Chess AI Implementation Project Proposal

CS 5100 Foundation of Artificial Intelligence

Dr. Robert Platt

Chenrong Su

Alex Naishuler

Problem Description:

In the field of game AI, chess has been a classic topic of implementation, various implementation strategy has been applied to such field, and the result of such implementation, such as the creation of AlphaZero and DeepBlue, along with their accomplishment, has proved the compatibility and established the potential of AI in the game of chess and established a foundation for the promising potential of game AI. If we view such a procedure from the foundation, which can be down to the root of chess AI, the implementation for such ideology is based on decision tree processing, a structure such as the implementation of Monte Carlo Tree Search, min-max search, alpha-beta pruning, iterative deepening, quiescence search with different heuristic and evaluation function that determines the potential of each state of the board. However, one issue always comes with such implementation is that the memory required for the tree to expand into the desired state is exponentially large. If the tree expands ultimately, it will contain approximately 10^120 different states; therefore, most, if not all, computational devices does not consist of enough space to store such information; therefore, for most chess game that is out on the internet, most of them utilize a limited depth function in their decision-making algorithm in some way or the other; however, in order to implement an efficient algorithm for chess while maintaining a cost-effect procedure, the understanding of the most efficient depth for each algorithm under a specific evaluation function and the cost, which is the execution the expansion that corresponds to under such properties is crucial to the implementation, and the analysis of such ideology is the main problem that this project is targeting to analysis and resolve. The input of such a problem will be agents that utilize different algorithms under specified evaluation functions and depth. The outcome of such input will be the number of nodes they explored under such conditions and the win rate of such algorithms under such conditions. The project does not require the input of outside data since the comparison data is collected when the agents are either playing against a human player or another agent. However, if necessary, for the reinforcement learning part we will utilize the data from Kaggle's article <Reinforcement Learning Chess> as a base template to train the reinforcement agent. The number of expanded states and the win rate will be stored and written under a file corresponding to their current condition. Such an issue is interesting since chess is a fundamental and classic problem in the field of game AI, and the outcome of a decision made by a game AI can change dramatically under a specific condition; therefore, it is fascinating to discover the option that can drive their maximum potential with allowed hardware requirements

Approach/Algorithms:

To resolve such issue, the algorithms utilized within the project are pure min-max search, min-max search with the utilization of alpha-beta pruning, min-max search with the utilization of quiescence

search, iterative deepening, best-first search, expectimax-search, A* search, and reinforcement learning. Furthermore, each search will be tested and analyzed under different conditions, such as the different heuristic/evaluation functions and depth. These algorithms are appropriate for such problems in that they all form a sort of pathfinding or decision-making procedure within them; most importantly, they can all be utilized in a multi-agent environment and adapt to a changing environment such as the chess board. Under a general situation, these algorithms are being utilized in various situations. The algorithm such as the min-max is being adopted in zero-sum games to calculate the best procedure when the opponent is playing under the most optimal strategy; in other words, it is a decision rule utilized in various fields such as the decision theory and game theory to minimize the lost in a worst-case scenario. The implementation such as the alpha-beta, expectimax, iterative deepening and quiescence search are functional utilities that can be built upon the min-max search to optimize the function and reduce unnecessary expansion. Furthermore, structures such as A* and best-first search are utilized when locating the shortest path from a given starting node to a goal in a graph. Moreover, on the subject of reinforcement learning, the agent will be able to capture the current stand of the board by different evaluation function, and through repeatedly completion of the game, it will be able to capture the optimal strategy to the end game base on the current circumstance of the board. Besides the utilization of algorithms, we will also utilize different depth and evaluation functions to evaluate the outcome of specific algorithms. The evaluation functions that will be utilized within this project are the number of enemy chess available, the sum of moves that each of the friendly chess is required to take to clear out the board, and the sum of moves that each of the friendly chess is required to take to checkmate the enemy, and checkmate status. Moreover, such results will be measured under the depth of 1-5. To reach the result, the project will first require an implementation of the chess board, which includes the board and the different chess and their action logic, along with the winning and losing conditions. After such, the project will require the implementation of different agents, with each agent having a similar function but different in the execution of moves; for instance, the min-max agent will have its strategy for determining which move to make, and best-first search agent will have another set of structure to follow when trying to determine their actions. In the past, there have been various implementations of chess algorithms, such as Deep Blue, and the analysis of its evaluation function and depth limiting strategy has successfully improved its action and structure.

Planned Comparison:

During the project, an agent under a specific condition will be faced with another agent with specific conditions or the same agent under different conditions, or even a real player, and the data of the number of the node that they have expanded during the process, and the total win rate of the agent will all be documented and written to a file, after the gathering process, the data will be formatted into a graphic form for analysis; therefore we can determine the efficiency of each agent under different circumstances, and comes up with a conclusion of which agent is best under what conditions, or which form of condition is the best fit for a specific agent in the field of chess.

Dividing of Work:

During the process of the project, the work will be split into half and half portion for each member of the group, when implementing the procedure for adversarial search, Chen will be responsible for the implementation of alpha-beta pruning, min-max search with the utilization of quiescence search and iterative deepening. On the other hand, Alex will be responsible for the implementation of best-first

search, expectimax-search, and A* search. Furthermore, for each of the search functions, everyone will be responsible for gathering data for their topic on different depth and evaluation function. Moreover, the implementation of reinforcement learning will also be a joint force of work, each one of the groups will be responsible for researching different part of reinforcement learning, such as policy iteration and Q-learning strategy that fits our project's topic, and in the end came up with a proper implementation.

Reference:

Campbell, M. S., & Hoane, J. A. (1999). *Search control methods in deep blue - association for the advancement ...* Search Control Methods in Deep Blue. Retrieved October 1, 2022, from https://www.aaai.org/Papers/Symposia/Spring/1999/SS-99-07/SS99-07-004.pdf

Cornell University. (2004). *AI Chess Algorithms*. Chess. Retrieved October 1, 2022, from https://www.cs.cornell.edu/boom/2004sp/ProjectArch/Chess/algorithms.html

freeCodeCamp.org. (2017, March 15). *A step-by-step guide to building a simple chess AI*. freeCodeCamp.org. Retrieved October 1, 2022, from https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/

Kaya, E. (2020, December 18). *Ai chess algorithms and chessus*. Academia.edu. Retrieved October 1, 2022, from https://www.academia.edu/44730471/AI_Chess_Algorithms_And_Chessus

Kumar, K. (2020, March 22). *Reinforcement learning chess 3: Q-Networks*. Kaggle. Retrieved October 13, 2022, from https://www.kaggle.com/code/jagannathrk/reinforcement-learning-chess-3-q-networks/notebook

Reinforcement learning. Reinforcement Learning - Chessprogramming wiki. (n.d.). Retrieved October 13, 2022, from https://www.chessprogramming.org/Reinforcement_Learning

University of Alaska Anchorage. (n.d.). *Problem and search spaces - University of Alaska System*. Retrieved October 1, 2022, from http://www.math.uaa.alaska.edu/~afkjm/cs405/handouts/search.pdf

Walker, L. (2020, August 21). *The anatomy of a chess ai*. Medium. Retrieved October 1, 2022, from https://medium.com/@SereneBiologist/the-anatomy-of-a-chess-ai-2087d0d565