# *Introduction to Artificial Intelligence*

*Andres Ponce*

*March 9, 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.

### *5.3 Alpha-Beta Pruning*