Research Review

Rivest, Ronald L – Game Tree Searching by Min/Max Approximation

In this paper, the author, Rivest, introduces a new technique for searching in game trees, using an approximation of the straightforward min and max operators. His new operator is more costly (i.e. computationally intensive), as it uses transcendental functions and potentially huge numbers as intermediate values, but it has the advantage of being differentiable.

Derivatives of the new function are used to assign weights, or penalties, to each edge in the game tree. This approach is therefore a *penalty-based* heuristic, wherein a game-playing agent will iteratively choose (until it has exhausted either the time or number of moves allocated to it) which one of the terminal nodes, or *tips*, of the game tree to expand, based on a penalty score. This differs from both iterative deepening, which expands *all* child nodes (alpha-beta pruning notwithstanding), level by level, and straightforward minimax, which considers only the results of the heuristic evaluation function, not results based on values computed from heuristic scores and values at parent or child nodes.

How does an agent efficiently choose which terminal node to evaluate? Thanks to the differentiable property of the new operator, it can identify the tip which has the greatest effect – the largest derivative – on the penalty value at the root node. The size of the derivative can also be thought of as the uncertainty in the penalty value at the root, and by expanding the tip with the largest derivative you can obtain more information from its children to help reduce this uncertainty.

The disadvantages of penalty-based schemes, however, is that you need to remember/store all previously computed penalties in memory – in essence, the full game tree being explored. The advantage is that after each penalty-based iteration, you now have a path, or *sequence* of moves, from your current game position (root node) down to the position with the best heuristic score. Contrast this with iterative deepening, where all nodes are re-computed at each iteration, and, after each iteration, you only have a single move – to the child with the best heuristic score. Also in contrast to alpha-beta pruning (which doesn't expand and evaluate children whose scores it knows will be redundant) penalty based schemes have to evaluate each child – but this is offset by the fact that they are more selective, or efficient, in choosing which children to evaluate.

A further feature of the new operator is that is parameterised; call this parameter p. P controls how closely the operator approximates the traditional min/max functions, and also the shape of the search tree. A large p results in trees which are deep and narrow, a small p in broader trees (with p=1, the search tree should match those produced by breadth-first search.) The parameter can be also varied as a game progresses, allowing an agent to focus or broaden its search depending on how near/far the game is from the end (and how many moves remain), for instance, or varied depending on the agent's confidence in the accuracy of the heuristic evaluation function.

When using minmax search with alpha-beta pruning, an agent using the new operator and the penalty-based approach can play better, when evaluating fewer positions, than an agent using the traditional min/max operators. If CPU time is the limiting resource, however, the new agent doesn't perform as well, largely due to the additional time required to calculate results using the more computationally intensive operator.