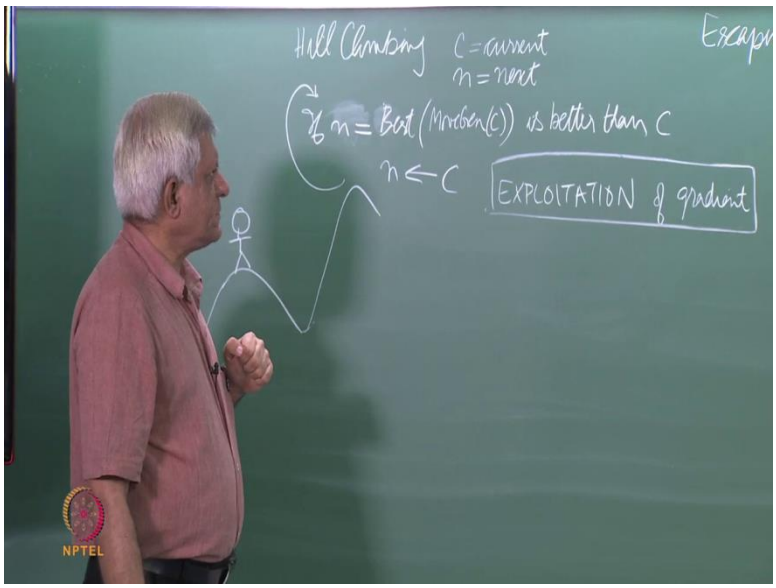
**Artificial Intelligence**
**Prof. Deepak Khemani**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Madras**

**Lecture - 13**
**Tabu Search**

(Refer Slide Time: 00:14)

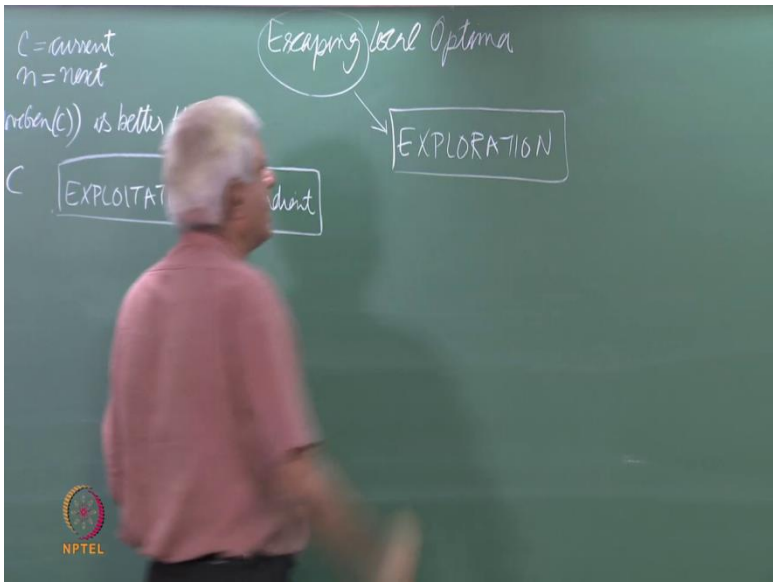



With our objective of escaping local maxima or local optima. So, remember our algorithm for hill climbing and it basically says that, if c is a current node and if n is a next node, whether you should move from a node to a next node. And algorithm essentially says that, if the best node that you get from move gen of c, which is basically in. So, you look at all the neighborhood of c, c is a current node and take the best amongst them, max or min value depending on what you are doing. And if that is better than c then you basically move to c and you put this in a loop essentially.

So, it basically looks at the neighborhood and so keeps this at problem in mind, all the T S C problem in mind, where we are looking at a candidate solution and the neighboring candidates solutions essentially. Now, what was happening with this algorithm is that, it along some let us say one dimensional problem, you would end up here and stop here, because all neighbors are worse in this and when in practices real optimum may be here or some were else bit further away essentially.

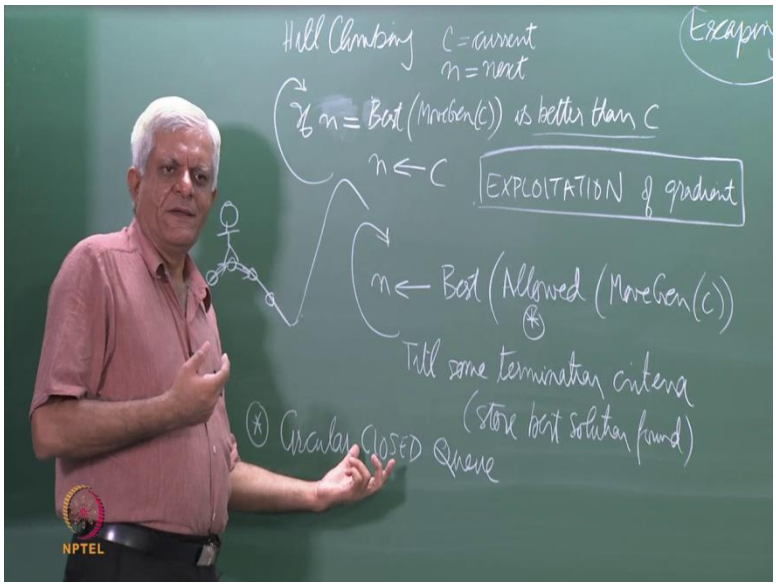
So, what do you want to now, look at this is, how do we explore this space more? So, that we do not get stuck at a local optima, in this case a local maxima essentially. So, what hill climbing does is exploitation of the gradient, it basically follows the gradient. If there is a neighbor which is better, it goes to that, otherwise, it gets stuck.

(Refer Slide Time: 03:28)




What explores, what this escape requires is the ability for exploration. So, the simplest, so first let us look at the deterministic algorithm, which will allow us to explore more of the space and what do you mean by explores, that we are allowed to go against the heuristic function. It does not means that you can only go to better states; you can go to states which are not necessarily better. So, of course then we have to work out the termination criteria and things like that.

(Refer Slide Time: 04:24)




So, let us look at the variation of this algorithm, which says that n, we get the best as before, but we will introduce another feature, which is allowable or allowed and we just simply put this in the loop. So, we are not checking whether it is better or not. Here, we have that condition, if the best neighbor is better than c, then you move from n to c. Here, we are simply saying just move to the best neighbor, but not just a best neighbor, but something called allowed essentially, which we will look at now essentially.

So, before we come to what is allowed means, it basically means that, you can move to the best neighbor, which means that, even if you are at maxima, you can go to a neighbor, which is not better than that essentially. So, that is the key first thing that you must remember essentially. Now, supposing we wanted to allow this, that always go to the best neighbor, which the criteria that you it does not have to be better than the current node.

Then how do we stop, that is one small problem, but there is a bigger problem and the bigger problem pertains the fact, that if I am stuck at this local maxima, how do I get to that maxima essentially. So, the first problem of how do we stop, we will say that put some other termination criteria. So, tell some, it can be simply time based or it can be simply is that we are not finding a better solution after a certain amount of time and so on. You can always store the best.

So, looking at in general in optimization terms, that in this process always keep track of the best solution that you have found ever, even if you have moved away from it, remember that, that was the best solution. So, always keep track of the best solution, so that you can always do. Given that, we are now exploring the states place and this algorithm says that for example, in this case, in a one dimensional word, they are two neighbors here and here. So, I am allowed to move to one of them essentially.

In a larger space of course, there are many neighbors, so you can move to the best amongst them essentially, but to illustrate the point, illustrate the difficulty that this algorithm has supposing I am allowed to, so I came from here to here. Then I am allowed to move here that is allowed, because I am no longer saying that, it should be better than what I am in, what will happen in the next step. Next step, this will have these two neighbors, this one and this one, what will the algorithm do?

Student: ((Refer Time: 07:56))

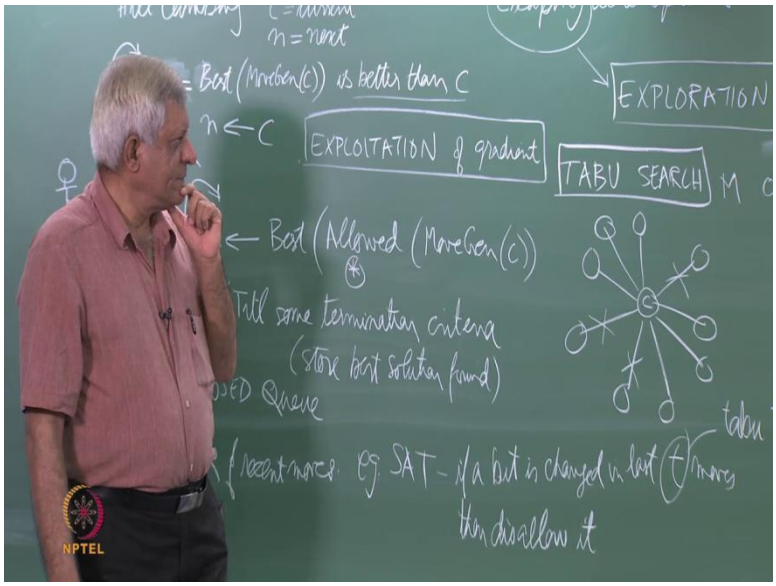
No, that we, that is the independent thing, we just now talking about the behavior of the algorithm, where will it go next.

Student: ((Refer Time: 08:06))

Unfortunately, it will go back to this place, even if it takes one step down, but this one say that, still that best criteria is still there, go to the best neighbor, so that one is still the best neighbor essentially. So, it will go back there, how do we get along this that is where this allowed thing is coming in. So, remember we had this idea for a close list, when we started doing the states space search and what close list said that, do not go back to the same node again that you have seen, so what we could do.

So, we are looking at this now, one way to do this is to maintain a circular list, maintain a circular cube of some finite size. So, you know what is a circular cube, that you keep overwriting as you go round and round the circle you allow to overwrite, but some k number of elements will always be stored in the cube, where you can you are allowed to overwrite. So, it is, this is like a short term memory. It is saying that these are the last k nodes that I went to and I am not allowed to go back to them, but I can I am allowed to go back to any other neighbor.

(Refer Slide Time: 09:40)




So, what with this allowed factor, what we are really doing is that, given a node c that we start with, we generate all the successors, all the neighbors, from this neighbor we disallow some. So, we say this move is not allowed, this move is not allowed and this move is not allowed. So, this is that allowed thing that we are trying to do here, but from the remaining move to the best one. Another way of doing, this is the following that keep track of what moves you made in the recent past essentially. So, for example if you are doing S A T and let us say for simplicity sake that, you are changing only one bit at the time.

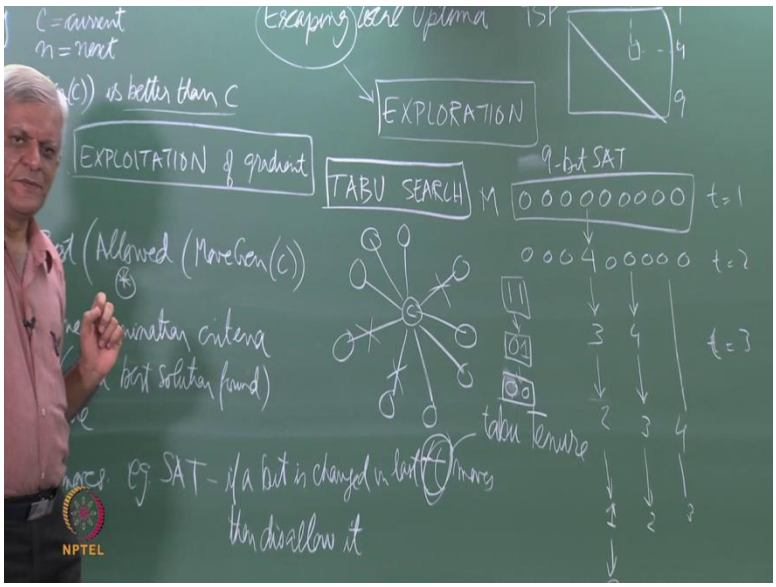
So, essentially move says that k in the highest bit where I could be any bit essentially. If a bit is changed in the last t moves then disallow it. Notice that, this is slightly different from maintaining a closed queue. In a close list, we are maintaining our candidates and saying we will not generate the same candidate again. In this example, we are saying that, if we are change, let us say the fifth bit now, then for the next t moves, I am not allowed to change that fifth bit essentially. So, what will happen, I can maintain for example, a memory and array called M, which will start with 0 for everything that is one way of doing it.

Let us say, I have a 9 bit problem, 9 bit S A T. So, how many does this have 3, 4, 7 let me add 2 more, I have a 9 bit S A T like this. I can change any one of those 9 bits and move to the best amongst them. Now, supposing I have change the fourth bit that is the

best, that I am allowing to do. So, what am I doing, I have a valuation function for every node or every candidate solution, I have a valuation function and we said that for S A T, it could be the number of clauses satisfied or something like that essentially. So, I have a valuation function and I generate all this neighbors and just move to the best amongst them, does not matter, whether it is better or not, but I am allowed to move to the best amongst them.

So, if I change this, now let us say, so this t is called the tabu tenure and this algorithm is actually called tabu search. So, this particular spelling of tabu I mean, we are more use to taboo t a b o o, but it the meaning is still the same. This comes from some I think Fiji Island or Tonga or somewhere, where they use this spelling of the word. And basically, it means disallowed essentially, it is a taboo move that you, you are not allowed to make that moves essentially. So, if I am allowed to, if I decide on tabu tenure of t, let us say in this example, let say 3 or something or 4 or something.

(Refer Slide Time: 14:43)




Then if I have changed this one, I could say that make this 4, everything else remains 0. So, this is like an array, which tells me whether I am allowed to change that bit or not, if that value is non zero, I am not allowed to change only. If the value is 0, I am allowed to change. So, after the first move I am allowed not allowed to change the fourth bit, but I am allowed to change any other bit. After the second move, this 4 will come down to 3

automatically and some other bit. Supposing, I change this bit, this will become 4 everything else will remain the same.

So, you understand what I am saying that, this is that t equal to 1, this is t equal to 2, then t equal to 3 not this t, we will use a term t t, which is a kind of a more standard term. So, t t stands for tabu tenure. So, my tabu tenure is 4 then and that is time, this t is time, the first cycle, the second cycle, the third cycle. Then in this next cycle, this will become 2, this will become 3 and some other bit will become 4 and eventually, this will become 1 and then become 0, which is when I am allowed to change it essentially.

So, this will go to 2 meanwhile, this will go to 3 and so on. So, after having change this here, then for 1, 2, 3, 4 cycles I am not allowed to change it, but now it, the value is become 0 I am allowed to change it essentially. There is just one way of implementing this, you can insert simply for every bit keep a time stamp of when it was last changed and do an explicit comparison with that, and decide whether you know, the current time is more than four units from that time when it was changed, you could do it in either ways actually. But, this is the kind of a standard way of doing this essentially.

So, for S A T you could maintain and array which traditionally we called it M, which stands for memory, for T S P we could maintain a triangular matrix. So, this is 1 to 9 and this is 1 to 9 for example, if it is a nine city problem and you could keep track of which edge that, remember that every square in this will correspond to an edge. So, let us say the seventh and the four, so the edge between 7 and 4 that I have removed that or something like that. So, I can keep track of that in a T S P like problem as well essentially.

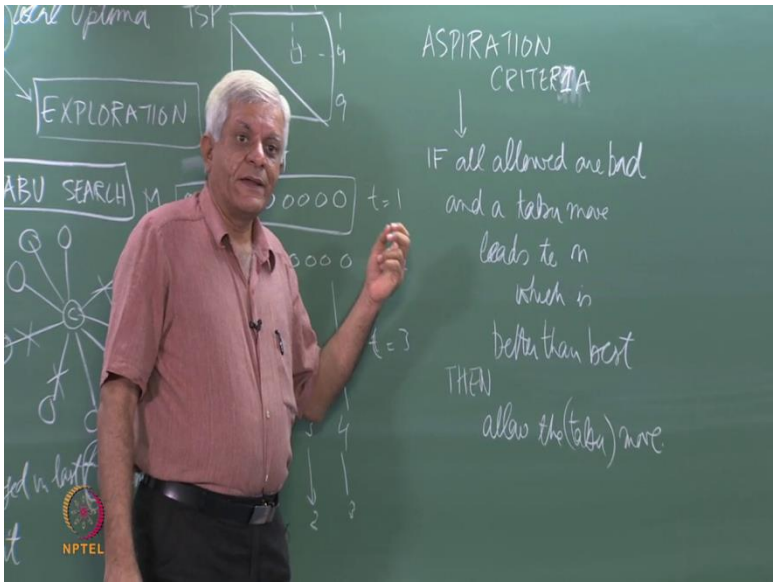
So, basic idea in tabu search is to have this notion that you do not allow moves we are made recently. So, that you do not go back to the same local maxima from which you are trying to escape that is the basic motivation, do not go back to the same maximum that would. Now, obviously if you look at this way of doing tabu search that you are controlling which bits to change, then given any two bits. So, for example, I start with let us say bits, two bits which are 1 1 to start with and then this gets changed to 0 1 essentially.

So, that means, I have changed the first bit and this is some substring here, I have changed this bit, I am not allowed to change it for four units. Then after this let us say I

change this other one, 0 1 0, 0 0, I change the other bit as well. So, if I made this two bits, where I change this bit, this substring has become 0 1, then I change the other bit, this is become 0 0, I have lost the ability to move to one combination which is the 1 0 combination, because of this tabu, that I am doing that I am not allowed to change either of this two bits for the next four rounds.

I cannot change this to 1 0 that will not be allowed essentially. So, I am moving I might lose out on something, but in general, experimentally it has been found that, this tabu search works well with these kinds of problem essentially, right. So, we just observe that, you cannot move to this 1 0 from here essentially, what if that 1 0 really happen to be the solution or something like that. So, actually the more detail tabu, I will search algorithm allows you to make an exception to this barring of certain moves essentially.

(Refer Slide Time: 19:47)



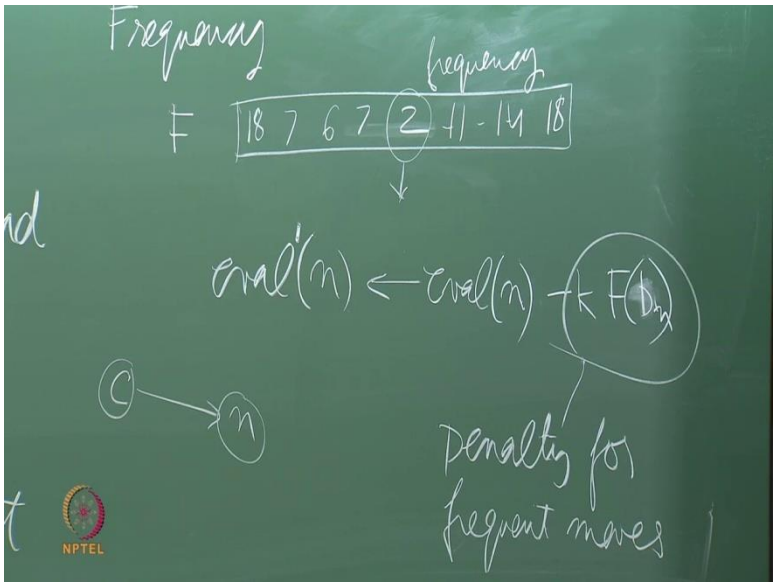

And the exception can be made and then what is called as an aspiration criteria, which confusion sometimes I get into spelling. So, the expression criteria's says that, if all allowed neighbors are bad. So, I will just write all are bad by bad we means worse than current and a tabu move leads to in which is better than best, then allow the move. So, obviously our goal is to optimize the valuation function or the objective function of that we are working on and if we are getting access to a good move then we should not lose it.

So, this aspiration criteria says that, if all this allowed neighbors, the once which are not crossed out are worse than my current node c. And if one of those barred neighbors, a one of the tabu neighbors is better than the best, by best I mean this thing that you store the best solution that you have found, if I can find the node, which is better than this also, than you allow that exception. So, tabu in general, recent moves are tabu, but we can make an exception, if one of them in a bad situation, when the other moves are bad, gives us a much better solution.

So, for example, if let us say value of this node is 27 according to some, let us say I have a 50 clause S A T to solve and this know this satisfying 27 out of those 50 clauses and all these nodes, which are disallowed or less than 27. And if these one of them happens to be let us say 40, somehow and the best that I have seen is only 35 or something like that. Then I will allow this move and that is an aspiration criteria. So, again you can see that somehow this design of such algorithms is kind of you know little bit of an art, you are trying to device algorithm, which will work which will give you good solutions and so on.

And what tabu search does is that, it basically gives you a deterministic mechanism to say that you can go past, local maxima and explore the state further essentially. So, it does this by having this tabu tenure, which says that for a certain period of time do not make the same move again essentially.

(Refer Slide Time: 23:25)



Another feature which has sometimes been used is called the frequency based method. So, I have a frequency table, so all these line bits. So, let us say this was changed 18 times, this was changed 7 times, in 6 times and so on essentially, how many times did I change that bit essentially? So, I can also bios the algorithm towards moves, which have been made less often essentially. So, for example, if somewhere here, there is a bit which has been change only twice in my whole this thing and everything else is large number of times.

Then I may want to say that try and change this bit and see, if something good comes out of it. Remember that, what these numbers are, these are the frequency of how many times you have changes that bit essentially. This is simply like a counter, which tells you whether you are allowed to change that bit or not. If you want to bios a algorithm, opposite towards those areas which the heuristic function is not taking it to, why did you not move this bit, because whenever this bit was generated, it is the noted generated was not the best amongst the neighbors and so, it never got changed essentially.

So, if you want to push the search into that direction, you can bios the tabu algorithm by sayings that modify your valuation function. So, eval of n, let us call it eval prime of n is eval of n, which is the function that we are using to compute the heuristic function that we were calling, but the out, but we are calling it evaluation function. Because it optimization community calls with an eval function minus some constant times frequency of n, this is not very good notation, because n is really known and here it is a bit that is being saying the index of the bit.

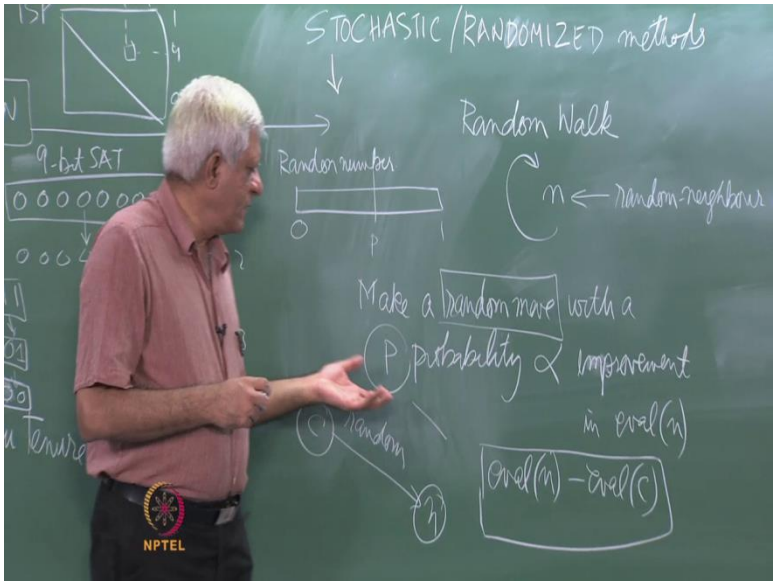
So, let me just call it frequency of b n, the bit which makes leads you to this n th essentially. So, as long as it is clear, we will use some notation here. So, essentially what it is saying is that, you are your moving from c, you are considering this move from c to n, from a node c to node n, when you are evaluating this node n, take into account. How frequently this particular move has been made, which means which bit has been changes and give a penalty for those bits, which have been change very often.

So, if the frequency is very high, the valuation value for the resulting node will be decrease more, if the frequency is less than this. So, this is like a penalty, in our case it is penalty for changing a bit too often, if you are changing that bit all the time then this valuation function kind of penalize it and says that, no, no you have changed this bit too

many times. I will reduce evaluation values to by this amount essentially. So, in the end of course, you can have this basic vanilla tabu search than you can add the aspiration criteria to say that, sometimes you do not make moves tabu.

And then you can have general increasing bios towards newer areas by saying that more frequent moves will be penalized essentially. So, this was the deterministic approach to trying to escape local maxima. So, let us move towards stochastic methods or randomized methods. So, what we have seen so far is the deterministic approach to escaping from local maxima.

(Refer Slide Time: 28:03)




Let us look at stochastic or randomized method. So, we will look at couple of them, we will start with some very simple thing today. So, the focus is still on exploration, how can we make the search go onto newer areas? So, you must keep in mind this two aspects of search, one is exploitation of the gradient, which is, what hill climbing does very interesting, that it just looks at the neighborhood and goes to the best neighbor. What tabu searched it was that it modify that little bit and allowed it to go to the best neighbor even if it was not better than that.

Exploration simply says that somehow you must allow the algorithm to go into different areas. And what stochastic or randomize method say is that give some degree of randomness to the movement. So, just imagine this earth, the algorithm searching through this earth space and exploitation simple says that the just follow the gradient and

now we are saying do not always follow the gradient, but do something different at sometime essentially. Now, the extreme example of randomized movement is a random walk.

And random walk, we can simply write as saying generate n is a random neighbor of c and put this in a loop. So, a random walk basically just takes says that, just take one step in some direction essentially. Of course, we can add on other stuff to this algorithm saying like keep track of the best node that you have seen so far and that kind of thing, but otherwise it is purely random essentially. From a given node c, it will just randomly choose one neighbor and go to that essentially, no comparison of evaluation function nothing essentially. But, of course you can keep track of the best one and so far and so far essentially.

Now, obviously a random walk is not going to be a great way for solving an optimization problem, because first of all, it is not even systematic, remember that we started out by saying that some searches are complete or systematic, which means that they explores the entire space. Hill climbing, tabu search they are not systematic, they do not guarantee that they will explore the entire space and this is an extreme example, it just go of in some random direction essentially. So, what we really need to find is ways which are somewhere between hill climbing, hill climbing is an extreme of exploitation and random walk is an extreme of exploration.

So, it is there is 0 exploitation in that, only exploration whereas here, in hill climbing there is 0 exploration, in a sense that it never waivers from the path that has been shown to it and there is complete exploitation. We want algorithms, which will be somewhere in between essentially. So, today I will just give you the intuition of the algorithm that we have going to study and in the next class, we will look at it in more detail. The basic idea is make a random move with a probability, which is proportional to improvement in eval n.

So, what I was saying here, that first of all I am talking about a random move. So, I am no longer saying that generate all the successors, I am just saying make a random move which means, you are at some given node c and choose a random successor n, just somehow generate one successor n. But, move to that with a probability, which is proportional to how better that move n is from c, as compare to c, how better that

candidate n is compare to c. So, the implication of this is the following that, I am not saying that n should be better than c.

I am only saying that, if n is better than c then there is a greater probability of making the move, if n is worse than c, there is lesser probability of making that move and secondly, it depends on the magnitude. So, if I look at this value, eval n minus eval c. So, let us say I am doing a maximization problem which means, the more positive this is, the better for me. I want to build an algorithm, which will say that the more positive, this is the greater should be the probability of making the move, but allow a move even if this is negative, just to include the exploration feature, but with lesser probability.

So, I want to bios my search towards better moves, but I do not want to stop it from making bad moves, bad moves meaning moves which decrease the evaluation function values essentially, which go against the gradient essentially. So, we will look at the details in the next class, but let me ask you one question at this moment, when I say make a move with the probability. So, let us say this probability is p and I gave you a value p, how will you make this move probably, when you if you want to implement this algorithm, how do you make a move probabilistically in an algorithm that you are writing

Student: ((Refer Time: 35:51))

You and

Student: ((Refer Time: 35:55))

So, you generate a random number in the range 0 to 1 and if that number happens to be greater than p, you make the move, if the number happens to be less than p, you will do not make the move. So, eventually the move is either made or is not made, but it is made with a probability p and that, for that you have to generate a random number. So, what are we after we are after an algorithm, which will have this stochastic play work, which means that it will look at a neighbor and may or may not make a move, but it should be bios towards better moves, but not barred from making bad moves.

So, somehow I have to devise a way of computing this p as the function of this difference in the evaluation function, in such a way that this behavior is manifested

essentially. So, maybe I will ask you to think about this and we will write it is not a very difficult thing to do. The only thing you have to be careful is that, this being probability it should come in the range 0 to 1 and it should satisfy this criteria that the larger this is.

So, if you want to plot this on x axis then the more you go to the right hand side, the probability should 10 to 1 and the more you go to the left hand side, the probability should 10 to 0 essentially. So, think about a function which will do that and we will specify it in the next class and the algorithm which is based on that. So, we will stop here.