2018

Keith Feeney

Introduction to Artificial Intelligence

15015556

Artificial Intelligence Strategies for Playing Chess



(Cover image: Public domain)

Contents

[Introduction 2](#_Toc525743806)

[Tree searching 2](#_Toc525743807)

[The Evaluation Function & Minimax 3](#_Toc525743808)

[Heuristics & Optimisations 3](#_Toc525743809)

[Conclusion 4](#_Toc525743810)

[Further reading 4](#_Toc525743811)

[References 5](#_Toc525743812)

# Introduction

This report looks at how an Artificial Intelligence (AI) machine could play a game of chess, either with another machine or human. We look at tree searching and how minor it is in comparison to the Montae Carlo tree search. We also look at the Evaluation Function and Minimax. These are ways of predicting moves from either the AI machine or an opponent and also both. After this, this report looks at how these strategies can be improved with heuristics and optimisations.

# Tree searching

A basic way of how an AI machine could play chess is with a Tree Search. A tree search is where every possible result is programmed into the computer. So when the chess board is a particular way, that is, where all the pieces are, are put into the system. The computer then searches using a Tree Search, to get the correct next move based on what pieces are currently on the board and where a next move would be allowed. The AI machine will continuously search for the move until the game is won. (Piech, et al., 2013) (Gupta, 2018) In a way, this is how a machine could seem intelligent, but it’s simply just searching. However, if the computer was able to have access to a large library of how humans play chess and incorporate it into its own learning, the computer would become very smart and should nearly always win.

An example of how AI uses tree searches is with the Monte Carlo tree search (MCTS). MCTS is an algorithm used to analyse a large sum of numbers while looking at the probability to obtain an estimated result that would not be doable by another way. (Investopedia, n.d.) MCTS is random, and it was designed to be random. This is a benefit as the chess game could become predictable if it was not random. Decisions that are not good and also blunders are less frequent. MCTS also uses a technique that allows it to look at the latest outcomes while also being able to utilise the outcomes it does know. (Champandard, 2017)

# The Evaluation Function & Minimax

The Evaluation Function is used to gather information of how valuable or how good a position a piece is. It evaluates the position. So, in a chess game, the function would assign a value to each of its own chess pieces. The lower the number, the less likely the move is a good one. However, one piece with the highest number would be the most playable piece to obtain checkmate. (Piech, et al., 2013) (Encyclopædia Britannica, n.d.)

Correctly predicting its own moves is a really good start to getting the opponent checkmated with the AI computer. But it would be hugely beneficial to be able to predict the opponent’s moves as well. This is where the Minimax algorithm comes into play. The Minimax algorithm is so-called because the players are named “maximiser” and “minimiser”. The player that is the maximiser is trying to obtain the most points, while the minimiser is trying to obtain the least points. The Minimax algorithm looks at the back and forth of the players, who, how each of them are playing and looks at the most ideal and best next move. (GeeksforGeeks; rootshadow, n.d.) (Piech, et al., 2013)

# Heuristics & Optimisations

Having the tree searches, the evaluation function and the minimax is an excellent beginning waypoint in AI learning the chess game and how to get the opponent checkmated. It works better on a small scale. But for larger computers and the likes of servers running hundreds or even thousands of chess games simultaneously, these strategies on their own may not be enough. Instructions can be created to optimise these strategies for the likes of increasing the computing speed or what is the maximum time an AI machine can wait before playing. For example, an instruction could be that an AI machine should stop looking for new moves when the opponent is checkmated as the opponent would not play that move. There would not be any point in continuing as the game would be over. Another instruction could be to that if the opponent is a few moves away from being checkmated, then the AI machine should set a time limit on looking for new moves, considering if the opponent is going to play the next piece, if any. (Piech, et al., 2013)

# Conclusion

This report looked the strategies on how a game of chess is played by an AI machine. We looked at how tree searching and the MCTS assist with the AI machine in playing a game of chess. We also looked at how the Evaluation Function and Mimimax greatly improve the AI Machine’s chances. This report also looked at further optimisations for these strategies.

# Further reading

* Montae Carlo Tree Search
* <http://mcts.ai/>
* Heuristic Evaluation Function
* [http://intelligence.worldofcomputing.net/ai-search/heuristic-evaluation-function.html#.W6ugTmhKiUk](http://intelligence.worldofcomputing.net/ai-search/heuristic-evaluation-function.html%23.W6ugTmhKiUk)
* Minimax Algorithm in Game Theory
* <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/>
* Heuristic Search in Artificial Intelligence
* <https://ac.els-cdn.com/S0004370201001114/1-s2.0-S0004370201001114-main.pdf?_tid=6a5a3758-b78a-4214-9583-5ac8baad2a71&acdnat=1537976945_7adae0285c511a703009014bba0e23e0>

# References

Champandard, A. J., 2017. *Monte-Carlo Tree Search in TOTAL WAR: ROME II’s Campaign AI.* [Online]   
Available at: http://aigamedev.com/open/coverage/mcts-rome-ii/  
[Accessed 26 September 2018].

Encyclopædia Britannica, n.d. *Chess Game.* [Online]   
Available at: https://www.britannica.com/topic/chess/The-time-element-and-competition#ref300066  
[Accessed 26 September 2018].

GeeksforGeeks; rootshadow, n.d. *Minimax Algorithm in Game Theory | Set 1 (Introduction).* [Online]   
Available at: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/  
[Accessed 26 September 2018].

Gupta, P., 2018. *Search Algorithms in Artificial Intelligence.* [Online]   
Available at: https://hackernoon.com/search-algorithms-in-artificial-intelligence-8d32c12f6bea  
[Accessed 26 September 2018].

Investopedia, n.d. *Monte Carlo Simulation.* [Online]   
Available at: https://www.investopedia.com/terms/m/montecarlosimulation.asp  
[Accessed 26 September 2018].

Piech, C., Joubert, N. & Liang, P., 2013. *Deep Blue.* [Online]   
Available at: http://stanford.edu/~cpiech/cs221/apps/deepBlue.html  
[Accessed 26 September 2018].