Artificial Intelligence
Prof. Masum
Department of Computer Science and Engineering
Indian Institute Technology Delhi

Lecture-26
Local Search: Hill Climbing With Random Walk and Random Restart part 5

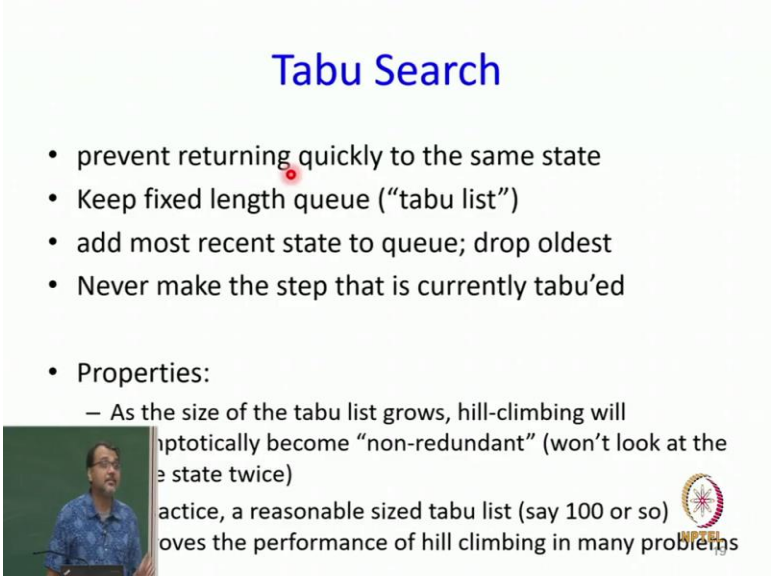
Today in the next class we will be looking at many, many different algorithms in this fold and in fact local search is sort of very intuitive. So, you can start making your own algorithms and most likely either somebody has made it or there is some problem where this is helpful. So, you know just think about it that way so what is a possible problem with sideways move like an immediate problem, yes moving in the wrong direction that is probably the whole of local search.

I do not know whether we can fix it unless we have additional information, yes algorithm might not terminate, why am I did not terminate? It is just going on and on looking at the same stage let us say with from state A I can reach state B and from state B I can reach state A and both of them have the same value. And suppose I do not impose a length of 100 limit then what would happen it will keep doing AB-AB-AB but it does not have to be.

Suppose A goes to B, C and D, B goes to C, D and E, D goes to and so on there is this fully connected kind of a shoulder what is going to happen? I will keep just oscillating between among these states what is a simple fixed limit on number of moves sure that is a that is a simple fix that we have already tried to add in sideways move. So, limit the maximum number of sideways moves. What else? One at a time, yes keep a track on the last node and keep a track of the last node and do not repeat it.

What is your name? Shreya, Shreya keep a track of the last node and do not repeat it. The last node is a specific example up until now we had complete amnesia. We had no idea where we came from so what Shreya suggests let us have a little idea where we came from let us remember the last node, we can generalize it to last k nodes whatever.

(Refer Slide Time: 02:31)

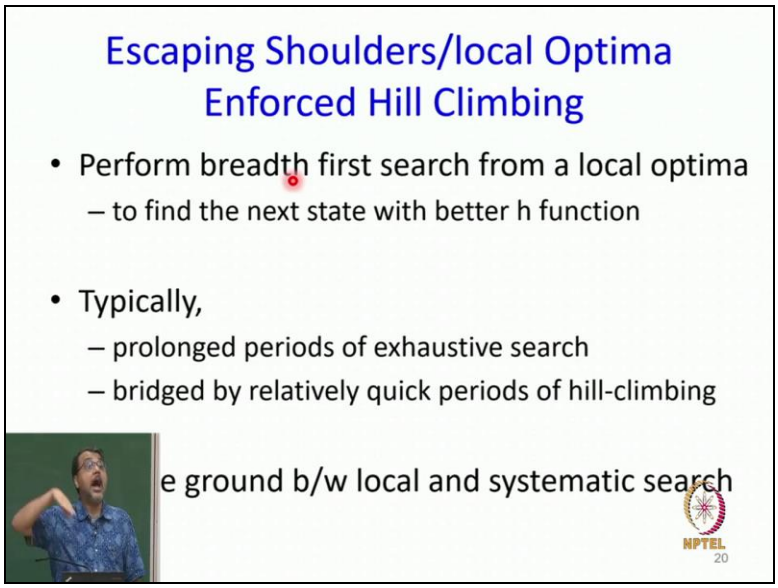


And so this algorithm is called Tabu search and this spelling is not mistaken this just how the algorithm is called. So, you create a Tabu list and Tabu list is a fixed length queue it basically says these were the nodes that were visited recently. Let us say the last 100 and you basically say you are not allowed to go back to the same node that is already in my Tabu list that is it. So, add most recent state to the queue drop the oldest.

Never the step that is currently the Tabu, now if you think about it as the size of the Tabu list grows, as you have a huge size of the Tabu list almost the full search space size of the Tabu list then you are essentially doing a systematic search you are saying do not repeat a state no redundancy whatsoever do not repeat a state look at all possible states that way. So, eventually this will become systematic search if my Tabu list becomes extremely large.

But in practice people use a small constant size Tabu list and that is about a 100 or so. Now we can extend, extend this idea why do we need a constant size Tabu list. Let us let us say that we will keep exploring until we hit a better optimum.

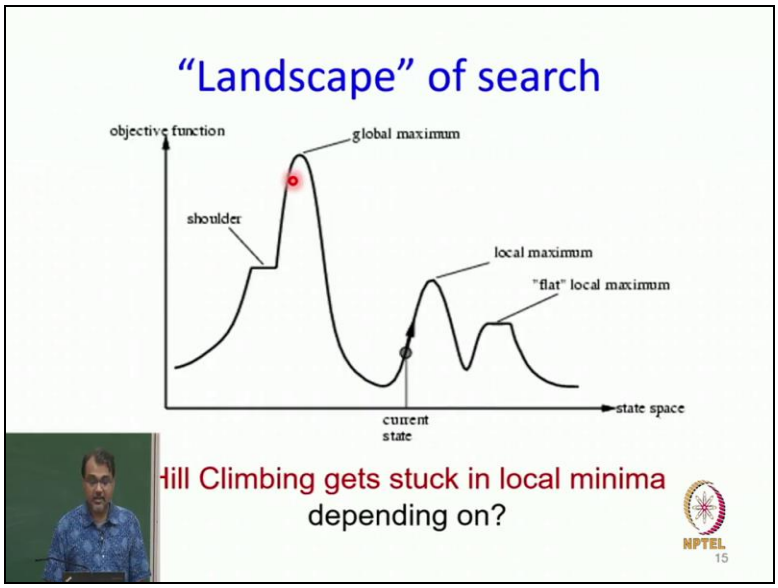
(Refer Slide Time: 04:06)



So, that particular algorithm is called enforced hill climbing the idea enforced means I am enforcing that I will always climb the hill what does that even mean. So, it means that I hit a local optimum I start doing great first search. So, now notice what are we doing from a point of complete amnesia that I do not have any memory of where I came from I am starting to add some memory in the system.

Initially I added a Tabu list which is a small memory and now I am saying let us start doing breadth-wise search. Let us start doing systematic search. So, I am doing hill climbing I do hill climbing I do hill climbing I reach a local optimum from the at local optimum I start doing breadth wise search in all directions I keep doing this bit for search what is going to happen is that at some point if I do this breath wise search long enough I will hit a state whose value is better than the root.

(Refer Slide Time: 05:06)



Let us think about why that is going to happen that is going to happen let us say I am stuck in this second local optimum. So, this particular local optima. or let us stuck in this flat local optimum third local optima. Now if I start doing breadth wise search from here I will be exploring states in all directions. So, at some point I will keep going keep going keep going and eventually hit this node on the second Hill whose value will be better than me and then I will jump.

And then I will forget that first search and start doing hill climbing all over again. So, I have enforced that if I am stuck in a local optimum I am enforcing that you have the jump off of it and this algorithm is called enforced hill climbing. In practice what happens is that the exhaustive search is exhausting it takes a lot of memory it takes a lot of time. So, in practice what ends up happening is that I am spending a lot of time doing breadth-first search and then suddenly I have found a better solution I jump and I started doing hill climbing.

I quickly looked go up the hill for hill climbing and then I am stuck and as soon as I am stuck I start doing breadth first search again and there is a prolonged period of breadth-first search before I get to the next. This is you can think of it as a middle ground between local search and systematic search. In systematic search the search space will become so large that you will not be able to make progress and memory requirements can be very high.

In local search memory requirement is nothing there is no memory requirement in practice. But I may not lead to an optimal solution and so there is a middle ground here where I will do local search and I will do systematic search then I get to a better local point I do local search and I get to systematic search then I get to a better local point and I came doing this. And this is a better middle ground where I will improve my solution but I will not be using a full memory and time as a typical systematic search solution.

And this algorithm avoids the algorithm of choice for a very well-known AA planner called the FF planner. FF was a planning system where the idea was to reach a start state to the goal state and output would be a part. Now the output is a part how can I do local search well I will be searching in the path space. Now I will be making local moves in the path space I will be removing one edge and adding another edge kind of a local move to my part, each state will be a full part.

And then on top of this I will do local search but I will be stuck in lots of local optima there were lots of local optima and so they came up with the info still climbing algorithm and that enforces climbing algorithm led to the best winning planner. By the way this idea that assignment should be competitions is very, very common in the field of AI. Pretty much all fields have competition all these research communities on that in that particular field participate their software in their in those competitions and whoever wins their technology becomes most famous.

So if you want to make an impact in any field look at what competitions they participated and try to win those companies. For example I worked in planning so we also submitted some probabilistic planners in the past and we could not win but we were close second. So, you know we had some influential was in a little 2010 on problems. The most famous planner for deterministic planning not probabilistic where the solution is a part there is no probability was the FF planner in the early 2000s from about 2001 to up to about 2010-11 FF planner forms Yogh Hoffmann.

He is now add I believe Max Planck and so group in Germany both the planner that everybody used to try to compete. Again it was using enforced hill climbing local search algorithm. Yes

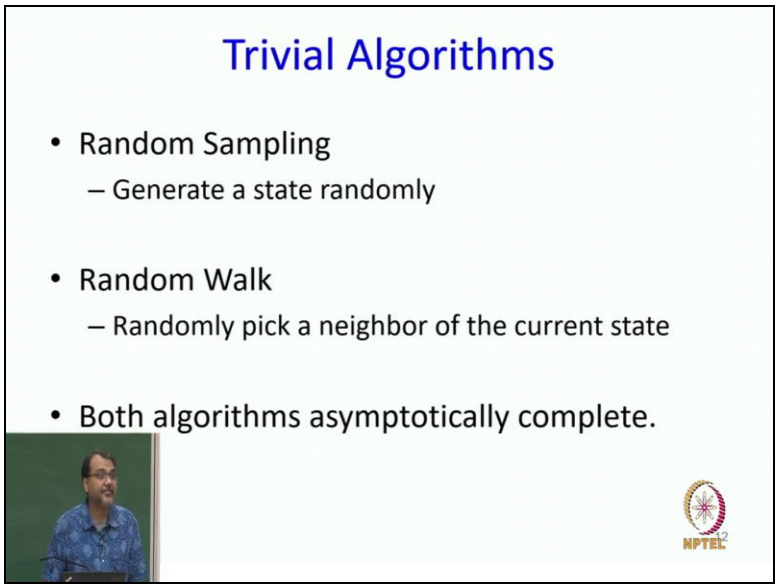
question very good question. The question is when you reach the global maximum you will keep doing breadth first search and you will never find a solution. So, what do you do at some point you will just say okay I am done with my time I am done with my memory I cannot search any further, so this is the solution I am going to return.

So in practice do you know when to stop local search now that you are doing so many variations of this it will never be clear. The best setting in which local search performs is I give you 2 minutes of time and your job is to come up with the best solution. So, your goal is not to think about when to terminate your goal is to say how do I use those 2 minutes most effectively and that is where local search shines because you can just search and you can do some tricks and search further do breadth-first search, search local, search further and before your 2 minutes are up you output the best solution found.

So far the best optimum found so far. But yes we have not discussed very much on termination condition in fact what we do next will completely take away termination condition. So, let us see what we can do next. Now let us up until now we have been doing mostly deterministic versions of local search there was a very little source of uncertainty which was in tiebreaking but modulo that we were always choosing to go greedy and no non determinism no probabilities no uncertainty.

Now very early in the class we discussed 2 algorithms which were asymptotically complete. Local search, greedy local search on the other hand is not complete it may stuck it may be stuck in local Optima what were the 2 algorithms trivial algorithms that we did which were asymptotically complete.

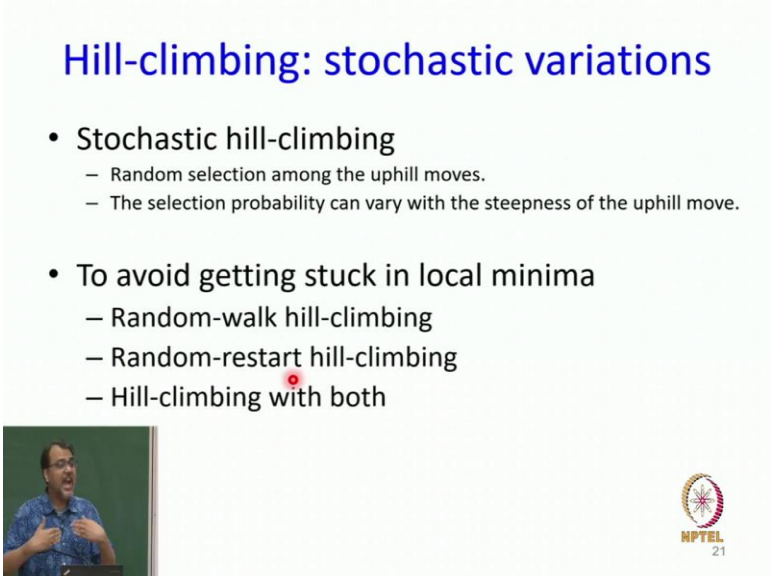
(Refer Slide Time: 11:09)



Those trivial algorithms were random sampling and random walk random sampling basically said give me a next give me a new state, give me a new state, give me a new state randomly and we discussed that eventually it may hit the optimum. Random walk however said let us just randomly the neighbors without thinking about the objective function and even that was guaranteed to hit the optimum eventually if the space search phase is connected.

So these algorithms were asymptotically complete but they did not have the intuitions of 3D such hill climbing greedy hill claiming on the other hand is not asymptotically complete. So, can I come up with some hybrid which is both asymptotically complete and has similar properties as greedy local search.

(Refer Slide Time: 12:11)

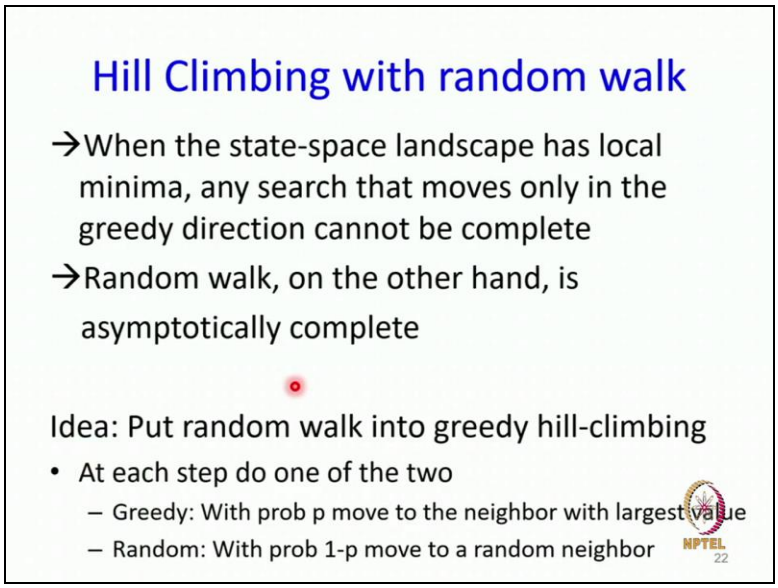


These would be called stochastic hill climbing algorithms. The goal is to avoid getting stuck in local optimum. What do you think we can do any suggestions, yes what is your name Prashant yes Prashant whenever we are stuck in local optimum randomly sampled a new neighbor or the new state, very simple. Whenever I am stuck in a local optimum the randomly sample a new neighbor if I say this what am I merging I am merging greedy local search with random sampling or random walk.

So, again if I am stuck randomly sampled a new neighbor, randomly sampled from one of the neighbors is this random walk or random sampling? Random walk on the other hand if I say whenever you are stuck randomly jump to a new state this would be random sampling. so we can try to add or both we can try to add ideas of randomness in hill climbing. So, we can do random walk hill climbing or random restart hill climbing or hill climbing with randomly start and random walk.

The algorithms are not deep at all in fact if you had thought of it for two minutes you may have been able to come up with them very simple ideas.

(Refer Slide Time: 13:51)



So, the idea is that whenever state-space landscape have local optimum any search that moves only in the video direction cannot be complete random work is asymptotically complete. So, let us put random walk into greedy hill climbing so let us do very simple we can do what Prashant said or the nicer version of that is with probability P move to an greedy neighbor with probability 1 - P move to a random neighbor.

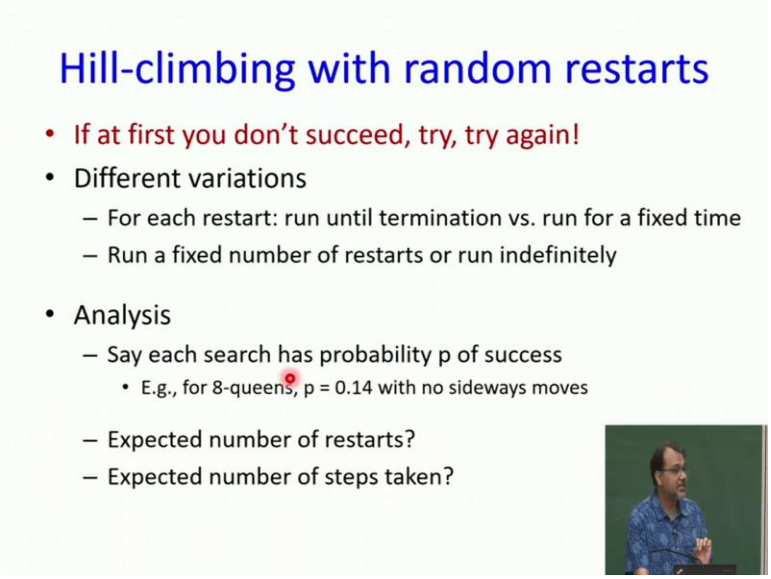
And for now we are keeping this probability P constant. But ideally what do you want should P change over time and if P should change should it increases should it decrease? It should increase, so as time goes on I should take more greedy steps and less and less non greedy steps. Initially it is ok to take many more non greedy steps why? Because I may have started in a bad Hill and I may need to jump off that bad Hill slowly.

And that changing of probability over time is called one way of doing this is called simulated annealing which will come to just in 2 minutes. Yes, so, the question is why are we doing it every step and not only local optimum and that is a fair question. See the questions the main issue is that if you were stuck at a local optimum getting of that local optimum is harder because it requires you to take lots of suboptimal moves again and again and again.

So, therefore these algorithms suggest that you should start taking suboptimal moves at every step just in case you mix left. So, now there are many algorithms that we will study but if you

have to remember only one of them like absolutely only one algorithm you want to take back from this class people will disagree with me but this is my personal choice.

(Refer Slide Time: 16:09)



It would be this algorithm hill climbing with random restarts and probably this is the way you would have thought I should fix hill climbing in the first place. Do hill climbing get stuck at a local optimum then randomly sample and you state and do hill climbing again and then randomly sample a new state and then do hill climbing again. And always keep the best solution found so far. The idea is if at first you do not succeed try, try again that is it.

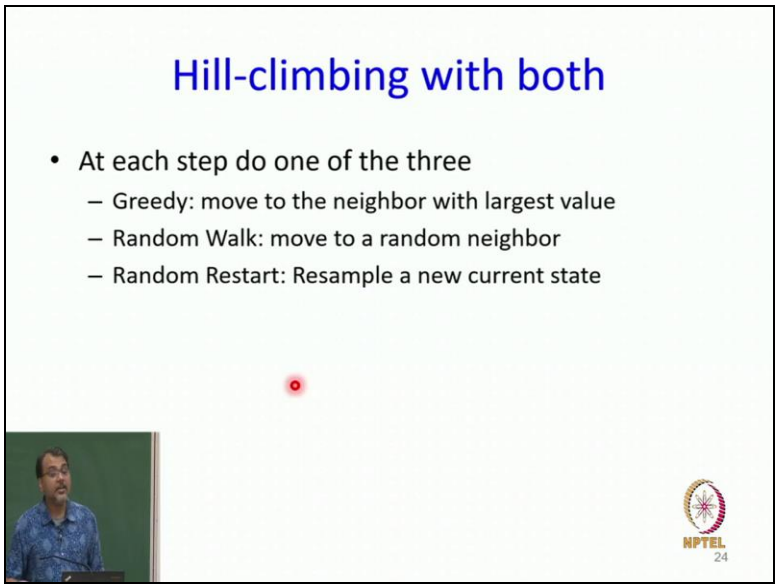
So for each restart now there choices so there are lots of parameters in all these algorithms. So, one parameter is for each restart do you go up to greedy optimum or you stop in the middle and then restart that is a choice. How many restarts you do, you do a fixed number of restarts or do you run indefinitely now that is not really a choice that depends on the total amount of time that is given to you. And you can also do some analysis and given the interest of time I will let you figure out this analysis.

But suppose each search has a probability P of success how many restarts would I need to do in order to hit a solution expected. How many expected number of the restart I will have to do this is math 101 for college, would be 1 by P. And you can figure out why. So, if it coincide ans succeeds 14% of the time and I am doing a random starting point every time in an expected

number of 7 restarts that means 6 into 3 18 + 4 22 hill climbing steps I will hit my solution in an expected set.

And as I said if you want to pick one local search algorithm to start with if you are given a new problem always you have to start defining your state space neighbourhood function to define your evaluation function you have to define a random function which will give you a new start stage random start state or possibly good random start state. But once you have done that the first algorithm I suggest you implement is hill climbing with random restarts and fine tune some parameters to figure out whether it gives you a good enough solution.

(Refer Slide Time: 18:46)



And of course you can do both so you can do hill climbing with both random walk and randomly starts at each step do one of the three either take a greedy move with some probability or you take a random walk move to a random neighbor with some probability or you do a random restart the resample a new current state. And a better algorithm which does this random walk with a slowly changing probability schedule it is called simulated annealing.

But in the interest of time we will start from here in the next class. Okay so we will start from simulated annealing in the next class we will then go to local beam search and its variations and one of the significant variations there is genetic algorithms which we will study in the next class.

After that our local search will be done and it will start with game playing as our topics next week okay let us stop.