The Role of AI in Board Games

[Review Paper]

Microprocessors and Interfacing (CSPE21)

[Divyansh Thakur]

Submitted By :-

Surendra Kumar

IIITU18123

Introduction

Board games have long been seen as an ideal test-bed for the study of AI. Many board games can be seen as simplified models which occur in real-life. Up to this point, a large portion of the scholastic work in the territory zeroed in on customary board games and card games, the test being to beat master human players. The future will see games with astute sympathetic characters who become more acquainted with you, and make a solid effort to streamline your experience while playing the game. Better game AI will lead than additionally engaging and vivid games and furthermore increase the value of the prospering genuine games market.

Traditional games are obliged by the capacity of the players to control the game items, for example, pieces on a chess-board or cards from a deck. The rules of a game determine how the game articles collaborate and essential combinatorics rapidly produce enormous game trees. Implementation of computer programs which are able to defeat almost any human opponent has been a hot topic for AI research. These programs expend most of their effort on game-tree search which has limited applicability when the game has a large branching factor or the game actions are continuous. Reinforcement learning with a general-purpose searching method, Monte Carlo tree search (MCTS) is used to learn a variety of games from scratch. To understand the difference between reinforcement learning and supervised learning, we can say, In supervised learning an agent is taught how to respond to a given situation, and on the other hand, in reinforcement learning the agent is not taught how to behave rather it has multiple options in how to behave.

Game conditions rearrange numerous parts of certifiable issues yet hold adequate unpredictability to challenge people and machines alike. Most projects for playing exemplary prepackaged games have been to a great extent human-designed. Modern pursuit techniques, complex assessment capacities, and an assortment of gamespecific stunts have permitted projects to outperform the best human players. More recently, a learning approach accomplished superhuman execution in the hardest of the exemplary games GO, yet was explicit for this game. But some new research removed the need for human knowledge, and algorithmic optimizations delivered further performance improvements. AlphaZero AI system can play three games (chess, go, shogi) at the highest levels.

Chess, Go, and Shogi are highly complex but have a number of characteristics that make them easier for AI models.The game actions and tricks are fully observable. Games with less observability like poker, can be considerably more challenging for AI.

AlphaZero AI approach still has limitations that could be addressed, for example, large computational requirements, brittleness, and lack of interpretability.

Initial Stage of AI and Games

At the start of the use of AI in board games, researchers looked for the evaluation function f(P) which can be applied to the current position P of the game. And then, the value of f(P) determines the category for the position P, that category can be won, lost or drawn. If the evaluation function f(P) found easily then, it is easy to design a machine capable of perfect play. In this case, the machine would never lose or draw to the opponent. Suppose:

f(P) = +1 for a won position
f(P) = 0 for a drawn position
f(P) = -1 for a lost position

Machine will calculate the f(P) for the various positions which can be obtained from the current position. After that, it will choose a move with the highest f(P). This strategy was applied to a game called NIM. And the machine never lost any match.

Another method was to have a dictionary of all the possible moves of game. For each and every position, there is an action mapped to that position stored in the dictionary. For the game of chess, there are roughly 10^43 positions. So, naturally it is impossible to store this much data.

So, we would like to play a skillful game , rather than a perfect play strategy using evaluation function or legal play strategy using the dictionary. Claude Shannon proposed a mixed strategy. According to the mixed strategy, number the legal moves using evaluation function and then, choose a random move from the list.
But, in the case of chess, there is no known simple and exact evaluation function f(P). Basically the evaluation function is very complex because it is based on the position, the number and kind of black and white pieces, pawn formation and mobility. Thats there is need of mixed strategy in the game of chess.

AI and Design of Balanced Board Games

Game designing is a challenging problem for the game designers. So, Here, AI plays a significant role. For this specific task, there are a lot of general game-playing (GGP) engines which play according to the rules. And these rules are specified through the general game-definition language. We can create the balanced games by changing the rule parameters and by changing the rule itself. This strategy is known as the automatic game design (AGD).
ZOG is a commercial GGP which uses ZRF as its scripting language. Here is an example of a ZRF for tic-tac-toe:

```
(players White Black)
        (turn-order White Black)
        (board
                (image "images\TicTacToe\TTTbrd.bmp")
                (grid
                        (start-rectangle 16 16 112 112) ; top-left position
                                (dimensions ;3x3
                                        ("a/b/c" (112 0)) ; rows
                                         ("3/2/1" (0 112))) ; columns
                                                (directions
                                                        (n 0 -1) (e 1 0) (s 0 1) (w -1 0)
                                                         (ne 1 -1) (nw -1 -1) (se 1 1) (sw -1 1))))
                (board-setup
                        (White (disk off 10))
                        (Black (disk off 10)))
                (piece
                        (name disk)
                        (image White "images\Reversi\WDisk.bmp"
                                Black "images\Reversi\BDisk.bmp")
                        (drops (add-to-empty)))
                (define add-to-empty ((verify empty?) add) )
                (draw-condition (White Black) stalemated)
                (win-condition (White Black)
                        (or     (relative-config disk n disk n disk)
                                (relative-config disk e disk e disk)
                                (relative-config disk ne disk ne disk)
                                (relative-config disk nw disk nw disk)))
```

Vincent Haum and Joe Marks proposed a AGD to search for balanced games in the space of board games that can be described by combining independent elements for board, piece, and victory specifications that are derived from known games and variations thereof.

The search algorithm combines various components to frame new games that are then tried in self-play utilizing the ZOG game engine. As an estimate to what in particular comprises challenge and interest, we look for games in which every player has generally equivalent possibilities and that infrequently end in draws.

Monte Carlo Tree Search

Monte Carlo search tree is an algorithm to predict the most promising game actions. This algorithm can be applied to any game which has finite length. This algorithm builds a tree of all possible next game positions and also uses them to predict next positions too. Basically, MCTS is a loop of four steps known as selection, expansion, simulation and back-propagation. In the selection step, it finds a position and then it looks for the next action according to the stored statistics. This step does two things simultaneously, first, it selects the best next position and second is that it also looks for the less promising actions which are unexplored. When the algorithm is stuck at the selection step, then it adds a new node and expands the tree. This step is known as the expansion step. After the expansion of the tree, it simulates all actions to end the game. After finding the end in the simulation step, the back-propagation step updates all nodes which were traversed during the game. And it also modifies the win/lose ratio of the game for the machine.

Now a days researchers prefer monte carlo search tree (MCST) over the alpha-beta algorithm. Because, the alpha-beta algorithm was unable to handle the high branching factor which is common in most games like chess and GO. And, second thing is that it is highly randomized, so, it creates the powerful AI models for the board games like chess.

References

1. Claude Shannon (1950). "Programming a Computer for Playing Chess" (PDF). Philosophical Magazine.

2. Samuel, Arthur L. "Some studies in machine learning using the game of checkers. II—Recent progress." IBM Journal of research and development 11.6 (1967): 601-617.

3. Hom, Vincent, and Joe Marks. "Automatic design of balanced board games." Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE). 2007.

4. Chaslot, Guillaume, et al. "Monte-Carlo Tree Search: A New Framework for Game AI." AIIDE. 2008.