Review: Game Tree Searching by Min/Max Approximation

Summary

This paper suggests a method which will always expand the node that is expected to have the largest effect on the value.

Techniques in use

1. Approximating the 'min' and 'max' operators by generalized mean-valued operators. Unlike min or max, generalized mean-value has continuous derivatives withe respect to each node value.
2. Use iterative search, introduced in our class, via a heuristic functions. The performance of original minimax pruning search depends on the sequence of the nodes. The purpose of this paper is to provide a new way to answer which expandable tip node should be expand.
3. Penalty-based iterative search method assign a nonnegative penalty to every edge in the game tree. Such that edges representing bad moves are penalized more than edges representing good moves.
4. min/max approximate method is a specific instance of penalty-based search method, where the penalties are defined in terms of the derivatives of the approximating functions.

Implementation

In consideration of the computational difficulty of computing th generalized p-mean, this paperuse the appropriate min or max values instead, for the main point of using the generalized mean values was for their derivatives, not for the values themselves. This idea is calld "reverse approximation".

For parameter `p`, the paper point out, choosing a large value of `p` corresponds to having a high degree of confidence in the accuracy of the values returned by the static evaluator, while a small `p` corresponds to a low degree of confidence. As `p` get large, the heuristic should grow very deep but narrow trees. For small `p`, the heuristic shoud grow rather boad trees.

Key Results

Experimental results based on Connect-Four game and compare this method with alpha-beta pruning method. Based on time usage alone, alpha-beta seems tobe superior to min/max approximation approach. However, if base comparison on move-based resource limits, min/max approximation is definitely superior.

When special-purpose hardware is used, or when move operator is expensive to implement, the move-based comparison would be more relevant. For software implementations more development of the

min/max approach is needed to reduce the computational overhead per call to the move operator.