CS323 Artificial Intelligence
Notes Assignment 3
Kartik Vyas B18CSE020

Adversarial Search

Till now, we have not considered opponents while achieving/optimizing our goal state. Now, we will consider opponents as well playing optimally/suboptimally and the players play according to their chance to do action according to their tasks. Algorithms have reached different levels [human, expert, solved] in various games such as Chess, Checker, Go. The game Pacman has not been explored to such an extent as of now.
There are various different types of games - single player or multiplayer, zero sum, etc. but the crux is that we focus on algorithms to come up with a suitable strategy (which can be termed policy) that decides our action for the next state. One suitable formalization for the games can be - states : representing the whole environment/situation of the game, players : the game can be single player/multiplayer, actions : the step that changes the state of the game, transitive functions : the next state is determined by the transitive functions based on what decision has been taken , terminal test : it determines if the game is finished and the player has lost/won, terminal utilities : it signifies the goodness of each state, it may vary for each player even if they are in same state and solution refers to the policy/strategy that leads us to action.

Zero Sum Games

These games are deterministic games in which the probability of chance is not accounted for. Here, there are two players who are opponents and they play turn by turn and maximizes and minimizes a certain target value accordingly. There is competition between these two players and they both play in accordance with opposite utilities. Another type of games are general games, where the players/agents may have independent utilities and their interaction with each other (that is cooperation and competition) is also accounted for. Zero sum games are basically a subset of general games only.
In zero sum games, both the agents/players alternatively (turn by turn) and they have opposite utilities. It is to be noted that here, the move of an agent depends not just on the environment but also on the position/status/move of their counter agent.

Single Agent Trees

In these trees, we assign a value to each state, this value decides how favourable the state is. More the value, the better the state is. The agent then decides which path is to be taken on the basis of these values which are basically the utility associated with that state. These values/utilities are assigned to the terminal states. For the non terminal states, the motive is to choose that path which results in the terminal state with maximum utility. In this way, we progress and with help of a recursive function of finding the path with maximum utility, we find our required path/policy/strategy.
In case of more than one agent (that is, if the opponent is there), the tree also needs to account for the actions of the opponent. Instead of taking maximum every time, when it is the opponent's

move, we should take the minimum of its successors' utilities in order to assign its own utility, this is because the opponent makes moves which minimizes the benefit of the main agent. Thus, we need to take minimum and maximum of the utilities at alternate levels to propagate the value to the root of the tree, from where we can decide the path.

Tic Tac Toe Game

This is a very good example of a zero sum deterministic game. Here, in the beginning we will have 9 states and then each of these states will have 8 states further extending from them. In this way, the whole tree would be very large, thus it is not feasible to check the whole tree and traverse to each terminal state and then propagate the answer. We can search till a specific depth and then we need to figure out the optimal path to follow. In the case of chess, we will have even more possibilities and a very large tree and a lot of terminal states, thus we should focus on how we can search on only till a particular depth and still get the same best results. Also, if we have a lot of levels, then like mentioned above, we need to propagate upwards in a recursive manner. If the action for a state is to be taken by the agent, then the utility assigned to it is the maximum of the utility of its successors (that is, the collection of possible next states from that state) , whereas if the action is to be taken by the opponent, then the utility assigned to the particular state is the minimum of the utility of its successors. And in case there are no successors, that is in case of terminal states, its utility is assigned on the basis of how favourable the state is for the agent.

Minimax Algorithm Efficiency : Let m be the depth of the tree and b be the branching factor, then we can think of the minimax algorithm as the DFS approach. Thus the time complexity is O(b^m) and the space complexity is O(b*m). This amount of computation is obviously not feasible. This leads us to the thought that do we need to search the entire tree. Also, it is interesting to note that in case the opponent does not play perfectly, then we take a path that might include some risk but also has great benefit. We can assign some probability to paths accordingly and then decide the path. This will be discussed further in much detail.

Resource Limits : In case of large trees, we need to make certain approximations and search only till a limited depth to make it practically feasible. Though, it is to be noted that now we cannot guarantee that the agent will make optimal moves. Also, we can make use of iterative deepening approach, in which we basically, search till a level if time allows, and if we further time, then we search on till next level, otherwise we stop the search and decide the path based on current level utilities which can be determined with the help of an evaluation function that is for non terminal states. This approach of iterative deepening is also called anytime algorithm since it determines the depth till search must continue based on the time that is allowed or is feasible. It is to be noted that these evaluation functions for non terminal states are imperfect and it is better if we could search at a deeper level instead of using a complex (that is compute intensive) evaluation function; this is because evaluation function give an estimate only, though in case of unsolved games such as chess, evaluation function play a very important role since the tree would be very large.

Evaluation Functions

The ideal/ perfect function returns the utilities based on actual minimax, and the utilities of terminal states with a less computing and accurate evaluation function. Though, usually in practice, we account for certain features and assign weight to them based on importance and then their linear weighted sum accounts for evaluation function; though this will also add to the computation but we will get more accurate function as we consider more features, however searching till a further depth, if possible is always to be preferred. The estimation of the evaluation function is not trivial, there are a lot of factors that need to be considered. For example, in case of Pacman, the number of food items already consumed, the relative positions of opponents, food items with respect to the agent, they all play a role. Sometimes, we need to wait for some moves in order to decide which state is more favourable. It is important to note that for the same number of food items left, we still need to account for the distance of food items from our agent, otherwise we might not take the most optimal path.
The opponents may coordinate among themselves as well in order to kill the pacman by working together and deciding their moves accordingly instead of just naive minimizing the distance approach.

Game Tree Pruning :
In this approach, we do not need to compromise over the guarantee of finding the most optimal path. In this approach, we check if a particular state' successors can even lead to the most optimal path or not, based on this, we decide if we need to further search that branch or not. In this way, without ignoring the paths, we have checked all the possibilities and saved computation time as well.
Alpha Beta Pruning : The approach is that, let us consider that we need to find the utility of a state which is governed by the agent's action, then we need to find the maximum of its successors' utility, which in turn is calculated by the minimum of their respective successors' utility. If we have found the utility of a successor, say x and then while computing another successor's utility, we find that its utility would be less than x, then in this case there is no need to search all the successors of that node, as this particular child will not be contributing to the utility of the required state. Similar observations can also be made for the states that are governed by the opponents. In alpha beta pruning, alpha is for computing the maximum best utility and beta is for computing the minimum best utility. In this way, we can recursively propagate to the top required state and thus identify the best possible path. It is also clear that the ordering of the successors is also a factor for the effectiveness of this algorithm. If the successors are in a proper order, then we can skip more searches. The time complexity reduces to O(b^(m/2)), which also means that effectively now we can double the depth till we could search feasibly otherwise.

Uncertainty and Utility
We earlier briefly discussed the case in which the opponent does not play perfectly, we will go in much detail now. The move of the opponent is not adversarial and is rather a chance. The opponent may choose a path based on its probability. The utility function to be computed will also change. For the computation, we should obviously consider the probabilities of the further

succession and then we can use the linear weight sum of all these probabilities as weights for the successors' utilities. It can be noted that this is for average case outcomes and thus named expectimax search, rather than minimax search which considers worst case outcomes. This expectimax search utility computation modification would only be for the opponents' governed states, otherwise for the agent's states, we will use the same minimax search approach.
Expectimax Pruning : Expectimax pruning gets a bit tricky. Let us consider that we need to compute the maximum utility for a state and we are searching one of its successor states, we cannot really skip the traversal, since it may be possible that we get a very large value at its last successor, which in turn makes this particular child favourable for the maximum utility of the original concerned state. However, we can always use depth limited expectimax where we can approximate the utilities after searching till a limited feasible death and then come up with the path..

Probability

A probability distribution is the allocation of probabilities (chance of outcome) to all the possible outcomes. A random variable signifies an event whose outcome is not done before the event actually occurs. It is to be noted that the probability of an event can never be negative and that the sum of probabilities for all different outcomes of an event always add to one. Probabilities are bound to change as we have more information, that is if we have some information about an event, and there is some dependence between these two events, then our associated probability will have a different value. The expected value of a random variable is the weighted linear sum of the outcome distribution with the respective probability as weight.
Informed Probabilities : In this case, the opponent is basically not randomly selecting the next successor state and may be partly intelligent. It is to be noted that in case the opponent is doing both (using minimax and random selection), we should use expectimax algorithm since there is some chance involved. Though this would take more time than the minimax algorithm since the tree could be very large and we would need to consider a tree for every opponent governed state; we can then decide the path according to the percentage of random selection and minimax selection that the opponent is using.

Modelling Assumptions

We will discuss that with which algorithm, that is minimax or expectimax, should be selected. Expectimax is not time efficient since we need to consider multiple trees and minimax works in the worst case, that is when the opponents play perfectly. It may so happen that sometimes we are optimistic but then the opponents' play is adversarial and we are pessimistic when the opponent does not play perfectly.
In case of minimax algorithm for Pacman game, the agent does not die (mostly) and the score in case of random ghost opponents is more than in case of adversarial ghost opponents; this also makes sense since in the case of adversarial ghost opponents, the opponents consider the worst case. In case of expectimax algorithm for Pacman game, in case of adversarial ghost opponents, the chances of defeat of pacman agent (that is the death of the agent) are quite high, since the expectimax algorithm would consider the average case but the ghost opponents

would select the worst case, which results in death of the agent. For random ghost opponents, the score is highest among the four cases which it should be as well since both the calculation of the action and the actual action of the opponents are considering the average case.

Other games

In case of a game such as the likes of Ludo and Backgammon, where there is a random chance and then we play best according to that outcome, or it is the other way round (that is we make some move intelligently and there is some random chance), then we use expectiminimax algorithm. In games in which there is a very big tree formation due to a lot of possibilities, the chance for each chance becomes less and thus the chance of reaching a particular search node also decreases, thus checking till limiting depth is less damaging.

Multi Agent Utilities : In case of multiple agents who play turn by turn and try to maximize their respective utilities, then at each level, the agent selects the utility according to its consideration discarding the utilities associated with the other agents. This will lead to indirect cooperation or competition between the agents. Overall, this may be a bit complex behaviour as there are many agents who try to maximize its own utility and upon several levels of operations, we might get some unexpected(undesirable/non accurate) results.

Utilities

Different agents might have different preferences over utilities, that is utilities are specific and their preference varies from agent to agent. We prefer to use average utilities instead of minimax utilities since it is generally better to take some risk in order to have a chance at very good utilities. An agent always tries to maximize its utility by its actions.

It is to be noted that the magnitudes play a significant role in the expectimax algorithm. In minimax algorithm, by applying a monotonic transformation (that is, higher value gets higher transformation and smaller value gets smaller transformation), the path to be selected still remains the same, though this need not be the case in expectimax search as then, the sum of these values might alter the path since the rate of increase of different utility values can be different. Utilities signify the state's closeness to the agent's goals. It puts the relatedness/similarity in terms of numeric terms, with the help of which the agent can make rational decisions. We can assign the utilities to the terminal states and then the agents evolve with the decision; we do not let the agents themselves decide the utilities in order to save up on the computation power. In case of uncertain outcomes, we take the help of utility values only and come up with the final decision.

Rational Preferences

We can make note of some constraints. Let us say that the agent prefers A over B and prefers B over C, then A is preferred over C. If A is preferred over B and B is preferred over A, then the states A and B are indifferent. These sets of constraints are the axioms of rationality. We can represent in the form of symbols. The symbol '>' denoted higher preference of the former and the symbol '~' denotes indifference between two states. The first axiom is orderability, that is

(A>B) U (B>A) U (A~B). The second axiom is transitivity, that is (A>B) ^[Intersection] (B>C) => (A>C). The third axiom is continuity, that is (A>B>C) => there exists a p such that [p, A; (1-p), C] ~ B. The fourth axiom is substitutability, that is (A~B) => [p, A; (1-p), C] ~ [p, B; (1-p), C]. The fifth axiom is monotonicity, that is (A>B) => ((p>=q) ⇔ [p, A; (1-p), B] >= [q, A; (1-q), B]). Our rational decisions will follow all these axioms and these signify maximization of the expected utility.

MEU Principle

It stands for Maximum Expected Utility Principle. We know that, if expected utility of A is greater than that of B, that it would also mean the prize/benefit of A would also be greater than B, that is A would be more preferable as compared to B. We can also infer that U([p1,S1; … ; pn,Sn]) = Summation of [pi * U(Si)], where i varies from 1 to n.

From this, we can observe in order to maximize the expected utility, we must choose that action/node which has the best utility of its own. This is known as MEU principle.

Utility Scales : It is to be noted that if the applied transformation is linear and the multiplying factor is non negative, the results would not change, that is linear transformation is invariant.

Human Utilities : Let's consider a lottery system against a definite prize A. And the best possible lottery prize is U+ and the worst is U-. We can consider U- as the death of the person who is involved in the lottery. Then for a p such that A ~ L; where L is expected lottery prize money and ~ represents indifference, where p is the best prize money probability and (1-p) in turn is the worst prize money probability. Also, now p would result in a utility in [0,1].

Money Utility : Money as a utility may seem as a good idea, though actually it is not the case. Let us consider that there is a lottery, prizes for which are X with probability p and Y with probability (1-p). Here the EMV [Expected Monetary Value] is [p*x + (1-p)*Y] and the utility of lottery will be [p*U(X) + (1-p)*U(Y)]. Now, in a considerable number of cases, U(L) < U(EMV(L)). Also, a particular amount of money would have different utilities for people with different amounts of wealth; that is for one person a particular amount of money would be a great deal, but for another person that may not be the case.

Insurance :

Let us consider a case of laundry which offers 1000 dollars prize upon with 0.5 probability and 0 dollars otherwise. In this case the expected monetary value is 500 dollars. Now, most likely a person agrees to selling the ticket at 400 dollars, since this money is definite that the person will receive and minimizes the risk, this amount is termed as certainty equivalent. The difference of 100 dollars between the expected monetary value and the certainty equivalent is termed as insurance premium. Since the insurance company buys a lot of tickets, we can likely assume that the insurance company earns around 100 per ticket, buying a lot of tickers under a lot of policies/schemes, the company ensures the chances of its profit. This thus is a win-win situation for both the parties, that is the person to whom initially the lottery ticket belonged to and the insurance company.

Human Rationality : Let us consider a comparison between two lotteries, the first one gives 4000 dollars with 0.8 probability and 0 otherwise and the second one gives 3000 dollars in every case. It has been observed that the majority of people would go on to take lottery 2, despite the fact that utility of lottery 1 is greater than lottery 2. However this trend is not observed in the case lottery 1 offers 4000 dollars with 0.2 probability and 0 otherwise and lottery 2 offers 3000 dollars with 0.25 probability. Here, people are likely to prefer lottery 1 and the fact that the utility of lottery 1 is greater than lottery 2 also supports it. This anomaly is observed due to the chance of getting 0 dollars in case 1's lottery 1. Hence, we must consider the case of the utility of 0 dollars as well.

References :

Attended class lectures of CS323 Course, IIT Jodhpur

I have not taken help besides this single source, I went through the class videos once again and then wrote these notes in my language based on my understanding from the above.