Strength Improvement and Analysis for an MCTS-Based Chinese Dark Chess Program

Chu-Hsuan Hsueh[1], I-Chen Wu[1], Wen-Jie Tseng[1], Shi-Jim Yen[2], and Jr-Chang Chen[3]

[1]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
{hsuehch,icwu,wenjie}@aigames.nctu.edu.tw

[2]Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan
sjyen@mail.ndhu.edu.tw

[3]Department of Applied Mathematics, Chung Yuan Christian University, Taoyuan, Taiwan
jcchen@cycu.edu.tw

Abstract. Monte-Carlo tree search (MCTS) has been successfully applied to Chinese dark chess (CDC). In this paper, we study how to improve and analyze the playing strength of an MCTS-based CDC program, named DARKKNIGHT, which won the CDC tournament in the 17th Computer Olympiad. We incorporate the three recent techniques, early playout terminations, implicit minimax backups, and quality-based rewards, into the program. For early playout terminations, playouts end when reaching states with likely outcomes. Implicit minimax backups use heuristic evaluations to help guide selections of MCTS. Quality-based rewards adjust rewards based on online collected information. Our experiments showed that the win rates against the original DARKKNIGHT were 60.75%, 70.90% and 59.00%, respectively for incorporating the three techniques. By incorporating all together, we obtained a win rate of 76.70%.

Keywords: Monte-Carlo tree search, Chinese dark chess, early playout terminations, implicit minimax backups, quality-based rewards.

1 Introduction

Chinese dark chess (CDC), widely played in Chinese community, is a two-player game and also a partially observable (PO) game with symmetric hidden information. The set of pieces in CDC is the same as those in Chinese chess; however, the pieces are faced down initially so both players do not know what the pieces are, until they are flipped.

In [8], the state space complexity of the game was estimated to be 10^{37} between those of Draughts and chess, while the game tree complexity was estimated to be 10^{135} between those of chess and Chinese chess without considering chance nodes. In [33], Yen *et al*. estimated the game tree complexity with chance nodes to be 10^{207} between those of Chinese chess and Shogi.

In the past, many CDC game-playing programs were developed. In the early stage, most CDC programs [8][27] were developed using *alpha-beta search*. Recently, Yen *et al*. [33] incorporated *Monte-Carlo tree search* (*MCTS*) into a CDC program, named

DIABLO, which won some tournaments [20][26][32][35]. Ones of the authors of this paper also implemented an MCTS-based CDC program, named DARKKNIGHT, which won two CDC tournaments [28][34], including that in the 17th Computer Olympiad.

This paper incorporates some recent techniques into DARKKNIGHT and analyzes how well these techniques can improve the game-playing strength. These techniques include the following:

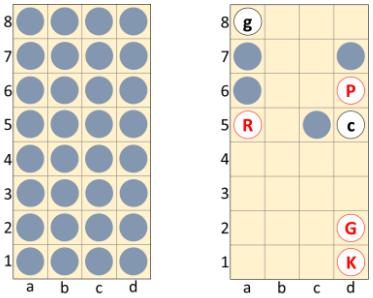
1. *Early playout terminations* [2][12][19][21][22][23][30][31]: Terminate a playout much earlier when it is very likely to win, lose, or draw.
2. *Implicit minimax backups* [19]: Guide MCTS selections by using heuristic evaluations of tree nodes together with the original simulated win rates.
3. *Quality-based rewards* [24]: Use simulation length and terminal state quality to adjust the rewards returning from simulations.

Our experiments showed that implicit minimax backups improved the most, which reached a win rate of 70.90% against the original DARKKNIGHT, serving as the baseline program. The improvements reached a win rate of 60.75% for early playout terminations, and 57.50% and 59.00% respectively when using simulation length and terminal state quality for quality-based rewards. By incorporating all together, we successfully improved DARKKNIGHT with a win rate of 76.70%.

This paper is organized as follows. Section 2 reviews the game CDC and the previous work for CDC game-playing programs, and briefly introduces the MCTS algorithm. Section 3 presents the above three techniques, including the reviews of these techniques and the incorporations into DARKKNIGHT. Section 4 shows the experimental results. Finally, Section 5 makes concluding remarks.

2 Background

2.1 CDC



Piece types			#	*Rank*
Name	Red	Black	*pieces*	
King	K	k	1	1
Guard	G	g	2	2
Minister	M	m	2	3
Rook	R	r	2	4
Knight	N	n	2	5
Cannon	C	c	2	6
Pawn	P	p	5	7

(a) (b) (c)

Fig. 1. (a) Initial board, (b) a board example and (c) piece information

CDC [8][33] is a two-player zero-sum non-deterministic game played on a 4×8 square board as illustrated in Fig. 1 (a) and (b). The two players, *Red* and *Black*, respectively own identical sets of sixteen pieces with different colors. The piece types are shown in

Fig. 1 (c). Each piece has two faces, one showing the piece type and the other showing the piece cover which is identical for all pieces. When the piece type faces up, the type is revealed and known by both players. When it faces down, it is unrevealed and unknown.

Two kinds of actions are allowed in CDC: *flipping* and *moving*. Each flipping action flips a piece, namely making the piece type revealed. Each moving action is to move a revealed piece by the player owning the piece. All pieces can be moved to empty neighboring squares (with one square up, down, left or right). Pieces except cannons can be moved to neighboring squares with capturing the opponent's pieces that have equal or lower ranks shown in Fig. 1 (c) with the following exceptions: Pawns can capture the opponent's king, but not the other way around. Cannons have a different capturing rule called *one-step-jump* in which any opponent's pieces can be captured as described in more details in [8][33].

Initially, all 32 pieces are unrevealed and placed randomly, as in Fig. 1 (a). So, the probability distribution of unrevealed pieces are all equal. The first player flips one of the 32 pieces and then owns the set of pieces of the revealed color, while the second player owns the other set. A player wins by capturing all of the opponent's pieces or making the opponent have no legal moves. The game also ends with a draw when both players play without any capturing and flipping within 40 plies, or when the same position appears three times.

2.2 Previous Work for CDC Game-Playing Programs

In the early stage, most CDC programs [8][27] were developed using alpha-beta search. Chen *et al*. published a paper [8] about alpha-beta search in their CDC program, FLIPPER. Some more research work for CDC includes opening books [7], endgame databases [9][10][25], solving smaller CDC [6] and game variations [15].

Recently, MCTS has been incorporated into CDC game-playing programs [16][17][33]. MCTS [5] is a best-first search algorithm on top of a search tree, named *UCT* [18], using Monte-Carlo simulations as state evaluations. It has been successfully applied to several games, such as Go [11][13][14], General Game Playing (GGP) [3], Backgammon [29] and Phantom-Go [4]. There are four phases in MCTS:

1. Selection. A path is traversed from the root to one of the leaves following a selection policy. A popular selection policy is based on an *upper confidence bounds* function [1]:

$$\text{argmax}_i \left\{ Q_i + C \sqrt{\frac{\ln N}{n_i}} \right\} \quad (1)$$

   where Q_i is an estimator for the value of node i, n_i the visit count of node i, N the visit count of the parent of node i and C a constant representing the weight of exploration. Commonly, Q_i is a win rate, w_i/n_i, or a mean value for rewards, where w_i is the win count of node i.
2. Expansion. One or more children are expanded from the selected leaf.
3. Playout. Following a playout policy, the game is played from the selected leaf to

some terminal state.

4. Backpropagation. The result of the playout, also called reward, is updated from terminal state back to all its ancestors in the selected path during selection.

Recently, several researchers applied MCTS to CDC. This paper briefly reviews the work in [33][34]. In [33], their program DIABLO used *non-deterministic nodes*, so-called in their article, to contain all *inner nodes* for all the possible outcomes of the flipping actions. In the rest of this paper, non-deterministic nodes are called *chance nodes*, since they are conceptually similar. Thus, inner nodes are simply children of chance nodes. The win counts and visit counts of chance nodes were the sums from all their children. In the selection phase, roulette wheel selection was applied to chance nodes to select one of children according to the probability distribution of unrevealed pieces.

Table 1. The piece weights

Piece type	K/k	G/g	M/m	R/r	N/n	C/c	P/p
Weights in [33]	5500	5000	2500	1000	800	3000	800
Weights in this paper	55	50	25	10	8	30	8

In the playout phase, they [33] used piece weights, as listed in Table 1, in their program DIABLO when capturing pieces. The program DARKKNIGHT [34] was developed independently, but it used almost the same weights as in [33], except for reducing them by a factor of 100. The program DARKKNIGHT serves as the baseline program in this paper.

3 Incorporated Techniques

3.1 Early Playout Terminations

For CDC, it is important to terminate playouts earlier for two reasons, speedup and accuracy of simulated results. The first reason is obvious. The second is that it is more accurate to terminate playouts at the time when they are very likely to win, lose, or draw. For example, it is normally a draw in a situation, Gmm, where Red has only one guard and Black has two ministers. In the case of keeping playing in the playout, Black may lose accidentally, while it is supposed to be a draw since G cannot capture either m. Thus, an early termination is actually more accurate than a continuous playout. Another example is KGmm. No matter where the pieces are placed and whether the pieces are revealed or not, Red always wins the game, since both K and G can capture m, but not *vice versa*. However, it may result in draws in playouts since Red may not be able to go to capture black pieces tactically during playouts.

In the rest of this subsection, we first review the previous work about early playout terminations and then describes our work to incorporate it into DARKKNIGHT.

Previous Work. In [12], a method was proposed to collect information online, and terminate playouts when the game obviously favors some player from the collected information. They used it to save the time for more simulations for GGP.

Some other researches were to return results after making a fixed number of moves in playouts. In [21][22][23], for Amazons, Breakthrough and Havannah, they used evaluation functions to determine the winners who the evaluated scores favor. In [2], they did a shallow minimax search with fixed depths and then returned the evaluated scores directly. Their method was tested for Othello, Breakthrough and Catch the Lion.

In [30][31], their methods for Lines of Action (LOA) were to check the scores by evaluation function every three moves, and then return the results in the following cases. If the scores exceeded some threshold, the results were wins. On the contrary, if the scores were below some threshold, the results were losses. Checking evaluation functions every three moves was for the sake of performance.

Our Work. Playouts are terminated earlier when detecting a *likely outcome* which is *win*, *loss*, or *draw*, from Red's perspective. The detection rules are based on a material combination [9][25], namely a set of remaining pieces on the board in CDC. For CDC, the total number of legal material combinations is 8,503,055 (= $(2^1 3^5 6^1)^2 - 1$). In our work, we analyzed all the combinations according to some heuristic rules. From the analysis, 750,174 are set to win, 750,174 loss, 108,136 draw, and the rest are unknown. For example, the likely outcomes for `KGmm`, `KGk` and `KGggm` are set to win, and those for `Gmm`, `CCccp` and `KPggm` are draw. Note that we call it a *likely* outcome since the outcome may not be always true in a few extreme cases as illustrated by the following example. For `KGggm`, Black can try to exchange one `g` with `G`, though it is hard to do so. If Black successfully exchanges `g` with `G` without losing `m`, then the game becomes a draw.

In playouts, if the likely outcome for a material combination is one of win, loss and draw, the playouts end immediately and return the outcomes. The overhead for the detection is little, since we can simply lookup a table to check this. So, the detection is done for every move in playouts.

3.2 Implicit Minimax Backups

This subsection reviews the previous work for implicit minimax backups and then describes our work to incorporate it into the baseline program.

Previous Work. In [19], the researchers proposed implicit minimax backups which incorporated *heuristic evaluations* into MCTS. The heuristic evaluations were used together with the estimator Q_i to guide the selections of MCTS. Namely, Q_i in Formula (1) was replaced by the following:

$$(1 - \alpha)Q_i + \alpha m_i \qquad (2)$$

where m_i is the minimax score of node i by heuristic evaluation and α is the weight of the minimax score.

In each leaf of UCT, a heuristic evaluation function was applied to compute the score,

scaled to [-1, 1] through a sigmoid function. Then, the scaled score was also backed up as classical minimax search. For consistency, Q_i was ranged in [-1, 1], too.

They improved the playing strength of the three games, Kalah, Breakthrough and LOA, and obtained win rates of around 60% to 70% in different settings of the self-play games. Their work also showed that even simple evaluation functions can lead to better performance. However, they found that this technique did not work on every game. They ran experiments on Chinese checkers and the card game Heart but obtained no significant improvement.

Our Work. For CDC, we use as the heuristic evaluation function the weighted material sum, namely the difference of the total piece weights between the player and the opponent. The piece weights are listed in Table 1. The heuristic evaluations are then scaled to [-1, 1] by a sigmoid function.

One issue to discuss is the heuristic evaluations for chance nodes, which were not mentioned in [19]. An intuitive way is to use the probability distribution for unrevealed pieces, when calculating the expected values for chance nodes. For example, suppose to have four unrevealed pieces, two `P`s, one `k` and one `m` (as in Fig. 1 (b)). The probability for `P` is 1/2, while those for the other two are 1/4. However, a problem occurs when unrevealed pieces are not in the UCT yet. For example, if `k` has not been flipped yet, the corresponding heuristic evaluation is unknown, making it hard to get the heuristic evaluations for chance nodes.

In order to solve this problem, we use the ratios of visit counts as the probabilities for all children of chance nodes. For the above example, assume to flip `P` 3 times and `m` once, but none for `k`. Then, we use 3/4 for `P` and 1/4 for `m`.

3.3 Quality-Based Rewards

This subsection reviews the previous work for quality-based rewards and then describes our work to incorporate it into the baseline program.

Previous Work. In [24], the researchers proposed new measurements to adjust the rewards obtained from simulations. Simulation length and terminal state quality were used as *quality assessments* of simulations to adjust the rewards from wins and losses.

In [24], simulation length, a domain independent quality assessment, was defined as the length of simulated game from the root to the terminal state in the simulation. Intuitively, in an advantageous position, the shorter the simulation length is, the better.

This technique maintained online sample mean and sample standard deviation of simulation length, denoted by μ_L^p and σ_L^p, for player p when p wins. With these statistical values, the reward R_L for simulation length was calculated as follows:

$$R_L = R + sign(R) \times a \times b(\lambda_L) \quad (3)$$

where R is the reward (e.g., 1 for a win and -1 for a loss), a the influence for this quality, $b(\lambda)$ a sigmoid function as shown in Formula (4) which scales the values to [-1, 1], λ_L a normalized value in Formula (5),

$$b(\lambda) = -1 + \frac{2}{1 + e^{-k\lambda}} \tag{4}$$

$$\lambda_L = \frac{\mu_L^p - l}{\sigma_L^p} \tag{5}$$

k a constant to be tuned and l is the length of that simulation.

For terminal state quality, a function describing quality of terminal states was needed. For example, for Breakthrough, the piece difference between the winning and losing players was used to measure the terminal state quality, which was scaled to [0, 1]. Let μ_T^p and σ_T^p respectively denote sample mean and sample standard deviation of terminal state quality for player p. Similarly, the rewards R_T for terminal state quality was calculated in a way similar to Formula (3) as follows.

$$R_T = R + sign(R) \times a \times b(\lambda_T) \tag{6}$$

where λ_T is a normalized value in Formula (7),

$$\lambda_T = \frac{t - \mu_T^p}{\sigma_T^p} \tag{7}$$

and t is the terminal state quality. In Formula (3) and (6), a is a constant or is calculated according to the data accumulated online. Their experimental results showed not much difference.

Our Work. We also incorporate into DARKKNIGHT the above two quality assessments, simulation length and terminal state quality, to adjust the rewards from wins and losses. For draws, the simulations are not sampled. The two quality assessments are measured when simulations end. Without early playout terminations, MCTS simulations end when one player wins. With early playout terminations, MCTS simulations also end when one obtains a likely outcome with win or loss.

At terminal states, simulation length is simply the same as the one described in [24], and terminal state quality is obtained in the following way. First, simply count the remaining pieces of the winner. Then, incorporate domain knowledge like piece weights. Namely, we use the following formula:

$$\sum_{i \in S} (1 + c_w \times \omega_i) = |S| + c_w \sum_{i \in S} \omega_i \tag{8}$$

where S is the set of remaining pieces of the winner, $|S|$ the size of S, c_w a coefficient and ω_i the weight of piece i as in Table 1. The larger the coefficient c_w is, the higher the influence of the piece weights. All values of terminal state quality are scaled to [-1, 1] according to a sigmoid function.

4 Experiments

In our experiments, each modified version played 1,000 games against the baseline, the original DARKKNIGHT. Among the 1,000 games, one played 500 games as the first and the other 500 as the second. For each game, one scored 1 point for a win, 0 for a loss and 0.5 for a draw. For a given version, the average score of 1,000 games was the win rate against the baseline.

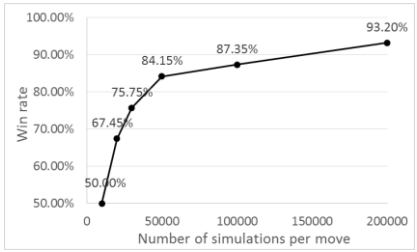


Fig. 2. Baseline with different numbers of simulations per move against that with 10,000

Initially, we performed a strength test for the baseline with different numbers of simulations per move against the one with 10,000, as shown in Fig. 2. In the rest of experiments, unless explicitly specified, we chose the one with 30,000, which was reasonable in the sense of both strength and computation time. Namely, it had a win rate of 75.75%, while it took about 3.2 minutes to run a game with one thread on machines equipped with Intel(R) Xeon(R) CPU E31225, 3.10GHz.

4.1 Experiments for Incorporating Individual Techniques

In this subsection, we incorporate techniques mentioned in Section 3 individually into the baseline to see how much they can improve.

Early Playout Terminations (EPT). As mentioned in Subsection 3.1, we used material combinations to detect whether a playout reaches a terminal state earlier with a likely outcome, one of win, loss or draw. The experimental result showed a significant improvement with a win rate of 60.75% against the baseline program. The result indicated that the accuracy of the simulated results was indeed increased with the help of the likely outcomes returned from EPT. In addition, the program with EPT also ran faster, at 32,000 simulations per second, than the one without EPT, at 27,000.

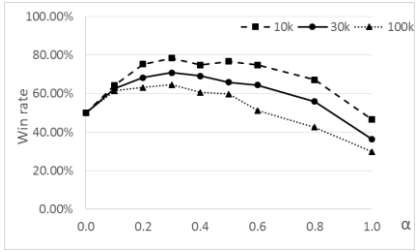


Fig. 3. Win rates for IMB with different α and different numbers of simulations per move

Implicit Minimax Backups (IMB). As mentioned in Subsection 3.2, we used heuristic evaluations to help guide the selections of MCTS more accurately. In our experiments, we tested different weights of minimax score, α, and different numbers of simulations per move. The experimental results are shown in Fig. 3 which includes three lines, respectively, for 10,000, 30,000 and 100,000 simulations per move. For fairness, the

corresponding baselines also ran the same numbers of simulations.

Fig. 3 shows that the win rates are the highest when $\alpha = 0.3$ for the three lines, and 78.45% for the one with 10,000 simulations per move, 70.90% for 30,000 and 64.60% for 100,000. The figure shows that IMB did significantly improve the playing strength. In the case that α was too high, the win rates went down for the following reason. The heuristic evaluations weighted too much higher than online estimation which is usually more accurate than heuristic evaluations for a sufficiently large number of simulations. On the other hand, in the case that α was too low, the win rates also went down for the following reason. Less heuristic information was used to help guide the selections of MCTS accurately, since short-term tactical information provided by the minimax scores was missing, as explained in [19].

Fig. 3 also has the following implication. For a higher number of simulations per move, the improvement was relatively smaller. This hints that IMB has less improvement for a sufficiently large number of simulations per move. The reason is: the help of minimax scores decreases, since simulated results become more accurate with more simulations.

In [19], they mentioned that IMB had no significant improvement for Heart, also a PO game. Interestingly, our results show that IMB did significantly improve the playing strength for CDC.

Quality-Based Rewards. As mentioned in Subsection 3.3, we used simulation length (SL) and terminal state quality (TSQ) to adjust the rewards from simulations to favor those with shorter length and higher terminal state quality.

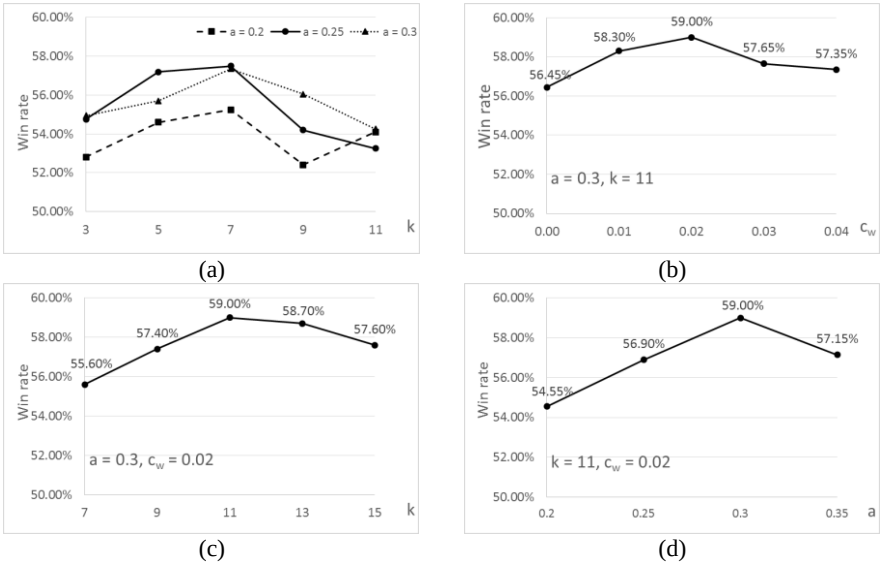


Fig. 4. The win rates for (a) SL, and for TSQ with different (b) c_w, (c) k and (d) a

For SL, we tested two parameters, the influence, a, and the sigmoid function constant, k, and obtained the results as shown in Fig. 4 (a). The highest win rate was

57.50% when $a = 0.25$ and $k = 7$.

For TSQ, we needed one more parameter, c_w, a coefficient of piece weights for measuring terminal state quality. In our experiments, we obtained the highest win rate of 59.00%, when $a = 0.3$, $k = 11$ and $c_w = 0.02$. By fixing $a = 0.3$ and $k = 11$, we obtained the highest at $c_w = 0.02$, slightly better than the one at $c_w = 0.01$, as in Fig. 4 (b). Similarly, by fixing a and c_w, we obtained the highest at $k = 11$ as in Fig. 4 (c). By fixing k and c_w, we obtained the highest win rate at $a = 0.3$ as shown in Fig. 4 (d).

4.2 Combinations of Techniques

In this subsection, we further improved the playing strength by combing the above techniques, each of which uses the best settings from the experimental results in the previous subsection. Table 2 (below) first lists the best results for incorporating each individual technique, EPT, IMB, SL and TSQ (from the previous subsection). Then, we combined both EPT and IMB first, denoted by EPT+IMB, since these two techniques improved the most when used alone. The result shows a win rate of 74.75% for EPT+IMB.

Table 2. The best win rates of EPT, IMB, SL, TSQ and their combinations

	EPT	IMB	SL	TSQ	EPT +IMB	EPT +IMB +SL	EPT +IMB +TSQ	EPT +IMB +SL+TSQ
Win rate	60.75%	70.90%	57.50%	59.00%	74.75%	76.00%	75.95%	76.70%

Furthermore, we incorporated SL, TSQ, and SL+TSQ respectively into the above EPT+IMB. Their win rates are shown in Table 2. By incorporating all techniques together, the win rates reached up to 76.70%.

5 Conclusion

To our best knowledge, this paper is the first attempt to incorporate the three techniques, early playout terminations (EPT), implicit minimax backups (IMB) and quality-based rewards (SL and TSQ) together and obtained significant improvement. We demonstrate this through an MCTS-based CDC game-playing program, DARKKNIGHT. Our experiments showed that all of these techniques did significantly improve the playing strength with win rates of 60.75%, 70.90%, 57.50% and 59.00% against the original DARKKNIGHT when incorporating EPT, IMB, SL and TSQ, respectively. By incorporating all together, the win rate reached up to 76.70%. The results demonstrated the effectiveness of the above techniques for an MCTS-based CDC program. Besides, the enhanced DARKKNIGHT, with more fine tunings, won the CDC tournament in the 18th Computer Olympiad.

Acknowledgements. The authors would like to thank the Ministry of Science and Technology of the Republic of China (Taiwan) for financial support of this research under contract numbers NSC 102-2221-E-009-080-MY2 and NSC 102-2221-E-009-069-MY2.

References

[1] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-Time Analysis of the Multiarmed Bandit Problem. Machine Learning 47(2-3), 235-256 (2002)
[2] Baier, H., Winands, M.H.M.: Monte-Carlo Tree Search and Minimax Hybrids with Heuristic Evaluation Functions. In: Cazenave, T., Winands, M.H.M., Björnsson, Y. (eds.) Computer Games, vol. 504, pp. 45-63. Springer International Publishing (2014)
[3] Björnsson, Y., Finnsson, H.: CadiaPlayer: A Simulation-Based General Game Player. IEEE Transactions on Computational Intelligence and AI in Games 1(1), 4-15 (2009)
[4] Borsboom, J., Saito, J.-T., Chaslot, G., Uiterwijk, J.: A Comparison of Monte-Carlo Methods for Phantom Go. In: Proc. BeNeLux Conf. Artif. Intell., pp. 57-64. Utrecht, Netherlands (2007)
[5] Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A Survey of Monte Carlo Tree Search Methods. IEEE Transactions on Computational Intelligence and AI in Games 4(1), 1-43 (2012)
[6] Chang, H.-J., Hsu, T.-S.: A Quantitative Study of 2 × 4 Chinese Dark Chess. In: van den Herik, H.J., Iida, H., Plaat, A. (eds.) Computers and Games, pp. 151-162. Springer International Publishing (2014)
[7] Chen, B.-N., Hsu, T.-S.: Automatic Generation of Opening Books for Dark Chess. In: van den Herik, H.J., Iida, H., Plaat, A. (eds.) Computers and Games, pp. 221-232. Springer International Publishing (2014)
[8] Chen, B.-N., Shen, B.-J., Hsu, T.-S.: Chinese Dark Chess. ICGA Journal 33(2), 93-106 (2010)
[9] Chen, J.-C., Lin, T.-Y., Chen, B.-N., Hsu, T.-S.: Equivalence Classes in Chinese Dark Chess Endgames. IEEE Transactions on Computational Intelligence and AI in Games PP, 1-1 (2014)
[10] Chen, J.-C., Lin, T.-Y., Hsu, S.-C., Hsu, T.-S.: Design and Implementation of Computer Chinese Dark Chess Endgame Database. In: Proceeding of TCGA Workshop 2012, pp. 5-9, Hualien, Taiwan (2012) (in Chinese)
[11] Enzenberger, M., Müller, M., Arneson, B., Segal, R.: Fuego: An Open-Source Framework for Board Games and Go Engine Based on Monte Carlo Tree Search. IEEE Transactions on Computational Intelligence and AI in Games 2(4), 259-270 (2010)
[12] Finnsson, H.: Generalized Monte-Carlo Tree Search Extensions for General Game Playing. In: The Twenty-Sixth AAAI Conference on Artificial Intelligence, pp. 1550–1556, Toronto, Canada (2012)
[13] Gelly, S., Silver, D.: Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go. Artificial Intelligence 175(11), 1856-1875 (2011)
[14] Gelly, S., Wang, Y., Munos, R., Teytaud, O.: Modification of UCT with Patterns in Monte-Carlo Go. Tech. rep., HAL - CCSd - CNRS, France (2006)
[15] Jouandeau, N.: Varying Complexity in CHINESE DARK CHESS Stochastic Game. In: Proceeding of TCGA Workshop 2014, pp. 86, Taipei, Taiwan (2014)
[16] Jouandeau, N., Cazenave, T.: Monte-Carlo Tree Reductions for Stochastic Games. In: Cheng, S.-M., Day, M.-Y. (eds.) Technologies and Applications of Artificial Intelligence, vol. 8916, pp. 228-238. Springer International Publishing (2014)
[17] Jouandeau, N., Cazenave, T.: Small and Large MCTS Playouts Applied to Chinese Dark Chess Stochastic Game. In: Cazenave, T., Winands, M.H.M., Björnsson, Y. (eds.) Computer Games, vol. 504, pp. 78-89. Springer International Publishing (2014)
[18] Kocsis, L., Szepesvári, C.: Bandit Based Monte-Carlo Planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) Machine Learning: ECML 2006, vol. 4212, pp. 282-293. Springer Berlin Heidelberg (2006)
[19] Lanctot, M., Winands, M.H.M., Pepels, T., Sturtevant, N.R.: Monte Carlo Tree Search with Heuristic Evaluations Using Implicit Minimax Backups. In: 2014 IEEE Conference on Computational Intelligence and Games, CIG 2014. pp. 1-8 (2014)
[20] Lin, Y.-S., Wu, I.-C., Yen, S.-J.: TAAI 2011 Computer-Game Tournaments. ICGA Journal 34(4), 248-250 (2011)

[21] Lorentz, R.: Amazons Discover Monte-Carlo. In: van den Herik, H.J., Xu, X., Ma, Z., Winands, M.H.M. (eds.) Computers and Games, vol. 5131, pp. 13-24. Springer Berlin Heidelberg (2008)
[22] Lorentz, R.: Early Playout Termination in MCTS. In the 14th conference on Advances in Computer Games (ACG2015), Leiden, the Netherlands (2015)
[23] Lorentz, R., Horey, T.: Programming Breakthrough. In: van den Herik, H.J., Iida, H., Plaat, A. (eds.) Computers and Games, pp. 49-59. Springer International Publishing (2014)
[24] Pepels, T., Tak, M.J., Lanctot, M., Winands, M.H.M.: Quality-Based Rewards for Monte-Carlo Tree Search Simulations. In: 21st European Conf. on Artif. Intell., Prague, Czech Republic (2014)
[25] Saffidine, A., Jouandeau, N., Buron, C., Cazenave, T.: Material Symmetry to Partition Endgame Tables. In: van den Herik, H.J., Iida, H., Plaat, A. (eds.) Computers and Games, pp. 187-198. Springer International Publishing (2014)
[26] Su, T.-C., Yen, S.-J., Chen, J.-C., Wu, I.-C.: TAAI 2012 Computer Game Tournaments. ICGA Journal 37(1), 33-35 (2014)
[27] Theory of Computer Games, a course in National Taiwan University taught by Tsu, T.-S., available at http://www.iis.sinica.edu.tw/~tshsu/tcg/index.html
[28] Tseng, W.-J., Chen, J.-C., Chen, L.-P., Yen, S.-J., Wu, I.-C.: TCGA 2013 Computer Game Tournament Report. ICGA Journal 36(3), 166-168 (2013)
[29] Van Lishout, F., Chaslot, G., Uiterwijk, J.W.: Monte-Carlo Tree Search in Backgammon. In: Computer Games Workshop, pp. 175-184, Amsterdam, the Netherlands (2007)
[30] Winands, M.H.M., Björnsson, Y., Saito, J.-T.: Monte Carlo Tree Search in Lines of Action. IEEE Transactions on Computational Intelligence and AI in Games 2(4), 239-250 (2010)
[31] Winands, M.H.M., Björnsson, Y., Saito, J.-T.: Monte-Carlo Tree Search Solver. In: van den Herik, H.J., Xu, X., Ma, Z., Winands, M.H.M. (eds.) Computers and Games, vol. 5131, pp. 25-36. Springer Berlin Heidelberg (2008)
[32] Yang, J.-K., Su, T.-C., Wu, I.-C.: TCGA 2012 Computer Game Tournament Report. ICGA Journal 35(3), 178-180 (2012)
[33] Yen, S.-J., Chou, C.-W., Chen, J.-C., Wu, I.-C., Kao, K.-Y.: Design and Implementation of Chinese Dark Chess Programs. IEEE Transactions on Computational Intelligence and AI in Games 7(1), 1-9 (2015)
[34] Yen, S.-J., Chen, J.-C., Chen, B.-N., Tseng, W.-J.: DarkKnight Wins Chinese Dark Chess Tournament. ICGA Journal 36(3), 175-176 (2013)
[35] Yen, S.-J., Su, T.-C., Wu, I.-C.: The TCGA 2011 Computer-Games Tournament. ICGA Journal 34(2), 108-110 (2011)