ReconChess Project Report

Tony Lin, Haoliang Jiang, Chen Liu
{tlin339, hjiang321, cliu324}@gatech.edu

1 Project Summary

Our approach was based on using the AlphaZero search algorithm with an evaluation function based on a weighted sum of a trained value network and the Stockfish library's evaluation engine (as necessary for the requirements of CS 7649) and an exploration strategy based on a combined trained policy network and the stockfish evaluations. To handle the state uncertainty of the board state, we tried two different representation strategies. First, we chose our state uncertainty representation as a belief tensor that contained the probabilities that a given square was one of the six class types for each side or an empty square (13 classes in total). From there, we could sample valid boards using the belief tensor and use the sampled boards as possible roots for us to perform the AlphaZero search. Then our selected action can be chosen as the action with the maximum expectation over all sampled boards. While the strategy for choosing the action stayed the same, our belief representation ended up changing due to issues we faced during integration. Initially, we chose the belief tensor strategy because this would allow us to retain the exact uncertainty distribution over possible boards instead of an approximation via particle filtering. Since our tensor would then be used to sample possible boards, the tensor would be very memory and computationally efficient since we wouldn't have to iterate over every possible board trace and every possible action (as in the particle filtering approach).

However, we discovered significant issues related to sampling valid boards from the belief tensor. For example, many sampled boards would be invalid or be strangely organized. Some of these invalid board samples could be rectified through careful sampling workarounds but others were unclear. In addition, as the number of opposing pieces was uncertain, we would often sample very strange boards that were dissimilar to the true board state. As a result, we switched to a particle filter approach to represent the belief uncertainty. Starting from a single particle with the initial board state, we iterate over all possible opponent move results (including a failed move) to generate all possible children particles. We can then weigh all candidate particles using scanning measurements or our own applied moves (e.g., rook traversal ensures no pieces can be along the rook trajectory). For efficiency, we only retain the 200 particles with the highest likelihood weights. To select our sensing location, we use a maximum entropy approach which can be computed from the current set of particles by iterating over each square and counting the occurrences of each piece type. This method allows us to compute an optimal sensing location with respect to an information gain metric without going through a costly training process (i.e., training another reinforcement learning agent to select the sensing location). Since the particle filter approach is an approximation of the distribution over the possible board states, we occasionally run into situations where none of our particles match the measurements collected, in which case we are forced to play randomly for the remainder of the game.

Using the stored particles, we then run the AlphaZero Monte Carlo Tree Search forward on the top five boards and find the action with the maximum UCT over each sampled board where our exploration strategy is governed by a policy network. Our AlphaZero search strategy incorporates an evaluation of the board state using a weighted sum of a trained value network of our design and the stockfish library's evaluation function, i.e. $V(s) = \alpha V_{nn}(s) + (1 - \alpha)V_{stockfish}(s)$ with $\alpha \in [0, 1]$. In our implementation, we use a value of $\alpha = 0.5$. Our value network is trained according to the AlphaZero supervised learning setup and our policy network is trained using the reinforcement learning update also used in the AlphaZero algorithm.

2 Learned Experiences

This project provided us with useful hands-on experience in understanding some of the major topics covered in class and why they are difficult to solve. In particular, this project helped us to improve our understanding of **i)** partially observable markov decision processes (POMDPs) and how we can solve planning problems in these domains, **ii)** implementing network-driven MCTS to learn better strategies over time, and **iii)** how we can convert POMDP problems into complete information domains so that we can apply MCTS. Our group first spent a significant amount of time and effort discussing how we could describe the uncertain state of the board. We first decided on representing the belief state as a $13 \times 8 \times 8$ tensor where each channel represented a particular class (6 unique black piece types, 6 unique white piece types, and an empty class) and planning could be done by sampling possible boards. By designing a transition matrix with the assumption that any valid move was equally possible, we could perform a transition update (as in a Markov chain) and perform measurement updates to this belief state whenever either a scanning result or a step action is taken. This strategy would notably provide us with a large memory improvement over particle filter based approaches as the "particles" could be extracted at each iteration. Unfortunately, we ran into major issues related to sampling valid boards and having the samplings provide reasonable board representations. One board restriction in particular we had difficulty with handling was the `BAD_CASTLING_RIGHTS` error. We could not identify a clean method to ensure our sampling strategy would satisfy this constraint and any invalid board would result in an invalid stockfish evaluation, making our approach lead to bad decisions. In the end, our strategy resulted in many odd workarounds that didn't result in a significant win-rate ($\approx 60\%$).

As a result, we re-wrote our belief representation to be through the particle filter approach instead. In using the particle filter approach, we found that, despite requiring more memory and computation to both propagate and store the particles, this approach was significantly more appropriate. Since our propagation retained possible board states based on previous valid board states, we were guaranteed to satisfy all board constraints. In addition, since subsequent boards could be eliminated, we could also easily retain an estimate of which opponent pieces were captured. Our initial approach of storing the exact distribution would be more appropriate for situations in which the number of agents/pieces in the game is fixed/exactly known, for example in a predator-prey style game in which the game ends when the prey is captured/removed. However, since the number of agents is also uncertain, it was difficult to sample boards that accurately represented the board state.

This allowed our agent to achieve significantly improved performance as our most likely belief state usually captured the true board. Our estimated win-rate performance with the particle filter belief was $\approx 100\%$. While it was unfortunate that the belief state representation did not achieve favorable results, we were able to understand significantly more about different representations of the state uncertainty and when they would be appropriate.

3 Individual Contributions

3.1 Tony Lin

Tony was responsible for code related to the belief propagation of the board state. He wrote helper functions to construct the Markov transition matrix based on all valid moves and the current state of our pieces. Under the assumption that any move is equally possible, the board belief state could be propagated by treating the board as a $12 \times 8 \times 8$ tensor that can be flattened to be a state with 768 elements. In addition, Tony wrote code to update the belief state based on either scanning measurements or valid taken moves. The updated belief state could then determine the next sampling location as the maximum entropy location over all possible squares. Tony also wrote the code to sample valid chess boards based on the belief state tensor which was used to perform the MCTS. Ultimately, our group used the particle filtering approach to representing the belief uncertainty over the board states.

3.2 Haoliang Jiang

Haoliang Jiang was responsible for the MCTS implementation and related integration. He also worked on testing the project with the team and improving the feasibility and efficiency of the implementation. He wrote a MCTS using a criteria based on iteratively updated Q value and predictions from a neural network. Besides, he dealt with the conversion between the FEN representation of the board, the belief tensor and the actual board presentation in the MCTS. Furthermore, he wrote APIs for selecting valid boards from samples, integrating the particle filters with MCTS and passing the NN prediction into the MCTS. Finally, Haoliang Jiang worked on testing the MCTS framework by real games, balancing the computation time for making decisions and the efficiency and feasibility of decisions, and handling the invalid belief sampling results with the team.

3.3 Chen Liu

Chen was responsible for implementing the policy network and the particle filter-based agent. He wrote a simple neural network model in PyTorch with two 2D convolution layers, a policy prediction head and a value prediction head, a dataset class that encapsulates game play simulation and a replay buffer, code for converting `chess.Board` and `chess.Move` objects to and from tensors, as well as code for the training pipeline. He was also responsible for training the network and tuning parameters to optimize for the loss function. In addition, Chen implemented a particle filter class that keeps track of possible board states as particles with associated weights, and updates particles by simulating moves by us and the opponent, as well as removing particles incompatible with sense results and move results. Chen was also responsible for creating a particle filter agent, integrating it with MCTS and tuning the max capacity parameter of particle filter to balance between accuracy and speed.