Game Tree Searching by Min/Max Approximation – Research Review[*]

Konrad Burnik

In his paper from 1987 [1], Ronald L. Rivest proposed a method for evaluating game positions by min and max operator approximated with *generalized means*. In the first part, this method was introduced and in the last part Rivest explains the details of the method as well as presenting the results of the experiments that compared the method to Minimax with Alpha-Beta pruning.

The main method of approximating min and max operators was by using *generalized means*. Let $\mathbf{a}$ be a vector. Then the *generalized mean* of $\mathbf{a}$ is defined as

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

To see how M_p can be useful in approximating the min and max operators, here we list here some basic facts about M_p.

$$\lim_{p \to \infty} M_p(\mathbf{a}) = \max(a_1, \dots, a_n) \tag{1}$$

$$\lim_{p \to -\infty} M_p(\mathbf{a}) = \min(a_1, \dots, a_n) \tag{2}$$

Furthermore, we have for all $p \leq q$ that $M_p(\mathbf{a}) \leq M_q(\mathbf{a})$.

Hence, for large positive or negative values of p the value $M_p(\mathbf{a})$ is a good approximation to $\min_i a_i$ and $\max_i a_i$ respectively. However, as Rivest suggests, the main point of introducing $M_p(\mathbf{a})$ is that $M_p(\mathbf{a})$ has an additional benefit of having continuous derivatives in all a_i and as such some standard calculus techniques can be applied to perform a certain kind of sensitivity analysis. In particular, the partial derivative of $M_p(\mathbf{a})$ with respect to a_i is

$$\frac{\partial M_p(\mathbf{a})}{\partial a_i} = \frac{1}{n} \left(\frac{a_i}{M_p(\mathbf{a})} \right)^{p-1}. \tag{3}$$

[*]submitted as part of Udacity AI Engineer Nanodegree

The paper then proceeds to introduce a new (at that time) *iterative heuristic technique*. In general, an iterative heuristic technique expands the game tree one step at a time. At each step, a leaf node is chosen and then expanded.

In the following, with C we denote the set of all nodes of the game tree and with $T(C)$ the set of all terminal positions. With s we denote the root of C. The partially expanded game subtree is denoted by E. We denote by $score(c)$ the actual score of the game position and by $score_E(c)$ the approximated score relative to the currently explored subtree E of C. With Max we denote the maximizing layer of nodes in the game tree and with Min the minimizing layer. The process of partially exploring a game tree by an iterative heuristic can be then described as follows:

1. Initialize E to $\{s\}$, and $score_E(s)$ to $score(s)$.
2. While $E \neq C$, and while time permits, do:
   (a) Pick an expandable leaf c of E.
   (b) Add all children of c to E.
   (c) Update $score_E(c)$ at c and all the nodes on the path from c to the root s, using the following equation:

$$score_E(c) = \begin{cases} score(c), & \text{if } c \in T(E) \\ M_p(score_E(d_1), \dots, score_E(d_k)), & \text{if } c \in Max \setminus T(E) \\ M_{-p}(score_E(d_1), \dots, score_E(d_k)), & \text{if } c \in Min \setminus T(E) \end{cases}$$

where $\{d_1, \dots, d_k\}$ are children of c. Here the unspecified part is step 2(a): which node c of E should we pick to expanded with its children? The article proposes an answer to this question by introducing a *penalty score* for each edge of the game tree. The method should pick the expandable node that has the least penalty. The "Min/Max Approximation" heuristic that Rivest introduced is therefore a special case of the penalty-based search method, where the penalties are defined in terms of the derivatives of the approximating functions M_p as follows.

For a subtree rooted at node x and y being a node of that tree, let:

$$D(x, y) := \frac{\partial score(x)}{\partial score(y)}. \tag{4}$$

So $D(s, c)$ measures the sensitivity of the root value $score(s)$ to the changes in the leaf value $score(c)$. For calculating $D(s, x)$ we have the chain rule:

$$D(s, x) = \prod_{c \in Path(x,s)} D(parent(c), c) \tag{5}$$

Hence, to calculate $D(x, y)$ where x is a root of a subtree containing y, we use equations (3) for $D(parent(c), c)$ and the rule (5).

We can now describe how the penalty score is calculated according to Rivest. First, each edge $(c, parent(c))$ of the game tree then gets assigned a weight

$$w(c) = -\log(D(c, parent(c)))$$

The total penalty of an expandable tip x can now be defined as:

$$P_s(x) = \sum_{c \in Path(x,s)} w(c)$$

Thus, the leaf x with the largest value $D(s, x)$ is the one with the least total penalty $P_s(x)$.

In his article, Rivest also gives a discussion on penalty-based iterative heuristic methods in general, since his proposed method falls into that category. The main drawback of these methods, he argues, is that they need a lot of memory to work as well as most of the time these methods traverse paths from the root to the leafs, while other methods that are based on depth-first search only consideres an implicit game tree and spends most of its time on the leafs. Rivest further argues that the efficiency of alpha-beta pruning can occur as well with his proposed scheme, once a move becomes refuted then its weight will increase and the corresponding subtree of the game tree will not be considered in the future. However, this depends on the way *score* gives meaningful estimates of the game positions. As well, there was a concern on which value of p to use in calculations. Choosing large values for p corresponds to having a high degree of confidence in values of the *score* function, while choosing small values of p corresponds to low degree of confidence.

Additionally, since there is much calculation involved in calculating the penalty scores, Rivest proposes a simpler way to calculate these scores. Namely, to use the a "reverse approximation" of w derived from (3) as follows

$$w(c) = \log(n) + (p - 1) \cdot (\log(score(d)) - \log(score(c)))$$

where c has n siblings and where d is the sibling with the most favorable value $score(d)$ for the player to move from $parent(c)$. Here the term $\log(n)$ was actually replaced with a constant. Rivest also implemented his method in C alongside with the Minimax with Alpha-Beta pruning and evaluated it in a series of experiments where both methods competed against each other in a series of matches playing a game of CONNECT 4.

The short overview of the results of the experiments which he performed are as follows. When the time bound is taken into account then his proposed

"Min/Max Approximation" heuristic performs poorly than Minimax with Alpha-Beta pruning (Wins: 186, Loses: 239, Ties: 65). On the other hand, when restricted on the total number of calls to the move procedure then "Min/Max Approximation" performs much better (Wins: 249, Loses: 190, Ties: 51).

Rivest concludes his paper with a series of open questions about his introduced method and a conclusion that the results look "promising". However, since the method was new, he suspects that, if his approach goes through further optimizations and experimental validation then that could improve its competitiveness even more.

References

[1] Rivest, R. L. *Game Tree Searching by Min / Max Approximation*, Artificial Intelligence. 31-1, Dec. 1987.