Employment of Monte-Carlo Tree Search and Minimax Algorithms in Connect4: A Literature Review

The traditional Connect Four game was published in 1974 by Milton Bradley (now Hasbro) and has also been known as Four-in-a-Row and Plot Four [11]. The game is a two-player game played on an upright board, with six rows of seven empty holes. Each player is given an equal number of pieces (twenty-one) to drop one at a time from the top of the board. Each player takes turns playing a piece until they make a straight vertical, diagonal, or horizontal line of four pieces with the first person doing so winning the game. Connect Four is categorized as a zero-sum game, meaning that the Minimax algorithm, as well as the Monte-Carlo tree search, can be applied. Through this literary review we explore the application of the Minimax algorithm in conjunction with the Monte-Carlo tree search to apply A.I. algorithms efficiently and effectively to the game, examining the findings of past research to expand our understanding of how effective both approaches are when applied to zero-sum, two player related games, namely Connect4 and Connect6.

Application of Improvised Monte-Carlo Tree Search Algorithm in AI Games

The two stage Monte Carlo Tree Search (MCTS) architecture for connect6 proposed by S. Yen[6] et al. has an unique dual staged approach which is more efficient than traditional MCTS. Threat space search (TSS), which is intended to address the sudden-death issue, is the primary emphasis of the first stage. This paper suggests a technique termed iterative threat space search (ITSS), which combines standard TSS with conservative threat space search, for the double-threat TSS in Connect6 (CTSS). The game-theoretic value of the starting location is estimated using MCTS in the second step which helps in selecting the most favorable move. Through the combination of ITSS and MCTS, two-stage MCTS proved to that for the locations where TSS solutions are available, search efficiency is much higher than that of conventional MCTS and for the locations where double-threat solutions exists, the ITSS search structure used in this study was unquestionably more effective than BFS, DFS, or conventional MCTS.

Yen, Shi-Jim et al.[7] proposed yet another improvised MCTS strategy where it takes sudden death scenarios into consideration. Games like connect4, connect6 have an option of determining a game's winner at any instance of the game (sudden death), and traditional MCTS do not consider sudden death situations. To overcome this drawback this paper offers a strategy of expanding leaf nodes. Traditional MCTS creates one leaf node in each simulation to regulate the growth of the search tree. Where the proposed improvised version of MCTS grows leaf nodes by two levels or more in every simulation, while limiting the depth of playout. The experiments conducted in the study demonstrates that for locations with TSS solution, proposed technique outperforms conventional MCTS.

In order to retain the general idea and the popularity of AlphaZero while also reducing hardware and computing costs, the purpose of research conducted by Colin Clausen et al [9] was to look at potential improvements to the efficiency of AlphaZero. Therefore, the research focuses on three potential expansions of the core AlphaZero algorithm: first being investigation of the game process as a tree, followed by representing the self-learning phase as an evolutionary process, and finally using network internal properties as auxiliary objectives. The first expansion focuses on how AlphaZero adopts a dispersed strategy when playing games: Instead of numerous computers playing independent games, just one plays all the games and asks from a service an MCTS assessment, which is dispersed over many machines. This helps in filtering out duplicate evaluations and hence in increasing the efficiency when compared to the extended baseline The second feature focuses on using domain specific knowledge-based features by embedding deep neural networks to auto add features. This second approach provided a small improvement in the test accuracy. For the final third approach, the authors use the results of played games to assess various hyperparameters; this tuning lead to the algorithm winning more games compared to the traditional approach.

Chaslot et al.[10] proposed Monte-Carlo Tree Search (MCTS) as a revolutionary, unified framework for game AI in this research. Randomized investigations of the search space are employed in the framework to forecast the most promising game actions. Chaslot et al.[10] claimed that MCTS can create effective game AI by using highly randomized and weakly simulated games. Chaslot et al.[10] demonstrated that MCTS may be used well in classic board games, modern board games, and video games in demonstrations. There are several research publications on Monte-Carlo Tree search for various games, however the approach is limited to a few boardgames. The technique is not restricted to specific board games and may easily be applied to modern

board games or video games. Furthermore, its implementation is extremely simple. They demonstrated in this research paper that MCTS may be used well to traditional board games, modern board games, and video games in a generic manner.

Alba Cotarelo et al.[12] proposed in this study that Monte Carlo Tree Search may be accomplished efficiently and generically using neural networks. The experiments with a unique counter for the game Dots and Boxes revealed that this technique is insufficient to win any game versus an MCTS domain-based implementation. Tests with a double counter and no Artificial Neural Network show an improvement in accuracy with 50% of victories but are 2.8 times slow. Finally, a universal implementation of MCTS with an Artificial Neural Network that lacks domain knowledge may outperform an MCTS implementation with heuristics in 81 percent of situations but takes 15.40 times longer to execute. A general representation of the board and the lack of heuristics over the MCTS algorithm degrade performance but allow this approach to be applied to various situations.

Marc Lanctot et al.[13] proposed a family of sample-based counterfactual regret minimization (CFR) methods for finding approximation equilibria in lengthy games in this study, which includes all earlier CFR versions. Marc Lanctot et al.[13] also presented two sampling schemes: outcome-sampling, which samples only one history every iteration, and external sampling, which samples a deterministic strategy for the opponent and chance. Researchers gave regret limits for both sampling types, demonstrating that external sampling with high probability provides an asymptotic computational time improvement over vanilla CFR. Then, in very diverse domains, Marc Lanctot et al.[13] demonstrated empirically that the reduction in iteration time balances the increase in necessary iterations, resulting in quicker convergence. Researchers suggested that their technique might be naturally expanded to circumstances when perfect recall is not assumed. Imperfect recall might be utilized as an abstraction technique for activities, where knowledge sets are organized by key partial sequences rather than their whole sequences.

The idea of MCTS complexity of an artificial player as the lowest number of iterations required for a vanilla MCTS to perform equivalently to the target player proposed by Pier Luca et al.[14] in this research. Pier Luca et al.[14] devised a bisection algorithm to compute such a value, as well as two criteria to use when two players perform similarly, namely Score Difference and True Skill. When the difference in the number of victories is less than a certain threshold, the former offers a definite barrier to end bisection. The latter calculates the match quality as an estimate of the match's draw probability. Pier Luca et al.[14] began by calculating the complexity of players built using MCTS with a known amount of iterations. The technique was then used to compute the MCTS complexity of players executing various strategies. Bisection can reliably detect the amount of iterations employed in target MCTS players, according to the data given. Pier Luca et al.[14] further show that when applied to players using unknown strategies, their technique gives results that are consistent with the underlying complexity of the evaluated strategies, with better players receiving higher MCTS complexity scores. Notably, by utilizing MCTS iterations to assess the strength of players, it is possible to compare the complexity of algorithms that would otherwise be incomparable.

Application of Improvised Minimax Algorithm in AI Games

Research study proposed by Rijul Nasa et al.[8] discusses the improvements in traditional Minimax algorithm with the use of alpha-beta pruning, which results in fewer nodes being assessed in the game tree.
With the inclusion of two new parameters, searching is now considerably faster, and the depth of the game tree search is much increased. Beta is the best value that a minimizer at that level or higher can now offer, whereas alpha is the highest value that a maximizer at that level or above can ensure. The study found that, for the same degree of difficulty, the two algorithms act substantially differently in terms of the time required and the number of iterations carried out, with alpha beta pruning generating the game state much faster and with a lot less iterations than Minimax.

Dual Integration of Monte-Carlo Tree Search with Minimax algorithm in AI Games

In the literary work Early Playout Termination in MCTS, Richard Lorentz [1] explores the differentiation between the Monte-Carlo Tree Search (MCTS) and the classical Minimax game-tree search. One of the immediate differences he highlights focuses on the lack of need for an evaluation function, meaning the random playouts in the MCTS act as a sampling of the possible outcomes from various board positions which in turn can be used to evaluate different positions[1]. As the article progresses, Lorentz presents an intriguing

perspective that lends itself to argue the implementation of the MCTS Early Playout Termination (EPT) as a more successful alternative to solely MCTS. Continuing to highlight early games such as Amazons, Breakthrough, and Havannah[1] being originally written using Minimax techniques, he proceeds to explain their increased win statistics following their conversion to MCTS EPT. After a brief history on the approach, the author focuses on the realization the random playouts were insufficient and by shifting the Amazons game approach to MCTS EPT how they were able to achieve a win rate of 80%, whereas it was previously 0%[1]. Lorentz thoroughly explains how MCTS programs prefer to record wins and losses at the end of playouts vs. keeping track of the margin of victory and finds the same true when integrating EPT. With considerable success being noted amongst MCTS EPT integrated games, Lorentz makes a point to highlight that it does not allow for very precise and sophisticated evaluation functions, but instead has a strong record of outperforming Minimax based programs.

AlphaGo: Mastering the ancient game of Go with Machine Learning by David Silver and Demis Hassabis[2] introduces the reader to the concept that games are a great testing ground for development of smarter, more flexible algorithms capable of tackling problems similar to humans. Through exploring the early history of programs capable of executing games, beginning with the first (Noughts and Crosses) in 1952, spanning almost 7 decades from checkers to Jeopardy and on to algorithms learning to play dozens of Atari games in 2014 from raw pixel inputs[2], the authors set the stage for exploring search algorithms, specifically the Monte-Carlo tree search application to the AlphaGo game. By employing the Monte-Carlo tree search utilizing deep neural networks, the authors found success suggesting intelligent moves to play while evaluating each position as it is reached, resulting in AlphaGo choosing the most successful move in simulation, but at high expense. The literary work continues, highlighting the computing power needed to drive the game to success and increase winning statistics. Upon conclusion the authors close with the reiteration that while games are a great testing ground for the implementation of algorithms and general machine learning techniques, ultimately this will lead the way to employing the same techniques to real world problems, addressing some of the most complex problems of our time[2].

The Monte-Carlo Tree Search and Minimax Hybrids literary work by Baier and Winands[3] focuses on the strategic advantage that the Monte-Carlo Tree Search algorithm has over traditional depth-limited Minimax search with alpha-beta pruning, while also highlighting its weaknesses. It is through the author's seemingly unbiased approach to evaluating the different algorithms available that they conclude to propose a MCTS-Minimax hybrid that employs shallow Minimax searches within the MCTS framework, reducing the occurrence of MCTS falling into traps within tactical situations due to the highly selective tree it builds[3]. As the authors build through strategy, highlighting the phases of their work, they continue to document the strategy's successes and failures and establish experimental results through testing in two different domains: The two-player, zero-sum games *Connect-4* and *Breakthrough*[3]. When evaluating the games against the implementation of regular MCTS as the baseline, they observed some weakness amongst the MCTS-Minimax hybrid due to increased computational cost. As the study progressed, the authors were able to conclude that while MCTS demonstrates strategic strength, allowing the search to observe distant consequences of actions, the tactical strength of Minimax lies with its exhaustive approach, guaranteeing to never miss a consequence of action[3]. Ultimately, by incorporating Minimax into the evaluation of hybrid algorithms, it could prove to potentially be more useful in search spaces with few terminal nodes.

The literary work Monte-Carlo Tree Search with Heuristic Evaluations Using Implicit Minimax Backups by Lancot, Winands, Pepels, and Sturtevant[4] introduces how the Monte-Carlo Tree Search (MCTS) has improved the performance of numerous game engines. However, the success found with MCTS is compounded when implicit Minimax backups are employed, leading to stronger game play performance[4]. Throughout their work, the authors take the time to explore how the MCTS simulation-based best-first search algorithm incrementally builds a tree in memory and the actions that take place to progress through the various states. By doing this, they set the stage for evaluating the practical performance of the implicit Minimax backup technique. Being motivated by the trap moves that pose problems in MCTS, the authors employed implicit Minimax backups to build fortress-esque structures that were handled better than standard MCTS, but without an increase to MCTS problem solving speed[4]. Ultimately finding that the technique improves overall performance/outcome. Upon conclusion the team communicates that they intend to apply the technique to other games, investigating and improving upon their initial evaluations.

Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search by Rémi Coulom explores the Monte-Carlo evaluation of estimating a position by averaging the outcome of random continuations at the leaves of

the Minimax tree. The author explores the algorithm's structure, selectivity, backup methods, and game results to reinforce that the accuracy of the Monte-Carlo Tree Search can be improved with tree search[5]. Ultimately the author is able to conclude that the new algorithm he implemented for Monte-Carlo tree search, to backup node values and uncertainties by applying the tree search algorithm presented in the paper, is an improvement over previous algorithms[5]. This improvement, he found, could mainly be attributed to the new efficient backup method he employed, outperforming a computer-based game of Go and winning a 100-game match against a Monte-Carlo only Go playing program.

Conclusion

The reviewed literature suggests that there are several advantages in integrating the Monte-Carlo Tree Search Algorithm and Minimax algorithm in conjunction with each other in AI games. There is evidence to suggest that the dual integration of Monte-Carlo tree search (MCTS) combined with the Minimax algorithm offers a strategic advantage[3]. While the MCTS has a strategic advantage over traditional depth-limited Minimax search with alpha-beta pruning, it is the combination of MCTS and Minimax, resulting in a MCTS-minmax hybrid, employing a shallow Minimax search within the MCTS framework, that reduces the occurrence of MCTS falling into traps within tactical situations due to the highly selective tree that it builds[3]. In addition, even though the MCTS by itself has been proven to improve the performance of numerous game engines[4], the success found with the MCTS is compounded when implicit Minimax backups are employed[4]. By employing Minimax as a backup, the MCTS has seen improvements[5]. It is through the evaluation of how the MCTS, simulation-based best-first search algorithm, incrementally builds a tree in memory[4] and the actions that take place through the various states, that the stage is set for evaluating more practical performance utilizing the implicit Minimax backup technique.

The reviews also came across multiple improvised versions of Monte-Carlo Tree Search Algorithms like dual stage MCTS [6],[7] with an improvement in effectiveness and higher search efficiency; Introducing three stages in AlphaZero algorithm [9] to increase test accuracy, decrease computation cost, and winning more percentage of games; The literature review also observed that using the Minimax algorithm with tuned alpha and beta values [8] resulted in lesser time required and lower the number of iterations while generating the game state.

This review also demonstrates that MCTS is not restricted to classic board games and can be employed in modern board games as well as video games [10] and the approach MCTS employing Artificial Neural Networks without Heuristics[12] is also mentioned in this review. We encountered many Monte-Carlo application scenarios in which we observed analyzing the complexity of players strategies utilizing Monte-Carlo tree search iterations[14] research and Monte Carlo sampling for regret minimization [13] research paper respectively.

This review provides a solid starting point for understanding the many processes involved in developing a Connect4 game utilizing the Minimax algorithm in conjunction with the Monte-Carlo tree search.

Bibliography

[1] R. Lorentz, "Early Playout Termination in MCTS," in The 14th Conference on Advances in Computer Games (ACG2015), Leiden, The Netherlands, 2015. 978-3-319-27992-3.pdf (springer.com) [Accessed 6 20 2022]

[2] D. Silver and D. Hassabis, "AlphaGo: Mastering the ancient game of Go with Machine Learning," 27 1 2016. [Online]. Available: https://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html [Accessed 6 24 2022]

[3] H. Baier and M. H. M. Winands, "Monte-Carlo Tree Search and Minimax hybrids," in 2013 IEEE Conference on Computational Intelligence in Games (CIG) Computational Intelligence in Games (CIG), 2013 IEEE Conference on., Niagara Falls, ON, Canada, 2013 paper 49.pdf (maastrichtuniversity.nl) [Accessed 6 22 2022]

[4] M. Lanctot, M. H. M. Winands, T. Pepels and N. R. Sturtevant, "Monte-Carlo Tree Search with heuristic evaluations using implicit Minimax backups," in 2014 IEEE Conference on Computational Intelligence and Games, Dortmund, Germany, 2014 im-mcts.pdf (du.edu) [Accessed 6 24 2022]

[5] Rémi Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. 5th International Conference on Computer and Games, May 2006, Turin, Italy. ffinria-00116992f Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search (inria.fr) [Accessed 6 18 2022]

[6] S. Yen and J. Yang, "Two-Stage Monte Carlo Tree Search for Connect6," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 3, no. 2, pp. 100-118, https://ieeexplore.ieee.org/document/5740585 June 2011

[7] Yen, Shi-Jim and Jung-Kuei Yang. "New Simulation Strategy of MCTS for Connect 6." ipsj.ixsq.nii.ac.jp (2010).

[8] Nasa, Rijul et al. "Alpha-Beta Pruning in Minimax Algorithm –An Optimized Approach for a Connect-4 Game." https://www.irjet.net/archives/V5/i4/IRJET-V5I4366.pdf (2018).

[9] Clausen, Colin & Reichhuber, Simon & Thomsen, Ingo & Tomforde, Sven. Improvements to Increase the Efficiency of the AlphaZero Algorithm: A Case Study in the Game 'Connect 4'. https://www.scitepress.org/Papers/2021/102459/pdf/index.html, (2021)

[10] Chaslot, Guillaume & Bakkes, Sander & Szita, Istvan & Spronck, Pieter. "Monte-Carlo Tree Search: A New Framework for Game AI" https://www.researchgate.net/publication/220978338_Monte-Carlo_Tree_Search_A_New_Framework_for_Game_AI(2008)

[11] Gamesver Team, A Brief History (Timeline) of Connect4 (Four-in-a-Row), https://www.gamesver.com/ , A Brief History (Timeline) of Connect 4 (Four-in-a-Row) - Gamesver [Accessed 5 30 2022]

[12]Alba Cotarelo, Vicente García-Díaz , Edward Rolando Núñez-Valdez , Cristian González García, Alberto Gómez and Jerry Chun-Wei Lin "Improving Monte Carlo Tree Search with Artificial Neural Networks without Heuristics".https://www.mdpi.com/2076-3417/11/5/2056 (2021)

[13]Marc Lanctot, Kevin Waugh, Martin Zinkevich, Michael Bowling "Monte Carlo Sampling for Regret Minimization in Extensive Games"https://papers.nips.cc/paper/2009/hash/00411460f7c92d2124a67ea0f4cb5f85-Abstract.html, NIPS (2009).

[14] Pier Luca Lanzi Politecnico di Milano Dipartimento di Elettronica, Informazione e Bioingegneria. "Evaluating the Complexity of Players' Strategies using MCTS Iterations". https://ieee-cog.org/2019/papers