AIND
Alexandru Oprisan

Review of
“Game Tree Searching by MIN/MAX Approximation (by Ronald L. Rivest)”

1. A brief summary of the paper's goals or techniques introduced (if any).

The Paper introduces an iterative method of searching min/max game trees.
The method presented in the paper is iterative and requires the search tree to be available (in memory) , as opposed to Iterative Deepening or Minimax algorithms which perform depth-first search.

The question the Article answers is “What node to explore next?”

The basic idea of the method is to approximate (replace) “min” and “max” operators by generalized mean-valued operators and iteratively expand the search tree (starting at the root) by including the leaf node that affects the value at the root the most.

The grown search tree will therefor most probably be not of uniform depth. (Some branches will grow more then others).

Each node the expanding search tree is assigned a value “v” based on the values of its descendants and weather it is a MIN on MAX node. This value “v” is calculated by applying a mean-valued operator with p>> 0 for MAX nodes (or p<<0 for MIN nodes) to the descendants of the node..

Using this value “v”, we can approximate the impact of each leaf node (tip) has on the root using the derivative of “v”.

At each iteration we choose to expand that tip which mas the greatest influence on the root node. Then we update the values “v” for the nodes along the path from root to chosen tip.

2. A brief summary of the paper's results (if any).

The method has high CPU overhead and requires significantly more memory compared to Minimax or Iterative Deepening.

The proposed method is evaluated against minimax search with alpha-beta pruning.

The authors use “Connect-Four” as the game to play starting in 49 different positions. Both algorithms where implemented in C and ran in parallel on 10 DEC MicroVax workstations . For each initial position, two matches were played, with each algorithm taking the first move once and the second move once. In total 980 matches were played.

When the constraint applied is “number of calls to <<move>> function”, the proposed method outperforms minimax. However, when the constraint applied is “CPU time”, minimax performs better.