Summary of “Game Tree Searching by Min/Max Approximation” (1988) by Ronald Rivest

Brian L. Cooley

February 15, 2017

1 Summary

The paper introduces a new technique for searching game trees by approximating the min and max operations with a generalized mean. In contrast to the static heuristic, uniform depth search of game trees that we have studied in class, the technique presented here uses an ”iterative” heuristic which grows the search tree one leaf at a time and may result in search trees without uniform depth (somewhat analogous to depth first search versus breadth first search).

The basic idea of the iterative heuristic is that the generalized mean:

$$M_P(a) = \left(\frac{1}{n} \sum_{i=1}^{n} a_i^p \right)^{1/p} \tag{1}$$

tends towards the max operator as $p \to \infty$ and towards the min operator as $p \to -\infty$. Good approximation of the operators for $|p|$ as low as 10 are shown in the paper. The usefulness of (**??**) is that its derivative is both easily computed and continuous.

The key idea of the paper is to use these generalized mean values to approximate the min and max functions and identify the leaf in the game tree that the root depends on most strongly by taking derivatives of (**??**) at each node and using the chain rule. That leaf is then chosen as the next leaf to expand as the game tree is searched. Its successors are added to the game tree and the process is repeated.

The biggest issue with the approximation is that it is (relatively) computationally difficult to compute the generalized mean and/or its derivative. The author makes some discussion of this, but as will be seen in the following results section, the additional computational burden is indeed impactful on the performance.

2 Results

The paper presents some experiments using the approach on the game of *Connect-Four*. At first glance, the results are somewhat disappointing because regular minimax search with alpha-beta pruning performs better when the calculations are bound by CPU time. However, when the calculations are limited by calls to the underlying ”move” operator, the author’s approach is superior.

However, the results do show that using the derivative of (**??**) is a superior way to decide which leaf to expand at each step of the algorithm. In fact, the standard minimax with alpha-beta pruning approach called the move operator over 4 times as often per second as the author’s approach (3500 vs. 800) and considered roughtly 3 times as many distinct positions when time rather than moves was the limiting factor. The author notes that when special-purpose hardware is used or when the move operator is expensive to implement, the move-based limit would be more relevant than the CPU time limit.