Game Tree Searching by Min/Max Approximation

Ronald L. Rivest MIT Laboratory for Computer Science Cambridge, Massachusetts 02139 USA March 29, 1995

The goal of this paper is to characterize an optimization technique for searching game trees using the Min/Max method by expanding the node that is expected to have the largest effect on the *value*. Exploring a game tree with Minimax and Alpha–beta pruning relies on the assumption that gameplay progresses optimally. The result is that with each new ply explored there is diminishing likelihood that the assumption of optimal play will lead to the highest *value* future game state. This paper addresses this issue of diminishing likelihood by first quantifying it and then maximizing it so that the searching of suboptimal paths is favored over increasing the search depth of the optimal path ad infinitum.

Generalized Mean Values:
The Min/Max function is approximated with the *generalized p-mean*

$$M_p(\vec{\boldsymbol{a}}) = \left(\frac{1}{n}\sum_{i=1}^{n} a_i^p\right)^{1/p}, \quad \frac{\partial M_p(\vec{\boldsymbol{a}})}{\partial a_i} = \frac{1}{n}\left(\frac{a_i}{M_p(\vec{\boldsymbol{a}})}\right)^{p-1}$$

where $\vec{\boldsymbol{a}}$ is a vector of values to evaluate Min/Max where:

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

The *generalized p-mean* is of interest here for use in "sensitivity analysis." The "sensitivity" comes from our choice of *p*. For large *p* the Min/Max function is sensitive to the extrema and for small *p* it is sensitive to the range of extrema and can be tuned anywhere in-between. The method of iterative search is then formalized by replacing Min/Max levels with evaluations of M_p

Penalty-based iterative search methods:
In general, with each successive ply of a game tree searched we must evaluate an exponentially larger number of game states with heuristic evaluation functions while simultaneously the individual game states we are evaluating tend to become less relevant to the current game state (root). In order to improve exploration of the game tree the authors introduce a method that discovers which leaf nodes of the explored game tree are most likely to affect game play at the root. For this method, we first back-propagate, using minimax, the heuristic evaluations of the leafs in our partially explored game tree using the *generalized p-mean* Min/Max. Then "we assign a nonnegative "penalty" (or "weight") to every edge in the game tree such that edges representing bad moves are penalized more than edges representing good moves." As an example, a "weight" may be calculated from a function of the squared difference in *generalized p-mean* value between two connected nodes or even from the gradient of the generalized p-mean. Thus, the moves that result in large changes to the value incur a large "penalty" and vice versa. The total "penalty" for exploring the game tree beyond any particular leaf is calculated as the sum of "penalties" between the leaf and the root. The game tree is then explored by expanding the leaf with the least "penalty" by one ply and then revaluating the game tree to find the next leaf with the least "penalty" in an iterative fashion. With proper tuning this method will explore the game tree by expanding the nodes that are expected to have the largest effect on the root value.

Experimental Results and Discussion:
To test this methodology the experimenters pit a version of the Minimax approximation with a penalty-based heuristic algorithm (MM) against Minimax search with alpha-beta pruning (AB) for the game of Connect-Four. Two experiments were conducted. The first limited the time (seconds) per turn and the second limited the number of calls (moves) to the evaluating functions during each turn.

A total of 490 games were played for each experiment. For time-limited search AB was superior with 239 wins and for call limited search MM won out with 249 wins. The differences are largely due to the number of computational steps involved in each algorithm. While MM does an excellent job of exploring the game tree it does so at a much slower rate than AB. However, there are numerous ways to approximate this methodology in order to balance compute time with accuracy and the computing functions themselves are tunable in such a way that the efficiency of game tree exploration can be tailored to specific game architectures.

TABLE 2. Experimental results

Resource bound per turn	MM wins	AB wins	Ties
1 second	41	46	11
2 second	40	42	16
3 seconds	36	44	18
4 seconds	39	52	7
5 seconds	30	55	13
Total	186	239	65
1000 moves	47	35	16
2000 moves	50	35	13
3000 moves	42	47	9
4000 moves	49	42	7
5000 moves	61	31	6
Total	249	190	51

Rivest, R. L. (1987). Game tree searching by min/max approximation.
Artificial Intelligence,*34*(1), 77-96. doi:10.1016/0004-3702(87)90004-x