Summary of

"Game Tree Searching by Min/Max Approximation"

Published in *Artificial Inteligence*, **34**, 77-96, 1988.

The paper aims to introduce a new method that delivers better results compared minimax search with alphabeta pruning – similar to what we learned here. To do that the author proposed using somewhat different mathematical expressions for min and max function. To describe the approximate min and max functions, a generalized *p*-mean of *a*, $M_p(a)$ is written as:

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

Where *a* is a vector of *n* real numbers and *p* nonzero real number. Note that if *p*==1, M_p becomes an arithmetic mean. From here min and max are rewritten as:

$$\lim_{p\to\infty} M_p(a) = \max(a_1, \ldots., a_n), \qquad \lim_{p\to-\infty} M_p(a) = \min(a_1, \ldots., a_n).$$

In a nutshell the idea is: instead of backing up alpha and beta values for every node per depth as it is done in alpha-beta, use a certain heuristics that allows to expand search on the specific branch of the search tree. That is achieved using penalty-based iterative search, where penalties being derived from derivatives of node values. The derivative in essence expresses "the sensitivity of the root value to the changes in the tip value". The search is 'guided' towards a brunch that has minimal penalization value. For a detailed explanation seek that article.

Author tested this technique on Connect-Four game with 7 columns. The game was run 980 times. Half the games had a limit on time needed to calculate the next move. The time interval was ranging from 1 to 5 seconds (compare that to our 150 ms!). Another half of games had a limit on number of times a call "move" function (similar to our game.forecast_move) can be issued. The ranges were from 1000 to 5000 at 1000 intervals.

Results of the experiment were following: when allotted time for next move calculation was a limiting factor, iterative deepening AB algorithm won. This has to do with derivatives and penalties calculations in the proposed method and they deemed to be computationally intensive. Interestingly, author noted that "…the number of distinct position considered by alpha-beta was approximately three times larger than the number of distinct positions considered by min/max..". By min/max here is referred to the new method introduced in this article

However, when the limiting factor was a call to the move function, the proposed approximate min/max method won against ID alpha-beta. According to the author: "…our implementation of minimax search with alpha-beta pruning called the move approximately 3500 times per second, while our implementation of the min/max heuristics called the move operator approximately 800 times a second."