Monte Carlo Tree Search in Chinese Dark Chess

Shi-Jim Yen [1], Cheng-Wei Chou [2], Tsan-Cheng Su[3],
Shun-Chin Hsu[4], Hsin-Hung Chou[5] and Jr-Chang Chen[6]

[1, 2, 3] Dept. Of Computer Science and Information Engineering, National Dong Hwa University, Taiwan
[4,5] Dept. of Information Management, Chang Jung Christian University, Tainan, Taiwan
[6]Dept. of AM, Chung Yuan Christian University, Taiwan
[1]sjyen@mail.ndhu.edu.tw, [2]d9721002@ems.ndhu.edu.tw, [3]d9821009@ems.ndhu.edu.tw,
[4]schsu@mail.cjcu.edu.tw, [5]chouhh@mail.cjcu.edu.tw, [6]jcchen@cycu.edu.tw

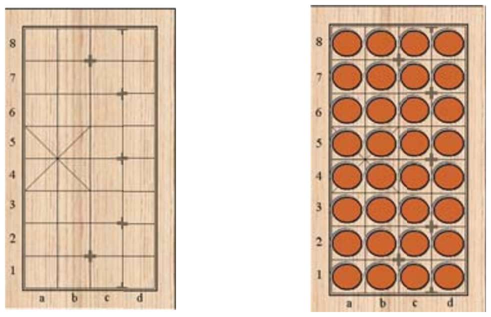
***Abstract*—In this paper we propose an efficient method to apply Monte Carlo Tree Search (MCTS) to Chinese dark chess (CDC) and get very well experimental results. CDC is an imperfect information variety of Chinese chess. This game is very popular in Chinese culture sphere. The key point of solving this game is how to handle the imperfect information part very well. Unfortunately, most of state-of-the-art CDC programs cannot do it. MCTS is a famous best-first search algorithm, which is suitable to make decisions in uncertain environments. We try to change the tree structure of MCTS to handle the imperfect information part of CDC, and obtain great improvement on CDC. Moreover, our program won every game in computer CDC tournament in TCGA 2011 computer game tournament.**

Keywords- Chinese Dark Chess, Computer Game, MCTS, Imperfect Information Game

I. INTRODUCTION

Chinese dark chess (CDC) is a very popular two-players board game in Chinese culture sphere. This game is a variety of Chinese chess. Although they use the same pieces and similar board (CDC only uses half of board of Chinese Chess), however, in the beginning of CDC, all pieces are placed conversely, so types of pieces are unknown. Therefore, CDC is an imperfect information game. The type of pieces can be known by flipping it and spending one round. If type of piece is known, piece can be moved as way of chess-like. Because it takes one round to switch the state of a piece from unknown to known, therefore deciding which piece and when to make unknown into known is a difficult problem of this game. Section II will show more details.

Until now, state-of-the-art program of CDC still uses min-max tree and alpha-beta pruning[1]. Because CDC is an incomplete information game, alpha-beta method cannot handle the flipping action satisfactorily. Monte Carlo Tree Search (MCTS) is an efficient algorithm for applying it to not only the game of Go [2], but also Amazons [3], even as some incomplete information games as Poker [4], Kriegspiel [5], Backgammon [6], and Phantom Go [7]. In This paper, we try to apply MCTS to CDC, change tree structure of flipping action, and make the flipping action and the moving action can be measured in the same way.



(a) (b)

Fig. 1. (a) The board of CDC, (b) The initial state of CDC.

II. Chinese Dark Chess

Chinese dark chess (CDC) uses the same pieces with Chinese chess, but half of the board of Chinese chess. 4×8 squares are used as the board of CDC. Figure 1 shows the board. In the beginning of a game, all the thirty-two pieces are placed conversely, so types of pieces are unknown, as Fig.1(b).

TABLE I. The Piece Types of CDC

Type of Pieces	Red Pieces	Black Pieces	Rank	Count
King	帥	將	1(Highest)	1
Guard	仕	士	2	2
Minister	相	象	3	2
Rook	俥	車	4	2
Knight	傌	馬	5	2
Cannon	炮	砲	6	2
Pawn	兵	卒	7(Lowest)	5

The pieces of CDC have two kinds of colors, black and red. There are seven types of pieces in this game, as Table I. The rank row shows the rank of every type of pieces. In general, a piece can capture the pieces with equal or lower ranks, but Pawn, the piece with lowest rank, can capture King, the piece with highest rank. It makes a cycle. All types of pieces except Cannon can only move or capture a piece one square up, down, left and right within the 4x8 grid area.

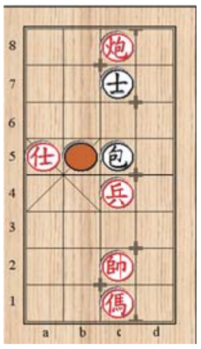


Fig. 2. An example of Cannon. Now turn is black. The black Cannon can moves to up or right square. However, it cannot capture the red Pawn. In contrast, the black Cannon can eat the red King, the red Guard and the red Cannon.

Cannon is a special piece. Although it moves as the same way of other pieces, but it cannot capture around pieces. If it wants to capture, it needs one piece, which can be any type, as a carriage to capture any opponent's pieces of any distance in the same row or column. Figure 2 shows the ability of this special piece.

III. Related Work

CDC can be seen as a chess-like game, therefore most of the programs in past tournaments [10] [11] use min-max algorithm, alpha-pruning, and some additional flipping heuristic. One of the best program is Flipper, which is developed by Bo-Nian Chen and Tsan-sheng Hsu [1]. Flipper combines the flipping move to the search tree. However, this method will cause very high branching factor. They use some methods to reduce the branching factor. The most efficient method is initial-depth flipping, which means only consider the flipping moves in the first level of search tree, as Fig. 3.

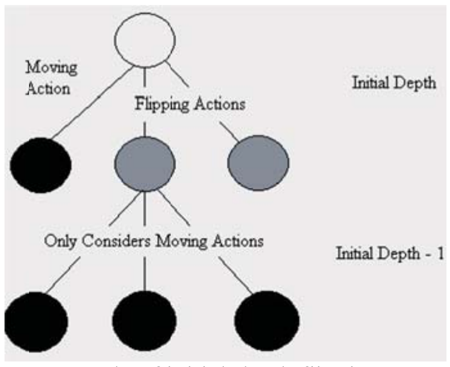


Fig. 3. An example of initial-depth flipping. [1]

Monte Carlo Tree Search (MCTS) is a best-first search. This search method builds an Asymmetric tree in memory incrementally. In general, MCTS have four stages as Fig.4. Those stages will be described below:

Selection : Every time, MCTS starts from the root, selects the best child in every level until reaches the leaf node of current tree. The strategy UCT [8], as formula 1, is the famous method to decide which child is the best. In formula 1, v_i means past rewards of node i, T_i means the visit counts of parent of node i, N means the visit counts of node i. *Bias* is a parameter to balance exploitation and exploration.

$$v_i + bias \times \sqrt{\frac{\ln N}{T_i}} \qquad (1)$$

Expansion : One (or more) new node(s) will be add the current tree. These nodes usually mean new boards by playing legal moves.

Simulation : This stage will play the game to end from the position of leaf node, and obtain result of the game.

Backpropagation : The result of simulation will update all nodes of the best sequence which are selected in *Selection* stage.

IV. Our Method

Our method is to apply MCTS to Chinese dark chess by changing tree structure of flipping actions. The four stages of MCTS applied to CDC will be described below:

Selection : In this stage, we use the famous strategy UCT [8] as formula 1. The goal of selection is to find a child which maximize formula 1 in every level of the game tree, and obtain a best sequence from root to leaf.

Expansion : In this stage, we divide nodes to two kinds. One is the moving actions, another is the flipping actions. If the action of the selected leaf node is moving action, then we expansion it by normal way. If the action of the selected leaf node is flipping action, we will create an inner node as the root of sub tree which flip some specific piece. For example,

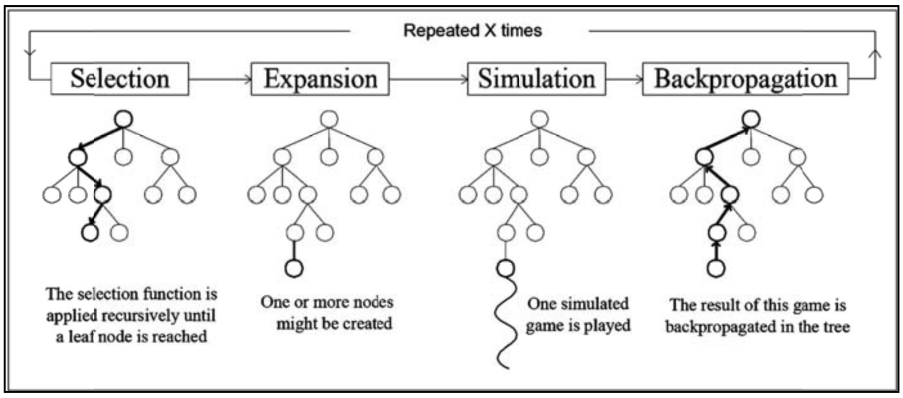

Repeated X times
Selection
Expansion
Simulation
Backpropagation
The selection function is applied recursively until a leaf node is reached
One or more nodes might be created
One simulated game is played
The result of this game is backpropagated in the tree

Fig. 4. Scheme of a Monte-Carlo Tree Search. [13]

there are two kinds of three pieces, one king and two cannon, are not revealed. If now we select a leaf node which action is flipping, and there is not any inner node in this node. We will create an inner node, and the real revealed piece will be chosen by probability distribution. In this example, there is one-third chance of choosing king, and two-thirds chance of choosing cannon. If next time the leaf node is chosen, and king is chosen again, then we will expansion the inner node and create its children. Figure 5 shows the process of our expansion stage.

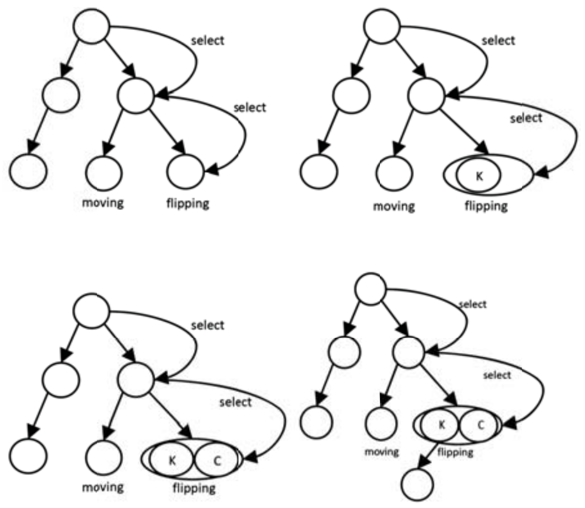


Fig. 5. The process of our expansion stage.

Simulation : In the simulation game, we only use one rule in simulation : if there are moves that can eat opponent's piece, then these moves will own the highest priority of choice. Otherwise, we randomly select a legal action.

Backpropagation : If the result is not draw, then the number of wins of winning nodes in the best sequence will be added 1. If the result is draw, then the number of wins of all nodes in the best sequence will be added 0.5.

V. EXPERIMENT RESULTS

We will measure our result by battling with another CDC program, BASIC, which uses nega-max algorithm and some simple flipping process as explained in [1]. Our experimental environment is formed by using Intel Core 2 Quad Q9450 CPU, 8G RAM, and the operation system is Windows 7. All of our experiment only used one core. Table II shows the scalability of our program. As general MCTS programs, winning rate of our program grows fast in less simulations, and meets plateau time after simulating fifty thousand times.

TABLE II. SCALABILITY OF OUR PROGRAM

Sims	Sec. of the first move	Win	Draw	Lose
2000	<1	24.25% (±2.1)	51.75% (±2.5)	24% (±2.1)
10000	1	55.5% (±2.5)	40% (±2.5)	4.5% (±1)
20000	2	62% (±2.4)	34.5% (±2.4)	3.5% (±0.9)
50000	6	72.75% (±2.2)	24% (±2.1)	3.25% (±0.9)
100000	12	72% (±2.2)	24.25% (±2.1)	3.75% (±0.9)

TABLE III. COMPARISON OF OUR PROGRAM AND STATE-OF-THE-ART PROGRAM[1]

AI	Sec. of the first move	Games	Win	Draw	Lose
MCTS_10k	1	400	55.5% (±2.5)	40% (±2.5)	4.5% (±1)
MCTS_50k	6	400	72.75% (±2.2)	24% (±2.1)	3.25% (±0.9)
VAR1	30	200	59.5%	39.5%	1%
VAR2	30	200	50%	49.5%	0.5%
VAR3	30	200	44.5%	55%	0.5%

Table 2 shows that our program perform state-of-the-art program of Chinese Dark chess [1] in both winning rate and spending time. MCTS_10k and MCTS_50k means simulating ten thousand times and fifty thousand times, respectively. VAR1~3 are three versions of state-of-the-art program [1].

VI. Conclusion

In this paper, we try to apply MCTS to the game of Chinese dark chess by changing tree structure of flipping actions. This method makes the flipping actions and the moving actions can be measured by the same way. The experiment results demonstrate that our method is efficiency. In addition, we only use one domain knowledge in simulation stage, and also obtain good result. Moreover, our program, Diablo, won the gold medal in Chinese dark chess at the TCGA computer game tournament which took place from June 25th to 26th 2011 at Tainan, Taiwan [9].

In the future, we will try to use RAVE [10], and add more domain knowledge to our program. We believe those techniques can improve the strength of our program greatly.

References

[1] Chen, B. N., Shen, B. J. and Hsu T. S. (2010) Chinese Dark Chess. ICGA Journal. Vol. 33, No.2, pp.93-106.

[2] Gelly, S., Wang, Y., Munos, R., & Teytaud, O. (2006). Modification of UCT with patterns in Monte-Carlo Go (Technical Report 6062). INRIA.

[3] Kloetzer, J., Iida, H., Bouzy, B. (2007) The Monte-Carlo Approach in Amazons. Proceedings of the Computer Games Workshop 2007 (CGW 2007), pp. 185–192. Universiteit Maastricht, Maastricht.

[4] Guy, V. B., Kurt, D., and Jan R. (2009) Monte-Carlo tree search in poker using expected reward distributions. Advances in Machine Learning, First Asian Conference on Machine Learning, ACML 2009, pp. 367–381.

[5] Ciancarini, P. and Favini, G.P. (2010) Monte Carlo tree search in Kriegspiel. Artificial Intelligence. Vol. 174, Jul. 2010, pp. 670-684.

[6] Van Lishout, F., Chaslot, G., Uiterwijk, J. (2007) Monte-Carlo Tree Search in Backgammon. Computer Games Workshop, pp. 175-184.

[7] Borsboom , J., Saito, J., Chaslot , G., and Uiterwijk, J. (2007) A comparison of Monte-Carlo methods for Phantom Go, Proc. 19th Belgian–Dutch Conference on Artificial Intelligence – BNAIC, Utrecht, The Netherlands.

[8] Kocsis, L., and Szepesvari, C. (2006). Bandit based Monte-Carlo planning. 15th European Conference on Machine Learning, pp. 282-293.

[9] TCGA tournament, http://tcga.ndhu.edu.tw/tcga2011/eng/p_10.htm.

[10] Gelly, S. and Silver, D. (2011) Monte-Carlo tree search and rapid action value estimation in computer Go. Artificial Intelligence. Vol. 175, Issue 11, July 2011, Pages 1856-1875

[11] TAAI tournament, http://taai2010.nctu.edu.tw/

[12] ICGA tournament, http://www.grappa.univ-lille3.fr/icga/

[13] Chaslot, G.M.J.-B., Winands, M.H.M., Uiterwijk, J.W.H.M., van den Herik, H.J., Bouzy, B.: Progressive strategies for Monte-Carlo Tree Search. New Mathematics and Natural Computation 4(3), 343–357 (2008)