Game Tree Searching by Min/Max Approximation

Goal:

The goal of this paper was to create an iterative method for searching Min/Max game trees, based on the idea of approximating Min and Max operators by generalized mean value operators. This method will attempt to always expand the node that is expected to have the largest effect on the current value (root of the tree being explored).

One of the key ideas in the paper is to use Generalized Mean Values (GMV) Mp(a), where p is a non-zero real number and a is a vector. The largest the value of p, the closer the GMV will be to the max function. The smallest the p value, the closer it will be to the min function. GMVs derivatives have a continuous behavior, which allows to trace which value of the array a has the most impact on Mp(a) (unlike the min and max function whose derivatives have a discontinuous behavior).

Given this, we can fix the p and n values, and use the derivatives of the generalized mean value functions and use the chain rule to see which value influences more the root of the tree we're analyzing.

The paper uses an iterative heuristic technique. This works just like the iterative deepening technique we talked in classes, but instead of searching each tree level equally, it chooses a node from the leaves (has to be a node which has successors in the complete game tree) to expand, adds the successors of that node to the tree, and evaluates them. The key question the paper tries to answer with its algorithm is: "which is the best node to choose for expansion?" This is where the GMVs derivatives come into play, in order to see which value has a bigger impact to the node which is the root of our tree. In this paper, this problem is implemented as a penalty based scheme, in which traversing down each node of the tree has a cost, and this cost is defined in terms of derivatives of the approximating functions.

Results:

The author's implementation of alpha/beta pruning with iterative deepening had superior results than the implementation of min/max approximation, when the limiting resource was computing time.
Nevertheless, the min/max approximation got better results when comparing the number of calls to the "move" operator.
In the end the conclusions taken are: The penalty based scheme only works well with large amounts of memory, it's oriented to find the better estimate of the evaluation function at the root, rather than picking the best move from the root (root is the node we are computing from). The author's also conclude that the penalty based scheme is less efficient than a depth search one, because it traverses the tree back and forth several times (while depth search spends most of the time near the leaves), and in particular, the scheme implemented evaluates all the successors of the expandable tip (while there are schemes that skip some evaluations). It is concluded that the approximating technique introduced could be implemented in an optimized scheme, in order to be more time efficient.