Game Tree Searching by Min / Max Approximation\*

Min/Max algorithm is used to evaluate the result of game. Min/max algorithm requires to create a full tree in memory. alpha-beta pruning helps to reduce the number of calculation required. This paper tries a solution for “A method is needed which will always expand the node that is expected to have the largest effect on the value”.

Paper introduces with the concept of generalized mean value. Using this we can identify a path from leave to root and root value is highly depended on this root.

We can add penalty on each in game tree. Penalty is a weight on edge. Penalty is depended attributes like, level of node and sibling of node etc. We penalize a bad edge by giving it high weight. At each level to evaluate a partial game tree a static evaluation function is used. This requires to store explored tree to be stored explicitly. Hence it increases system memory requirement.

Now at each level this is penalty is assigned. Node with least penalty is picked to expand the game tree. This steps are repeated iteratively to find best path.

Before taking a move penalty approach would first define the complete path. This is overkill if you have only one move at root.

The results were encouraging.