Game Tree Searching by Min / Max Approximation

This paper presents a new way of evaluating a move in a MinMax game, using penalty based iterative heuristics.

In a regular minmax game, a static heuristic evaluation function is used to compute the value *v* for every node in the game tree. At the max (min) level, maximum(minimum) among the node's children values is used to select the next move.

In minmax. approximation, similar static heuristic evaluation function is used to compute heuristic at each node. But the maximum / minimum values of the children are computed using *generalized p-mean* approach, with *p* sufficiently negative for minimum operation and *p* sufficiently positive for maximum operation. Instead of taking max(min) of children to determine the next move, this algorithm selects the child whose heuristic fluctuations would have the most impact on the heuristic *v* of parent node. In other words, the algorithm computes the "derivative" of parent node *v* with respect to child node *v* for each child, and picks the child with the highest derivative to be the best move.

Rather than computing derivatives directly, a mathematical operation is performed to compute a new weight, *w* for each parent node - child node edge. This weight *w* is called *penalty*, because it shares an inverse relationship with the derivative. As a result, the goal at each level now becomes to determine the move that results in lowest penalty.

This penalty based scheme assigns a penalty value *P* to every edge (between 2 nodes) of the game tree as the tree starts to expand. But the penalty function is computed in such a way that the bad moves are penalized much more than the good moves. Penalty at an expandable node is zero, while the penalty of terminal node is infinity.

At each step, a tip node (or leaf) of the current tree is chosen, and the successors of that tip node are added to the tree. Then the values provided by the static heuristic function v at the new leaves are used to provide new backed-up values to the leaves' ancestors using min. at min levels and max. values at maximum levels.

The goal in this approach can be summarized as follows:

1) Start at root s.
2) Pick an expandable tip of the game tree (c) that has the lowest penalty
3) Expand the tree at this node c
4) Update the tree at node c and its father and ancestors all the way up to root S,
5) based on min-max rules.

Step 2 above is determined as follows:

At every node c, starting from root node, three values need to be stored.

1) Value v, computed by a heuristic function v.
2) Node to descend to (that would eventually lead to the expandable tip with least penalty)
3) The penalty value as a summation of penalty of all the edges from the expandable tip back up to to the current node.

All 3 values for each node, c, are computable from the corresponding values for c's children. The root node would begin with the following (v, root, 0).

The paper then applies this min/max approximation approach to *Connect-Four* game where it compares the results of this approach against minmax with alpha-beta pruning and iterative deepening. It concludes that alpha-beta is the superior one based on time usage alone. However, when comparison is based on move-based resource limits, the min/max approximation turns out to be the better choice.