PAPER REVIEW: GAME TREE SEARCHING BY MIN / MAX APPROXIMATION

In this paper, the author attempts to solve the minimax game tree by using the min/max approximation as a heuristic to determine the next node to expand along the tree. This is a special case of a more general method called penalty-based search method, whereby non-negative penalties are assigned along the edges of the tree to penalize bad moves. The leaf to expand is obviously the one which has least penalty.

To implement min/max approximation the author has to rely on the generalized mean value operator to approximate min and max operator. Penalties in this case takes the form of derivatives of the generalized mean value function.

Unlike max/min operators, the generalized mean value operator has continuous derivatives which makes it suitable for finding the leaf whose value at the root depends most strongly, by applying chain rules to get the derivatives.

The generalized mean-value operator however happened to be very costly to compute. To avoid that the author skipped the computation and used approximated max and min values instead. This is called “reverse approximation”, since generalized mean value is initially supposed to approximate max and min values anyway.

To compare min/max approximation with alpha beta pruning, benchmark had been run on the Connect-Four game with 2 different resource bounds: CPU time which represents timely bound and number of calls to move function which represents computational power bound.

The experiment was setup like this. Considering 49 different starting positions, where for each of them one game is played with min/max approximation moving first and one game is played with alphabeta moving first. There will be 49*2=98 games in total. For each of 5 time bounds and 5 move bounds, one such experiment will be run. Therefore the whole experiment consists of 98*5*2 = 980 games. The author observed that while alphabeta showed superiority with time bound, min/max approximation dominates on move bound however.

A few more observations have also been discussed: First, min/max approximation takes a lot of memory since it has to store the whole tree that is being explored. Second, min/max approximation maybe inefficient when there is only one move to make from the root, since it focus on improving the value of the estimation at the root rather than finding the best choice to make from root. Third, min/max approximation requires all the successors of a node being evaluated, which makes it less efficient than other techniques that can skip this step, for example alphabeta pruning.

The paper concludes by emphasizing on the victory of min/max approximation using move bound, and leave some open questions for readers such as how to find the best penalty function to use, how can this be combined with traditional techniques such as depth first search to get the best of both worlds, how to parallelize and how well will it work in games where sacrifices has an important impact on the outcome.