Review of "Game Tree Searching by Min/Max Approximation" by Ronald L. Rivest

Abderrahim Kitouni

June 26, 2017

The paper presents a technique for searching a game tree based on approximations of the min and max operations by generalized mean values.

Given n positive numbers $a_1, \ldots, a_n$, the generalized p-mean of $a_1, \ldots, a_n$ is defined by

$$M_p(a_1, \ldots, a_n) = \left(\frac{1}{n} \sum_{i=1}^{n} a_i^p \right)^{1/p}.$$

For p large enough (say $p = 32$), M_p is a good approximation of max, and M_{-p} is a good approximation of min. They can be used alone as a replacement for max and min in a minimax playing agent, and have some potential for better play. A branch which has multiple good moves will have greater generalized mean than a branch which have only one best move even though the max would be the same. (For example, $M_{32}(35, 40) = 39.16$ and $M_{32}(30, 40) = 39.14$, while $\max(35, 40) = \max(30, 40) = 40$.)

However, the real advantage of generalized mean values is that they have continuous derivatives, which can be used to find which values have more effect on the value of the generalized mean. This can be used to find which leaf of the tree has more effect on the value at the root of the tree.

This can be used as part of what the paper calls iterative search: given a partial game tree, we pick one of its leafs that is not a terminal state (called expandable tip in the paper), and expand it one level deeper. As the partial tree grows larger, we get more accurate evaluation of the outcome of different moves.

The question is how to choose which tip to expand. To this end, the author presents a general penalty based method for choosing a tip. It consists of assigning a "penalty" or "weight" to every edge of the game tree, such that bad moves are more penalised. The penalty of a leaf would then be the sum of the weights of every edge between it and the root.

The weight proposed by the paper is the negative logarithm of the derivative of the value of the parent node with respect to the value at the child node. This way, the total penalty of a leaf would be the negative logarithm of the derivative of the value at the root with respect to the value at the leaf. (This comes from the chain rule for derivatives).

Then the author goes on to suggest using an "inverse approximation"; that is, to compute the values using ordinary min and max, and only use the generalized means to compute derivatives. The advantage here is that a min or max needs less CPU time than taking powers and roots.

The paper ends with experimental results comparing the performance of the proposed approach (using the inverse approximation) against iterative deepening with alpha-beta pruning. The results is that min/max approximation gives better results when visiting

the same number of tree nodes. When the limiting factor is CPU time, however, iterative deepening with alpha-beta pruning gives better results.

I have tried implementing the algorithm proposed by the paper for the knights-isolation game, and noticed the same thing (less performance with the same CPU time). So even with modern hardware, the result is the same.