Assignment Part B

Rohan Hitchcock and Patrick Randell

Overview of our algorithm.

1 Searching and pruning

- Used minimax with alpha-beta.
- Additional pruning heuristics including pruning states we had already encountered before (helps avoid states which our evaluation function overestimates how good they are), and (others considered / used)
- Move evaluation: static, [explain what it does]. Found it provided moderate speed improvements, but leaves a lot to be desired. Believe that a solution which dramatically improves upon this would need to be a lot more sophisticated, but would potentially have big gains.
- Investigated principle variation search but found it did not offer significant improvements over alpha-beta because our move evaluation order was not good enough.
- Discuss approaches which vary depth to allow for some critical states to be searched very deep, such as doing this at the start. Why we didn't use it: when combined with our opening book, we believe the early states are strategically very important. Strong play early can make or break games. Don't want to use heaps of time at the start however in case this doesn't work. Late game is simple, its either lost or just a matter of tracking down opponents pieces. We continue to use our time budget for as long as possible but will revert to a shallow search if we run out of time.

2 The evaluation function

2.1 Feature selection

- Focused on 3 player behaviours: taking tokens, stacking, controlling space on the board. Taking tokens is obvious: we need to do this to win, stacking improves mobility which is important for both offence and defence. Controlling space allows for more strategic play.
- Selected signed squared token difference to encourage taking tokens, as well as the number of opponent stacks (to encourage trading 1-1 if necessary).

- Experimented with many features to encourage stacking, including having the number of each stack as a feature. Eventually found that the number of our player's stacks was the feature most amenable to training.
- Used distance between our player and opponent's center of mass to cover space. This is a good trade off between something which is easy to calculate and something which captures a lot of information. Talk about how sum of distance from each stack to nearest opponent token would be better but completely intractable. CoM also encourages aggressive play in the late game when the opponent only has a few tokens remaining.

2.2 Training the evaluation function

- Trained using TD-Leaf(lambda) with lambda = [insert lambda value]. Learning rate was gradually decreased (how?) as performance improved.
- Ideally we would be able to play against a series of varied opponents increasing in difficulty (see paper). We aimed for as much variation as possible in training. This included play against a player making random moves, play against other players with different features, self-play, and play against other players on the battlefield (ideally we would have done this a lot more but battlefield games were at times hard to come by). This was done to avoid overfitting any particular players strategy. We observed that against players with simple features it could learn behaviour that it could learn to manipulate that opponents evalution function and to manufacture a draw due to the 4-repeated-state gamerule.
- Initially used the same weights for white and black. This resulted in a player which was reasonably good as white but worse as black (long games with lots of draws). Using seperate weights when playing as white and black resulted in performance improvements when playing as both colours.
- Initial manual tuning of weights was important. For example when training a player with initally random weights, it found a local maximum where games were very long and it drew frequently (can be thought of as a highly defensive player). TD-leaf doesn't favour winning faster, but this is important for our resource constraints.

3 Other aspects

3.1 Opening book

We included a sequence of (how many) fixed opening moves. These moves were found by playing games in real life, and chosen so they are at least not possibly detrimental. Initially this was done to help manage our resource budget. With a relatively shallow minimax search, the early game is hardest to navigate since the tokens are so far apart and all stacks are size 1 (the opening moves were some of the longest to evaluate). It also served to make the transition from early-to-mid game quicker (where our evaluation function performs better, good moves are at a depth our minimax can reach, our move generation order is better optimised so search speed improves). We found this had a big impact on our performance too. We believe this was because we entered the mid-game on our terms, rather than because our opponent had developed their position and advanced.