Homework Assignment 3

601.464/664 Artificial Intelligence Spring 2020

Due: April 10, 2020

Mou Zhang

March 18, 2020

Game Playing

In the past weeks, we discussed intelligent agents and how they can use tree searching techniques to solve abstracted problems. In this assignment, you will implement chess-playing agent and answer some theoretical questions.

Question 1. Open the following google colaboratory notebook. Follow all the steps specified in it. Include link to your solved notebook in your submission. Optional: implement your own chess-playing agent and we will run a small competition between agents of other students (you can work in teams).

https://drive.google.com/open?id=1QfxhGnMP-V42sJCeGcY44sPETeUKx818

Question 2. Why do we assume that we play against an optimal opponent in the minimax algorithm. What happens otherwise?

Because we assume that our opponent is doing his best in the game playing. If we assume that we play against an non-optimal opponent while we are calculating in the minimax algorithm, then we it is likely that we will lose to an opponent with optimal algorithm. If in fact the opponent chooses a non-optimal stratary, then we will do better than against an optimal opponent.

Question 3. What kind of node exploration is the minimax algorithm using? Depth-first or breadth-first?

Depth-first search.

Question 4. What is the time complexity of the naive minimax algorithm? Prove it.

$O(b^m)$

Proof. As we can see from the search tree, in each step, there are b options for 1 option in the previous step and there are total m steps. So in each step, the options are b times of the options in the previous step. In the final step, there are total $1 + b + b^2 + b^3 + \ldots + b^{m-1} = O(b^m)$ nodes.

Question 5. Explain why minimax algorithm with α–β pruning is more efficient than naive minimax. What is the complexity and why it depends on the ordering of the elements?

It is more efficient because that alpha–beta pruning eliminate the unpromising branches in the search tree. This makes sure that serach only happens to more promising subtrees, which is more efficiency.

The complexity of minimax algorithm with $\alpha - \beta$ pruning is $O(b^{\frac{m}{2}})$ with perfect ordering.

It depends on the order because if your serach is optimal at first, this will helps you to eliminate large numbers of unpromising branches in the beginning, which can largely improve the algorithm.

Question 6. Why did we introduce EVAL function instead of UTILITY for some games? Explain what is a good EVAL function for chess and how it affects the minimax algorithm.

To overcome the resource limits, especially the time limits. Since there are too many possibilities while searching, we use EVAL to find the next step quickly.

There are three standards for a good EVAL function:

(1)The evaluation function should order the terminal states in the same way as the true utility function: states that are wins must evaluate better than draws

(2)The computation must not take too long

(3) For nonterminal states, the evaluation function should be strongly correlated with the actual "chances of winning."

The EVAL function affects the minimax algorithm so that it tends to keep more high-value pieces, therefore, it tends to have a better chance to win.

Question 7. What is a Horizon effect and Quiescencia?

The horizon effect is a problem which can occur when all moves from a given node in a game tree are searched to a fixed depth. Horizon effect means that minimax algorithm can only foresee several steps, but it has no long-term consideration. As a result, it can not see the big problem unless the problem approaches. By the time the problme must be dealt with, it has become too hard to solve.

Quiescencia state is that the state unlikely to exhibit wild swings in value in the near future.

Quiescencia search is an algorithm used to extend search at unstable nodes in minimax game trees. It is an extension of the evaluation function to defer evaluation until the position is stable enough to be evaluated statically, that is, without considering the history of the position or future moves from the position. It mitigates the effect of the horizon problem. This can provide the deeper search of game-changing moves.

Question 8. Under what kind of transformation the behaviour of minimax algorithm is preserved in case of a game with no chance nodes? In case of a game with chance nodes?

In case of a game with not chance nodes, behaviour is preserved under any monotonic transformation of EVAL.

In case of a game with chance nodes, behaviour is preserved only by positive linear transformation of EVAL.

Logic

Translate the following English sentences into *propositional logic*

Question 9. A and B are both true.

$IsTrue(A) \wedge IsTrue(B)$

Question 10. If A is true, then B must be true as well.

$IsTrue(A) \rightarrow IsTrue(B)$

Question 11. If a student studies for a test, they will do well on it. We can also tell that if a student did well on a test, then they must have studied for it.

$\forall x \in students, (Study(x) \rightarrow DoWell(x)) \wedge (DoWell(x) \rightarrow Study(x))$

(Which is the same as $\forall x \in students, Study(x) \Leftrightarrow DoWell(x)$)

Question 12. If a student is completely dry and it is raining outside, it is because they have an umbrella or a hoodie and it is not raining heavily.

$\forall x \in students, (Dry(x) \wedge Rain) \Rightarrow (HaveUmbrella(x) \vee HaveHoodie(x)) \wedge \neg RainingHeavily$

Question 13. Simplify and translate the following *propositional logic* sentence into English: $A \vee (A \wedge B) \Leftrightarrow \neg(A \wedge B \wedge C)$

Since $A \vee (A \wedge B) \Leftrightarrow A$, the original *propositional logic* sentence is simplified to
$A \Longleftrightarrow \neg(A \wedge B \wedge C)$

In English it's : If A, then not A and B and C, vice versa.

Question 14. Is the following sentence valid? $A \vee B$

No. A sentence is valid if it is true in all models. In this case, $A \vee B$ is not true when A is false and B is false.

Question 15. Is the following sentence satisfiable? $A \Longrightarrow B$

Yes. A sentence is satisfiable if it is true in some model. In this case, when

A is true and B is true, this sentence is true.

Question 16. Is the following sentence unsatisfiable? $(A \wedge (B \vee C)) \wedge ((A \wedge B) \vee (A \wedge C))$

No. A sentence is unsatisfiable if it is true in no models. In this case, if A ,B and C are all true, the sentece is true.