Artificial Intelligence for Economics

Prof. Adway Mitra

Artificial Intelligence

Indian Institute of Technology Kharagpur

Week – 02

Lecture - 09

Lecture 09 : Multi-objective Heuristic Search and Game Trees

Hello everyone, welcome to this course on Artificial Intelligence for Economics. I am Adway Mitra, an Assistant Professor at Indian Institute of Technology, Kharagpur. Today we are on the 9th lecture of this course, like in the last lecture we discussed about the heuristic search algorithms in graph and the A^* algorithm. Now in today's lecture 9, we will carry forward some of those concepts. So, specifically we instead of just that is heuristic search we will extend that to multi objective heuristic search and we will also discuss the very important concept of game tree. So, that topics which we are going to discuss today we will start with that I will introduce what is meant by game tree.

And like how we can like find optimal paths in a game tree or optimal sequence of strategies game tree, then we will discuss the algorithms of alpha beta pruning and we will also extend like this the game tree as well as the this search techniques to the multi objective case. Now, let us start with an example. So, let us consider the game of chess. So, chess is a game which is played sequentially that is there are two players who play in turn in like by taking turns.

and their two their aims are exactly the opposite that is there is an initial configuration from which I like let us say typically the player who is playing with the white pieces they make the first move then the player who is playing with the black they make the next move and so on and so forth. So, that is the usual sequence of steps. when the first player one who is playing with white pieces their aim is something and the basically their aim is to win the game and similarly the player who is playing black their aim is also to win the game which means that the their two objectives are exactly opposite of each other that means if you consider a particular configuration of the chess board one which is very good for the white is automatically is quite bad for the black and the reverse also. So, that means, that as we are try like as they play in turns. So, we like we try to change the configuration of the chess board or the system in general to like to a configuration which is favorable to us and unfavorable to the other person by opponent player also does the same thing when they get their turn.

So, now why is this game hard? This game is hard because like when I am taking a move I want to understand whether this move will take me to a like a desirable state for me a goal state for me a state where I win or it will not take me. So, the usually the any move which I take it like it is not guaranteed to take me to either the victory or defeat. It all depends on what happens in the subsequent steps specifically what moves my opponent gives and what moves I give in the next step. So, like like ideally I would think that if I make me this move. then most likely my opponent will move that make that move and in which case I will make that move and in response to that my opponent will make such and such move and I will have to see it several times into the future and see whether I am reaching a goal state or not what is the goal state in this case a state where I win the game of chess But in of course, like in chess is such a complex game with so many different possibilities it is very difficult or impossible for a human brain to take to think of all possible scenarios that is what will happen if I make this move now what response my object my opponent can give after this what can I do.

So, like there are so many possibilities at every step I usually will not be able to think of so much. like typically a human it is almost impossible for a human being to think more than two three steps ahead. In fact, even then we may make mistake because we may overlook some opportunities right. The similar situation happens in a game of tic tac toe also except that the in this case the it is much easier because the possibilities are much limited. that is there are only 9^2 where we can which we can fill up and as we as the game proceeds the number of squares left that also decreases.

So, like the number of options in front of us becomes much smaller. So, we can it is easier for us to evaluate it. So, now this concept we can formulate again in the like as a graph where each graph indicates a state of the system. So, we have a starting state let us say the state from which the p_1 moves after that we like. So, like the only thing is that in this case the graph is arranged in the form of a tree.

So, a tree like now what is a tree we all know a tree is a graph which has no cycles, but like specifically the this tree is arranged in the like in clear layers like this. that is the we have a goal state sorry a start state which is the starting of the game that represents when the first player p_1 they make the move. Now, let us say that three choices are available in front of them when they make the first move. So, those three choices are represent I mean or the rather the outcome of those three choices are represented by these three variables. Now so when the first player makes a move they start with here and they may come to any of these like these things let us say it comes to this node.

Now when it comes to this node this is the state of the system then the second player p_2

it is their turn to move. So, now they will choose like they will have these two options in front of them. So, they will move to this one and so on and so forth this will go on until we reach the situation where no further moves are possible. So, like we reach the end of the game and then we see who has won and like and what was the cost spell like acquired in reaching that solution. So, now at any given point when I am starting initially like how will these are the three options available in front of me.

So, how will I choose where to go. So, the thing is at any give like every step just like we are talking about heuristic function earlier. What is heuristic function? Heuristic function is a estimate of like the distance of the goal state from that state. So, like in this case also for I am currently at the starting point. I will move to one I have three I can move to these three states I will move to that state which appears to be the closest to the gold ok.

So, like I do not know the actual distance to the gold. So, again I will have to use apply a heuristic function. So, let us consider a heuristic function like this. at each of these nodes I have again once again I have put some heuristic values. Now, the aim is I like that is when I am trying to move the I am player p_1.

So, I my aim is to find the shortest path to the to the to a goal node. So, I will now these heuristic functions they indicate the distances of these options. So, I like looking at these heuristic functions I think that if I come to this one I will probably be the closest to the goal node because this has the value heuristic value of 6 while the other two have much higher 9 and 7. So, I think it is best for me to come here. Now, player p_2 will move.

Now, when player p_2 moves they also have access to the heuristic function. Now, player p_2's objective is exact opposite of mine. I that is if I my aim is to reach the goal state my the player p_2's aim is to make sure that I am not able to reach my goal state. So, they will try to take the system to a state which is as far away from the goal state as possible. So, now so in my aim is to minimize the heuristic function or to choose that configuration for which the heuristic function is low in for my opponent the aim is just opposite it wants to or they want to change the system to such a state for which the heuristic function is the highest because that is disadvantageous for me and by extension advantageous to him or her.

So, like in this case assuming that I have come here there are two possible my adversary sees that there are two possible choices in front of them either they come here for which the heuristic is 2 or they go there for which the heuristic is 8. So, sees they want to maximize the heuristic. So, they will obviously take me here. Now, as we are doing this that is from here to here I as I moved the made this step I may have incurred a loss or I may have incurred a cost which is associated with these edges. So, I take care of that is I the price which I paid here is 2 and similarly when my adversary took the system from 6 to 8 it also paid a cost which is 7.

So, right now my objective I mean my adversary has taken the system to a situation which is disadvantageous for me, but in the process they have also paid a high cost. Now the same thing let us see it in a like a bit more elaborately. So, what do I do? So, like I mean the tree which we are like forming in this case this is what is known as the game tree. So, what happens in a game tree at like at every stage like the game tree is basically arranged into stages like this one stage is for the player 1 the other stage is for player 2. Of course, we can have more than two players also in which case like we will have let us say $p_1, p_2, p_3, p_1, p_2, p_3$ this is how the stages we are going to be organized.

Now each at each of these stage there are these possibilities and each possibility is associated with a heuristic value which indicates how good or how bad that possibility is for a particular player. Now it like the we follow a convention. that for player p_1 the aim is to minimize the heuristic function and for player p_2 the aim is to maximize the heuristic function ok. So, at every step the p_1 it tries to reach a vertex with the least value of heuristic function and when p_2 gets to turn gets to move they try to reach a vertex with the maximum value of the heuristic function. Now, in this case this process goes on till we reach a leaf node that is from which no further transitions are possible.

And then when such a situation is reached then every leaf node has a value function associated with it which indicates whether it is good for the player p_1 or good for the player p_2. So, as already mentioned player p like the low value of means good for player p_1. So, and high value means good for player p_2. So, these are the goal the leaf nodes from which it is no longer possible to move. So, I like if I reach any of these nodes I will try to understand whether like p_1 can be considered to have won the game or p_2 will have won the game.

So, like these are the values associated with the goals. So, I mean these values 4, 8, 0, 1 etcetera I have just put them. So, it suggests that these two nodes 0 or 1 these two goal node leaf nodes. they are good for player p_1 whose aim is to minimize and these two nodes 4 and 8 they are good for the player p_2 whose aim is to maximize. In fact, we can also say that like 0 is best for player p_1, 8 is best for player p_2, 1 and 4 you can say represent draw kind of conditions.

So, now like as we mentioning earlier. So, like player p_1 moved and brought it to this step after that player p_2 again saw that by checking the heuristic functions here of these internal nodes brought the system to this state. Then again it is turn for p_1 to move. So, p_1

sees that there are two options in front of them one of them has the heuristic value 6 the other has heuristic value 1. So, since they want to minimize they come here and in the process they incur a cost of 9.

Again p_2 moves p_2 sees there are two options in front of them like one is 8 the other is 1. So, they choose 8 and they come here in the process they incur a cost of 5. Now this 8 this turns out to be a goal state or a leaf node from here no further movement is possible and this 8 is considered favorable to player p_2. So, we say that the Player p_2 has won, but at a higher price. Player p_1 has played a price of 11, player p_2 has placed a price of 12 ok.

So, now the we the we may ask that ok. So, both player p_1 and player p_2 they made some decisions. Now, were they necessarily the right decision? That is if instead of choosing so player p_1 to move to this node based on this heuristic function which could have been wrong also heuristic function is just an estimate. So, it is possible that if instead of this being 7 if instead this had been 3 that then player p_1 would have come here. So, maybe in that player p_1 by coming here ended up losing the game maybe if they had come here instead then maybe they could have done better.

So, who knows. So, to know that we can go for the optimal path identification. So, for optimal path identification that is the our task is basically to estimate the value of these heuristic functions. Earlier we had just put this randomly. Now, the question is can we actually calculate them so that finding the optimal path becomes easier for both the things. So, we follow this algorithm called the minimax.

So, whenever we are considering the level in which like p_2 moves then we choose the maximum value and whenever we are considering the heuristic functions at the level where p_1 moves we are considering the minimum value. Why? Because see like if like we suppose we have come to this node and p_2 is going to move then obviously p_2 will come to here and he will never go here why because this is like a desired state for p_2 and this is not a desired state for p_2. So, from p_2's perspective when they come here it does not make any sense to go this side at all. So, they will always go to this side. So, like we can say that if it is here then the objective I mean the heuristic value for p_2 is 8.

like in this case it is I mean coming here is as good as coming here for p_2 the same goes for this node also it is the two possibilities are 8 and 1. So, if they come to this node obviously, they will come here they will never go there. So, it like the value I mean you can say that the heuristic of this function is 8 it means that if the if p_2 is able to reach this state somehow then they are guaranteed to get 8. on on the say on the other hand if you consider this step say for example, here like the p_1 has to make a move. Now, p_1 that is if

p_1 is here then they either they can come to this state whose value is 0 or they can go to this state whose value is 8.

Now, p_1 of course, wants to minimize. So, of course, they will choose the value come to this place of 0. So, like we can say that we like if p_1 comes here, it is guaranteed that they will come to this that that is the they will get the valuation of 0. So, with respect to p_1, I choose the minimum of these two values and I place the value of 0 as the heuristic for this node. Similarly, for this node also. So, like but in this case like only one possibility is there that is 8.

So, I like for both of these two nodes they get the value of 8. In fact, for this node there are two possibilities both of them have the value of 8. So, this node is also going to have the value of 8. that means, that if I if player p_1 reaches in this node no matter what they do they will like reach an unfavorable situation that is where the final result is going to be 8 which is not a favorable for them. This way we keep on moving upward that is like each leaf node already has a utility measure use it to update the heuristic values of the remaining vertices.

Consider each pre-leaf vertex if p_2 is going to make the move they will surely move to the leaf with the higher value. So, you copy the same value to the pre-leaf vertex and in for p_1 in the previous round p_1 will surely move to the pre-leaf node with the lower value. So, accordingly you update the estimates as we did here. So, for like so this is why it is called a minimax algorithm for optimal path finding. So, at the when p_2 is moving we choose the maximum that is like in the in the in the level where p_2 has to move the heuristic function is estimated by calculating the maximum of the of of their the maximum value of their children which in this case are the leaf nodes.

Similarly, for those layers in which p_1 is going to be is going to make the move the heuristic function is calculated by considering the minimum value of the their children and in so on and so forth. So, this is the way you calculate the in the heuristic function for the entire tree. Now, what is the heuristic the optimal path? So, in the original like if you consider the starting point its valuation is 0. So, p_1 is making a move. So, like the these are the three possibilities in front of them 8, 8 and 0.

So, obviously, they will move to 0. So, now p_2 it has has to make a move, but it has two possibilities in front of him both of them have the value of 0. So, like since both of them seem to be equally bad for the player p_2 they may choose the one which cost less. So, one cost 6 the other cost 8. So, they come to this one. now again p_1 has to move like there are two possibilities here either 8 or 0.

So, p_1 obviously will again choose the 0. So, they are like so I am basically blindly trusting my heuristic function now earlier I like I when I the heuristic function had been set So, randomly I did not I mean by trusting the heuristic function could have resulted in me going to a like a suboptimal solution or losing the game in other words, but in this case I have actually chosen the I have computed the correct values of the heuristic functions. So, now if I follow them I will reach what is the optimal result for me. So, I reach so player p_1 reaches this which is a leaf node and with the value of 0. So, player p_1 thinks that they have won the match. So, note that player p_2 they never really got any chance, player p_1 took the first move and at that time itself it was clear that they are going to win the game, player p_2 could not have done anything differently.

Now, so the p_1 we can say they had the first movers advantage, but if p_2 had the first movers advantage then like it can be like you can see they would have taken a different path altogether. And they would have reached this goal state which has a value of 4 which is like you can say an intermediate value which is neither too good for p_1 not too good for p_2. So, like this is what is known as the game tree. Now, when I am calculating this game or this applying this minimax algorithm to find the optimal like the optimal heuristic values for a game tree. I can actually like make this process bit more efficient that is I may not always have evaluate the entire game tree to calculate all the intermediate values.

So, it turns out that in some situations some parts of it become irrelevant that is if I like I will let us look at this example. So, I am calculating from the leaf nodes upwards. So, I see 5. So, like I want to like copy that value to the previous node here 5, but that also depends on this leaf node. So, but if the value of this is 6 which is lower than which is higher than 5, but my aim is in this layer is minimize.

So, I will not consider this 6 at all and I will stick to 5. now here again like I have found this value of seven and I copy that value seven in this intermediate node but then I see that it depends on these two leaf nodes also so here I consider this leaf node four and I place it here now it is already four Now if I want to place the value of 5 now when I see the value of 5 here then I do not that is I do not need to consider it because I have already found a value which is less than 5. So, if this 5 if it had children I would not need to consider this part of the tree at all because I have like already that is I know that. like the values which are below this they will not be any less than 5 because if this is anyway a max layer right. So, in that case so that is I I can just ignore this part all together and I can focus on 4.

The similar situation happens here say like here from this node onwards I have come to this node I have copied the value here this does not have any other children. So, the same value is propagated again here and then and this also has no other children so the this value is 5 is copied here now I want to see what about here now I so like I have to calculate this value that is I have to calculate the max of sorry the mean of 5 and this value but this value I have not yet calculated So, now I compare the sub tree this I compare this sub tree and I look at its leaves they have the values of 9, 8 and 6 all of which are greater than 5. So, that means, that like after applying the mean and max to this whatever value I get it will be either 9, 8 or 6 in any case it is going to be less than 5. So, I do not really need to calculate the value of any of these intermediate nodes I can just prune of this thing. So, this is known as the like alpha pruning where you just prune the sub trees with high values.

That is you have found a sub tree where you are convinced that its value will always be higher than the value which you currently have, but it is a mean node. So, mean node means you are trying to minimize, but you have found a sub tree whose value will always be higher than what you have found. So, it is not necessary to evaluate it. So, you prune that tree. Similarly, similar situation can arise in the max layer also you may have found a sub tree whose value is obviously going to be lower than what you currently have.

So, you do not have to evaluate that sub tree. So, you prune that sub tree. So, that is called as beta pruning where you prune the sub trees with low values. Now, we have already we have several times we have discussed the concept of heuristic search and on like we have considered the A^* algorithm as well as the game tree. Now, it may happen that the cost function is is is vector valued that is what is known as the multi objective heuristic search that is as you are moving like when we are considering the shortest path like we had only one objective that is to like the total path length, but suppose the path length has like multiple criteria associated with it.

So, the so in other words the cost path cost is a vector. So, when we are considering a heuristic function $h(x)$ though or the like $f(x)$ $g(x)$ etcetera those also then will have to be vector valued to like make sure you have the compatibility with the h cost. So, we have to change the A^* algorithm accordingly. So, remember that when we are considering the A^* algorithm one crucial point was at any given point when we are trying to move from the current state to the next state we were comparing the f values of the open vertices and whichever vertex had the least f value we move to that one. But now when we now that we are dealing with vectors identifying the smallest vector is that is often difficult because f is now vector valued. So, now it may happen that there are 3 nodes for the y_1 y_2 and y_3 for $f(y_1)$ is 348 $f(y_2)$ is 257 and $f(y_3)$ is 458.

Now, if you see these 3 4 8 and 2 5 7 these two they form a non dominating set that is like 3 is of course, that is 2 is of course, better than 3, but then 4 is better than 5 and 7 is better than 8. So, with respect to some of the vector dimensions y_2 has a smaller value of f and for some other dimensions y_1 has a smaller value of f. So, it is not very clear between y_1 and y_2 which I should choose, but if you see y_3 it is like in all the dimensions it is like at least its f value is at least as large as the other two. So, f y so we can say that y_1 and y_2 they both of them dominate y_3. So, y_3 I can ignore for now, but y among y_1 and y_2 there is no obvious winner.

So, I may like I now I cannot go to both y_1 and y_2 I cannot visit both of these nodes. So, I will have to choose one of them, but I cannot ignore the other also I cannot close the that vertex because I mean the vertex x from which I have got these neighbors because I that is like I have not been able to rule out the y_2 it is possible that y_2 will become useful for me later. So, that is it I cannot place it in the close list as I did in the original A^* algorithm. So, so and finally, the like A^* algorithm is guaranteed to find a path which is like I mean a path to the goal if it is like the if the I mean a shortest path to the goal if the if the heuristic function is admissible. In this case we cannot say that we will find the shortest path to the goal, but it is guaranteed that we will find a path to the goal which is non-dominated.

So, like like an example you can see here like the where the A^* algorithm is applied here for the lack of time I am not going to go into the details of this. So, but I will just show you the first two steps. So, we start at this node s. So, I have these two possibilities in front of me node n_1 and node n_2. So, for n_1 you will see that it has like so I am I am trying to calculate the heuristic function at each of these.

So, at like if you remember at the time when we discuss A^* algorithm we said that the heuristic function at any for any node can be simply the minimum value of its or the minimum cost of its neighbors. So, now this one it has two neighbors 3 and 4, one of them has the path cost of 3 2, the other has the path cost of 2 3. So, 3 2 and 2 3 however, I am not able to compare. So, they are non dominating set. So, I consider like both of them that is like as the as the choice of their heuristic.

The heuristic in this case is not a single value, but a set of values. So, this is what sets the a multi objective heuristic search different from the original heuristic search. In case of n_2 however, there is no such problem. In n_2 has two neighbors 4 and 5, one for one the path cost is 3 2, the other also has path cost of 3 2. So, either way the heuristic value of the heuristic is 3 2. Now, if you consider this n like n_3 or let us consider n_4 it has 2 neighbors one of them is 1 3 the other one is 1 1.

So, 1 1 however dominates 1 3. So, I consider the heuristic value of n_4 as just 1 1 and so on and so forth. So, this way we consider construct the heuristic function. Now, the same A^* algorithm we apply using these kinds of heuristic using these values of the heuristic. I start at n the next node is n_1.

So, the g the cost for reaching n_1 is of course, 2 comma 3. Now, like at n_1 the heuristic is like we have a set of heuristic values 2 3 and 3 2. Now, from n_1 if I go to 3 the in that case the value that is if I am choosing this value in that case the value of f will become the current value of g which is 2 3 plus the heuristic which is 3 2. So, 2 3 plus 3 2 is 5 5. right.

On the other hand I could also have chosen this as the value of my heuristic. So, 2 3 and 3 2 are both possible values of heuristic. If I say 2 3 then the value of f would be 4 6, but if I say 3 2 then the value of the heuristic would be 5 5. So, 4 6 and 5 5 both are possible values of f that is for n_1. for n_2 however like the path cost that is the g is 4 2 now I have already identified that the heuristic is 3 2 so the f value of n_2 is 4 2 plus 3 2 which is 7 4 so like I have got n_1 and n_2 so now like I may like you can see once again that these are non dominated sets So, like there is no obvious it is not clear to me which one I should expand.

So, let us say I decide I will expand n_1. So, again like when I expand n_1 I encounter n_3 and n_4. So, I like add them to the open list n_2 is still there. now again I compare their f values and this time I decide to expand n_4 now you will see that n_4 however called dominates n_3 even though it may not sorry I see that n both n_4 and n_2 they dominate n_3 so now between them I decide to expand n_2 first So, I expand n_2 and n 5 gets added and now I decide to expand n_4 and so on and so forth. So, this is how the multi objective heuristic works.

Similar situation I can use in case of multi heuristic objective game trees also. So, let us say that I am at a like I am trying to maximize like that is I am player p_2 who is trying who wants to maximize. Now, there are three like the options available in front of them 11 5 9 4 and 7 3. So, 11 5 of course, dominates 9 4 and 7 3. So, I just choose 11 5. If I am considering the player p_1 who wants to minimize then the like we see that from the perspective of minimization 7 3 dominates.

So, like for the player p_1 the obvious choice is to choose 7 3. So, the like the game tree accordingly can also be like modified in this way some of the and like as happened earlier some of the heuristic values they will not be single vectors, but actually collections of vectors and it will this will happen if there is a non dominated non dominating situation available like 11 5 and 5 7 you see that they these are non dominating solutions with respect to minimization. So, both of them are stored at possible heuristic values. So, so when do we where do we like encounter these situations in economics or why are these relevant for economics. So, game trees can represent many strategic interactions among various stakeholders in any market scenarios let us say competing companies or competing suppliers and consumers. So, each of them can be considered as different players each of them playing with certain objectives like which are selfish to themselves and they make moves turn wise.

So, like one company for one company making a move may be lowering the prices of something for another company it may mean raising the prices of something and so on and so forth. So, the whole situation is like the competition the market competition between the players is modeled as game tree. And like in case of behavioral economics also game trees allows the economics to predict how the rational agents are going to behave in various strategic situations such as like the competition or bargaining and or negotiations and so on and so forth. In case of macroeconomic policy making also like which we have the concept of multi objective optimization which we just saw. So, like for example, bank has to make a monetary policy and it has to keep changing its monetary policy like to and while doing so it has to like it has so many objectives to consider inflation, unemployment and so on and so forth.

So, with this we come to the end of this lecture. In the coming lecture which is lecture 10, we will start again shift tracks and we will start on data driven models or which we can call as machine learning models and how they can be like useful in various tasks related to economics. So, we will start with unsupervised learning. So, till then all of you please take care and stay well. See you soon. Bye.