Parallel Monte Carlo Tree Search in Perfect Information Game with Chance

Junkai Lu, Xiaoyan Wang, Dayi Wang, Yajie Wang

Shenyang Aerospace University, Shenyang 110136
E-mail: junkai-lu@outlook.com

Abstract: Perfect information game with chance is a special form of perfect information game, which includes exogenous uncertainty from chance events. In this paper, we discuss the application of Monte Carlo tree search in perfect information game with chance and propose a parallel Monte Carlo tree search for perfect information game with chance.

Key Words: Monte Carlo Tree Search, Perfect Information Game with Chance, Parallelization

1 INTRODUCTION

Perfect information game with chance is a special form of perfect information game, in which each player can see all of the pieces on the board at all times like perfect information game such as chess and go, but it includes exogenous uncertainty from chance events such as rolling dice. The most famous example of perfect information game with chance is Backgammon, in which each player's decision-making process accompanied by a chance event. The player need to roll dice twice before they make their move in current state, and the set of the moves that player can choose is decided by the result of chance event, which is an important attribute of perfect information game with chance.

In recent years, the Monte Carlo tree search (MCTS) [1,2] ,a well-known game search method, has been successfully applied in many perfect information games such as Go and Hex[3]. In some perfect information games with chance MCTS has also shown to increase performance, such as the backgammon agent McGammon[4] and Einstein würfelt nicht!(EWN) agent OneStone[5]. However, there are not essential differences between the MCTS applied in those agents and standard MCTS. The main improvement of the search method includes chance nodes into standard MCTS to adapt to those extra chance events.

In this paper, according to the characteristic of perfect information game with chance, we present a variant of MCTS called s-MCTS, which is more effective than standard MCTS in perfect information game with chance. We implement s-MCTS in EWN game and the result corroborates our improvement.

This article is organized as follows. In Section 2, we explain the background of perfect information game with chance and Monte Carlo Tree Search method. Then we describe how to enhance MCTS with parallelism in games with chance event and evaluate the improvement empirically in Sect.3 and Sect.4 respectively. Finally, Sect. 5 gives conclusions and an outlook on future research.

The work presented in this paper is supported by Liaoning Innovation and Entrepreneurship Program for College Students under Grant DS514101

2 BACKGROUND

Monte-Carlo Tree Search (MCTS) is a recent paradigm for game-tree search, which is a state-of-the-art approach in many games. However, there are some problems of the standard MCTS in perfect information game with chance, mostly due to the special structure of game tree. The prominent issue is how to deal with chance events such as rolling dice in search process; a simple way to solve this problem is to divide the nodes into two kind of nodes: choice node and chance node. The game tree that consist of those two kinds of nodes is completely differ from the game tree in perfect information game, no chance.

2.1 Perfect Information Game with Chance

In game theory, Zermelo's theorem says that in any finite sequential game of perfect information in which chance affects the decision-making process, each player has a certain best move but the result of it is not be guaranteed, except for those moves by which player can win the game directly.

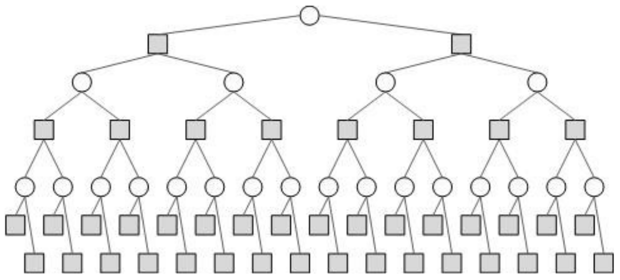

Fig1. A simplified 3-lays game tree, the rectangles represent chance node and the circles represent choice node.

As shown in Figure 1, as a part of a typical game tree of perfect information game with chance, it contains chance nodes (shown as grey rectangles) and choice nodes (shown as circles). Chance node represent chance event, which decides the player's decision-making space in the next step.

Considering the move h ∈ H(H is the set of moves that player may choose in next step), the actual decision-making space of player HC is a subset of H decided by chance events. A chance subtree is a subtree of the game tree with a chance node as its root node; the probability of the chance subtree be visited is equal to the probability of its root node being visited.

Due to the existence of chance events, the prune of searching tree is more difficult. Only after a complete search for their branches, that we can prove one move is better than another one, which leads to a huge burden of search algorithm.

2.2 Monte Carlo Tree Search

Monte Carlo Tree Search, as a best-first search guided by the results of Monte-Carlo simulations, it performs a single simulation from the root node of the search tree to a terminal node at each iteration. During the iterative process, a game tree is progressively built by adding a new leaf node to current game tree on each iteration, whose nodes maintain statistical estimates such as average payoffs. Each new simulation would improve and help future simulations.

Four steps are applied per search iteration [8]:

1) Selection: Starting at the root node, a child selection policy is recursively applied to descend through the tree until the most urgent expandable node is reached. A node is expandable if it represents a non- terminal state and has unvisited (i.e. unexpanded) children.

2) Expansion: One (or more) child nodes are added to expand the tree, according to the available actions.

3) Simulation: A simulation is run from the new node(s) according to the default policy to produce an outcome.

4) Backpropagation: The simulation result is "backed up" (i.e. backpropagated) through the selected nodes to update their statistics.

During the iterative process, the tree selection allows the algorithm to favor those more promising nodes that is more possible to be the best move than others, leading to an asymmetric tree over time. Figure 2 shows asymmetric tree growth using the BAST variation of MCTS. The promising nodes would be visited frequently and the visited times of unpromising nodes are much less.

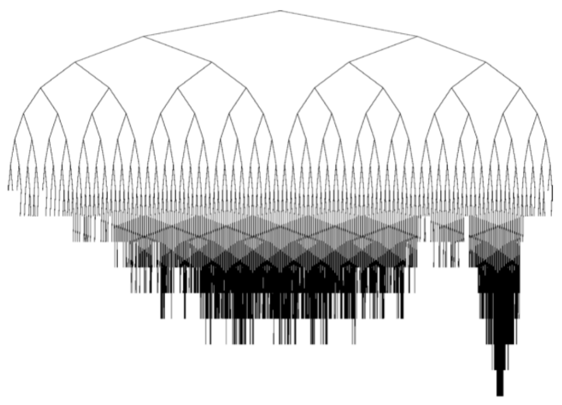

Fig 2: Asymmetric tree growth [9]

3 MCTS Enhancement

When the Monte Carlo tree search is applied to the perfect information game with chance, how to solve the chance events in the search tree is a sophisticated problem. A convenient method is to consider different result of chance events as independent chance nodes, and each chance node is root node of a chance subtree in search tree, which can avoid complex calculation. Figure 3 shows an iterative process from root node to the leaf node that is most urgent and expandable than other nodes that have the same parent. An important point is that the leaf node is not the most urgent expandable node in the tree, but it is truly more urgent than other nodes that have same parent node. This is obviously different from the standard MCTS. For a chance subtree, the probability that a new node is added to expand it equals to the probability that its root node being visited. This attribute results in that the size of the chance subtree is proportional to the probability that their root node being visited.

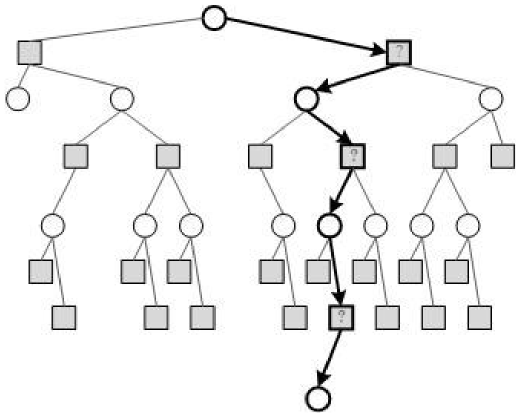

Fig3. One choice nodes is added to expand the tree after several chance events, which is the most urgent childe node of its parent.

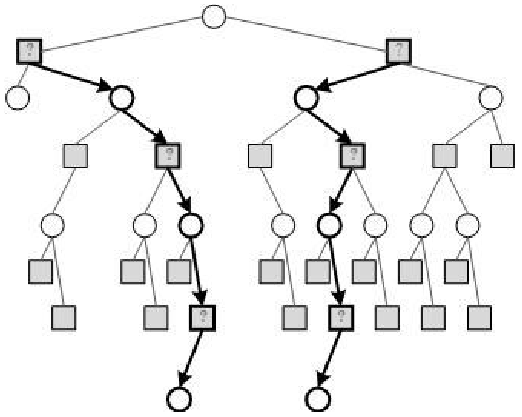

Fig4. Multiple choice nodes are added to expand different chance subtree in the game tree.

3.1. Simultaneous Search

When we make moves in a state that is not terminal state, any child choice node that also is the root node of a chance subtree may be chosen. Although the same state may exist in different chance subtree, most of them can be ignored. The process of adding a new node to expand a chance subtree would not be effected by other chance subtree or effect the growth of another chance subtree, so adding multiple nodes to different chance subtree simultaneously do not affect the growth of game tree. Figure 4 have shown the process that add two nodes to expand different chance subtree, that can be comprehended as the superposition of two process which is showed in Figure 3.

3.2. Parallelization

When adding multiple nodes simultaneously to expand the game tree, it can also be considered to execute multiple standard Monte Carlo tree search simultaneously in different chance subtree whose root node is different. After all searches in different chance subtree have finished, the payoff of their root node would be backpropagated. In order to avoid dead lock, the root node of the independent search process in a chance subtree should not be the root node of another independent search process or any its parent node. Nevertheless, it is not absolute because this problem can also be solved by using global lock. As shown in Figure 4, the two independent search process do not begin from the root node of the game tree but from its two child nodes.
This algorithm is easy to be parallelized. When we implement it, each thread can independently maintain a chance subtree and backpropagate the payoff of chance subtree after all threads have finished

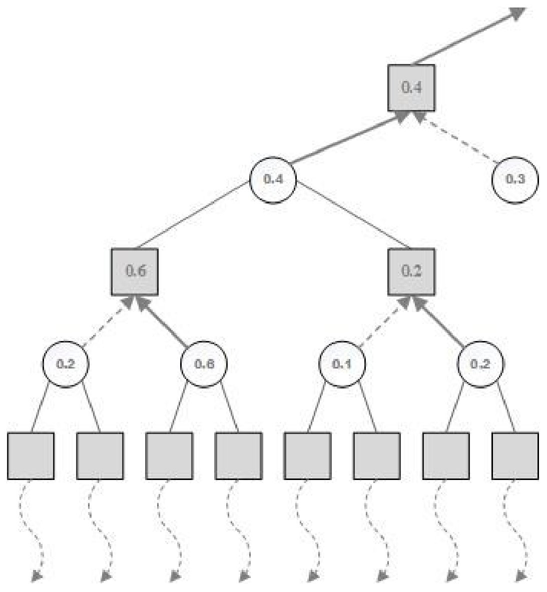

Fig5: A part of simplified minimax process, note that the below circles represents the actual leaf node of this tree. We assume the probability of each chance node is equal.

Suppose that all chance nodes of a certain layer of the game tree are the root nodes of different independent search process, we should backpropagate their payoff by minimax instead of the average payoff. For those nodes in upper layer, average payoff is meaningless. They should be considered as a minimax tree whose leaf node is the root node of each independent search process. This method is similar to Coulom's original maximum backpropagation[1]. This method of backpropagation suggests switch to propagating the maximum value instead of the simulated (average) value after a number of simulations to a node have been reached. In our algorithm, each node has been reached with enough times, so propagating the maximum value is better than the average value. Figure 5 is a part of simplified minimax process. From those leaf nodes that also are the root nodes of an independent search process, the minimax value would be propagated to update the value of their parent nodes.

4 EXPERIMENT

In this section, we evaluate empirically the addition of the parallel MCTS. We choose Einstein würfelt nicht!(EWN), a typical perfect information game with chance, as the platform to test the algorithm.

4.1. Einstein würfelt nicht!

Einstein Würfelt Nicht! (EWN) is invented in 2004 by Ingo Althöfer[10], It is played on a 5 × 5 board where each player (called Blue and Red) controls 6 playing pieces, which are stones with the numbers 1 to 6 marked on them and placed in a triangular pattern (shown in Figure 6). The first player that occupy the corner field of his opponent with a playing piece would be the winner, and an alternate way to win the game is to capture all of the opponent's pieces. In each turn, the piece to move is decided by the roll of a die, the piece can only move horizontally, vertically or diagonally to the corner field of his opponent. If a piece is decided to move to a field occupied by another piece, then the another piece would be removed from play no matter which player it belong to.

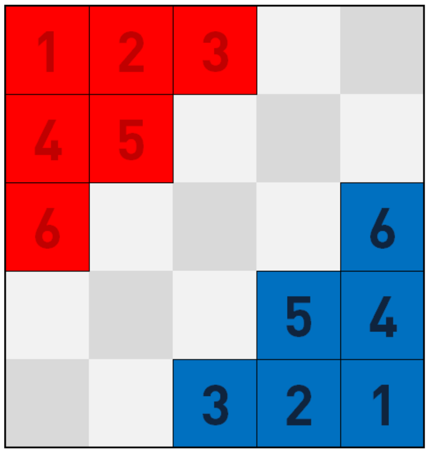

Fig6: A typical EWN starting position, each player controls 6 playing pieces numbered 1 to 6.

Each algorithm has been tested more than 100 times, the starting position of each player is decided randomly, and alternate to on the offensive.

4.2. Result

We test the algorithm with 18 threads and 48 threads, and each thread maintains an independent search tree. We match them with standard MCTS that only expand one node in each iteration. Figure 7 shows the result under different time limit.

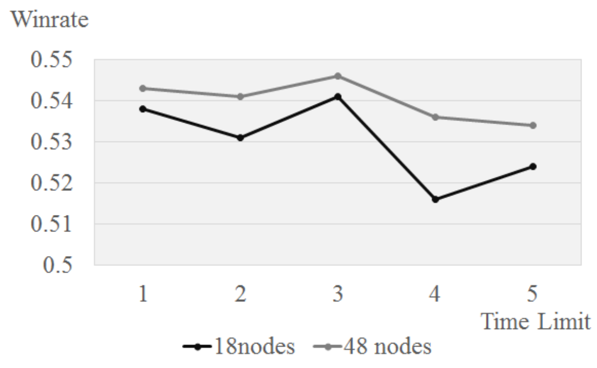

Fig 7: The win rate under different time limit.

The results of the tournaments are given for searches with a thinking time of 1, 2, 3, 4, 5 seconds per move, respectively. As the thinking time rises, the win rate of parallel MCTS is decreasing slightly, but consistently higher than the standard MCTS. This can be explained by the fact that the number of threads do not bring a significantly change to the depth of search, even though the algorithm has multiplied the size of the search tree. Based on the results, we can confirm the algorithm is available and with the increase in the number of threads, the result of this algorithm would be more accurate.

5 FUTURE WORK

We have demonstrated that the parallel MCTS for perfect information game with chance is available. This algorithm is also available for GPU, which has more core to maintain more independent search process. However, how to manage those search process, such as releasing unpromising search tree, exploring the child node of promising search tree, remain a challenging issue. The GPU algorithm for perfect information game with chance is another problem worthy to explore, and that may be our future work.

REFERENCES

[1] R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In 5th International Conference on Computers and Games, volume 4630 of LNCS, pages 72–83, 2007.

[2] L. Kocsis and C. Szepesvári. Bandit-based Monte Carlo planning. In 15th European Conference on Machine Learning, volume 4212 of LNCS, pages 282–293, 2006.

[3] Broderick Arneson, Ryan B. Hayward, and Philip Henderson. Monte-Carlo tree search in Hex. IEEE Transactions on Computational Intelligence and AI in Games, 2(4):251–258, Dec 2010.

[4] F. Van Lishout, G. Chaslot, and J. W. H. M. Uiterwijk, “Monte-Carlo tree search in backgammon,” in Comput. Games Workshop, Amsterdam, The Netherlands, 2007, pp. 175–184.

[5] Richard J. Lorentz. An MCTS Program to Play EinStein Würfelt Nicht!. In Advances in Computer Games, volume 7168 of LNCS, pages 52–59, 2012.

[6] Schwalbe, Ulrich, and Paul Walker. "Zermelo and the early history of game theory." Games and economic behavior 34.1 pages: 123-137, 2001.

[7] Wolfe, Philip. "The strict determinateness of certain infinite games." Pacific Journal of Mathematics 5.1 pages: 841-847, 1955.

[8] G. M. J.-B. Chaslot, S. Bakkes, I. Szita, and P. Spronck, “Monte-Carlo Tree Search: A New Framework for Game AI,” in Proc. Artif. Intell. Interact. Digital Entert. Conf., Stanford Univ., California, 2008, pp. 216–217. 5.

[9] R. Coquelin, Pierre-Arnaud and Munos, “Bandit Algorithms for Tree Search,” in Proc. Conf. Uncert. Artif. Intell. Vancouver, Canada: AUAI Press, 2007, pp. 67–74.

[10] Althöfer, I. On the origins of Einstein würfelt nicht!. http://www.althofer.de/origins-of- ewn.html

[11] J Lu, X Wang, Y Li, "Einstein würfelt nicht! Strategies Research and Algorithm Optimization." 27th Chinese Control and Decision Conference, 2015, 5818 – 5821.