[1] Bruce D. Abramson. *The Expected-outcome Model of Two-player Games*. PhD thesis, New York, NY, USA, 1987. AAI8827528.

The dissertation is about the expected-outcome Model of Two-player games, which considers the game and tree nodes instead of board positions. The author gives a detailed review of the problems related to the games, such as the ambiguity in the definition of the static evaluation function, the problems related to Artificial Intelligence in board games, the ways of handling evaluation errors. The model, which the author proposes and explains thoroughly, calculates an expected-outcome for a player from the random number of games; furthermore, the author describes his model in the board game Othello. The work influenced future authors to successfully implement search algorithms based on random simulations in other board games, such as Go, Hex, Havannah, and the game of Amazon.

[2] Bernd Brügmann. Monte Carlo Go. 1993.

Bernd Brügmann's research is the first paper about the usage of Monte-Carlo Search in a specific board game. Presenting theoretical basics and areas of Monte-Carlo Search is successfully applied, the author provides comprehensive properties of his Go program Gobble, which used random Monte-Carlo Search and focused only on the 9x9 Go board due to computation restrictions. Despite that Gobble performed weaker against Many Faces of Go, the work, which predicted the replacement of heuristic methods and gave possible improvements such as pattern recognition, inspired future authors to generate novel Monte-Carlo Search methods in different areas.

[3] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo Planning. In *ECML-06. Number 4212 in LNCS*, pages 282–293. Springer, 2006.

The innovative research is about a novel algorithm for Monte-Carlo Tree Search called Upper Confidence Bounds applied for Trees, which is practiced efficiently in other board game programs, such as Go, Havannah, Hex, Lines of Actions. The authors provide theoretical both basis and analysis of their algorithms from the point of Bandit Problems and Markov Decision Processes (MDP), testing the algorithm with MDP and random game trees. The results illustrated that UCT needs fewer samples to reach the same error level comparing to both ARTDP or PG-ID, which makes UCT powerful for large samples.

[4] Guillaume Chaslot, Mark H. M. Winands, H. Jaap van den Herik, Jos W. H. M. Uiterwijk, and Bruno Bouzy. Progressive Strategies for Monte-Carlo Tree Search. 2008.

The brief research is about progressive strategies for Monte-Carlo Tree Search, such as Progressive Bias, Progressive Unpruning, and Using Time-Expensive Heuristics. The authors give a brief explanation and work mechanisms of the strategies realized in their Go program called MANGO. The distinctive feature of their experiments is the fact that unlike other authors, they used 13x13 Go board; the reasons are the speed of played games, which is higher than in 19x19 Go boards, and the level of programs which are lower than in 9x9 Go board. Based on the results, the Progressive Strategies dramatically grow the strength of the MCTS search.

[5] Rémi Coulom. Computing “Elo Ratings” of Move Patterns in the Game of Go. *ICGA Journal*, 30:198–208, 2007.

The research is about reinforcement learning in Go program Crazy Stone by Monte-Carlo Tree Search. The program also used pattern recognition as in previous works, but unlike programs in past works that added frequent patterns, the author used the Bradley-Terry model to predict Elo ratings of patterns and Bayesian inference to estimate the outcome of simulations between them. The

author gives a detailed description of theoretical aspects related to his work, experimental results of reinforcement learning with the detailed results of pattern recognition. The results against GnuGo indicate the simplicity and strength of the provided algorithm, which might perform better with a high number of games.

[6] Fabien Teytaud and Olivier Teytaud. On the huge benefit of decisive moves in Monte-Carlo Tree Search algorithms. *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 359–364, 2010.

The research is about the Monte-Carlo Tree Search in connected games, such as Hex and Havannah. The authors give a detailed theoretical review of the data structures and complexity of games and provide experiments with two types of forced moves policies in Havannah: a decisive move that leading to the immediate win and the anti-decisive move avoiding the decisive move one step later. The primary importance of results is not only the improvement of the MCTS but also the possible increase of efficiency by reduction of computational overhead from Tlog*(T) to Tlog(T).

[7] Yizao Wang and Sylvain Gelly. Modifications of UCT and sequence-like simulations for Monte-Carlo Go. *2007 IEEE Symposium on Computational Intelligence and Games*, pages 175–182, 2007.

The groundbreaking paper is about the Go program named MoGo, which is the first program using the UCT algorithm in its MCTS. The authors give a theoretical review of previous works and their improvement based on sequence-like simulations within a limited time instead of revising all nodes iteratively. Additionally, the simulations in a group mode, which is the search within the recognized groups, are also tested. The results are providing a great perspective of the UCT algorithm, which might be improved by a pattern recognition, influenced future authors in the implementation of the algorithm for other board games. As a result, nowadays, almost all MCTS programs use the UCT search.

[8] Mark H. M. Winands and Yngvi Björnsson. Evaluation Function Based Monte-Carlo LOA. In *ACG*, 2009.

The work that is similar to [14] except that authors skip the characteristics of Lines and Actions and briefly explain the MCTS Search. The distinguishing feature that the authors try to find the best value of a bound for Evaluation Cut-Off strategy, testing against MCTS with No-bound Strategy that plays all simulations till the end, and different MIA versions.

[9] Mark H. M. Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. Monte-Carlo Tree Search Solver. Pages 25–36, 2008.

The paper is about the enhancement of the MCTS called Monte-Carlo Solver, which proves a theoretical value of the node and optimizes the selection and backpropagation mechanisms. Giving a detailed explanation of the MCTS, the authors explain how the Solver affects both selection and backpropagation stages. Additionally, the authors provide both diagrams and empirically evaluates the MCTS with a Solver against both default MCTS and different versions of MIA in Lines Or Actions. The results present a considerable rise in the strength of MCTS after the implementation of the Solver, which might be sufficiently implemented in other board games.

[10] Bruno Bouzy and Bernard Helmstetter. Monte-Carlo Go Developments. In *ACG*, 2003.

The journal article is about random Monte-Carlo Search in Go. In the work that based on [1] and [2], the authors provided several improvements, such as another definition of eyes, constant

temperature to regulate the number of games in simulations. The authors test the MCTS programs OLGA based on little-dependent knowledge, and OLEG based on random simulations. While tests indicate that both programs have the same play strength against knowledge-based programs GNUGo and INDIGO and might be designed with little knowledge, the programs still have several drawbacks, such as the underestimation of positions and tactical deficiency. Nevertheless, the authors point out the possible solutions, such as using domain-dependent knowledge in random games, using statistics to explore the locality, and adding tactical modules to a program.

[11] Bruno Bouzy. Associating domain-dependent knowledge and Monte Carlo approaches within a go program. *Inf. Sci.*, 175:247–257, 2005.

The paper is based on [10], where the authors pointed out possible enhancements of Monte-Carlo Search. In his work, Bruno Bouzy tests pseudo-random move generator based on a little domain-dependent knowledge, and the preprocessing of the generator's results with the INDIGO's knowledge-based move generator to select some number of best moves in simulations. The results describe that additional knowledge and the preprocessor significantly increases the strength of Monte Carlo Search and slightly inferior against knowledge-based Go programs; the difference grows by enlarging the board size of Go.

[12] Rémi Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Computers and Games*, 2006.

The article about the Go program named Crazy Stone using a new search method, which stores the value of nodes and selectively allocates simulations of moves that might be better than the current best move. The author gives detailed review of Crazy Stone algorithm, their experiments with different Backup methods, the results against programs GNU Go and Indigo, and possible directions for future enhancements that might improve his algorithm, such as the improvement of selectivity algorithm and uncertainty-backup operator, overcoming tactical weaknesses, scalability of the approach to larger boards.

[13] Sylvain Gelly and David Silver. Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go. Artif. Intell., 175:1856–1875, 2011.

The article is about novel methods of Monte-Carlo Tree Search invented by authors and later widely used in other programs: Rapid Action Value Estimation (RAVE) and heuristic MCTS. The authors provided a comprehensive review of not only known MCTS search methods and board game Go, but also their novel methods and its combinations. Experiments conducted by authors in the MoGo program with different parameters and search algorithms against GNUGo describe the total superiority of the methods against alpha-beta pruning. It also might be seen that MCTS programs, which do not require any human knowledge, dominate against programs with other algorithms in different board sizes.

[14] Mark H. M. Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. Monte Carlo Tree Search in Lines of Action. *IEEE Transactions on Computational Intelligence and AI in Games*, 2:239–250, 2010.

The research is about implementing various Evaluation-Based Strategies in Monte-Carlo Tree Search for the board game Lines Or Actions. The authors give a detailed explanation of the game rules, the pseudo-code of algorithms, how their program MC-LOA handles the steps of the MCTS using the UCT and Progressive Bias, and tests of various evaluation improvements, such as the Evaluation Cut-Off, the Corrective Strategy, the Greedy Strategy, and the mixed Strategy containing all of them. Evaluating all improvements against each other, the authors test both MC-LOA and a

Progressive Bias improvement against MIA 4.5, the program based on alpha-beta pruning. The results describe that MCTS search in Lines Or Actions is an equipollent alternative to alpha-beta pruning.

[15] Broderick Arneson, Ryan B. Hayward, and Philip Henderson. Monte Carlo Tree Search in Hex. *IEEE Transactions on Computational Intelligence and AI in Games*, 2:251–258, 2010.

The article is about Monte Carlo Tree Search in Hex by the authors of the MoHex program, which won gold in the 2009 Computer Olympiad. Giving a detailed description of possible move types of move decisions, such as the analysis of inferior cells and the computing of connection strategies, the authors later explain the algorithms' specification of MoHex. It is attractive that MoHex calculates not the actual position, but the position with the patterns of fill-in. Conducted experimental results indicate that both growth of time per move and the different types of knowledge, such as a heuristic, tree, and opening book, grow the power of MCTS, which proved with results against different programs.

[16] Tristan Cazenave and Abdallah Saffidine. Monte-Carlo Hex. 2010.

Another research about Monte Carlo Tree Search in Hex, where authors try to combine Monte Carlo methods, which they implemented to their Hex program in 2000, with Tree Search. The distinctive feature that authors tried to use both UCT and RAVE methods in their YOPT program. The experiments conducted by authors show the sufficiency of Monte-Carlo Tree Search with the RAVE algorithm in Hex and the dependency between the strength of program and simulation numbers: while the increase of simulation numbers and recognizing templates makes MCTS search with more powerful, the recognition of Ziggurats reduces the search speed.

[17] Jan A. Stankiewicz, Mark H. M. Winands, and Jos W. H. M. Uiterwijk. Monte-Carlo Tree Search Enhancements for Havannah. In *ACG*, 2011.

The article is about Monte-Carlo Tree Search in board game Havannah, where authors try to enhance MCTS by various variations of RAVE search, such as Last-Good-Reply and N-grams. Provided experiments described that the best results in the Last-Good-Reply algorithm where a last good reply causes removing the response from all simulations, which named Last-Good-Reply with Forgetting-1 (LGRF-1). While the implementation of N-grams gives the same growth in the strength of MCTS, combining the two methods give only a slight increase in performance. The most significant rise might be achieved by adding the heuristics for patterns, such as recognizing Joint and Neighbour Moves, Local Connections, and Edge and Corner Connections.

[18] Joris Duguépéroux, Ahmad Mazyad, Fabien Teytaud, and Julien Dehos. Pruning Playouts in Monte-Carlo Tree Search for the Game of Havannah. In *Computers and Games*, 2016.

Another research is about board game Havannah, where the authors tested their novel enhancement of the MCTS against PoolRave, LGRF-1, Move-Average Sampling Technique (MAST), and N-gram Average Sampling Technique (NAST) with the different values of parameters, biases nodes not in the selection step but the simulation step of the MCTS algorithm. The distinctive part of the research is the novel algorithm named Playout Pruning with Rave (PPR), which focuses on the simulation step on the moves with RAVE scores higher than the preassigned threshold. Tests have shown the supremacy of PPR in the different number of playouts.

[19] Julien Kloetzer, Hiroyuki Iida, and Bruno Bouzy. The Monte-Carlo Approach in Amazons. 2007.

The brief research is about Monte-Carlo methods in board game Amazons. After the explanation of rules, the authors review the details of Monte-Carlo Tree Search in the game of Amazons and the heuristic knowledge named a liberty rule that is the move decision when there are less or equal than two possible moves for Amazons, trying to use various search techniques. Experiments described the superiority of MCTS against default Monte-Carlo search, which might be improved by different tricks, such as splitting the movements and firing moves from each other and using combinational evaluation from both average scores and Win-Loss ratio of nodes.

[20] Oleg Arenz. Monte Carlo Chess. Master's thesis, Knowledge Engineering Group, TU Darmstadt, 2012. Bachelor's Thesis.

The thesis work is about Monte Carlo Tree Search in chess, which challenges the assumption that MCTS was not suitable for chess. Giving brief definitions of minimax, alpha-beta pruning, Monte Carlo Tree Search with its enhancements, the author compares Monte Carlo Tree Search enhancements, discussing the possible benefits and drawbacks of each MCTS enhancement from the view of chess, giving essential assessments to each of them. After the realization of MCTS instead of alpha-beta pruning in chess program Stockfish and series of experiments, it becomes clear that Monte Carlo Tree Search is not suitable for chess due to the high number of search traps at the moment, however, possible improvements in the future researches might be implemented.

[21] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *ArXiv*, abs/1712.01815, 2017.

The innovative work by a group of authors is about the AlphaZero program using MCTS for reinforcement learning in board games Shogi and Chess. The research describes the learning mechanisms of AlphaZero and significant differences between AlphaZero and its predecessor AlphaGo Zero. The learning speed of AlphaZero, according to the Elo scale, the frequency of chess openings chosen by AlphaZero for reinforcement learning, and some wins against Stockfish are also provided. Despite that experiments demonstrated sufficient superiority of AlphaZero against alpha-beta programs Stockfish and Emmo, the results cause skepticism due to test conditions, which reduced the strength of alpha-beta programs.

[22] Komodo MCTS (Monte Carlo Tree Search) is the new star of TCEC, http://www.chessdom.com/komodo-mcts-monte-carlo-tree-search-is-the-newstar-of-tcec/.

A site article is about the TCEC tournament for chess engines, where Komodo MCTS placed second in Division 1. The article contains not only notable games of Komodo MCTS that give a picture of a game-style of Komodo MCTS but also an interview with the authors. They explain the improvements and particular features of their engine's work, such as how and why they have combined MCTS with alpha-beta pruning, and the skepticism about neural networks in chess due to test condition of the match between AlphaZero vs. Stockfish and the power consumption of GPU.

[23] Yoshikuni Sato, Daisuke Takahashi, and Reijer Grimbergen. A Shogi Program Based on Monte-Carlo Tree Search. *ICGA Journal*, 33:80–92, 2010.

The article is about the Shogi program based on plain MCTS. Describing the main methods implemented in the program in detail, the authors analyze the fail of their program that has scored inferior against the alpha-beta pruning program TOHMI. The possible reasons are related to features of Shogi that is a tactical game: the wrong evaluation of minor tactical losses and the game during

disadvantage periods. Nonetheless, the article gives a little optimism regarding the MCTS in the long-distance because it reveals the benefits and strategical perspectives of the MCTS.