFIter,F,f] = TLBO(prob,lb,ub,Np,T)
% Starting of TLBO
F = NaN(Np,1); % Vector to store the fitness function value of the population members
bestfitIter = NaN(T+1,1); % Vector to store the best fitness function value in every iteration
D = length(lb); % Determining the number of decision variables in the problem
P = repmat(lb,Np,1) + repmat((ub-lb)/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
bestfitIter(1) = min(F);
% Iteration loop
for k = 1:T
    % Teacher Phase
    Xmean = mean(P); % Determining mean of the population
    [I,ind] = min(F); % Determining the location of the teacher
    Xbest = P(ind,:); % Copying the solution acting as teacher
    TF = randi([1,2],1,1); % Generating either 1 or 2 randomly for teaching factor
    Xnew = P(i,:) + randi(2)*Dbest - TF*Dmean; % Generating the new solution
    Xnew = min(lb,Xnew); % Bounding the violating variables to their upper bound
    Xnew = max(ub,Xnew); % Bounding the violating variables to their lower bound
    fnew = prob(Xnew); % Evaluating the fitness of the newly generated solution
    if (fnew < f(i))
        % Greedy selection
```

Now, that we know how to handle the constraints, let us solve the problem using teaching learning based optimization right

So, we have the TLBO code here; this is the same code that we have explained you previously right. So, we are not going to change anything over here. Remember, the ways still we are handling problem is this is our algorithm right and this is our problem. Whether it is constrained optimization or unconstrained optimization right so, right now the communication is the algorithm is going to give the decision variables and the problem is going to give the fitness function value right.

So, if it was unconstrained optimization problem it would have returned just the objective function value. In the case of constrained optimization problems, it will return the fitness function value which will also consider the violation of the constraints right. So, we are not going to change anything in the algorithm. Algorithm is going to remain as such right. So, whatever constraints we have right is part of the problem and will be handled over here.

(Refer Slide Time: 55:10)

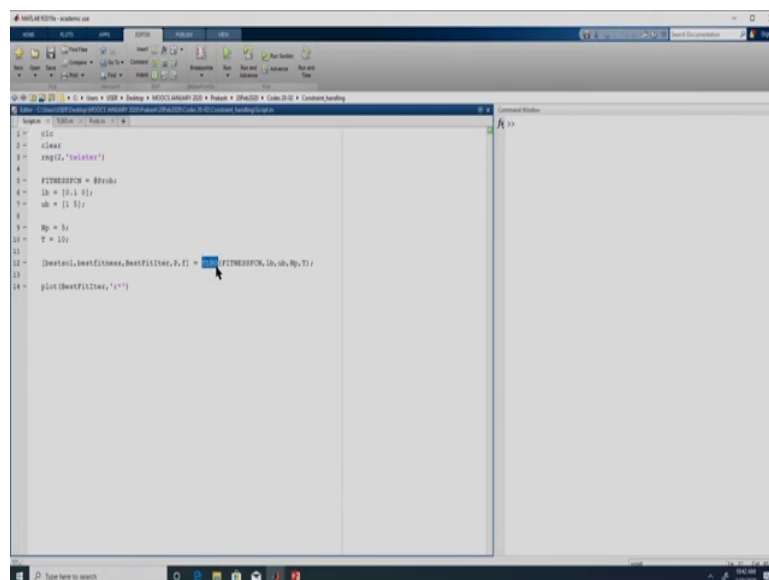
```

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

```

So, that is why we will not be making any changes to the algorithm as such right. So, the algorithm remains the same.

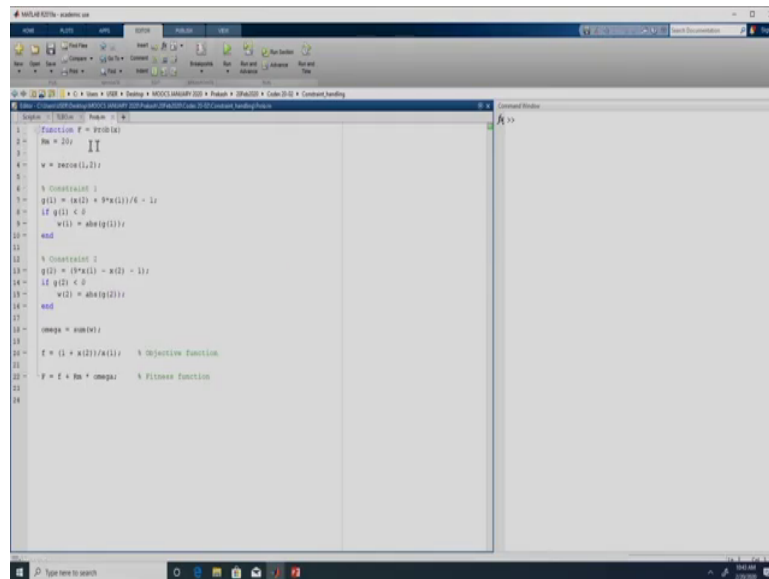
(Refer Slide Time: 55:17)



```
1 % Problem: Minimize f(x) = x1^2 + x2^2
2 % subject to: x1 in [0.1, 1], x2 in [0, 5]
3 % This is a two-variable problem.
4
5 % Define the fitness function
6 f = @(x) x(1)^2 + x(2)^2;
7
8 % Set the lower and upper bounds
9 lb = [0.1 0];
10 ub = [1 5];
11
12 % Call the optimization function
13 [bestfit, bestfval, BestFitter, F] = fmincon(f, lb, ub, [], [], [], [], [], [], []);
14
15 % Plot the results
16 plot(BestFitter, 'r*')
```

This is the script file right so, which is going to call the algorithm to solve the problem right. So, we need to provide the lower bound, upper bound. So, what we will do is we will take this same problem to solve it is a two variable problem. The lower bounds are 0.1 and 0 and the upper bounds are 1 and 5.

(Refer Slide Time: 55:40)



```
1 function f = Prob(x)
2
3   Nn = 20;
4   w = zeros(1,2);
5
6   % Constraint 1
7   g(1) = (x(2) + 9*x(1))/6 - 1;
8   if g(1) < 0
9       w(1) = abs(g(1));
10  end
11
12 % Constraint 2
13 g(2) = (9*x(1) - w(2) - 1);
14 if g(2) < 0
15     w(2) = abs(g(2));
16 end
17
18 omega = sum(w);
19
20 f = (1 + x(2))/x(1); % Objective Function
21
22 F = f + Nn * omega; % Fitness Function
23
24
```

So, that is what we are defining over here 0.1, 0, 1, 5. So, lower and upper bound is defined right.

And, this is the function file in which we are going to define the problem. So, we would say fitness function is equal to at Prob. So, the fitness function is now a function handle right. So, whenever we are accessing fitness function, we would actually be accessing this file Prob right.

So, to TLBO we passed the fitness function which is the optimization problem that we want to solve, the lower bounds, the upper bounds. Here we have taken a population size of 5 and the number of iterations to be 10 right. So, again no changes in this script file right. So, here we need to define the problem. So, as we know from the algorithm, we are going to get the

decision variable x . So, in this case since there are two decision variables, we will have x of 1 and x of 2 right.

So, this penalty factor we are fixing it to be R_m is equal to 20 right. So, w is a rho vector of two columns right because we have two constraints, remember this 2 is not related to the number of decision variable it is because we have two constraints right. So, for each constraint we are going to check what is the amount of violation right and we are going to assign it to w .

So, first constraint that we have here is $9x_1$ plus x_2 by 6 minus 1 right. So, that is what we have here $9x_1$ plus x_2 by 6 minus 1. So, we are calculating the left-hand side. So, if this left-hand side happens to be greater than 0, then we do not need to penalize, but if it is less than 0 we need to calculate the omega values right. So, what we are doing is exactly what we had seen previously. So, this is what we are calculating as of now using that w .

So, we transformed the constraint into this form the normalized form and then we are calculating this absolute. So, if it is otherwise so, if the constraint is satisfied if it is greater than or equal to 0, we do not need to assign any value of w because already it has been initialized to 0.

But, if it is less than 0, we need to calculate absolute of the left-hand side of the constraint provided the right-hand side is 0 right. So, that is what we are calculating g of one. So, this is the left-hand side, if it is less than 0 we calculate w_1 , if it is greater than or equal to 0 we do not do anything because w of 1 is anyway 0 right. So, similarly the second constraint we are handling. So, the second constraint is $9x_1$ minus x_2 minus 1 should be greater than equal to 0 right so, $9x_1$ minus x_2 minus 1.

So, we are calculating the left hand side again if it is greater than or equal to 0 we do not need to do anything else if this value happens to be less than 0 right, we need to take absolute of this g of 2 as the amount of violation right. So, we have calculated both the w 's then we calculate this omega variable, which is nothing, but some of the w vector right. So, the

violation of first constraint plus the violation of second constraint that is what we will get by sum of w .

This f is our objective function right. So, objective function in this case is $1 + x^2$ by x^1 right. So, here we write $1 + x^2$ divided by x^1 again you need to take care of this brackets properly right. So, x^1 is in the denominator and the numerator is $1 + x^2$. So, that is our objective function f , lower case f , the upper case F is the fitness function since it is a minimization problem objective function plus R_m . So, R_m we have as of now taken to be 20; again, R_m has to be tuned accordingly and would vary from problem to problem, R_m into ω this violation. So, that is what we had seen over here.

So, this is what we are calculating F of x is equal to f of x plus R_m into ω right. So, that will return the fitness function value. So, as an when we get a solution x it will see if there is any violation in the constraint, if there is any violation in the constraint it will calculate the appropriate violation like what is the magnitude of violation and then with the help of penalty factor we add it to the objective function right. We added to the objective function because our algorithm is a minimization algorithm and the problem is also a minimization problem right. So, we add over here right. So, this will return the fitness function value.

Let us just execute this and see the determination of fitness function for one solution right. So, all this steps I will just skip because I presume you know them right. So, the first solution is being sent to this prob right. So, this prob is nothing, but this Prob function right. So, it will come to this file. So, we define R_m as 20 right, w as 0 right. So, g of 1 we are calculating, so g of 1 turns out to be 0.0139. So, which is greater than 0 right. So, there is no violation of this constraint. So, it will not go into this loop right. So, it skipped this right.

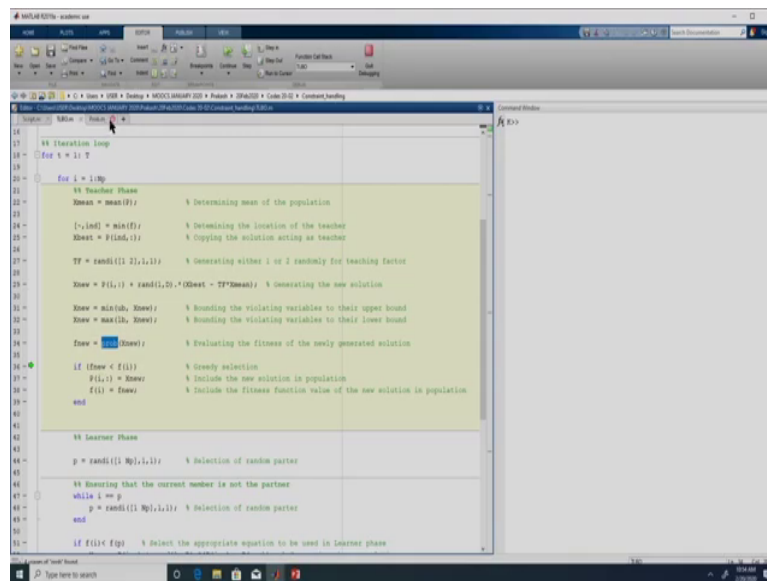
In this case if we see g of 2 is greater than 0. So, it will again not get inside this f condition right. So, w will still be 0 because w^1 was not calculated, w^2 was also not calculated because the solution satisfied both the constraint right. So, in this case ω is equal to 0. This is the objective function f 5.3853. So, the fitness function will also be 5.3853 because this ω is

0 right. So, right now what we have got is the randomly generated solution is also a feasible solution over here. So, the fitness function of the first solution has been calculated.

Let us look at the second solution. So, for this solution if we see it is 0.1233 and 1.0232 right. So, let us see if it is getting into this loop right. So, if we calculate this value g turns out to be minus 0.6445 right. So, that is actually violating the constraint. So, this right hand side over here is supposed to be greater than or equal to 0, but since it is less than 0, we need to calculate what is the violation. So, violation is nothing, but g of 1 would be negative. So, absolute of that right so, that is w .

So, in this case it violates the second constraint also right. So, the second constraint values minus 0.9132 whereas, it is suppose to be greater than or equal to 0 right. So, again w is calculated right some of both the penalty values will give us this ω which is 1.5577 right. So, this is the objective function which is 16 right. So, now, we need to calculate the fitness. So, the fitness function is going to be f which is 16.4046 plus this 20, R_m into this ω .

(Refer Slide Time: 62:58)



```
24
25 % Iteration loop
26 for k = 1:T
27     for i = 1:Np
28         % Teacher Phase
29         Xmean = mean(F); % Determining mean of the population
30         [r,ind] = min(F); % Determining the location of the teacher
31         Xbest = F(ind,:); % Copying the solution acting as teacher
32         TF = randi([1,2],1,1); % Generating either 1 or 2 randomly for teaching factor
33         Xnew = F(i,:) + randi([20,-20],1,20) * TF * Xmean; % Generating the new solution
34         Xnew = min(Xnew, Xmax); % Bounding the violating variables to their upper bound
35         Xnew = max(Xnew, Xmin); % Bounding the violating variables to their lower bound
36         Fnew = F(i,Xnew); % Evaluating the fitness of the newly generated solution
37         if (Fnew < f(i)) % Greedy selection
38             F(i,:) = Fnew; % Include the new solution in population
39             f(i) = Fnew; % Include the fitness function value of the new solution in population
40         end
41     end
42     % Learner Phase
43     p = randi([0],1,1); % Selection of random partner
44     % Ensuring that the current number is not the partner
45     while i == p
46         p = randi([0],1,1); % Selection of random partner
47     end
48     if f(i) < f(p) % Select the appropriate equation to be used in learner phase
```

So, here if we see the fitness of the second solution is 47.5586. The first one was a feasible solution right, the second solution which was randomly generated is actually a infeasible solution right. Though it is a constraint optimization problem we have still managed to get back a scalar value from the problem which is the fitness function value right.

So, similarly this will be calculated for all the five members right. So, let me just skip that right. So, here if we see it has calculated for all the five members right and the rest of the procedure is same. This is the iteration loop, the first solution that is being created right. So, this is the new solution which has been created. Fitness function of the new solution is also to be evaluated right. Again, it will go into this function file. So, every time this is being accessed with this prob, it will go into this problem file and then this is the greedy selection right.

So, if the new solution is better, it would be taken into the population right. This is the learner phase again a new solution would be generated. So, the new solution generated here is 0.4735 and 0.5171 right. So, the bounding procedure and again the new fitness function value is obtained by passing the solution to this prob right. And, then again we have a greedy selection mechanism right and this iteration loop continues right.

(Refer Slide Time: 63:53)

The screenshot shows the MATLAB R2019a environment. The script editor on the left contains the following code:

```

1 = tic
2 = clear
3 = rand(2, 'single')
4 =
5 = FITNESSFcn = @randz
6 = lb = [0, 0];
7 = ub = [1, 1];
8 =
9 = Np = 5;
10 = T = 10;
11 =
12 = [bestval, bestFitness, BestFitIter, F, f] = TLABF( FITNESSFcn, lb, ub, Np, T);
13 =
14 = plot(BestFitIter, 'r')

```

The command window on the right shows the following output:

```

>> bestval
    bestval =
        0.7126     0
>> bestFitness
    bestFitness =
        1.4033
>> BestFitIter
    BestFitIter =
         5
         4.9433
         4.9693
         4.9792
         4.9352
         4.4371
         3.9429
         3.0987
         2.6445
         2.0429
         1.4033
f >>

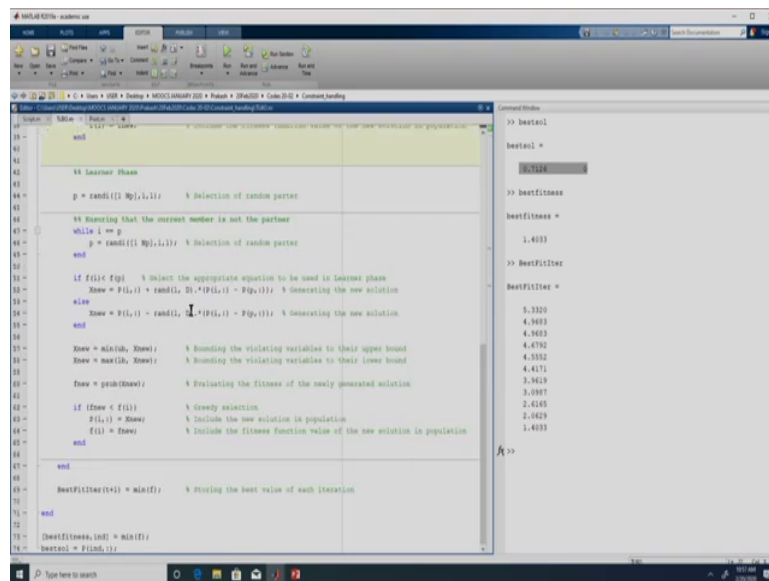
```

So, if we just remove this debug and then if we just do this continue it will complete all the iterations. So, now, if we see the best solution that it has determined is x 1 is equal to 0.7126 and x 2 is equal to 0 right. The fitness function corresponding to this solution is 1.4033 right. And, if we look at this BestFitIter right so, we had 10 iterations, but there would be 11 values over here because we also store the best fitness function value in the initial population. So, that was 5.3320 right.

So, as we can see as the iteration progresses TLBO for an constrained optimization problem. Previously, we had seen something similar right, but that was an unconstrained optimization problem, here for a constraint optimization problem same thing happens. As far as the algorithm is concerned whether it is a constraint problem or unconstrained problem does not really matter right. The way we are using it, it does not matter whether the problem is an constrained optimization problem or unconstrained optimization problem because in both cases the algorithm sends the solution and the optimization problem is supposed to send back the fitness function.

If there are no constraints the objective function is the same as fitness function but if there are constraint then the violation has to be considered along with the objective function value to determined the fitness function value. So, when we get the solution 0.7126 and 0 we do not really know whether it is feasible or not right. So, what we need to do is go back and plug the solution into this file and actually see what is g of 1 or g of 2.

(Refer Slide Time: 65:24)



```
17 = bestsol = find(i);
18 =
19 =
20 =
21 =
22 =
23 =
24 =
25 =
26 =
27 =
28 =
29 =
30 =
31 =
32 =
33 =
34 =
35 =
36 =
37 =
38 =
39 =
40 =
41 =
42 =
43 =
44 =
45 =
46 =
47 =
48 =
49 =
50 =
51 =
52 =
53 =
54 =
55 =
56 =
57 =
58 =
59 =
60 =
61 =
62 =
63 =
64 =
65 =
66 =
67 =
68 =
69 =
70 =
71 =
72 =
73 = [bestFitness, ind] = min(f);
74 = bestsol = P(ind,:);
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 =
```

Command Window

```
>> bestsol =
bestsol =
0.7424

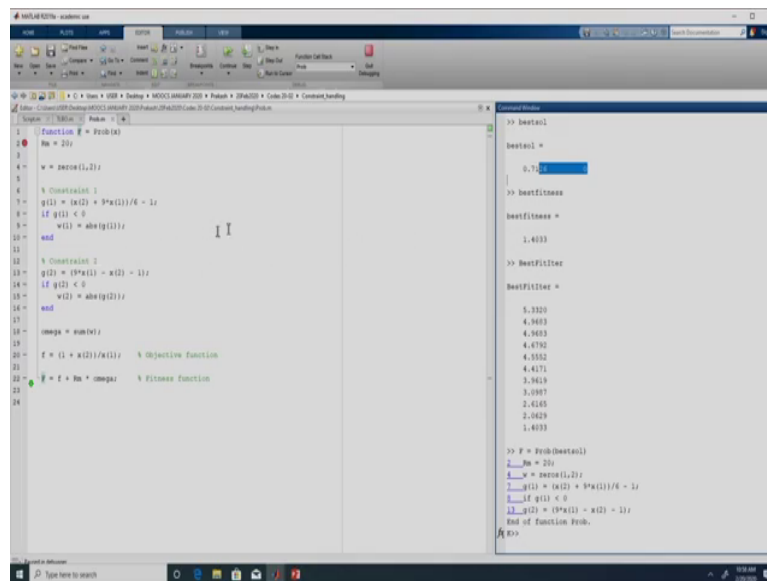
>> bestfitness =
bestfitness =
1.4033

>> BestFitter =
BestFitter =
5.3320
4.9693
4.9693
4.6792
4.5552
4.4171
3.3419
3.0987
2.4245
2.0619
1.4033

h>>
```

Or we can slightly modify this TLBO code, so that we also get to know whether the reported solution is a feasible solution or not right.

(Refer Slide Time: 65:33)



```
function f = Prob(x)
1  Nn = 20;
2
3  w = deconv(1,2);
4
5  % Constraint 1
6  g(1) = (x(2) + 9*x(1))/6 - 1;
7  if g(1) < 0
8      w(1) = abs(g(1));
9  end
10
11
12 % Constraint 2
13 g(2) = (9*x(1) - x(2) - 1);
14 if g(2) < 0
15     w(2) = abs(g(2));
16 end
17
18 omega = sum(w);
19
20 f = 1 + x(2)/x(1); % Objective Function
21
22 f = f + Nn * omega; % Fitness function
23
24
```

```
>> Bestsol
Bestsol =
    0.7126
         0

>> BestFitness
BestFitness =
    1.4033

>> BestFitter
BestFitter =
    5.3320
    4.9033
    4.9033
    4.6792
    4.5552
    4.4171
    3.3419
    3.0987
    2.4245
    2.0629
    1.4033

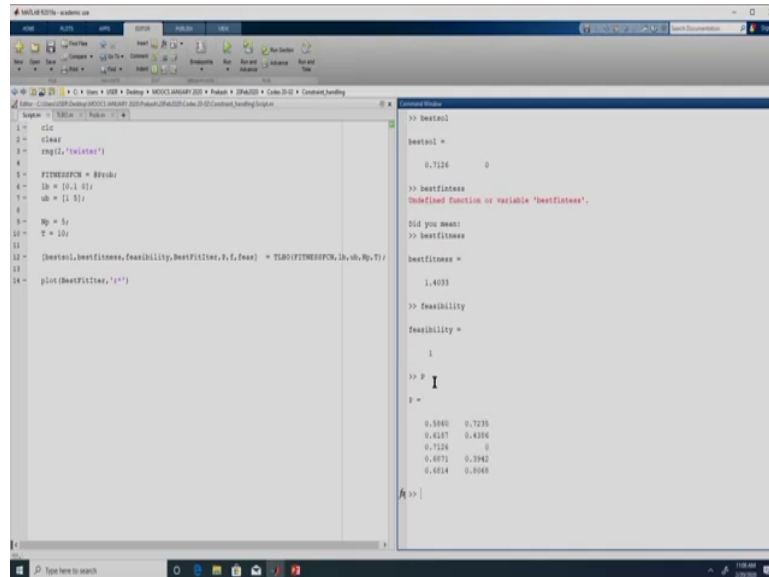
>> f = Prob(Bestsol)
f = 20
w = 20
g(1) = (x(2) + 9*x(1))/6 - 1;
g(2) = (9*x(1) - x(2) - 1);
End of function Prob.
>>
```

So, in this case what we can do is we can directly use this file right. So, let me just put a debug over here. Let me just execute this function file from here right. I am just checking whether the solution which we have obtained 0.7126 and 0 is feasible or not right. So, if we plug this solution right. So, I am calling this function now Prob and the input I am giving is Bestsol right. So, Bestsol will become x inside this function file right and x of 1 is going to be 0.7126 and x of 2 is going to be 0.

And, since I am putting a debug button over here I am going to have control over this function right. So, if we do just f 11 right it did not go to the line 9, it did not go to the line 15. So, in this case both the constraints are satisfied. So, omega would be 0, f would be the same as what we have observed here 1.4033 because this omega is 0 right. So, 1.4033 and this f will also be 1.4033. So, this solution what we have got is actually a feasible solution right. So, in the later

part what we did is we just plugged that solution and just checked whether the solution is a feasible solution or not.

(Refer Slide Time: 66:50)



```
clear
x0 = [1; 0];
T = [1; 0];
[bestsol,bestfitnes,feasibility,BestFitter,P,f,feas] = fmincon(FITNESSFcn,lb,ub,My,T);
plot(BestFitter,'r')
```

Command Window

```
>> bestsol
bestsol =
    0.7126    0

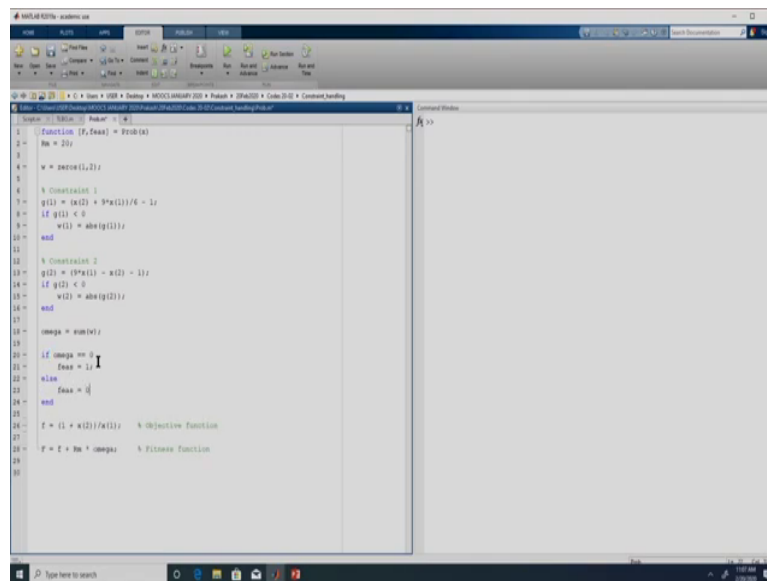
>> bestfitnes
Undefined function or variable 'bestfitnes'.
Did you mean:
>> bestfitnes
bestfitnes =
    1.4033

>> feasibility
feasibility =
    1

>> P
P =
    0.5940    0.7235
    0.4187    0.4396
    0.7126    0
    0.0572    0.3942
    0.0514    0.0048

>> |
```

(Refer Slide Time: 66:54)



```
function [F,feas] = Prob(x)
1
2  Nm = 20;
3
4  w = zeros(1,2);
5
6  % Constraint 1
7  g(1) = (x(2) + 9*x(1))/6 - 1;
8  if g(1) < 0
9      w(1) = abs(g(1));
10 end
11
12 % Constraint 2
13 g(2) = (9*x(1) - x(2) - 1);
14 if g(2) < 0
15     w(2) = abs(g(2));
16 end
17
18 omega = sum(w);
19
20 if omega == 0
21     feas = 1;
22 else
23     feas = 0;
24 end
25
26 f = (1 + w(2))/x(1); % Objective Function
27
28 F = f + Nm * omega; % Fitness Function
29
30
```

So, if you have reasonable coding skills what you can do is you can also vary this value of Rm right; either you can manually vary every time and then execute or you can just add another loop over here right. So, remember, right now what we have done is we have only executed one run right. Remember that since this is a stochastic algorithm, we need to execute it for multiple times with different seed right. So, we can also change this Rm value and study the impact of this Rm on the problem right.

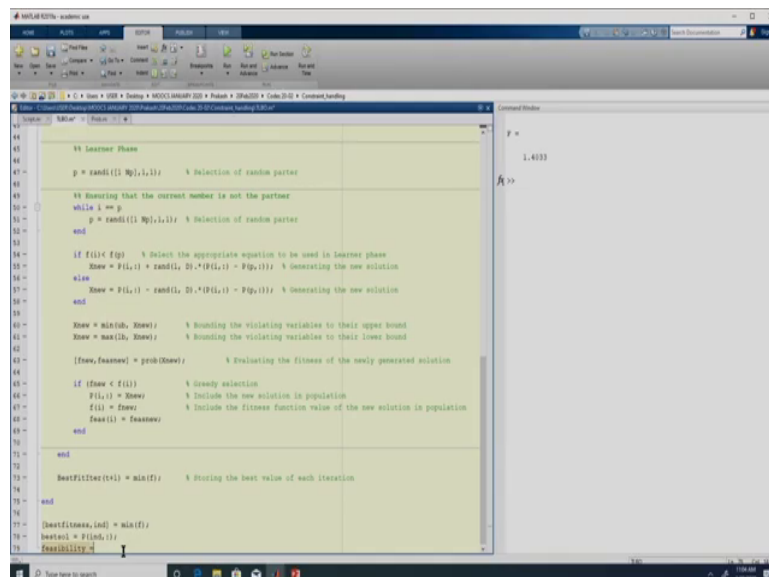
So, we leave that to you as an exercise right. So, if you have any doubts over there you can post it your query in the forum and we will clarify it. So, if you are interested that you can do is you can have a counter right. So, we have this f of p which stores the fitness function value right. So, similarly I can create another variable, let us say something as feas. So, the name of

the variable is `feas` all right. So, what this is going to store is if the solution is feasible it is going to store 1; if the solution is infeasible it is going to store 0 right.

So, over here what we will expect the problem to give the fitness function as well as whether the problem is feasible or not right. So, same thing I will require in three places right this is the initial evaluation of the population over here and the same thing I will require over here. So, what we are expecting the problem to provide is not only the fitness function value, but also explicitly tell whether the solution is feasible or not right.

So, here we can do `feas = new` or let us say just `feas`. So, here we will put a if condition, if `omega` is equal to 0 right. So, we are checking whether `omega` is equal to 0; if it is equal to 0 that is a feasible solution so, we will return a value of 1. We will just say it is either 1 or it is 0 over here right.

(Refer Slide Time: 69:02)



```
44
45
46 % Learner Phase
47 p = randi([0,1],1,1); % Selection of random partner
48
49 % Knowing that the current number is not the partner
50 while i == p
51     p = randi([0,1],1,1); % Selection of random partner
52 end
53
54 if f(i) < f(p) % Select the appropriate equation to be used in learner phase
55     Xnew = P(i,:) + randi, D1.*(f(i,:) - f(p,:)); % Generating the new solution
56 else
57     Xnew = P(i,:) - randi, D1.*(f(i,:) - f(p,:)); % Generating the new solution
58 end
59
60 Xnew = min(Xb, Xnew); % Bounding the violating variables to their upper bound
61 Xnew = max(Xl, Xnew); % Bounding the violating variables to their lower bound
62
63 [fnew,feasnew] = prob(Xnew); % Evaluating the fitness of the newly generated solution
64
65 if (fnew < f(i)) % Greedy selection
66     P(i,:) = Xnew; % Include the new solution in population
67     f(i) = fnew; % Include the fitness function value of the new solution in population
68     feas(i) = feasnew;
69 end
70
71 end
72
73 bestFitness(i+1) = min(f); % Storing the best value of each iteration
74
75 end
76
77 [bestFitness, ind] = min(f);
78 bestsol = P(ind,:);
79 %feasibility
```


So, we also need to replace this feasibility check. So, if the new solution is accepted whether it is feasible or not that will also have to be updated. So, here we are updating the population, here we are updating the objective function value, here we are updating whether the solution is feasible or not. So, this line will also be required over here.

Well, let me just have a look at it once. So, whether the solution is feasible or not is stored over here, and when we get a new solution, we would also know whether it is feasible or not and that is updated over here. So, at the end of the algorithm what we can do is we are identifying the best member. So, here what we can do is. So, whether the best solution is feasible or not, this vector `feas` contains whether a solution is feasible or not. So, we are interested in this particular value `ind` because that is the best solution. This will return feasibility. So, here we are returning feasibility.

So, over here I can also return this feasibility. So, `bestsol` tells us what is the solution; `bestfitness` tells us what is the fitness function value of this `bestsol` and this feasibility will be either 1 or 0. So, if it is 1 that means, this `bestsol` the solution that we get in this `bestsol` is feasible else it is infeasible. So, we do not have to go back and substitute in this problem again and check.

So, over here, if we come so, this feasibility has been added. I can also pass back this `feas` as a vector. So, this is the entire population in the last generation, this is their corresponding fitness function value, this vector will have either 1 or 0. So, the size of this vector will be the same as the size of the objective function value or the fitness function value.

So, this will tell us as to how many solutions are feasible in the population. So, since we have changed things on the site. So, let me just copy this and put it over here. So, here if we see `bestsol` should be the same which we got previously; `bestfitness` should also be the same and now if we see feasibility we know we had previously explicitly checked this solution by plugging it back into this problem. So, right now we do not need to do that we can just say feasibility so, it is 1. So, this solution which we have is actually a feasible solution.

From the fitness function it may not be possible to always tell whether it is feasible or not. So, we need to explicitly check whether it is feasible or not right. So, this is the population; our population size was 5, so these are the five members right.

(Refer Slide Time: 72:01)

```

1 = clear
2 =
3 = rng(1,'twister')
4 =
5 = FITNESSFcn = @randz
6 = lb = [0, 0];
7 = ub = [1, 5];
8 =
9 = Np = 5;
10 = T = 10;
11 = [bestof, bestFitness, feasibility, BestFitter, F, f, Feas] = fmincon(FITNESSFcn, lb, ub, Np, T);
12 =
13 = plot(BestFitter, 'r')
14 =

```

```

>> feasibility
feasibility =
     1
>> F
F =
    0.5840    0.7235
    0.6187    0.4394
    0.7124     0
    0.4071    0.3942
    0.0514    0.9040
>> f
f =
    2.3479
    2.3252
    1.4093
    2.0291
    2.4917
>> Feas
Feas =
     0
     1
     1
     1
     1

```

They are corresponding fitness is this right and whether they are feasible or not can be obtained from here. So, out of five solutions the last four are actually feasible whereas, this is an infeasible solution right. For constraint optimization problem you should not only stop with finding out the best fitness function value, but also should ensure that whatever solution is being reported is actually a feasible solution right. So, that call has to be made whether the solution which we have obtained at the end of the specified number of iterations from any algorithm is actually feasible or not right. So, that has to be checked.

So, here we demonstrated it for teaching learning-based optimization right. So, you can directly plug this file into any of the other four metaheuristic techniques which we have studied right. So, we have already given you the codes of GA, ABC, DE as well as PSO right. So, you can take this file prob and execute those algorithms to see if you are able to solve constrained optimization problems with those algorithms. With that we will conclude the session on constrained optimization problem.

Thank you.