Introduction to AI Technology (2009)

Paper Summary

The reason why I chose this paper is because this is my first time learning AI technique while joining AIND, so in addition to the content introduced in the nanodegree, I want to make myself more familiar with the basic concept of how to apply AI techniques to game playing.

This paper describe serval ways of how human had tried to apply basic AI techniques to game playing. As described in the paper, the easiest way to teach a computer how to play a game like tic-tac-toe is to manually specify some rules for the computer to follow, like "if there is a winning move, take it" or "If your opponent has a winning move, take the move so he can't take it". The downside of this approach is obvious: the human involved have to manually specify all the rule so it doesn't scale to the number of all possible outcome of games.

Instead of rule-based approach, we may instruct the computer to "simulate" all the possible moves that the computer and its opponent can take and make the best move corresponding to best simulated result. This approach work well when we have superior computing power and relatively limited possibility of outcome, but it still has downside because the we can't always search to the "bottom" (of the game tree) in the limited time constraint, which is common for game playing.

In order to solve the problem that we can't alway search to the bottom, we have two directions for improvement when doing searching: (1) try to search minimum number of nodes as possible as we can given existing knowledge using techniques like Alpha-Beta Pruning. (2) try to guide the computer toward goal (which is to win game) using a game-specific heuristic scoring function when we only search for limited depth of the game tree and thus the final outcome remain unknown.

Although we can let the computer make "smarter' decision by specifying a better heuristic score function, what inspire me most is the idea that wouldn't it be perfect that we can let the computer learn the optimal score function that enable computer to make the best moves every game state? For example, it's common to design a heuristic scoring function that compute score for a given game state using the formula "(1)#available_moves_for_computer_player - (2)#available_moves_for_opponent". In this case, the weight for (1) and (2) is the same, that is, equal to 1. What if we can set the weight for (1) and (2) as w1, w2 respectively and let the computer learn the value of the weights w which lead to the best strategy? There is no discussion about it in the paper but it allow me to think further when trying to design heuristic functions for a specific game like the isolation described in the nanodegree.