Comp 424 Final Project Report

Jonathan Pearce, 260672004

April 8, 2018

1 Introduction

Tablut is an ancient Scandinavian board game played on a 9x9 square board. Tablut is a two player game in which one side (The Swedes) tries to move their king from the center of the board to one of the four corners, while the opposing side (The Muscovites) attempts to prevent this and to capture the king. The game shares similarities with chess, in that all game pieces move equivalently to rooks (only horizontally and vertically), as well as checkers in that every piece follows the same movement rules. However, the game as a whole is very different from chess, checkers and other popular board games as Tablut is an asymmetric game, since each side wins the match in a different way. This asymmetry in the game means that both sides have different strategies during the entirety of the game. This distinction in strategy between the two sides makes it more complex to iteratively improve an AI agent and posed an interesting challenge in this project. The agent I developed to play Tablut evaluates board configurations using custom features deemed relevant to the game strategy, along with separate feature weights for each side we play as and makes use of a traditional search algorithm and pruning technique to select the optimal next move. Through testing against different opponents I was able to find suitable feature weights for my agent to use when playing as the Swedes, as well as a separate but equally effective set of weights for when we play as the Muscovites. Along with strong predetermined opening moves these distinct set of weights support the asymmetry of Tablut and make my agent more versatile. During final testing my agent was capable of beating the random player in every match and beat the greedy player 98% of the time.

2 The Agent: Motivation and Theory

To begin every match my agent has a planned opening move when playing for each side, pictured below.

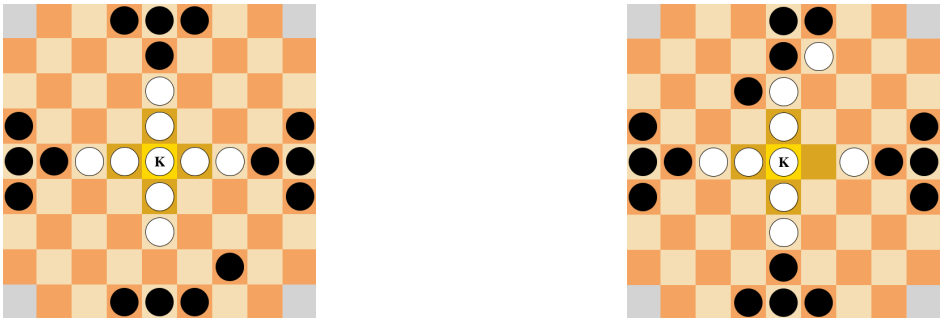

Figure 1: Agent's Muscovite opening (left) and Swede opening (right)

The Swede opening move is effective by setting a block at the top of the board and forcing black to pick whether to capture the piece at $(7, 5)$ or make an alternative move, in which case we have now provided the king an opportunity to move out from the center and will potentially be able to make a significant move up or down the board on the next move. This opening move is designed to try and get the king away from the center and close to a corner as quickly as possible. The Muscovite opening move is more conservative, it simply begins the formation of a blockade at the lower right hand corner of the board, forming these types of walls around the corner is one tactical pursuit of the Muscovite side. Both of these opening moves were inspired by a set of videos discussing Tablut strategy [1], with slightly different game rules.

After the opening moves, my agent switches its decision making process to a more extensive and practical method, in the form of an exhaustive search algorithm. The search method used by my agent is a Minimax search with Alpha-Beta Pruning [2]. Alpha-Beta Pruning, preserves the results of Minimax but reduces the number of game states explored and allows my agent time to explore deeper levels in the search tree. With over 250 students in the class, my agent will come up against a massive diversity of strategies and I felt that a complete search algorithm was the best option to make my agent competitive for every match, since it can look ahead in the match and try to figure out what the opponent is trying to do as well as plan out its own winning strategy. In testing the agent demonstrated that this search method could look ahead 3 moves in the game (depth = 3 in the search tree) while still adhering to the two second time limit safely.

When my agent's search reaches the desired depth in the game tree the agent uses an evaluation function to assign a score to the current game state, the higher the score the better it is for my agent, and vice versa with lower scores. The evaluation function is a summation of board features multiplied by the corresponding feature weight. My evaluation function makes use of four distinct features (six total):

1. Manhattan Distance of King to closest corner

2. Number of possible moves the King can make

3. Number of pieces on board (Calculated for both Swedes and Muscovites)

4. Number of rows and columns with a piece on it (Calculated for both Swedes and Muscovites)

Feature 1 (Manhattan distance) can be translated into the Swedes winning or losing the game, in order for a win, the Swedes must reduce this factor to 0. Similarly for the Muscovites, Feature 2 (king options) can translate directly to a win or lose, in order for the Muscovites to win they must reduce the number of move options for the King to zero. These two features demonstrate why Tablut is an asymettric game, each side has a different goal to reach and needs to focus on different aspects of the game in order win. Therefore it seems quite clear that you need two distinct sets of feature weights, one for when playing as the Swedes, and one for Muscovites. Feature 3 ensures that we try to keep our pieces safe, while capturing the other teams pieces. Feature 4 helps with board layout and developing a good structure early in the game, having your pieces spread out between all the rows and columns means that they have a better chance of accessing more of the board, which can aid greatly with capturing pieces. If we let f_i be the ith feature from the list above and $\omega = (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6)$ be the set of feature weights, then the exact formula used to calculate the score S of an arbitrary board B [3,4] is as follows:

$$S(B) = \omega_1 f_1 + \omega_2 f_2 + \omega_3 f_3^{Swedes} + \omega_4 f_3^{Muscovites} + \omega_5 f_4^{Swedes} + \omega_6 f_4^{Muscovites}$$

As mentioned above the asymmetry of Tablut required the development of two distinct sets of feature weights ω^{Swedes} and $\omega^{Muscovites}$. Because of my knowledge of Tablut and the strategy of both sides I felt that hand tuning the weights would be the most effective option. This was supported by the fact that previous successful board game playing agents have had their evaluation function weights tuned by hand, an example would be the world champion Checkers agent Chinook developed at the University of Alberta [5]. Further the final performance of my agent will be based on how it fairs in the class tournament, therefore in order to evaluate an agent's skill one has to be able to find a reasonable approximation to the class tournament, this is very difficult to do, in fact nearly impossible without developing multiple agents with different strategies and different features and even then it will do a poor job of representing the diverse population of strategies that will be on display in the class tournament. Due to this inability to properly evaluate an agents performance I decided to not pursue any optimization techniques to train the feature weights such as genetic algorithms and simulated annealing. Instead I iteratively tested my agent with the chosen weights against the random player, greedy player and the agent itself but with randomly selected weights (to make sure my weights were better than randomness). After every set of matches I inspected the game play, tried to find the imbalances in strategy that my agent was demonstrating and find a more well suited set of weights. After about 20 iterations of this process I was confident with my choice of weights and felt my agent was capable of solid and sound strategy when playing as the both the Swedes and Muscovites.

3 The Agent: Strengths and Weaknesses

After finishing hand tuning the feature weight parameters I completed a final validation of the weights by playing my agent against the same three opponents as during the tuning procedure. My agent played each opponent 200 times (100 for each side).

Agent vs. Random Player			
	Wins	Draws	Losses
Swedes	100	0	0
Muscovites	100	0	0

Agent vs. Greedy Player			
	Wins	Draws	Losses
Swedes	100	0	0
Muscovites	98	0	2

Agent vs. Agent with random weights			
	Wins	Draws	Losses
Swedes	76	12	12
Muscovites	73	7	20

These results show that my selected sets of weights work quite well. I was happy to have perfect results against the random player as that is the benchmark for grading my agent. Further I faired nearly perfectly against the greedy player. This result shows that smart heuristics do not fair well against game tree search algorithms. Any agent that I face that does not do a similar game tree search will be at a disadvantage because it will be very difficult to plan out sequences of moves, where as my search algorithm with a max depth of three makes it possible for me to constantly plan/anticipate my next two moves, and thus my agent's strategy is just slightly more complex and well thought out. In the tournament I expect to do quite well against other agent's that built upon the greedy player's framework. In matches against search algorithm based agents with only minimax implemented my agent should have an advantage as well since my testing showed that minimax alone can only reach a depth of two in the game search tree, which again favours my agent since it can plan further ahead. Agents with search algorithms that go further than a depth of three will leave me at a disadvantage assuming the heuristics they use to reach this greater depth are practical and do not actually reduce the skill of there agent. Finally, in matches where I go up against other agents with search algorithms that go to a depth of three the results of the matches will come down to how well developed the evaluation functions are both in terms of feature weighting and feature selection. From my testing I'm confident in my feature weighting and I think the features I choose to use in the evaluation function are all practical as they all relate to very relevant aspects of the game strategy.

Looking specifically at my agent's strategy, for both sides it is designed to be quite aggressive and tends to push for early wins. This works well for playing as the Swedes because if the king is too slow to escape the Muscovites may have time to surround him completely. Once near a corner though my player is not great at executing game finishing

moves because usually any type of game winning sequence takes more than 3 total moves in the game and therefore my agent is unable to see this possibility. Eventually my player is able to piece together a winning sequence of moves if it is there or it waits until the opponent makes a mistake and leaves an obvious opening for the win. Although closing out games is not perfect for the Swede side adding Alpha-Beta pruning and increasing the search depth from two to three had a huge impact on the performance. With a search depth of two, the agent could only plan one of its own moves at a time and therefore it had to be in line with the corner in order to finish a match which was very difficult to safely achieve unless the opponent made a mistake. Being able to plan three game moves ahead allows the agent to at least execute basic game finishing sequences. The agent's aggressive playing style is both good and bad for the Muscovite side, my agent focuses on minimizing the number of move options the king has, in an attempt to bring it to zero and win the match. However the downside to this is that occasionally the Muscovite side allows its pieces to be captured slightly too easily, the risk of my agent's Muscovite strategy is that if we play against a patient Swede agent that simply waits to go for the win and continues to move the king into open space, slowly my pieces will be eliminated and I fear towards the 100 move limit the king will have a much easier path to a corner with a severe reduction in Muscovite pieces on the board. The aggressive strategy is definitely less effective when playing as the Muscovites. The Muscovite side plays better when it is patient and fixes blockades in the corners or encloses the King, however I found it very difficult to manage these defences properly, because if you put too much emphasis on building these defensive structures the King may simply escape to whichever corner is least guarded and you will have no defence. The difficulty in having my agent build solid defensive structures lead to me sticking with the aggressive strategy for the Muscovites, I figure that with this strategy I will at least be able to pin any king that is not moved from the center immediately and I will still have a good opportunity to capture the king near the corners early in the match with my full set of Muscovites pieces.

4 Alternate Tested Techniques

As mentioned above making the shift from a search depth of two to a search depth of three made a huge impact on the skill of my agent as it was now able to plan two of its own moves as well as one opponent move. An obvious extension of this is that the greater depth you can reach in your search tree the more sophisticated your agent becomes. Therefore I considered applying more aggressive heuristics to my search algorithm to reduce the branching factor and permit my search algorithm to go to a greater depth in the allotted time. I tried two different heuristics. The first was to calculate the board score before and after any move at any level of the search tree and only add it to the tree if this move was beneficial to the team making the move. In other words this heuristic assumed that teams only made locally good moves, i.e. they would never sacrifice a piece or move there king away from the corner in pursuit of a more long term strategy. This heuristic proved to be very difficult to handle in the search algorithm because the branching factor varies a lot during the course of the match. The heuristic usually trims very few moves from the tree near the start of a match and at the end of the match the heuristic would remove most moves from the search tree.

Because of this variation with the branching factor I did not consider this heuristic very seriously because it was very tough to find a consistent way to prevent it from timing out while still reaching a greater depth. A similar heuristic i considered was also calculating the local change in board score given a move and then only adding the x best local moves to the search tree, were x was a constant. This strategy solved the variation in branching factor size that the first heuristic presented as I was able to choose and fix the branching factor x and I would be able to adjust x whenever I wanted raise or lower my desired depth. I decided to not go with this heuristic because to make it worthwhile I needed the search to reach a depth of five (three of my moves in the future), but in this case I would need to trim most moves at each level at which point I felt I was missing far too much of the match and effective strategies would be missed by my search algorithm.

Further I experimented with a few additional features for the evaluation function. The first was a feature that calculated the number of Muscovite pieces directly surrounding the king, this was to indicate to the Swedes when the king was under attack, and to help the Muscovites trap the king. After some analysis I found that this feature was highly correlated with the 'king options' feature while providing slightly less information and thus I decided to exclude from my final evaluation function. Another feature I considered was one to help with the spacing of my agent's pieces, this feature calculated how much of the board each side could cover in one move, in other words it checked how many of the open board squares your pieces could reach in the next move. Later in the development process I found that the row and column count that ended up in my evaluation function was much better at developing effective board structures for my agent's piece's and thus I dropped this feature from the evaluation function. The last feature I excluded from the evaluation function calculated how clear each corner was of a blockade, it looked at the 4×4 chunk of the board in every corner and checked every possible path from the corner to the edge of the subsection of the board to calculate how easy it would be for the king to navigate into a particular corner. This feature proved to take too long to compute since it needed to check nearly 20 paths per corner per board.

5 Future Improvements

Through testing I found that my agent plays perfectly against the random player and near perfect against the greedy player. Further, I feel that it will be competitive in the class tournament, but of course there are lots of places to improve the agent's abilities and tactics. The easiest way to improve the agent would be to find more effective features to use in the evaluation function, these features would need to make the agent's strategy more robust while being quick to compute across any board configuration. However, the areas that I would like to improve in the future would be creating a more extensive opening game library and working on more sophisticated strategies for closing out matches. The opening move library is critical for a game such as Tablut because of how quickly a game can end, one bad move with the king or one Muscovite out of place and the match may be over the next turn. An extensive opening move library would potentially have the optimal move for both sides of the game up to move number 7 or 8 in the match, but anything above 2 total moves into the match would be an improvement for my agent. The opening sequence of moves is

so critical for both sides that this fixed library would prevent any foolish moves that could ruin our strategy down the road. In the beginning the Muscovites have to work to set-up blockades in front of the corners in attempt to close off parts of the board from the King, similarly to avoid the King getting trapped by the Muscovites it is essential for the Swedes to move the King out of the center towards an unguarded corner very early in the match. The single opening moves I have planned for my agent support these early game strategies, but it would be very nice to run extensive game simulations with different permutations of moves two,three and four for each side to see what is effective and should be added to our opening move library. On the other end of the matches I think a separate end game strategy would be a massive benefit. The most common and effective way for the Swedes to win a match is with 3 moves, an example below.

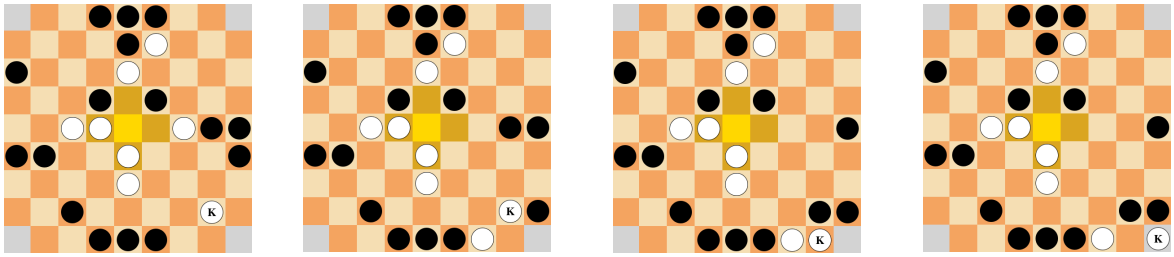

Figure 2: Common match finishing sequence of moves for Swede side

My search algorithm can only plan two of its own moves ahead and so executing this type of closing move is nearly impossible without a bit of luck with the board formation. Finding a heuristic to determine when to switch from the exhaustive search (depth = 3) to a more focused end game search (depth $\geq$ 5) would improve my Agent's win rate when playing as the Swedes. This end game search would have to focus on king moves and moves of pieces near the king to set the initial block, as discussed before a full branching factor would be too large if we needed to reach a depth of five in the game search tree. My agent's ability to close out matches with the Muscovites is fairly effective thanks to the 'King Options' feature, however a more advanced finishing strategy would be to slowly form a closed shape around the king and then compress that shape until the King is captured. This technique is very ambitious and would be fairly tough to execute. Finally as a stretch goal that goes beyond the scope of this course slightly, it would be interesting to experiment with a neural network where each square, row and column on the board form the input to the network [6]. The hidden layers would be trained through constant game play and the network would return the optimal move given the features. It would be very interesting to see how many input nodes and how many hidden layers the network could support while still guaranteeing a result in the two second move limit. Recent advancements in this area of research as well as applications that make use of the technology such a DeepMind's AlphaGo and AlphaGo Zero have shown how powerful neural networks can be at teaching an agent how to play complex board games very well [7].

6 Conclusion

I am pleased with how my agent developed over the course of the project. I think the agent has strong opening moves for both sides that will get the match started in our favour. The full game tree search will be very important and will help the agent deal with the 250+ different strategies it will face in the class tournament, and with the well tuned evaluation function my agent should be competitive in most matches. I look forward to hearing the results of the tournament. This project has given me a great appreciation for the amount of work that must have gone into developing successful AI agents in the past such as Chinook, Deep Blue, AlphaGo and many others and I look forward to having some time in the near future to try my hand at another AI agent for a different game or to further develop my agent for Tablut.

7 Sources

1. https://www.youtube.com/watch?v=d9kc2hehj48
2. https://www.cs.cornell.edu/courses/cs312/2002sp/lectures/rec21.htm
3. http://ai.cs.unibas.ch/˙files/teaching/fs16/ai/slides/ai42-handout4.pdf
4. http://www.cs.unca.edu/ bruce/Spring98/csci373/ch5/game-lecture.html
5. https://www.aaai.org/ojs/index.php/aimagazine/article/viewFile/1208/1109
6. https://pdfs.semanticscholar.org/ba25/8085a8ff64e296bbd176cec42bf710784d7c.pdf
7. https://arxiv.org/pdf/1712.01815.pdf