Comp 424 - RAVE and Heuristic Based Tablut Player

Aidon Lebar - 260668812

April 8, 2018

1 Approach and Motivation

The agent mainly relies on a minimax search of all legal moves from the current board state for move selection. Using soft-fail alpha-beta pruning[1] to speed up the search, the agent will look to a depth of three to evaluate which move is the most advantageous. Due to the branching factor of the game and the overhead of cloning board states, the search could not look any deeper, as it took too long to evaluate such a large number of moves. To further ensure that the move would not exceed the time limit, at the start of each move the time would be logged and passed down recursively to the alpha-beta pruning. If the move took longer than a presepecified amount of time, the search would break and return the best move it had seen so far. Through experimentation, 1995 milliseconds was determined to be an appropriate time limit, as it was usually enough time to complete the minimax search, while giving enough time for the final operations after the search and a reasonable buffer.

Upon reaching the maximum depth, each move is evaluated using a weighted linear function instead of being expanded. As a design choice, positive scores correspond to the move favouring the Swedes and negative score favouring the Muscovites. This allowed the same evaluation function to be used for both sides, and gave a clear deliniation between a good move

[1] A modified and unified version of the pseudocode found in Russell and Norvig, Artificial Intelligence: A Modern Approach (2nd ed.)

and a bad one for the agent. This fuction is based on six heuristics for how advantageous or disadvantageous a move seems given the current board state. The simplest factor was whether or not the move ended the game. If it did, and the Muscovites won, the move would be weighed overpoweringly in the direction of the Muscovites, and correspondingly, would weigh it heavily in favour of the Swedes if it caused them to win. If a move clearly caused the agent to win, that move would be taken immediately, and if it clearly caused the agent to lose it would be discounted.

The other factors do not immediately discount or choose a move. They give smaller penalties and bonuses used for deciding whether or not a move is worth taking. The foundation of the evaluation function is a move value established through random simulation. Since it was too slow to run the simulations in game, two million random games were run in advance using a faster autoplay class that played two random players against each other. These games were then preprocessed, giving +1 to every move in a game where the Swedes won, -1 to every move where the Muscovites won and 0 to the moves in a drawn game. These totals were then divided by the number of games the move appeared in to get a standardized value for each move. Moves that appeared rarely (less than 25 times) were given the average value among all the moves, to avoid the misleading values caused by a small sample size. These values were the stored in a HashMap, which was serialized and can be read in from data during the first move of the game.

Several heuristics were then added to this base value to take the board configuaration into account when considering a move. Each of these factors was given a hand-selected value that was determined through expertimentation in games against random, greedy and fellow student players. Three general heuristics were added as well as two that specifically consider the king. The evaluation function gives a bonus for each of the opponents pieces that are taken and a penalty for each piece lost. This is to try to avoid board configurations where the agent is heavily outnumbered. The final general factor is a penalty against moves that have been taken before given identical board states. This is to avoid looping given repeated board states, as this was a problem encountered in test games. The two king-specific heuristics were its proximity to the closest corner and how many Muscovite pieces it is surrounded by. The king's closeness to the corner offers two advantages. The first being that it is close to the victory condition and the second being that since the edges have low manhattan distance from the corners the king will be more likely to move to them, which is a safe position as it reduces

the number of directions it can be captured from.

2 Theoretical Basis for Approach

The theoretical motivation behind the original approach (which will be discussed later in Other Approaches Tried section) was based in Rapid Action Value Estimation(RAVE) and adversarial search (pruned with alpha-beta pruning), and later updated to include heuristic search, all of which were intoduced in class. Many of the world's best game-playing AI agents use RAVE to quickly determine the value of an action by assuming that a move has the same value regardless of when it is played. This is a big assumption, but works well in practice given adequate simulations. The calculation of the the base move value was implemented with the same idea in mind. In typical RAVE algorithms, the calculations are done in game to speed up Monte Carlo game playing algorithms. Due the time restriction on moves, not enough simulations coud be generated in game for effective Monte Carlo or RAVE search. To solve this problem, the simulations were run beforehand and preprocessed to make evaluation faster in game.

Prior to the addition of heuristics, the evaluation did not consider the context in which the move was made, which required large assumptions about board state to be made, which is the major flaw in a RAVE-based approach. Heuristics helped to offset these assumptions by considering simple features of the board states. This allows for much more thorough evaluation of a move, with only minor slow downs from increased computation. The slow down can then be mitigated by the now more informed alpha-beta pruning, which will stop evaluating branches of the search tree if they look worse than alternatives that have already be considered.

3 Advantages and Disadvantages

The largest advantage of this approach was that it was fast. The relatively simple evaluation function did not take much time per move, but still gave a good estimation of the result. This meant that the pruning rarely hit the imposed buffer limit and therefore would usually consider all possible branches of the game tree. It would only get close to time out in certain,

partcularily open board configurations. It also performed well experimentally. The agent had a good win rate against both a random and greedy player.

The biggest disadvantage of this approach as that it could have done a more thorough evaluation. There are several heuristics to consider and their relative advantages and disadvantages are not always easy to discern. Given that the agent usually had time to spare during its moves, it could have instead used it for more complex heuristics on the board state, giving a potentially better estimation of move value. This meant that the agent sometimes made what seemed like a suboptimal move to a human observer, implying that there are factors of the game that are not being considered, or possibly not appropriately weighted in the agent.

4 Other Approaches Tried

The original encarnation of the agent only used the RAVE evaluation and alpha-beta pruning to decide on its move. The rational behind it was that it would allow fo significanty faster evaluation, which in turn would allow for deeper search. However, due to the branching factor of the game and the significant overhead of cloning the board state, it became apparent that move evaluation was not the botleneck time-wise. This meant that the agent had an evaluation function that made heavy use of RAVE assumptions, which do not necessarily hold, and it did not provide enough of a speed up for deeper search. At this point focus was shifted to heuristics, to give the agent more contextual information. This allowed for a more informed search in place of a deeper search.

5 Improvements

The best improvements that could be made to this algorithm would be more informed weights and better ordering of moves during the search. To choose better weights, a hill climbing, simulated annealing or possibly a genetic algorithm could be employed. Given appropriate step sizes and parameters, the weights should converge to an optimal configurations given the heuristics I consider. This could lead to a performance improvement

over the hand-selected weights, since they will be more fine tuned. The other major improvement that could be made is a better ordering of the moves for alpha-beta pruning. This could be achieved through an ordering heuristic or possibly an alpha-beta variant such as NegaScout or MTD(f) algorithms. Both would allow for better moves to hopefully be seen early, which would allow more branches to be pruned from the game tree. This could lead to a considerable speed up in searching through moves allowing for either deeper search or more thorough searching.