Introduction to Artificial Intelligence

Andres Ponce

March 17, 2020

Suppose we have a game with two agents. The two agents each make moves that are optimal for them, according to some constraints. They may not know all the environment, but they can still make optimal decisions from what they do know.

Chapter 5: Adversarial Search

5.2 Optimal Decisions in Games

Because adversarial circumstances allow the agents to influence each other's decisions, we need some way to still arrive at an optimal solution for the agent we wish to "win". To recap, we have a **Min-Max** situation.[1]

[1] In this situation, the **max** agent takes the option with highest value, and the min agent will choose the one with lowest value as a counter move.

Remember the UTILITY function tells us how useful a certain end state is for our given configuration. The utility is defined by adding the utility of the leaf node and the sequence of min-max moves from the node to the leaf node.

The MINMAX algorithm backs up from a leaf node, adding either the max or the min utility values as it backs up, depending whether it is **Max**'s turn or **Min**'s. This assumes that both agents always play optimally and know what the best move is at every state.

Although this algorithm will search every state in DFS fashion, it will clearly be too slow for anything resembling a real game. However, other algos use this as their starting point.

When we have **multiple agents** in a single game, we need a vector of utility values for each state and every agent. Two weaker agents could form **alliances** to bring down the bigger agent, however, since this is purely selfish behavior, the alliance will probably dissovle when there is no more threat from that agent.

5.3 Alpha-Beta Pruning

The idea behind **pruning** is that we can remove one of the large exponents behind the search. The way we remove the steps is by realizing that for a node n somewhere in the tree, if there is a better move anywhere above it, then n will never be reeached by the minmax algorithm. This means that we can safely not even search it. Since we continue to switch between Max and Min nodes, if the node to be chosen by a Max node is greater than the node to be chosen by a Min node, we node that the node chosen by a Min node will never be

chosen by a Max node, so we can safely discard it.[2]

When we are dealing with search trees, we have to be careful to avoid **transposition**, since that can really throw off our attempt. [3]

5.4 Imperfect Real-Time-Decisions

[2] The **alpha** and **beta** values refer to the maximum value of the Max and Min players, respectively. If $\alpha > \beta$, then we can say we will never choose β as our best move.

[3] Transposition refers to multiple move sequences that lead to the same end result.