Ronald L. Rivest: Game tree searching by Min/Max Approximation

1. Introduction

The paper introduces an **iterative heuristic** for searching min/max game trees by replacing the min/max nodes with **generalised mean value nodes** which provide an **approximation of min/max nodes**.

Iterative heuristics algorithms **grow the game tree one node at a time**. This is different from iterative deepening which searches the tree uniformly as deep as allowed by some constraint (e.g. time). Instead, iterative heuristics pick a node which is then expanded [if possible] and its children become new tree leaves. Iterative heuristic **search trees don't necessarily have uniform depth**!

Min/max approximation algorithm guides the game tree search towards game paths which the **root node** is the **most "sensitive"** to. It uses generalised mean values nodes which are more suitable to analyse the root node sensitivity to different game paths than the min/max nodes since they are **continuous functions which also have continuous derivatives** with respect to all arguments - this is not the case with min/max functions. Derivatives of the generalised mean value functions play crucial role in selecting which tree node is expanded next during the search.

2. Min/Max Approximation

Min/Max Approximation search is a **"penalty" based iterative heuristic**. Penalties are non-negative **weights assigned to every edge** in the game tree so that the edges representing bad moves are penalised more than the edges representing good moves. The penalty of a tip (expandable) node is defined as a sum of the penalties of all the edges between the tip and the tree root. The algorithm **expands the tip node which has the smallest penalty** and then updates the score estimates of each node in the expanded subtree path as well as the edge penalty weights. When the algorithm terminates then it has traced a path from the root node down to the best expandable tip. The min/max approximation defines the **penalties in terms of the derivatives of the generalised mean value functions**.

3. Analysis

Penalty based algorithms put a lot of requirements on memory as they **require the tree to be explicitly stored,** they can **often** be **inefficient** as they spend a lot of time computing the estimates of the node trees and then traversing between the root and leaves. **Min/max approximation** algorithm allocates the resources in a sensible manner, **searching shallowly in unpromising parts** of the tree, and **deeper in promising sections**.

Generalised mean values impose a **high computational cost** during the search as it requires calculating powers and roots of picked dimension. The author chooses a method called "**reverse approximation**" which avoids computing the generalised mean values (for each tree node) and instead uses the min and max values directly. The whole point of using generalised mean values was their derivatives, not their values, so using min and max values won't introduce much error. It's not clear how big the **power of the generalised mean value** should be. Large values of power grow deep but narrow trees and correspond to high degree of confidence in the accuracy returned by evaluation function, small values grow broad trees and correspond to lower confidence in returned evaluation function values. Author also suggests using different values at different tree levels.

4. Results And Discussion

The new algorithm was tested on a game called **Connect-Four** and benchmarked against minimax with alpha-beta pruning. Each game playing strategy was allocated a **fixed amount of CPU time and number of calls to “move” subroutine**. It’s worth noting that penalty calculation uses an experimental constant whose value significantly influences the performance of the algorithm. Overall **490 games were played for each resource bound**; 980 altogether.

Based on CPU time usage alone, alpha-beta seems to provide superior results in comparison to the min/max approximation algorithm. However, **if the performance is measured based on move-based resource limits the min/max approximation algorithm is superior**.

Number of nodes considered by alpha-beta was approximately three times larger than the number of nodes considered by min/max when CPU time bound was in effect. When a move bound was in place the number of positions considered by each algorithm was roughly equal.

Minimax search with alpha-beta pruning called the move operator approximately over four times more than the min/max approximation algorithm. The author suggests this can be improved even more if using specialised hardware.

Overall the min/max approximation introduces **a novel approach to game tree searching**, which **outplays alpha-beta pruning minimax when it comes to number of move operators** used, however it carries larger CPU overhead.