# An Investigation of Strength Analysis Metrics for Game-Playing Programs: A Case Study in Chinese Dark Chess

Chu-Hsuan Hsueh [a], I-Chen Wu [a,*], Tsan-sheng Hsu [b], and Jr-Chang Chen [c]

[a] *Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan*
*Emails: hsuehch@aigames.nctu.edu.tw, icwu@aigames.nctu.edu.tw*
[b] *Institute of Information Science, Academia Sinica, Taipei, Taiwan*
*Email: tshsu@iis.sinica.edu.tw*
[c] *Department of Computer Science and Information Engineering, National Taipei University, New Taipei City, Taiwan*
*Email: jcchen@mail.ntpu.edu.tw*

**Abstract**. For developers of game-playing programs, it is an important but non-trivial task to measure the strengths of the programs, especially with respect to the theoretically optimal players. The optimal players are hard to obtain in practice since many real-world games are too hard to solve entirely. In this paper, a reduced version of Chinese dark chess (CDC), 2×4 CDC, which has been solved, is used as a testbed to analyze the strengths of game-playing programs for CDC, a non-deterministic game. Absolute strength is then defined to be the win rates against the optimal player. Experiment results show that win rates against designated baseline programs are a good metric, since these win rates are highly correlated to absolute strengths. The stronger the designated baseline programs are, the higher the correlations of win rates are to the absolute strengths. In addition, it is shown that prediction rates and mean squared errors are other good metrics and have high correlations to the absolute strengths. The experiment results also show that the win rates obtained in 2×4 CDC have a high correlation to those in the original 4×8 CDC, except for those using the technique of progressive bias.

Keywords: Chinese dark chess, strength analysis, optimal player, game-playing program

## 1. INTRODUCTION

When developing game-playing programs, one important thing is to measure the strengths of the programs. Assuming that an *optimal player* exists, a program's *absolute strength* can be measured by the win rate playing against the optimal player. However, it is impractical to expect access to such optimal players in many real-world games since these games are too complex to solve entirely (Kishimoto & Mueller, 2015; van den Herik, Uiterwijk, & van Rijswijck, 2002). Thus, for the strength analysis of game-playing programs of these games in practice, many researchers used win rates as well as some other metrics such as prediction rates and mean squared errors to measure their strengths.

The first goal of this paper is to analyze the correlation between these practically measured strengths and absolute strengths for the game of Chinese dark chess (CDC), a non-deterministic game normally played on a 4×8 board, as a case study. A strong correlation implies that the practically measured strengths can be used to represent the absolute strengths. The analysis was done on the solved 2×4 CDC (Chang & Hsu, 2014), a reduced version of CDC. The experiment results showed that for MCTS-based programs that utilized techniques such as early playout terminations, implicit minimax backups, quality-based rewards, and progressive bias (Hsueh, Wu, Tseng, Yen, & Chen, 2016), the win rates playing against available baseline players indeed had high correlations to their absolute strengths in 2×4 CDC. The stronger the designated baseline player (with respect to the optimal player), the higher the correlation coefficient was. Thus, it can be concluded that the win rates playing against a reasonably strong player can reflect the absolute strengths. The experiments also showed that prediction rates to expert moves and mean squared errors of values of positions were also good metrics of the programs' strengths, though these metrics were less accurate compared with win rates against designated players.

The second goal is to investigate whether the win rates obtained in reduced and small-sized games can be used to predict those in the full-sized games, since games with smaller sizes usually share similar properties to the

* Corresponding author. E-mail: icwu@aigames.nctu.edu.tw

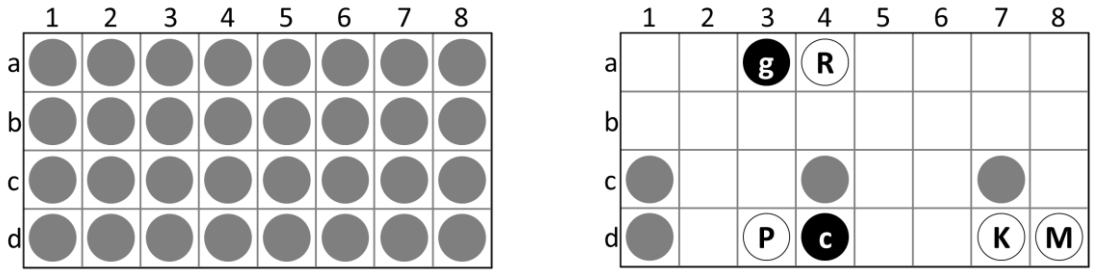
original ones. A strong correlation in this case implies that new algorithms can be tested in small-sized games instead of full-sized games to save time. The experiment results showed that most of the win rates of the MCTS-based programs (Hsueh *et al*., 2016) obtained in 2×4 CDC can be used to predict those in 4×8 CDC, except for the variations that used the technique of progressive bias.

The rest of the paper is structured as follows. Section 2 describes background knowledge including the game of CDC, the reduced 2×4 CDC and how it was solved, and Monte-Carlo tree search. Section 3 illustrates in more detail three metrics: win rates against a designated player, prediction rates to expert moves, and mean squared errors of values of positions. Section 4 then contains the experiment results and discussions on the three metrics in 2×4 CDC. Section 5 analyzes projecting the win rates obtained in 2×4 CDC to those in 4×8 CDC. Finally, Section 6 makes concluding remarks. Detailed experiment results are included in Appendix A, and simple linear regression is described in Appendix B.

## 2. BACKGROUND

### 2.1 Chinese dark chess (CDC)

Chinese dark chess (CDC) (B.-N. Chen, Shen, & Hsu, 2010; Hsueh *et al*., 2016; Yen, Chou, Chen, Wu, & Kao, 2015) is a two-player zero-sum non-deterministic game which uses the same set of pieces as Chinese chess. The two players, red and black, respectively own one king (`K`/`k`), two guards (`G`/`g`), two ministers (`M`/`m`), two rooks (`R`/`r`), two knights (`N`/`n`), two cannons (`C`/`c`), and five pawns (`P`/`p`). The uppercase and lowercase letters are abbreviations for red and black pieces respectively.




(a) (b)

Fig. 1. (a) The initial position and (b) an example of position for CDC.

The game is played on a 4×8 square board. At the start of the game, all 32 pieces are randomly shuffled, then placed face-down in the squares, as shown in Fig. 1 (a). Pieces are referred to as *unrevealed* pieces if they are faced-down, or *revealed* pieces when they are facing up. The first player begins by revealing any unrevealed piece. The revealed piece, and the set of pieces with the same color as this piece, is now owned by the first player. The second player owns the other set. Information conveyed by the revealed pieces are observable by both players. Both players in their own turn can either reveal an unrevealed piece, which is called a *flipping action*, or move one of their own revealed pieces, which is called a *moving action*. Note that it is possible for a player to flip and reveal enemy pieces, which the player's opponent can move immediately in the proceeding turn. A flipping action is denoted by S(?) where S is the source square. For example, c4(?) indicates a flipping action at c4 in Fig. 1 (b). A moving action is denoted by S-D where S and D are source and destination squares.

All pieces can be moved to any of its adjacent empty squares. Pieces other than cannons can be moved to an adjacent square to capture an opponent piece. Whether a piece can capture another piece depends on its and its opponent piece's ranks, which, from the highest to the lowest, are kings, guards, ministers, rooks, knights, cannons, and pawns. Generally, pieces can capture opponent pieces with equal or lower ranks. Three exceptions are illustrated as follows. First, kings cannot capture pawns. Second, pawns can capture kings. Third, cannons can capture all types of opponent pieces but need to jump over exactly one piece in the same row or the same column, which is called *one-step-jump*. An example for one-step-jump is shown in Fig. 1 (b), where the black cannon (`c`) at d4 can capture the red rook (`R`) at a4 or the red minister (`M`) at d8. Finally, when one player captures all the opponent pieces or when the opponent has no legal move, the player wins. When both players neither capture nor reveal any piece within 40 moves or repeat the same position three times, the game ends as a draw.

CDC is one of the games played in the Computer Olympiad since 2010 (Hsueh & Wu, 2015; Tseng, Chen, & Wu, 2017; van den Herik, Plaat, & Hellemons, 2012; Yen, Chen, Chen, & Tseng, 2013; Yen, Chiu, & Wu, 2010). Many programs were developed based on alpha-beta search (Chang, Fan, Chen, Hsueh, & Hsu, 2017; B.-N. Chen *et al*., 2010; J.-C. Chen, Fan, Chang, & Hsu, 2018; Yen *et al*., 2010). MCTS was first applied to CDC by Yen *et al*. (2015). Hsueh *et al*. (2016) further improved an MCTS-based game-playing program by incorporating four techniques, described in more detail in Subsection 2.3.

## 2.2 The solved 2×4 CDC

A reduced version of CDC, 2×4 CDC, has been devised and solved by Chang and Hsu (2014). The major difference between 2×4 CDC and the original version lies in the board size and the set of used pieces. In addition to the board size, the used pieces in 2×4 CDC only considered 24 *fair* and *equivalent* material combinations (combinations of pieces) as listed in Table 1. By fair, both players have the same set of pieces at the beginning of a game. For example, "KGGM vs. kggr" is not listed in Table 1 since two players have different sets of pieces. By equivalent (J.-C. Chen, Lin, Chen, & Hsu, 2015), material combinations with the same relations between pieces are only counted once. For example, "KGGR vs. kggr" is not listed in Table 1 since it has the same piece relations as "KGGM vs. kggm."

Table 1
24 fair and equivalent material combinations in 2×4 CDC

| | | | |
|---|---|---|---|
| KGGM vs. kggm | KGGC vs. kggc | KGGP vs. kggp | KGMM vs. kgmm |
| KGMR vs. kgmr | KGMC vs. kgmc | KGMP vs. kgmp | KGCC vs. kgcc |
| KGCP vs. kgcp | KCCP vs. kccp | KCPP vs. kcpp | KPPP vs. kppp |
| GGMM vs. ggmm | GGMR vs. ggmr | GGMC vs. ggmc | GGCC vs. ggcc |
| GGCP vs. ggcp | GMCP vs. gmcp | GCCP vs. gccp | GCPP vs. gcpp |
| GPPP vs. gppp | CCPP vs. ccpp | CPPP vs. cppp | PPPP vs. pppp |


(a) (b)

Fig. 2. Examples of positions in 2×4 CDC from a game of "KGCP vs. kgcp" (a) without and (b) with chance events.

All legal positions were assigned *theoretical values* from retrograde analysis (Chang & Hsu, 2014). In their work, expected win rates were used as theoretical values. If a position does not contain any unrevealed piece or only one unrevealed piece type remains, the game becomes deterministic. Thus, the theoretical value is 1, 0, or -1 which represents a win, a draw, or a loss respectively from the view point of the player to move. An example position is shown in Fig. 2 (a), where the result is a win for the black player. Since CDC is actually a non-deterministic game – a flipping action with different piece types may result in different game outcomes – for each position, the theoretical value ranges from -1 to 1 where -1 and 1 were 100% losses and 100% wins respectively from the view point of the player to move. For example, assuming that it is the red player's turn in the position in Fig. 2 (b), the theoretical value is 0.0958 according to Chang and Hsu (2014), which shows a slight advantage toward the red player. Note that a theoretical value of 0 does not represent a necessary draw, it may also represent the situation with 50% wins and 50% losses.

## 2.3 Monte-Carlo tree search (MCTS)

Monte-Carlo tree search (MCTS) (Browne *et al*., 2012) is a kind of best-first search where states are evaluated by Monte Carlo simulations. One popular implementation in practice is *upper confidence bounds applied to trees* (UCT) (Kocsis & Szepesvári, 2006) which applies *upper confidence bounds* (UCB) (Auer, Cesa-Bianchi, & Fischer, 2002) at selection phase, one of the four phases in MCTS. The UCB function is shown as follows:

$$UCB_i = Q_i + C\sqrt{\frac{\ln N}{n_i}} \quad (1)$$

where $Q_i$ is the estimated value of node $i$ which is usually a win rate or a mean value of rewards for the position corresponding to node $i$, $C$ the coefficient balancing exploration and exploitation, $N$ the visit count of the parent of node $i$, and $n_i$ the visit count of node $i$.

Table 2
Piece weights for CDC used by Hsueh *et al*. (2016)

| **Piece type** | K/k | G/g | M/m | R/r | N/n | C/c | P/p |
|---|---|---|---|---|---|---|---|
| **Weight** | 55 | 50 | 25 | 10 | 8 | 30 | 8 |

To generate a move, MCTS usually consists of several iterations (or simulations). Each iteration contains four phases, selection, expansion, playout, and backpropagation. The selection phase follows the above UCB formula to traverse UCT until a leaf is reached. During the expansion phase, one or more children from the leaf is expanded. A playout policy is then used to play the game to the end, obtaining a game outcome in the process. Backpropagation then refers to the process of updating this outcome of the playout to all nodes traversed during selection. As for playout policies, the simplest one is to randomly select a move. More sophisticated playout policies were used to make the playouts cleverer for CDC (Hsueh *et al*., 2016; Yen *et al*., 2015). They used piece weights, as an example listed in Table 2, to calculate scores for moves when captures and/or escapes happened. Roulette-wheel selections were then applied to select moves according to their scores.

Hsueh *et al*. (2016) further incorporated four techniques into an MCTS-based CDC program to improve the playing strength. The four techniques were early playout terminations, implicit minimax backups, quality-based rewards, and progressive bias, respectively introduced in the following four subsections.

### *2.3.1 Early playout terminations*

The technique of early playout terminations (EPT) terminates playouts before the end of games. For CDC, Hsueh *et al*. (2016) terminated playouts immediately and returned the corresponding outcomes whenever a material combination of a position is very likely to win (e.g., the player has a king and a guard while the opponent has only two ministers), or very likely to draw (e.g., the player has only one guard while the opponent has two ministers). In their article, all material combinations were analyzed manually for EPT, and labeled as *likely wins*, *likely draws*, *likely losses*, or *unknown*.

The advantages of EPT were to speed up and make the outcomes of playouts more accurate. In their experiments, they also reversed the outcomes for those positions with likely wins and likely losses with a probability of $\varepsilon$ to show the importance of the accuracy of the outcomes of playouts. Clearly, when $\varepsilon$ was 0, EPT performed best and obtained a win rate of 60.75% (±2.76%) against the original program. Note that these programs were run with 30,000 simulations per move, and the same for all experiments mentioned in the rest of this section.

### *2.3.2 Implicit minimax backups*

The technique of implicit minimax backups (IMB) was proposed by Lanctot, Winands, Pepels, and Sturtevant (2014). The objective was to help MCTS deal with tactical positions by introducing minimax scores from heuristic evaluations of positions. The $Q_i$ in the UCB formula (Formula (1)) was modified as

$$Q_i^{IMB} = (1-\alpha)Q_i + \alpha m_i \quad (2)$$

where $m_i$ is the minimax score of node $i$ which is mapped to the same range of $Q_i$, and $\alpha$ the weight of the minimax score.

The heuristic evaluation function for CDC used by Hsueh *et al*. (2016) was the weighted material sum, namely the difference of the total piece weights between the player and the opponent. The piece weights were also the ones in Table 2. The minimax scores of chance nodes were weighted according to visit counts instead of

probabilities of unrevealed pieces. The experiment results showed that with $\alpha = 0.3$, IMB had a win rate of 70.90% (±2.42%) against the original program.

### *2.3.3 Quality-based rewards*

The technique of quality-based rewards was proposed by Pepels, Tak, Lanctot, and Winands (2014). The rewards of wins and losses obtained in simulations were adjusted according to simulation lengths (SL) or terminal state qualities (TSQ) collected online. Simulation length was the length of the game in the simulation from the root to the end. Terminal state quality used by Hsueh *et al.* (2016) for CDC was the remaining number of pieces with considering piece weights,

$$|S| + c_w \sum_{i \in S} \omega_i \tag{3}$$

where $S$ is the set of remaining pieces of the winner, $|S|$ the number of remaining pieces of the winner, $c_w$ the coefficient representing the importance of piece weights, and $\omega_i$ the weight of piece $i$ as listed in Table 2. The adjusted rewards were then calculated by

$$R^{new} = R + \text{sign}(R) \times a \times b(\lambda) \tag{4}$$

where $R$ is the original reward (1 for a win and -1 for a loss), $a$ the coefficient deciding the largest range of adjustment, and $b(\lambda) = -1 + 2/(1 + e^{-k\lambda})$ a function scaling the value of $\lambda$ to $[-1, 1]$ where $k$ is a parameter. In Formula (4), $\lambda$, normalized values, for SL and TSQ are $\lambda_{SL} = (\mu_{SL}^p - l)/\sigma_{SL}^p$ and $\lambda_{TSQ} = (t - \mu_{TSQ}^p)/\sigma_{TSQ}^p$ respectively where $\mu_{SL}^p$, $\sigma_{SL}^p$, $\mu_{TSQ}^p$ and $\sigma_{TSQ}^p$ are sample means and sample standard deviations of simulation lengths and terminal state qualities for player $p$, and $l$ and $t$ the length and the terminal state quality of the simulation. The experiments by Hsueh *et al.* (2016) showed that SL ($a = 0.25$ and $k = 7$) and TSQ ($a = 0.3$, $k = 11$ and $c_w = 0.02$) had win rates of 57.50% (±2.77%) and 59.00% (±2.78%) against the original program.

### *2.3.4 Progressive bias*

The technique of progressive bias (PB) was proposed by several researchers with different forms (Chaslot, Winands, van den Herik, Uiterwijk, & Bouzy, 2008; Ikeda & Viennot, 2014; Nijssen & Winands, 2011). The idea was to use available heuristics to guide selections among nodes with few simulations. Hsueh *et al.* (2016) extended the UCB formula (Formula (1)) to:

$$Q_i + C\sqrt{\frac{\ln N}{n_i}} + c_p \times \frac{H_i}{count_i} \tag{5}$$

where $H_i$ is the heuristic score of the move to node $i$, $c_p$ the coefficient representing the influence of the heuristics, and $count_i$ a counter corresponding to node $i$. Hsueh *et al.* (2016) experimented six forms of $count_i$, including $n_i$, $\sqrt{n_i}$, $\ln(n_i + 1)$, $l_i$, $\sqrt{l_i}$, and $\ln(l_i + 1)$ where $l_i$ is the loss count of node $i$.

The heuristic scores of moves designed by Hsueh *et al.* (2016) for CDC used some basic heuristics including capturing opponent pieces, escaping the player's pieces, and suiciding the player's pieces. These basic heuristics are collectively referred to in the remainder of this paper as $\mathbf{H^B}$. In addition, they also used advanced heuristics including curbing opponent pieces, which will be referred to as $\mathbf{H^C}$, and calculating scores for flipping actions, referred to as $\mathbf{H^F}$. The heuristics were experimented incrementally as follows, denoted by $\mathbf{H^B}$, $\mathbf{H^B + H^C}$, and $\mathbf{H^B + H^C + H^F}$. The experiment results showed that with $\mathbf{H^B + H^C + H^F}$ and the settings, $count_i = \sqrt{n_i}$ and $c_p = 0.1$, PB had a win rate of 82.10% (±1.98%) against the original program.

## 3. INVESTIGATED STRENGTH ANALYSIS METRICS

### 3.1 Win rates playing against a designated player

For game-playing programs, win rates against designated players are commonly used as a metric to measure the strengths of the programs. It is usually claimed that the higher the win rate obtained, the stronger the program is (Tesauro, 1995). Playing against human players is also a way (Campbell, Hoane, & Hsu, 2002; Silver *et al*., 2016; Silver *et al*., 2017; Tesauro, 1995); however, since it is too slow to obtain statistically significant results (Tesauro, 1995), usually, the designated players are computer programs, often called *baseline programs*.

Baseline programs in practice can be a random player (randomly make a move), an early version of the tested program, or other available programs. Usually, random players are weak and are easily defeated. In the past, when demonstrating the strengths of new algorithms or heuristics, usually they were made to play against early versions of the tested programs (Chaslot *et al*., 2008; B.-N. Chen *et al*., 2010; Gelly & Silver, 2007; Hsueh *et al*., 2016; Lanctot *et al*., 2014; Nijssen & Winands, 2011; Pepels *et al*., 2014; Runarsson & Lucas, 2014; Silver *et al*., 2016; Silver *et al*., 2017; Tian & Zhu, 2016; Yen *et al*., 2015). Sometimes, they were made to play against other openly available programs (Chaslot *et al*., 2008; Coulom, 2007; Gelly & Silver, 2007; Ikeda & Viennot, 2014; Lanctot *et al*., 2014; Silver *et al*., 2016; Silver *et al*., 2017; Tian & Zhu, 2016; Yen *et al*., 2015).

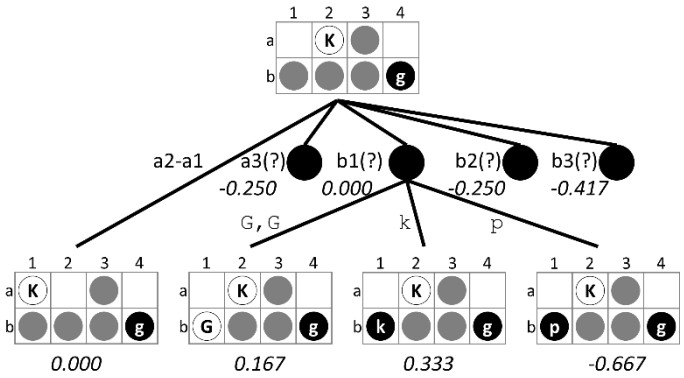



Fig. 3. A partial tree with theoretical values of positions in the red player's view from a game of "KGGP vs. kggp" where one of the two g's as well as the P are captured and it is the red player's turn at the root position.

However, it is still a question whether and how much the win rates can reflect the absolute strengths of the tested programs. In the solved 2×4 CDC, since theoretical values of all positions are known, there exists an optimal player which always selects *best moves*, defined as the moves leading to the highest theoretical value. This optimal player can serve as the baseline program. More specifically, theoretical values of moves are determined as follows. For a moving action, the theoretical value directly comes from the position after moving. For a flipping action, the theoretical value is the weighted average for unrevealed pieces based on the probability distribution of these unrevealed pieces. For example, the move b1(?) in Fig. 3 has a theoretical value of 0.000 which is the weighted average of the theoretical values of the three possibilities G ($0.167 \times 2/4$), k ($0.333 \times 1/4$), and p ($-0.667 \times 1/4$).

Note that, for deterministic positions with wins or losses, the theoretical values did not provide information of distances to win or loss (Chang & Hsu, 2014). Thus, the optimal player does not take into consideration shortest wins and longest losses. In addition, the optimal player does not take into consideration whether a move is risky or not. For the example in Fig. 3, the move b1(?) has a theoretical value of 0.000 from the red player's view, which is as good as the move a2-a1. Thus, the optimal player just randomly chooses from the two moves. However, the move b1(?) is riskier since the red player has a probability of 1/4 to flip the p at b1 which will lead to a highly disadvantageous situation.

### 3.2 Prediction rates to expert moves

Assuming that a set of positions with moves played by experts (optimal players at best) are available, *prediction rates* to those expert moves are commonly used to analyze and measure strengths of programs (Chang, Hsueh, & Hsu, 2015; Coulom, 2007; Lai, 2015; Liskowski, Jaśkowski, & Krawiec, 2018; Runarsson & Lucas, 2014;

Silver *et al*., 2016; Silver *et al*., 2017; Tian & Zhu, 2016). The prediction rate of a program is the probability that the moves selected by the program match expert moves. For example, moves from game records played by human players were used as expert moves in the game of Go (Coulom, 2007; Silver *et al*., 2016; Silver *et al*., 2017; Tian & Zhu, 2016) and Othello (Liskowski *et al*., 2018; Runarsson & Lucas, 2014). In some cases, the moves generated by a program with much deeper search also served as expert moves. For example, Lai (2015) used the moves from a program with time-limited searches as expert moves in chess.

In 2×4 CDC, since theoretical values of all positions are known, best moves can be used to derive the prediction rate. It is possible that a position has more than one best move. An example is shown in Fig. 3 where the moves a2-a1 and b1(?) are all best moves. In this case, selecting any of these moves is said to match the best moves. As explained in Subsection 3.1, the theoretical values cannot distinguish shorter wins or longer losses from wins or losses. Thus, if more than one move leads to a win, all are considered as best moves. Prediction rates of best moves were measured by Chang *et al*. (2015) for a basic MCTS program in 2×4 CDC. However, only positions within the first two moves of a game were tested.

If a program has a high prediction rate to the expert moves, it can be said that the strength of the program is close to the experts' and thus can be inferred as a strong program. However, any moves not matching expert moves do not necessarily indicate sub-optimal plays for the following two reasons. First, the expert moves (from human players or programs) may not be the theoretically best moves. Second, it is possible that there exist several equally good moves given a position, but only the one selected by the expert is counted as an expert move. An example that stronger programs have lower prediction rates is shown by Silver *et al*. (2017) for the game of Go, where AlphaGo Zero had much higher win rates but lower prediction rates to moves by professional players (cf. Figures 3 and 4 by Silver *et al*. (2017)).

### 3.3 Mean squared errors of values of positions

Assuming that a set of positions with *ground-truth values* (theoretical values under best circumstances) are available, *mean squared errors* between the given ground-truth values and the values estimated by game-playing programs can be used to analyze the strengths of the programs (Gelly & Silver, 2007; Silver *et al*., 2016; Silver *et al*., 2017). The mean squared error of a program on a given set of positions $P$ and the corresponding ground-truth values $\{v_p\}_{p\in P}$ is calculated as

$$MSE = \frac{1}{|P|}\sum_{p\in P}\left(\hat{v}_p - v_p\right)^2 \tag{6}$$

where $|P|$ is the number of positions in $P$ and $\hat{v}_p$ the value of position $p$ estimated by the program.

In 2×4 CDC, theoretical values of all positions are known and can be used to serve as the ground-truth values. However, since it is hard for many games to obtain their theoretical values, usually the outcomes of the games played by experts or reasonably strong programs were used instead (Silver *et al*., 2016; Silver *et al*., 2017; Takeuchi, Kaneko, & Yamaguchi, 2010). Using the outcomes of the games meant that $v_p$ for all positions in a game were set to the outcome of the game. In addition, Takeuchi *et al*. (2010) labeled the values of positions by the theoretical values if exhaustive search can be finished in a reasonable time.

If a program has a low mean squared error of values of positions, it means that the program understands the positions better and thus is usually stronger. Mean squared errors were shown to have a strong correlation to the strengths of programs (Gelly & Silver, 2007; Silver *et al*., 2017).

## 4. EXPERIMENTS FOR 2×4 CDC

In the experiments, all analyzed CDC programs were based on MCTS with different techniques or different parameters (Hsueh *et al*., 2016). Although the programs were originally written to play 4×8 CDC, it was generalized to different board sizes. All techniques incorporated by Hsueh *et al*. (2016) for 4×8 CDC were directly applied to 2×4 CDC. In total, the strengths of 129 program settings, as listed in Table 3, were analyzed. The program settings generally followed the experiments by Hsueh *et al*. (2016). All of the experiments were run on machines equipped with Intel (R) Core(TM) i5-3570K CPU 3.40GHz with 8GB memory.

Table 3
Details of the 129 analyzed program settings for 2×4 CDC

| **Incorporated technique** | **Number of Simulations Per Move** | **Settings of Parameters** |
|---|---|---|
| None | 1, 10, 100, 500, 1,000, 2,000, 5,000, 10,000, 20,000, 30,000, 50,000, 100,000, 200,000 | None |
| EPT | 1,000<br>30,000 | $\varepsilon \in \{0, 0.05, 0.1, 0.2, 0.3, 0.4\}$ |
| IMB | | $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ |
| SL | | $(a, k) \in \{0.2, 0.25, 0.3\} \times \{3, 5, 7, 9, 11\}$ |
| TSQ | | $a = 0.3,\ k = 11,\ c_w \in \{0, 0.01, 0.02, 0.03, 0.04\}$<br>$a = 0.3,\ c_w = 0.02,\ k \in \{7, 9, (11,)\ 13, 15\}$<br>$k = 11,\ c_w = 0.02,\ a \in \{0.2, 0.25, (0.3,)\ 0.35\}$ |
| PB | | $\mathbf{H^B},\ count_i = \sqrt{n_i},\ c_p \in \{0.001, 0.01, 0.1, 1, 10\}$<br>$\mathbf{H^B} + \mathbf{H^C},\ count_i = \sqrt{n_i},\ c_p \in \{0.001, 0.01, 0.1, 1, 10\}$<br>$\mathbf{H^B} + \mathbf{H^C} + \mathbf{H^F},\ count_i = \sqrt{n_i},\ c_p \in \{0.001, 0.01, 0.1, 1, 10\}$ |

In the following subsections, for each of the settings of MCTS above, win rates are collected given five different baseline programs in Subsection 4.1, prediction rates are collected given three different sets of expert moves in Subsection 4.2, and mean squared errors are collected given three different sources of ground-truth values in Subsection 4.3. All the detailed values of win rates, prediction rates, and mean squared errors are presented in Appendix A.

## 4.1 Win rates

This subsection investigates whether the win rates against practical and available baseline programs are correlated to the win rates against the optimal player (absolute strengths). In the experiments, the win rate of a program against a designated baseline program was obtained in the following way. The program played against the designated baseline program 1,000 games, with altering turns playing first, in each of the 24 combinations as listed in Table 1. Its win rate was the average score of the 24,000 games where a win was counted as 1, a draw as 0.5, and a loss as 0.

In 2×4 CDC, five baseline programs were considered. The first was the optimal player (OPT) as described in Subsection 3.1. The second was a four-ply[1] expectiminimax player (EMM) (Russell & Norvig, 2009) using weighted material sum (as described in Subsection 2.3.2) as heuristic evaluation. EMM had a win rate of 35.22% (±0.57%) against OPT. The remaining three baseline programs were the original MCTS with 1,000, 5,000, and 30,000 simulations per move (MCTS-1k, MCTS-5k, and MCTS-30k), which had win rates of 37.67% (±0.57%), 43.41% (±0.57%), and 46.78% (±0.58%) against OPT respectively. The five baseline programs from the weakest to the strongest were EMM, MCTS-1k, MCTS-5k, MCTS-30k, and OPT. In the experiments, EMM was intentionally included to have a baseline program not based on MCTS but also practically available.

[1] Six-ply EMM ran about 150 times slower than that of four-ply. Due to time constraints, experiments on EMM with more plies were not conducted.

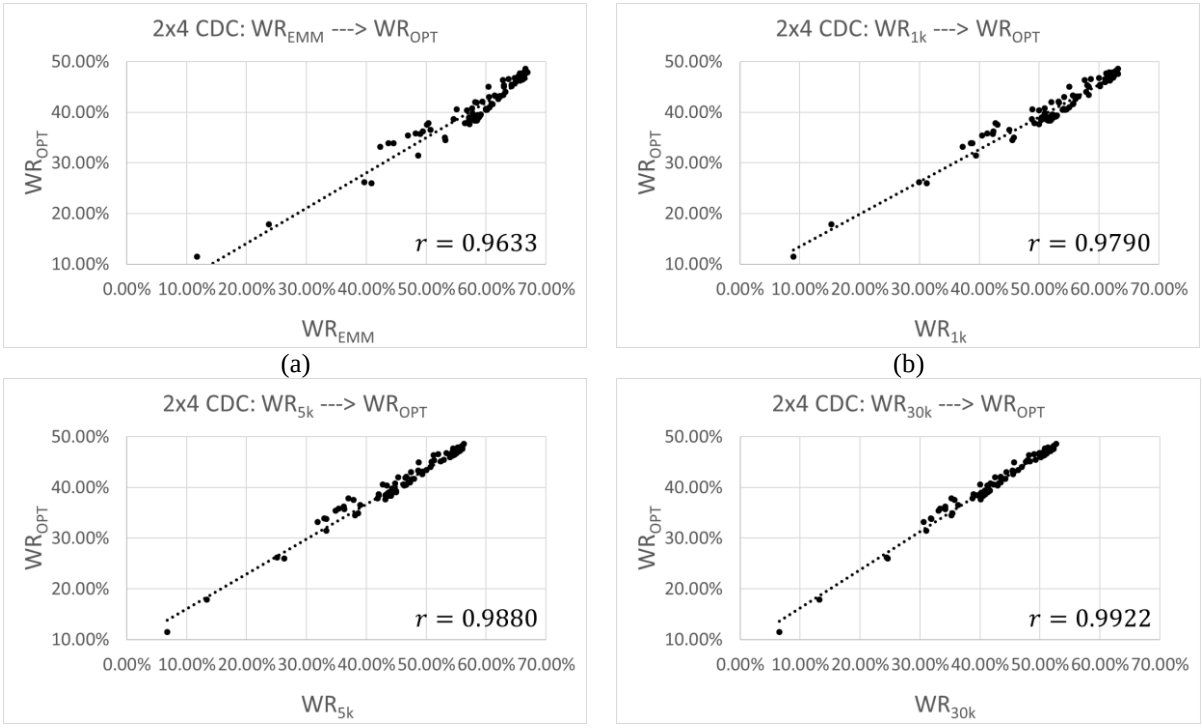


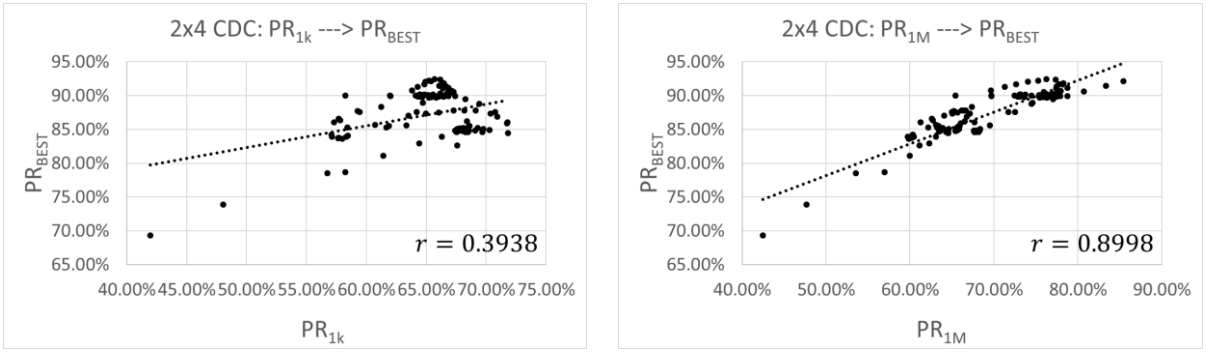

Fig. 4. Scatter plots for (a) $WR_{EMM}$, (b) $WR_{1k}$, (c) $WR_{5k}$, and (d) $WR_{30k}$ to $WR_{OPT}$.

In order to analyze the correlation between win rates against practical and available baseline programs and those against OPT, each of the 129 program settings (listed in Table 3) played against the five baseline programs respectively. For these program settings, let $WR_{OPT}$, $WR_{EMM}$, $WR_{1k}$, $WR_{5k}$ and $WR_{30k}$ be the win rates respectively against OPT, EMM, MCTS-1k, MCTS-5k, and MCTS-30k. Fig. 4 (a)-(d) show respectively $WR_{EMM}$, $WR_{1k}$, $WR_{5k}$, and $WR_{30k}$ with respect to the absolute strength $WR_{OPT}$ for all program settings. These plots show very highly positive correlations whose correlation coefficients $r$ (Asuero, Sayago, & González, 2006; Lane *et al*., 2014), described in more detail in Appendix B, were 0.9633, 0.9790, 0.9880, and 0.9922 respectively. The stronger the baseline program was, the higher the correlation coefficient obtained. The sample standard errors (Lane *et al*., 2014) with respect to the predicted lines (the dotted lines in Fig. 4) were 1.57%, 1.19%, 0.90%, and 0.73% respectively. Appendix B describes more on the sample standard errors and the predicted lines. The above observations suggested that the baseline program should be as strong as possible in order to obtain accurate strengths of game-playing programs when the optimal player was not available. However, this would be a tradeoff between the accuracy of the metric and the time needed to perform experiments, since a stronger baseline program usually implies more computation time.

## 4.2 Prediction rates

This subsection investigates whether the prediction rates to practical and available expert moves are correlated to those to the best moves (moves by OPT), and whether all of these prediction rates are correlated to the absolute strengths (win rates against OPT). A set of 2,400 games, 100 in each of the 24 material combinations listed in Table 1, were collected and about 47,000 positions in the game set were used for experiments on prediction rates.

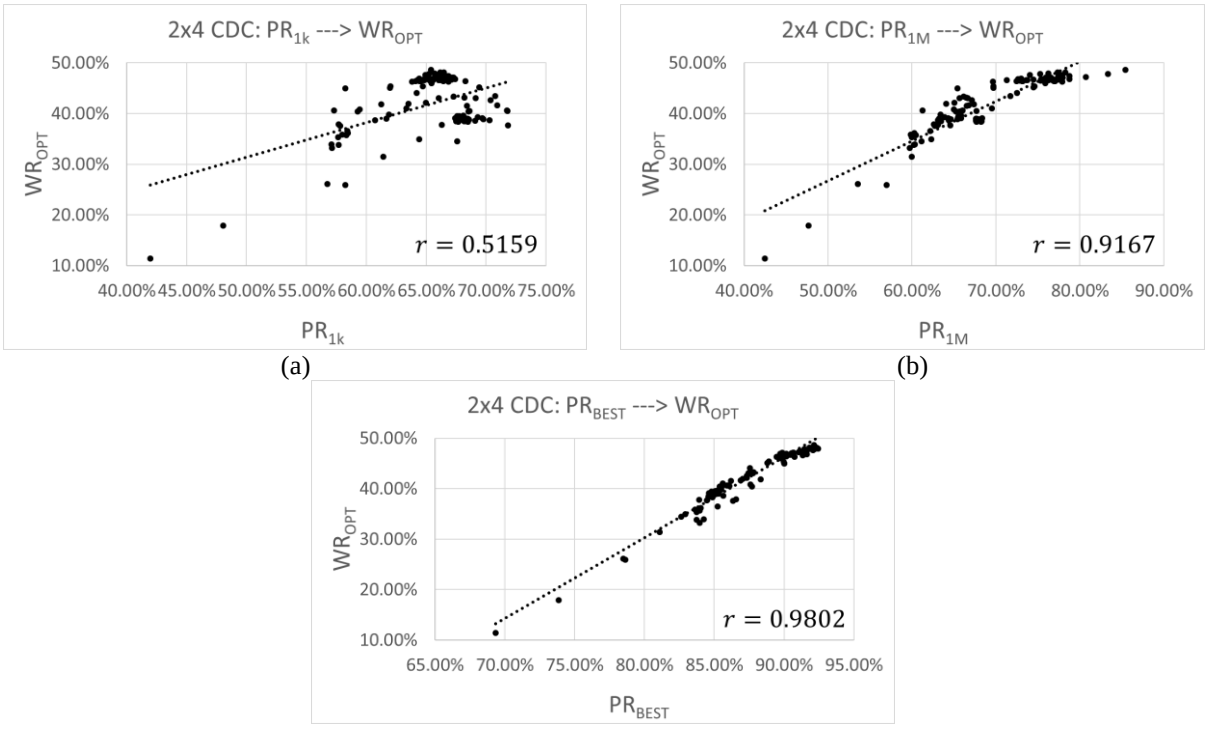
Three sets of expert moves were considered, the best moves as described in Subsection 3.2, the moves by MCTS-1k, and the moves by MCTS-1M (the original MCTS with 1,000,000 simulations per move). MCTS-1k and MCTS-1M had win rates of 37.67% (±0.57%) and 49.21% (±0.58%) against OPT respectively. Note that MCTS-1k was intentionally included to investigate the cases where the quality of the collected expert moves was low. To obtain the prediction rate of a program, the program played moves for the positions in the selected game set. In the experiment, both MCTS-1k and MCTS-1M obtained prediction rates respectively to the best moves with 84.46% (±0.33%) and 93.31% (±0.23%). The results clearly showed that MCTS-1M played the best moves more often than MCTS-1k.




(a) (b)

Fig. 5. Scatter plots for (a) $PR_{1k}$ and (b) $PR_{1M}$ to $PR_{BEST}$.

In order to analyze the correlation of prediction rates, each of the 129 program settings (listed in Table 3) played moves for the positions in the selected game set. For these program settings, let $PR_{BEST}$, $PR_{1k}$ and $PR_{1M}$ be the prediction rates respectively to the best moves, the moves by MCTS-1k, and the moves by MCTS-1M. Fig. 5 (a) and (b) show respectively $PR_{1k}$ and $PR_{1M}$ with respect to $PR_{BEST}$ for all program settings. Note that the values in $PR_{BEST}$ tended to be higher than those in $PR_{1k}$ and $PR_{1M}$, for the reason that it was easier to match best moves since a position may have multiple best moves as described in Subsection 3.2. From the experiment results, as expected, $PR_{1k}$ could not reflect $PR_{BEST}$ well, which only had a correlation coefficient of 0.3938. On the contrary, $PR_{1M}$ had a highly positive correlation to $PR_{BEST}$, which had a correlation coefficient of 0.8998. The sample standard errors with respect to the predicted lines were 3.29% and 1.56%. The results showed that the quality of expert moves influenced the results of prediction rates greatly, and that prediction rates to moves by a stronger program can be used to approximate those to the best moves.




(a) (b)

(c)

Fig. 6. Scatter plots for (a) $PR_{1k}$, (b) $PR_{1M}$, and (c) $PR_{BEST}$ to $WR_{OPT}$.

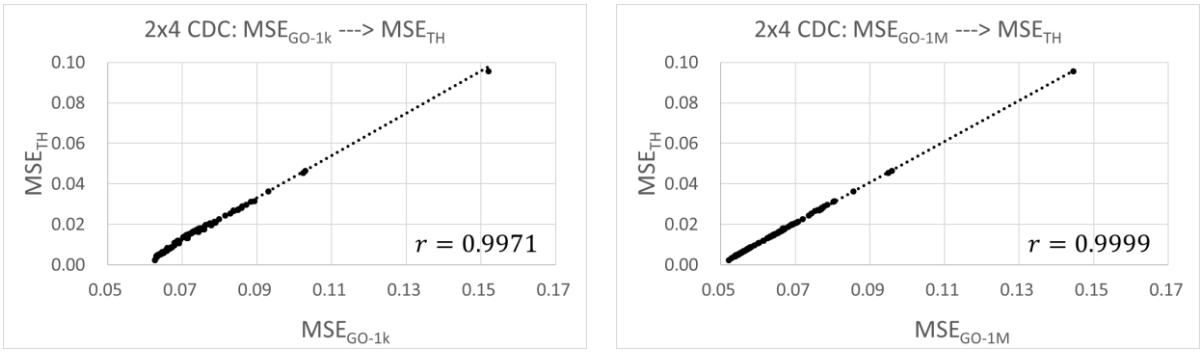
Next, the correlation between the prediction rates and the absolute strengths (win rates against OPT) was investigated. Fig. 6 (a)-(c) show the prediction rates, $PR_{1k}$, $PR_{1M}$ and $PR_{BEST}$, with respect to $WR_{OPT}$. Similar to those shown in Fig. 5, $PR_{1k}$ was not a good metric to the absolute strengths either, which only had a correlation coefficient of 0.5159. $PR_{1M}$ had a very highly positive correlation coefficient of 0.9167 to the absolute strengths. The sample standard errors with respect to the predicted lines were 5.00% and 2.33%. These suggested that when expert moves with high quality were available, prediction rates to those expert moves were good metrics to the strengths of game-playing programs, though not as good as win rates playing against a baseline program

as shown in Subsection 4.1. In addition, $PR_{BEST}$ had a higher correlation to absolute strengths than both $PR_{1k}$ and $PR_{1M}$. Namely, the correlation coefficient of $PR_{BEST}$ to $WR_{OPT}$ was 0.9802, and the sample standard error with respect to the predicted line was 1.16%. The analyses showed that the strengths of game-playing programs had strong correlations to the ability to select the best moves.

### 4.3 Mean squared errors

For 2×4 CDC, it is straightforward to use the theoretical values (as described in Subsection 3.3) as the ground-truth values to analyze mean squared errors of given programs. In addition to this set, let the moves played by MCTS-1M be regarded as expert moves, and thus the outcomes of these games be another source of the ground-truth values. The correlation of mean squared errors with different sources of ground-truth values is then analyzed. In addition, this subsection also studies whether all of these mean squared errors are correlated to the absolute strengths (win rates against OPT).

In the experiments, the same set of about 47,000 positions as in Subsection 4.2 was used. Two sources of ground-truth values for these positions were considered, the theoretical values and the outcomes of games from self-plays by MCTS-1M. Namely, a self-play game was performed by MCTS-1M from a given position to the end of the game. The game outcomes, consisting of wins, draws, and losses, were converted to 1's, 0's and -1's respectively, and subsequently used as a source of ground-truth values of the positions. Without loss of generality, all of the values were from the red player's point of view. To obtain a mean squared error of a program, the program ran MCTS on each of the positions in the game set. The value of position $p$ estimated by the program, $\hat{v}_p$, was $Q_i$ in Formula (1) where $i$ was the root of the search tree corresponding to $p$. Mean squared errors were then calculated based on the given ground-truth values and the collected $\{\hat{v}_p\}$. Note that mean squared errors were scaled by a factor of 1/4 to the range of [0, 1] in this paper. To investigate whether game outcomes with a lower quality influenced the results of mean squared errors, those from self-plays by MCTS-1k were included as the third source of ground-truth values.




(a) (b)

Fig. 7. Scatter plots for (a) $MSE_{GO\text{-}1k}$ and (b) $MSE_{GO\text{-}1M}$ to $MSE_{TH}$.

In order to analyze the correlation of mean squared errors, each of the 129 program settings (listed in Table 3) ran MCTS on all positions in the game set. For these programs, let $MSE_{TH}$, $MSE_{GO\text{-}1k}$ and $MSE_{GO\text{-}1M}$ be the mean squared errors respectively with theoretical values, game outcomes frm self-plays by MCTS-1k, and game outcomes from self-plays by MCTS-1M. Fig. 7 (a) and (b) show respectively $MSE_{GO\text{-}1k}$ and $MSE_{GO\text{-}1M}$ with respective to $MSE_{TH}$ for all program settings. Note that the values in $MSE_{TH}$ tended to be lower than those in $MSE_{GO\text{-}1k}$ and $MSE_{GO\text{-}1M}$, for the reason that the granularity of theoretical values was lower than that of game outcomes. The experiment results showed very highly positive correlation coefficients, 0.9971 and 0.9999, and the sample standard errors with respect to the predicted lines were 0.0008 and 0.0002. These showed that mean squared errors with different sources of game outcomes could predict those with respect to theoretical values well even when the *expert* had a lower quality such as MCTS-1k.

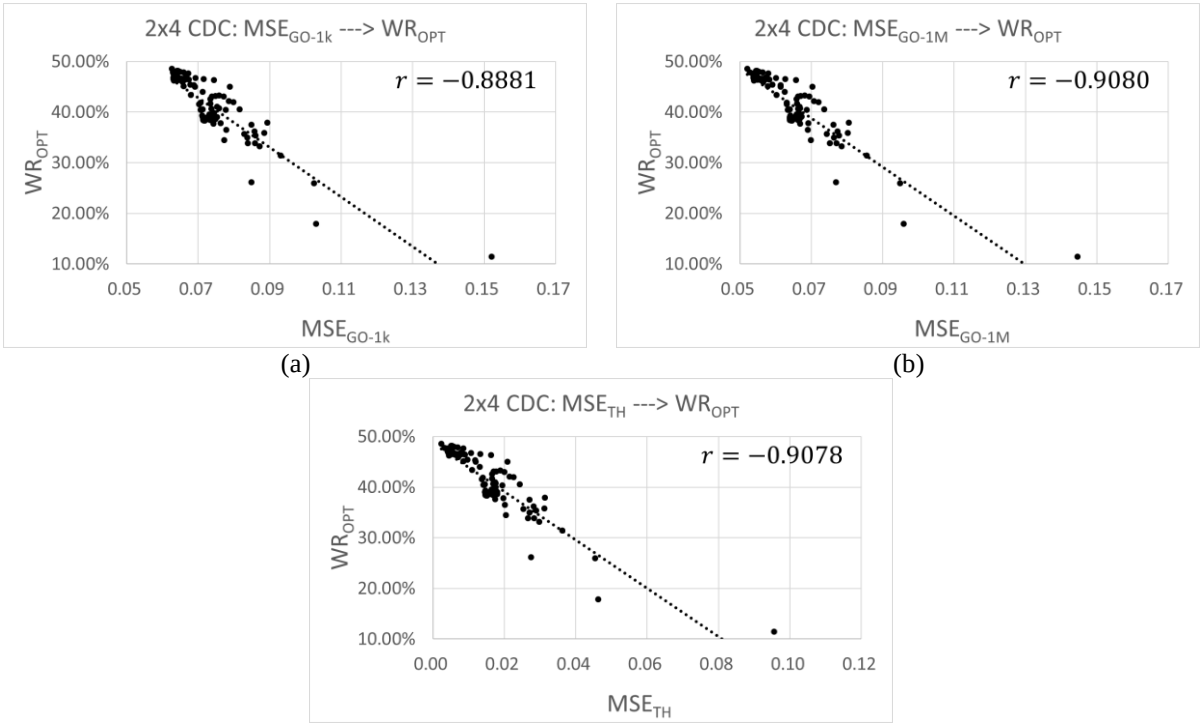



Fig. 8. Scatter plots for (a) $MSE_{GO-1k}$, (b) $MSE_{GO-1M}$, and (c) $MSE_{TH}$ to $WR_{OPT}$.

Next, the correlation between the mean squared errors and the absolute strengths (win rates against OPT) was investigated. Fig. 8 (a)-(c) show the mean squared errors $MSE_{GO-1k}$, $MSE_{GO-1M}$, and $MSE_{TH}$ with respect to $WR_{OPT}$. All showed highly negative correlations, which had correlation coefficients of -0.8881, -0.9080, and -9078 respectively, and the sample standard errors with respect to the predicted lines were 2.68%, 2.45%, and 2.45%. Note that the results here showed negative correlations since stronger game-playing programs tended to have lower mean squared errors. Generally, mean squared errors were good metrics to the strengths of game-playing programs, though not as good as win rates playing against a baseline program as shown in Subsection 4.1. However, it was a relatively stable metric compared to prediction rates, since game outcomes with a lower quality did not influence this metric too much.

## 5. EXPERIMENTS ON PREDICTING WIN RATES IN 4×8 CDC FROM 2×4 CDC

In this section, the absolute strengths of game-playing programs in 2×4 CDC were used to help analyze the strengths of game-playing programs in 4×8 CDC. Since games with different board sizes may have similar properties, it is worth investigating whether the experiment results obtained in smaller sizes can be used to predict those in the original size or a bigger size.

Table 4
Details of the 66 analyzed program settings for both 2×4 and 4×8 CDC

| **Incorporated technique** | **Number of Simulations Per Move** | **Settings of Parameters** |
|---|---|---|
| None | 2,000, 5,000, 10,000, 20,000, 30,000, 50,000, 100,000, 200,000 | None |
| EPT | 30,000 | $\varepsilon \in \{0, 0.05, 0.1, 0.2, 0.3, 0.4\}$ |
| IMB | | $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ |
| SL | | $(a, k) \in \{0.2, 0.25, 0.3\} \times \{3, 5, 7, 9, 11\}$ |
| TSQ | | $a = 0.3,\ k = 11,\ c_w \in \{0, 0.01, 0.02, 0.03, 0.04\}$<br>$a = 0.3,\ c_w = 0.02,\ k \in \{7, 9, (11,)\ 13, 15\}$<br>$k = 11,\ c_w = 0.02,\ a \in \{0.2, 0.25, (0.3,)\ 0.35\}$ |

| | | |
|---|---|---|
| PB | | $\mathbf{H^B}$, $count_i = \sqrt{n_i}$, $c_p \in \{0.001, 0.01, 0.1, 1, 10\}$<br>$\mathbf{H^B} + \mathbf{H^C}$, $count_i = \sqrt{n_i}$, $c_p \in \{0.001, 0.01, 0.1, 1, 10\}$<br>$\mathbf{H^B} + \mathbf{H^C} + \mathbf{H^F}$, $count_i = \sqrt{n_i}$, $c_p \in \{0.001, 0.01, 0.1, 1, 10\}$ |

The details of the 66 program settings for both 2×4 and 4×8 CDC are listed in Table 4. For each program setting, let the two win rates $WR_{2x4}$ and $WR_{4x8}$ be obtained as follows. $WR_{2x4}$ was the win rate against OPT (the optimal player) in 2×4 CDC which was the average score of 24,000 games as described in Subsection 4.1. $WR_{4x8}$ was the win rate against MCTS-30k (the original MCTS with 30,000 simulations per move), also a baseline program used by Hsueh *et al*. (2016). Most of the win rates in $WR_{4x8}$ were presented by Hsueh *et al*. (2016), where each win rate in 4×8 CDC was the average score of 1,000 games.

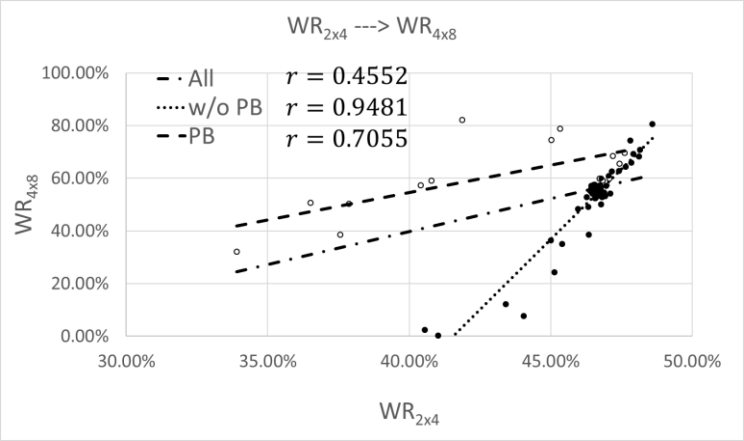



Fig. 9. Scatter plot for $WR_{2x4}$ to $WR_{4x8}$ where the hollow points are related to program settings with PB.

Fig. 9 shows that $WR_{2x4}$ were not highly correlated to $WR_{4x8}$ for all program settings listed in Table 4. The correlation coefficient of 0.4552 showed a low positive correlation only, and the sample standard error with respect to the predicted line (the dot-dashed line in Fig. 9) was 14.51% which was high. In addition to a lower correlation coefficient, another reason that the sample standard error here was relatively higher than those in Section 4 was that the number of sample data were lower here (66 vs. 129). After checking the distribution of points in Fig. 9, the points at the top-left part (hollow points) were all from the program settings with PB. By removing those related to PB, a much higher correlation was obtained with a correlation coefficient of 0.9481. The sample standard error with respect to the new predicted line (the dotted line in Fig. 9) was 5.27%. In addition, the correlation coefficient for the program settings with PB was 0.7055. Although a high positive correlation was obtained, only 15 sample data were included. The sample standard error with respect to the predicted line (the dashed line in Fig. 9) was 10.15%, which was still high.

In brief, $WR_{2x4}$ can be roughly used to predict $WR_{4x8}$ except for PB. In order to investigate the inconsistency of PB, the win rates from different program settings with PB, shown in Appendix A, were further analyzed. It was observed that PB in 2×4 CDC obtained higher win rates with $c_p = 0.01$ for all of the three experimented heuristics, $\mathbf{H^B}$, $\mathbf{H^B} + \mathbf{H^C}$ and $\mathbf{H^B} + \mathbf{H^C} + \mathbf{H^F}$. However, it was $c_p = 0.1$ for 4×8 CDC. Judging by this, it was speculated that heuristic scores of moves might not generalize well to different board sizes. Thus, the heuristics used in PB were further experimented. More specifically, in both 2×4 and 4×8 CDC, the strengths of the strongest heuristics $\mathbf{H^B} + \mathbf{H^C} + \mathbf{H^F}$ were measured by the original MCTS with different numbers of simulations per move. In 2×4 CDC, the program using the heuristics without search had a playing strength approximate to the original MCTS with only 30 simulations per move, where the win rate was 50.41% (±0.52%). In contrast, the program using the same heuristics without search had a playing strength approximate to the original MCTS with 1,800 simulations per move in 4×8 CDC, where the win rate was 50.30% (±2.08%). The results showed that the advantages brought by heuristic scores of moves were much higher in 4×8 CDC than in 2×4 CDC. From the above, the inconsistency of PB was conjectured to be a result of the used heuristic scores of moves. However, this should be justified with further research, and therefore is left as an open problem in the future.

## 6. CONCLUSIONS

In this paper, three metrics measuring strengths of game-playing programs are analyzed in a solved game, 2×4 CDC. The commonly used metric, win rates playing against a baseline program, has been shown to have the highest correlation to the absolute strengths of programs (win rates against the optimal player). Moreover, using stronger baseline programs reflects the absolute strengths better. The analysis in this paper offers evidence that win rates against a baseline program – the stronger the better – can be used to approximate the absolute strengths of programs.

In addition, the second metric, prediction rates to expert moves, also has a high correlation to the absolute strengths, as long as expert moves with high qualities were collected. The third metric, mean squared errors of values of positions, also has a high correlation to the absolute strengths, which is influenced less by the quality of the collected data. The results show that the strengths of game-playing programs can be roughly measured by these two metrics.

In addition, the win rates for game-playing programs with the same settings in both 2×4 and 4×8 CDC are analyzed. Except for those related to PB, the win rates obtained in different board sizes also have a high correlation. A conjecture for the inconsistency of PB is related to the used heuristic scores of moves. Further research is needed to study the condition that the win rates obtained in smaller sizes can or cannot be used to predict those in the full or larger sizes. In the future, it is also worth applying the analysis presented in this paper to other solved games, even deterministic games such as Hex (Gao, Mueller, & Hayward, 2017; Henderson, Arneson, & Hayward, 2009; Pawlewicz & Hayward, 2014).


## 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 MOST 106-2221-E-009 -139 -MY2, 106-2221-E-305-016-MY2, 106-2221-E-305-017-MY2, 107-2634-F-009-011, and 107-2634-F-259-001.


## APPENDIX A. DETAILED EXPERIMENT RESULTS

Table A.1 lists the win rates of the 129 program settings against five 2×4 CDC baseline programs, OPT, EMM, MCTS-1k, MCTS-5k and MCTS-30k, in the five columns $WR_{OPT}$ ($WR_{2x4}$), $WR_{EMM}$, $WR_{1k}$, $WR_{5k}$, and $WR_{30k}$, and the 66 program settings against MCTS-30k in 4×8 CDC in the column $WR_{4x8}$. The 95% confidence bounds for win rates in $WR_{OPT}$, $WR_{EMM}$, $WR_{1k}$, $WR_{5k}$, and $WR_{30k}$ were less than 0.59%, and those in $WR_{4x8}$ were less than 2.89%.

Table A.1
Detailed win rates for the 129 and 66 analyzed program settings in 2×4 and 4×8 CDC

| Incorporated technique | Number of simulations per move | Parameters | $WR_{OPT}$ ($WR_{2x4}$) | $WR_{EMM}$ | $WR_{1k}$ | $WR_{5k}$ | $WR_{30k}$ | $WR_{4x8}$ |
|---|---|---|---|---|---|---|---|---|
| | 1 | | 11.45% | 11.70% | 8.92% | 6.77% | 6.53% | |
| | 10 | | 17.89% | 23.71% | 15.30% | 13.36% | 13.23% | |
| | 100 | | 26.15% | 39.69% | 29.94% | 25.10% | 24.48% | |
| | 500 | | 34.49% | 53.15% | 45.54% | 38.14% | 35.23% | |
| | 1,000 | | 37.67% | 57.24% | 50.00% | 43.19% | 40.09% | |
| | 2,000 | | 40.56% | 60.09% | 54.58% | 46.66% | 42.92% | 2.45% |
| None | 5,000 | None | 43.41% | 62.84% | 58.27% | 50.00% | 46.39% | 12.20% |
| | 10,000 | | 45.13% | 64.26% | 60.19% | 52.46% | 48.38% | 24.25% |
| | 20,000 | | 46.35% | 64.97% | 61.21% | 53.96% | 49.98% | 38.60% |
| | 30,000 | | 46.78% | 65.45% | 61.89% | 54.58% | 50.00% | 50.00% |
| | 50,000 | | 47.16% | 65.73% | 62.42% | 55.37% | 51.54% | 62.55% |
| | 100,000 | | 47.81% | 66.28% | 62.99% | 56.03% | 52.26% | 74.35% |
| | 200,000 | | 48.60% | 66.54% | 63.20% | 56.24% | 52.75% | 80.60% |
| | | $\varepsilon = 0$ | 47.06% | 65.34% | 61.76% | 54.60% | 51.04% | 60.75% |
| | | $\varepsilon = 0.05$ | 46.59% | 65.07% | 61.05% | 54.12% | 50.36% | 55.30% |
| EPT | 30,000 | $\varepsilon = 0.1$ | 45.96% | 64.92% | 61.03% | 53.94% | 50.15% | 48.25% |
| | | $\varepsilon = 0.2$ | 45.41% | 64.28% | 60.18% | 52.90% | 49.25% | 35.00% |
| | | $\varepsilon = 0.3$ | 44.04% | 63.24% | 57.89% | 50.70% | 46.95% | 7.65% |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\varepsilon$ = 0.4 | | 41.02% | 60.60% | 55.01% | 47.30% | 43.46% | 0.20% |
| IMB | | $\alpha$ = 0.1 | | 47.42% | 66.14% | 62.60% | 55.43% | 51.76% | 62.80% |
| | | $\alpha$ = 0.2 | | 48.13% | 66.65% | 62.87% | 55.87% | 52.28% | 68.30% |
| | | $\alpha$ = 0.3 | | 48.16% | 66.66% | 62.84% | 55.98% | 52.33% | 70.90% |
| | | $\alpha$ = 0.4 | | 47.92% | 66.91% | 62.35% | 55.95% | 52.08% | 69.25% |
| | | $\alpha$ = 0.5 | | 47.86% | 66.32% | 61.74% | 55.24% | 51.36% | 65.85% |
| | | $\alpha$ = 0.6 | | 47.65% | 65.64% | 61.19% | 54.44% | 50.85% | 64.45% |
| | | $\alpha$ = 0.7 | | 46.78% | 64.76% | 60.01% | 53.36% | 49.86% | 59.80% |
| | | $\alpha$ = 0.8 | | 46.57% | 63.72% | 58.66% | 52.02% | 49.04% | 55.90% |
| | | $\alpha$ = 0.9 | | 46.33% | 62.77% | 57.63% | 51.21% | 48.20% | 49.10% |
| | | $\alpha$ = 1 | | 45.00% | 60.36% | 54.99% | 48.76% | 45.72% | 36.35% |
| SL | | $a$ = 0.2 | $k$ = 3 | 46.28% | 65.64% | 61.83% | 54.41% | 50.55% | 52.80% |
| | | | $k$ = 5 | 46.71% | 65.79% | 61.95% | 54.66% | 51.08% | 54.60% |
| | | | $k$ = 7 | 46.41% | 65.45% | 61.88% | 54.51% | 50.66% | 55.25% |
| | | | $k$ = 9 | 46.59% | 65.81% | 61.55% | 54.56% | 50.55% | 52.40% |
| | | | $k$ = 11 | 47.10% | 66.04% | 62.19% | 54.38% | 50.65% | 54.10% |
| | | $a$ = 0.25 | $k$ = 3 | 46.80% | 65.95% | 62.09% | 54.79% | 50.82% | 54.75% |
| | | | $k$ = 5 | 46.96% | 65.92% | 61.82% | 54.84% | 51.08% | 57.20% |
| | | | $k$ = 7 | 46.56% | 65.62% | 61.94% | 54.56% | 50.86% | 57.50% |
| | | | $k$ = 9 | 46.71% | 65.98% | 61.91% | 54.41% | 50.68% | 54.20% |
| | | | $k$ = 11 | 46.92% | 65.62% | 61.72% | 54.37% | 50.74% | 53.25% |
| | | $a$ = 0.3 | $k$ = 3 | 46.74% | 65.81% | 61.76% | 54.47% | 50.59% | 54.95% |
| | | | $k$ = 5 | 46.63% | 65.83% | 61.95% | 54.57% | 50.74% | 55.70% |
| | | | $k$ = 7 | 46.74% | 65.59% | 61.94% | 54.45% | 50.87% | 57.35% |
| | | | $k$ = 9 | 46.76% | 65.49% | 61.84% | 54.64% | 50.81% | 56.05% |
| | | | $k$ = 11 | 46.48% | 65.71% | 61.78% | 54.72% | 50.76% | 54.25% |
| TSQ | | $a$ = 0.3<br>$k$ = 11 | $c_w$ = 0 | 46.81% | 66.20% | 61.48% | 54.48% | 50.86% | 52.80% |
| | | | $c_w$ = 0.01 | 46.89% | 66.11% | 61.41% | 54.76% | 51.06% | 54.60% |
| | | | $c_w$ = 0.02 | 46.43% | 66.06% | 61.35% | 54.54% | 50.86% | 55.25% |
| | | | $c_w$ = 0.03 | 46.57% | 65.95% | 61.61% | 54.46% | 50.69% | 52.40% |
| | | | $c_w$ = 0.04 | 46.62% | 65.98% | 61.73% | 54.58% | 50.84% | 54.10% |
| | | $a$ = 0.3<br>$c_w$ = 0.02 | $k$ = 7 | 46.55% | 65.98% | 61.70% | 54.39% | 50.88% | 54.75% |
| | | | $k$ = 9 | 46.43% | 65.89% | 61.72% | 54.37% | 50.73% | 57.20% |
| | | | $k$ = 13 | 46.53% | 65.63% | 61.85% | 54.56% | 50.77% | 57.50% |
| | | | $k$ = 15 | 46.51% | 65.87% | 61.79% | 54.62% | 50.68% | 54.20% |
| | | $k$ = 11<br>$c_w$ = 0.02 | $a$ = 0.2 | 46.65% | 66.26% | 61.85% | 54.78% | 50.80% | 53.25% |
| | | | $a$ = 0.25 | 46.59% | 65.91% | 62.00% | 54.38% | 50.77% | 54.95% |
| | | | $a$ = 0.3 | 46.42% | 66.01% | 61.59% | 54.31% | 50.76% | 55.70% |
| PB | | $\mathbf{H^B}$ | $c_p$ = 0.001 | 47.08% | 66.02% | 62.23% | 55.09% | 51.24% | 59.85% |
| | | | $c_p$ = 0.01 | 47.20% | 66.39% | 62.43% | 55.62% | 51.77% | 68.40% |
| | | | $c_p$ = 0.1 | 45.03% | 62.94% | 58.26% | 50.88% | 47.69% | 74.45% |
| | | | $c_p$ = 1 | 40.41% | 56.82% | 49.94% | 43.43% | 41.23% | 57.30% |
| | | | $c_p$ = 10 | 37.88% | 50.34% | 42.73% | 36.99% | 35.21% | 50.30% |
| | | $\mathbf{H^B + H^C}$ | $c_p$ = 0.001 | 46.84% | 66.06% | 62.43% | 54.99% | 51.24% | 59.00% |
| | | | $c_p$ = 0.01 | 47.44% | 66.32% | 62.70% | 55.69% | 52.06% | 65.55% |
| | | | $c_p$ = 0.1 | 45.34% | 62.98% | 58.04% | 51.30% | 47.94% | 78.90% |
| | | | $c_p$ = 1 | 40.79% | 57.62% | 50.96% | 44.78% | 41.78% | 59.00% |
| | | | $c_p$ = 10 | 37.56% | 50.08% | 43.00% | 37.85% | 35.72% | 38.65% |
| | | $\mathbf{H^B + H^C + H^F}$ | $c_p$ = 0.001 | 46.73% | 65.72% | 62.08% | 55.05% | 51.00% | 59.90% |
| | | | $c_p$ = 0.01 | 47.62% | 66.51% | 63.13% | 56.03% | 52.30% | 69.55% |
| | | | $c_p$ = 0.1 | 41.88% | 58.45% | 53.34% | 46.54% | 43.43% | 82.10% |
| | | | $c_p$ = 1 | 36.52% | 50.68% | 44.99% | 39.01% | 36.35% | 50.70% |
| | | | $c_p$ = 10 | 33.90% | 43.66% | 38.55% | 32.98% | 31.79% | 32.15% |
| EPT | 1,000 | $\varepsilon$ = 0 | | 39.33% | 58.61% | 52.33% | 44.86% | 41.71% | |
| | | $\varepsilon$ = 0.05 | | 38.44% | 57.17% | 50.35% | 43.18% | 39.56% | |
| | | $\varepsilon$ = 0.1 | | 37.81% | 56.42% | 49.17% | 41.92% | 38.77% | |
| | | $\varepsilon$ = 0.2 | | 34.96% | 53.11% | 45.75% | 38.65% | 35.38% | |
| | | $\varepsilon$ = 0.3 | | 31.45% | 48.68% | 39.46% | 33.28% | 31.08% | |
| | | $\varepsilon$ = 0.4 | | 25.94% | 40.84% | 31.21% | 26.33% | 24.63% | |
| IMB | | $\alpha$ = 0.1 | | 40.63% | 59.97% | 54.13% | 46.17% | 42.28% | |
| | | $\alpha$ = 0.2 | | 41.66% | 60.92% | 55.70% | 47.99% | 44.20% | |
| | | $\alpha$ = 0.3 | | 42.60% | 62.04% | 56.10% | 49.35% | 45.52% | |
| | | $\alpha$ = 0.4 | | 43.10% | 62.10% | 56.55% | 49.30% | 45.72% | |
| | | $\alpha$ = 0.5 | | 43.14% | 62.39% | 56.19% | 48.88% | 45.65% | |
| | | $\alpha$ = 0.6 | | 43.31% | 61.45% | 55.65% | 48.66% | 45.47% | |
| | | $\alpha$ = 0.7 | | 43.04% | 60.47% | 54.17% | 47.47% | 44.42% | |
| | | $\alpha$ = 0.8 | | 42.13% | 59.37% | 53.21% | 46.68% | 43.49% | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | $\alpha$ = 0.9 | | 41.96% | 58.18% | 51.99% | 45.30% | 42.49% |
| | | $\alpha$ = 1 | | 40.60% | 55.08% | 48.85% | 42.75% | 40.01% |
| SL | | $a$ = 0.2 | $k$ = 3 | 38.54% | 58.03% | 51.23% | 43.57% | 40.23% |
| | | | $k$ = 5 | 38.35% | 58.45% | 51.48% | 44.12% | 40.53% |
| | | | $k$ = 7 | 38.59% | 58.15% | 52.01% | 43.94% | 40.39% |
| | | | $k$ = 9 | 38.97% | 58.01% | 51.31% | 43.84% | 40.68% |
| | | | $k$ = 11 | 38.95% | 58.18% | 51.39% | 43.67% | 40.23% |
| | | $a$ = 0.25 | $k$ = 3 | 38.77% | 57.74% | 51.43% | 43.47% | 40.33% |
| | | | $k$ = 5 | 38.87% | 57.96% | 52.05% | 43.81% | 40.57% |
| | | | $k$ = 7 | 38.64% | 58.27% | 51.20% | 43.79% | 40.29% |
| | | | $k$ = 9 | 38.70% | 58.08% | 51.75% | 43.94% | 40.36% |
| | | | $k$ = 11 | 38.52% | 57.95% | 51.33% | 43.80% | 40.41% |
| | | $a$ = 0.3 | $k$ = 3 | 38.45% | 58.15% | 51.61% | 43.71% | 40.11% |
| | | | $k$ = 5 | 39.05% | 58.27% | 51.85% | 43.86% | 40.78% |
| | | | $k$ = 7 | 38.46% | 58.50% | 51.39% | 43.91% | 40.46% |
| | | | $k$ = 9 | 38.73% | 58.01% | 51.91% | 44.08% | 40.16% |
| | | | $k$ = 11 | 38.34% | 58.04% | 51.90% | 43.89% | 40.46% |
| TSQ | | $a$ = 0.3<br>$k$ = 11 | $c_w$ = 0 | 39.13% | 58.81% | 52.11% | 44.58% | 41.39% |
| | | | $c_w$ = 0.01 | 39.31% | 58.90% | 52.52% | 44.56% | 41.10% |
| | | | $c_w$ = 0.02 | 39.20% | 58.84% | 52.82% | 44.68% | 40.88% |
| | | | $c_w$ = 0.03 | 39.49% | 58.99% | 52.11% | 44.75% | 41.19% |
| | | | $c_w$ = 0.04 | 39.24% | 58.61% | 52.69% | 45.01% | 41.10% |
| | | $a$ = 0.3<br>$c_w$ = 0.02 | $k$ = 7 | 39.58% | 59.06% | 52.22% | 44.47% | 41.21% |
| | | | $k$ = 9 | 39.37% | 58.69% | 52.96% | 44.70% | 40.97% |
| | | | $k$ = 13 | 39.41% | 58.57% | 52.53% | 44.63% | 40.81% |
| | | | $k$ = 15 | 39.49% | 58.63% | 52.27% | 44.52% | 41.29% |
| | | $k$ = 11<br>$c_w$ = 0.02 | $a$ = 0.2 | 38.92% | 58.74% | 52.37% | 44.35% | 40.55% |
| | | | $a$ = 0.25 | 39.14% | 58.57% | 52.17% | 44.39% | 40.93% |
| | | | $a$ = 0.3 | 39.09% | 58.61% | 52.22% | 45.00% | 41.35% |
| PB | | $\mathbf{H^B}$ | $c_p$ = 0.001 | 38.68% | 57.65% | 51.09% | 43.84% | 40.93% |
| | | | $c_p$ = 0.01 | 40.43% | 60.13% | 53.94% | 46.31% | 42.94% |
| | | | $c_p$ = 0.1 | 38.99% | 57.14% | 50.44% | 44.06% | 40.61% |
| | | | $c_p$ = 1 | 36.18% | 49.43% | 42.35% | 36.27% | 34.23% |
| | | | $c_p$ = 10 | 35.84% | 48.20% | 41.33% | 35.39% | 33.38% |
| | | $\mathbf{H^B + H^C}$ | $c_p$ = 0.001 | 38.92% | 57.64% | 51.55% | 44.02% | 40.76% |
| | | | $c_p$ = 0.01 | 40.47% | 59.96% | 54.34% | 46.56% | 42.95% |
| | | | $c_p$ = 0.1 | 39.80% | 57.70% | 50.94% | 44.26% | 41.37% |
| | | | $c_p$ = 1 | 35.70% | 49.00% | 42.30% | 36.34% | 34.20% |
| | | | $c_p$ = 10 | 35.39% | 46.91% | 40.47% | 34.89% | 33.12% |
| | | $\mathbf{H^B + H^C + H^F}$ | $c_p$ = 0.001 | 39.08% | 58.43% | 51.94% | 44.34% | 41.04% |
| | | | $c_p$ = 0.01 | 41.57% | 61.06% | 55.59% | 47.55% | 43.59% |
| | | | $c_p$ = 0.1 | 38.64% | 54.55% | 48.75% | 42.02% | 38.95% |
| | | | $c_p$ = 1 | 33.84% | 44.52% | 38.88% | 33.32% | 31.91% |
| | | | $c_p$ = 10 | 33.21% | 42.30% | 37.23% | 31.90% | 30.64% |

Table A.2 lists the prediction rates of the 129 program settings for 2×4 CDC with respect to the best moves, the moves played by MCTS-1k, and the moves by MCTS-1M in the columns $PR_{BEST}$, $PR_{1k}$, and $PR_{1M}$, the mean squared errors with respect to the theoretical values, the game outcomes from self-plays by MCTS-1k, and the game outcomes from self-plays by MCTS-1M in the columns $MSE_{TH}$, $MSE_{GO\text{-}1k}$, and $MSE_{GO\text{-}1M}$ respectively. The 95% confidence bounds for prediction rates in $PR_{BEST}$ were less than 0.42%, and those in $PR_{1k}$ and $PR_{1M}$ were less than 0.45%. The 95% confidence bounds for mean squared errors in $MSE_{TH}$ were less than 0.0012, and those in $MSE_{GO\text{-}1k}$ and $MSE_{GO\text{-}1M}$ were less than 0.0017.

Table A.2
Detailed prediction rates and mean squared errors of the 129 analyzed program settings in 2×4 CDC

| **Incorporated technique** | **Number of simulations per move** | **Parameters** | **$PR_{BEST}$** | **$PR_{1k}$** | **$PR_{1M}$** | **$MSE_{TH}$** | **$MSE_{GO\text{-}1k}$** | **$MSE_{GO\text{-}1M}$** |
|---|---|---|---|---|---|---|---|---|
| None | 1 | None | 69.34% | 41.98% | 42.49% | 0.0955 | 0.1521 | 0.1445 |
| | 10 | | 73.86% | 48.04% | 47.67% | 0.0464 | 0.1029 | 0.0958 |
| | 100 | | 78.50% | 56.76% | 53.55% | 0.0275 | 0.0849 | 0.0769 |
| | 500 | | 82.65% | 67.58% | 61.16% | 0.0205 | 0.0773 | 0.0699 |
| | 1,000 | | 84.46% | 71.82% | 64.54% | 0.0175 | 0.0743 | 0.0669 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2,000 | | | 86.08% | 71.80% | 67.69% | 0.0145 | 0.0713 | 0.0640 |
| | 5,000 | | | 87.59% | 70.76% | 71.70% | 0.0109 | 0.0679 | 0.0603 |
| | 10,000 | | | 88.79% | 69.42% | 74.49% | 0.0085 | 0.0660 | 0.0579 |
| | 20,000 | | | 89.46% | 68.27% | 77.04% | 0.0065 | 0.0646 | 0.0559 |
| | 30,000 | | | 89.94% | 67.42% | 78.78% | 0.0055 | 0.0640 | 0.0549 |
| | 50,000 | | | 90.65% | 66.84% | 80.71% | 0.0044 | 0.0634 | 0.0539 |
| | 100,000 | | | 91.43% | 66.07% | 83.36% | 0.0033 | 0.0630 | 0.0529 |
| | 200,000 | | | 92.15% | 65.38% | 85.43% | 0.0024 | 0.0627 | 0.0521 |
| EPT | | $\varepsilon$ = 0 | | 90.54% | 67.27% | 75.91% | 0.0051 | 0.0637 | 0.0547 |
| | | $\varepsilon$ = 0.05 | | 90.23% | 66.18% | 76.26% | 0.0060 | 0.0646 | 0.0556 |
| | | $\varepsilon$ = 0.1 | | 89.78% | 65.46% | 75.88% | 0.0071 | 0.0656 | 0.0566 |
| | | $\varepsilon$ = 0.2 | | 88.91% | 64.72% | 74.62% | 0.0097 | 0.0679 | 0.0591 |
| | | $\varepsilon$ = 0.3 | | 87.57% | 64.18% | 72.53% | 0.0131 | 0.0712 | 0.0625 |
| | | $\varepsilon$ = 0.4 | | 85.62% | 63.36% | 69.50% | 0.0175 | 0.0752 | 0.0668 |
| IMB | | $\alpha$ = 0.1 | | 91.12% | 66.84% | 78.75% | 0.0052 | 0.0640 | 0.0546 |
| | | $\alpha$ = 0.2 | | 91.81% | 66.49% | 78.25% | 0.0051 | 0.0641 | 0.0546 |
| | | $\alpha$ = 0.3 | | 92.34% | 66.17% | 77.33% | 0.0053 | 0.0644 | 0.0548 |
| | | $\alpha$ = 0.4 | | 92.45% | 65.68% | 76.27% | 0.0059 | 0.0650 | 0.0554 |
| | | $\alpha$ = 0.5 | | 92.18% | 65.24% | 75.28% | 0.0070 | 0.0659 | 0.0564 |
| | | $\alpha$ = 0.6 | | 92.07% | 64.95% | 74.02% | 0.0086 | 0.0672 | 0.0580 |
| | | $\alpha$ = 0.7 | | 91.64% | 64.85% | 72.62% | 0.0107 | 0.0692 | 0.0601 |
| | | $\alpha$ = 0.8 | | 91.32% | 64.28% | 71.32% | 0.0133 | 0.0715 | 0.0627 |
| | | $\alpha$ = 0.9 | | 90.74% | 63.83% | 69.63% | 0.0163 | 0.0744 | 0.0657 |
| | | $\alpha$ = 1 | | 90.03% | 58.23% | 65.39% | 0.0209 | 0.0789 | 0.0702 |
| SL | | $a$ = 0.2 | $k$ = 3 | 89.93% | 66.83% | 77.88% | 0.0045 | 0.0632 | 0.0540 |
| | | | $k$ = 5 | 90.27% | 66.66% | 77.43% | 0.0045 | 0.0633 | 0.0540 |
| | | | $k$ = 7 | 89.87% | 66.41% | 76.91% | 0.0047 | 0.0634 | 0.0541 |
| | | | $k$ = 9 | 89.97% | 66.03% | 76.74% | 0.0049 | 0.0636 | 0.0543 |
| | | | $k$ = 11 | 89.85% | 66.04% | 76.55% | 0.0051 | 0.0637 | 0.0544 |
| | | $a$ = 0.25 | $k$ = 3 | 89.88% | 66.48% | 77.46% | 0.0044 | 0.0631 | 0.0538 |
| | | | $k$ = 5 | 89.73% | 65.81% | 76.34% | 0.0046 | 0.0633 | 0.0540 |
| | | | $k$ = 7 | 89.99% | 65.55% | 76.24% | 0.0049 | 0.0637 | 0.0543 |
| | | | $k$ = 9 | 89.90% | 65.34% | 75.91% | 0.0052 | 0.0640 | 0.0547 |
| | | | $k$ = 11 | 90.16% | 65.57% | 76.12% | 0.0055 | 0.0642 | 0.0548 |
| | | $a$ = 0.3 | $k$ = 3 | 90.34% | 66.18% | 76.90% | 0.0044 | 0.0631 | 0.0538 |
| | | | $k$ = 5 | 89.87% | 65.45% | 75.81% | 0.0048 | 0.0635 | 0.0541 |
| | | | $k$ = 7 | 89.71% | 65.23% | 75.46% | 0.0052 | 0.0640 | 0.0546 |
| | 30,000 | | $k$ = 9 | 90.11% | 65.14% | 75.38% | 0.0057 | 0.0645 | 0.0551 |
| | | | $k$ = 11 | 89.94% | 65.33% | 75.33% | 0.0061 | 0.0648 | 0.0555 |
| TSQ | | $a$ = 0.3<br>$k$ = 11 | $c_w$ = 0 | 89.89% | 64.41% | 73.02% | 0.0080 | 0.0665 | 0.0573 |
| | | | $c_w$ = 0.01 | 89.93% | 64.40% | 73.01% | 0.0079 | 0.0664 | 0.0572 |
| | | | $c_w$ = 0.02 | 90.18% | 64.48% | 73.19% | 0.0078 | 0.0663 | 0.0571 |
| | | | $c_w$ = 0.03 | 89.83% | 64.41% | 73.05% | 0.0077 | 0.0662 | 0.0571 |
| | | | $c_w$ = 0.04 | 89.97% | 64.30% | 73.03% | 0.0076 | 0.0661 | 0.0570 |
| | | $a$ = 0.3<br>$c_w$ = 0.02 | $k$ = 7 | 89.78% | 64.67% | 73.60% | 0.0066 | 0.0652 | 0.0560 |
| | | | $k$ = 9 | 90.13% | 64.46% | 73.13% | 0.0073 | 0.0658 | 0.0566 |
| | | | $k$ = 13 | 89.89% | 64.32% | 72.83% | 0.0082 | 0.0668 | 0.0576 |
| | | | $k$ = 15 | 90.14% | 64.46% | 73.04% | 0.0085 | 0.0670 | 0.0579 |
| | | $k$ = 11<br>$c_w$ = 0.02 | $a$ = 0.2 | 90.02% | 65.31% | 74.45% | 0.0061 | 0.0646 | 0.0554 |
| | | | $a$ = 0.25 | 90.12% | 64.83% | 73.74% | 0.0068 | 0.0654 | 0.0562 |
| | | | $a$ = 0.3 | 90.04% | 64.09% | 72.45% | 0.0089 | 0.0674 | 0.0582 |
| PB | | $\mathbf{H^B}$ | $c_p$ = 0.001 | 90.68% | 67.16% | 77.96% | 0.0050 | 0.0636 | 0.0545 |
| | | | $c_p$ = 0.01 | 91.04% | 66.38% | 77.38% | 0.0041 | 0.0633 | 0.0538 |
| | | | $c_p$ = 0.1 | 90.02% | 61.95% | 69.69% | 0.0120 | 0.0691 | 0.0615 |
| | | | $c_p$ = 1 | 87.69% | 59.27% | 65.22% | 0.0195 | 0.0778 | 0.0687 |
| | | | $c_p$ = 10 | 86.57% | 57.69% | 62.58% | 0.0314 | 0.0894 | 0.0804 |
| | | $\mathbf{H^B + H^C}$ | $c_p$ = 0.001 | 90.52% | 66.98% | 77.81% | 0.0049 | 0.0636 | 0.0544 |
| | | | $c_p$ = 0.01 | 91.36% | 66.63% | 77.46% | 0.0041 | 0.0632 | 0.0537 |
| | | | $c_p$ = 0.1 | 89.96% | 62.03% | 69.73% | 0.0118 | 0.0688 | 0.0613 |
| | | | $c_p$ = 1 | 87.58% | 59.44% | 64.96% | 0.0176 | 0.0758 | 0.0668 |
| | | | $c_p$ = 10 | 86.33% | 57.82% | 62.73% | 0.0271 | 0.0849 | 0.0761 |
| | | $\mathbf{H^B + H^C + H^F}$ | $c_p$ = 0.001 | 90.48% | 67.13% | 77.73% | 0.0049 | 0.0635 | 0.0544 |
| | | | $c_p$ = 0.01 | 91.68% | 66.36% | 77.80% | 0.0040 | 0.0631 | 0.0536 |
| | | | $c_p$ = 0.1 | 88.33% | 61.23% | 67.38% | 0.0140 | 0.0706 | 0.0632 |
| | | | $c_p$ = 1 | 85.25% | 58.42% | 62.15% | 0.0201 | 0.0779 | 0.0691 |
| | | | $c_p$ = 10 | 84.26% | 57.09% | 60.33% | 0.0283 | 0.0859 | 0.0771 |
| EPT | 1,000 | $\varepsilon$ = 0 | | 85.25% | 69.28% | 63.88% | 0.0147 | 0.0711 | 0.0641 |
| | | $\varepsilon$ = 0.05 | | 84.71% | 67.50% | 63.44% | 0.0168 | 0.0734 | 0.0662 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\varepsilon$ = 0.1 | | 83.90% | 66.29% | 63.07% | 0.0197 | 0.0763 | 0.0691 |
| | | $\varepsilon$ = 0.2 | | 82.93% | 64.40% | 62.30% | 0.0270 | 0.0838 | 0.0763 |
| | | $\varepsilon$ = 0.3 | | 81.10% | 61.40% | 59.96% | 0.0362 | 0.0931 | 0.0855 |
| | | $\varepsilon$ = 0.4 | | 78.65% | 58.25% | 56.99% | 0.0455 | 0.1025 | 0.0949 |
| IMB | | $\alpha$ = 0.1 | | 85.92% | 71.73% | 66.01% | 0.0169 | 0.0737 | 0.0663 |
| | | $\alpha$ = 0.2 | | 86.88% | 70.94% | 66.79% | 0.0166 | 0.0734 | 0.0659 |
| | | $\alpha$ = 0.3 | | 87.32% | 70.37% | 67.12% | 0.0165 | 0.0734 | 0.0658 |
| | | $\alpha$ = 0.4 | | 87.76% | 69.13% | 66.66% | 0.0170 | 0.0739 | 0.0663 |
| | | $\alpha$ = 0.5 | | 87.77% | 68.19% | 66.44% | 0.0178 | 0.0748 | 0.0671 |
| | | $\alpha$ = 0.6 | | 87.81% | 67.28% | 66.16% | 0.0189 | 0.0759 | 0.0681 |
| | | $\alpha$ = 0.7 | | 87.46% | 66.04% | 65.69% | 0.0201 | 0.0772 | 0.0694 |
| | | $\alpha$ = 0.8 | | 87.32% | 64.98% | 65.04% | 0.0214 | 0.0786 | 0.0707 |
| | | $\alpha$ = 0.9 | | 87.06% | 63.51% | 64.09% | 0.0227 | 0.0799 | 0.0720 |
| | | $\alpha$ = 1 | | 86.03% | 57.28% | 61.28% | 0.0243 | 0.0816 | 0.0735 |
| SL | | $a$ = 0.2 | $k$ = 3 | 84.81% | 69.09% | 67.61% | 0.0155 | 0.0723 | 0.0649 |
| | | | $k$ = 5 | 84.60% | 68.50% | 67.71% | 0.0150 | 0.0717 | 0.0644 |
| | | | $k$ = 7 | 84.63% | 68.29% | 67.79% | 0.0149 | 0.0715 | 0.0642 |
| | | | $k$ = 9 | 84.76% | 68.37% | 67.61% | 0.0148 | 0.0714 | 0.0642 |
| | | | $k$ = 11 | 84.80% | 68.42% | 67.69% | 0.0148 | 0.0715 | 0.0642 |
| | | $a$ = 0.25 | $k$ = 3 | 84.82% | 68.55% | 68.03% | 0.0150 | 0.0718 | 0.0644 |
| | | | $k$ = 5 | 84.81% | 68.41% | 67.93% | 0.0147 | 0.0713 | 0.0639 |
| | | | $k$ = 7 | 84.76% | 68.06% | 68.09% | 0.0147 | 0.0714 | 0.0641 |
| | | | $k$ = 9 | 84.75% | 68.08% | 67.94% | 0.0147 | 0.0715 | 0.0640 |
| | | | $k$ = 11 | 84.80% | 68.17% | 68.17% | 0.0148 | 0.0716 | 0.0642 |
| | | $a$ = 0.3 | $k$ = 3 | 84.83% | 68.18% | 68.29% | 0.0148 | 0.0715 | 0.0642 |
| | | | $k$ = 5 | 85.03% | 67.80% | 68.33% | 0.0145 | 0.0712 | 0.0639 |
| | | | $k$ = 7 | 84.62% | 67.55% | 68.15% | 0.0147 | 0.0714 | 0.0641 |
| | | | $k$ = 9 | 84.68% | 67.58% | 68.17% | 0.0149 | 0.0717 | 0.0643 |
| | | | $k$ = 11 | 84.87% | 67.65% | 68.24% | 0.0151 | 0.0718 | 0.0645 |
| TSQ | | $a$ = 0.3<br>$k$ = 11 | $c_w$ = 0 | 85.05% | 67.94% | 65.84% | 0.0175 | 0.0741 | 0.0669 |
| | | | $c_w$ = 0.01 | 85.07% | 67.81% | 65.74% | 0.0174 | 0.0739 | 0.0668 |
| | | | $c_w$ = 0.02 | 84.84% | 67.75% | 65.50% | 0.0172 | 0.0738 | 0.0666 |
| | | | $c_w$ = 0.03 | 85.13% | 67.76% | 65.66% | 0.0171 | 0.0737 | 0.0665 |
| | | | $c_w$ = 0.04 | 85.11% | 67.86% | 65.72% | 0.0168 | 0.0733 | 0.0661 |
| | | $a$ = 0.3<br>$c_w$ = 0.02 | $k$ = 7 | 85.19% | 68.11% | 65.85% | 0.0165 | 0.0731 | 0.0659 |
| | | | $k$ = 9 | 85.02% | 67.65% | 65.70% | 0.0169 | 0.0735 | 0.0662 |
| | | | $k$ = 13 | 84.81% | 67.88% | 65.62% | 0.0175 | 0.0740 | 0.0668 |
| | | | $k$ = 15 | 85.07% | 67.58% | 65.54% | 0.0176 | 0.0742 | 0.0670 |
| | | $k$ = 11<br>$c_w$ = 0.02 | $a$ = 0.2 | 84.74% | 68.51% | 65.40% | 0.0162 | 0.0728 | 0.0656 |
| | | | $a$ = 0.25 | 84.89% | 67.94% | 65.67% | 0.0166 | 0.0731 | 0.0659 |
| | | | $a$ = 0.3 | 84.84% | 67.42% | 65.46% | 0.0181 | 0.0745 | 0.0674 |
| PB | | $\mathbf{H^B}$ | $c_p$ = 0.001 | 84.92% | 70.27% | 64.40% | 0.0161 | 0.0727 | 0.0655 |
| | | | $c_p$ = 0.01 | 85.40% | 68.54% | 65.66% | 0.0143 | 0.0708 | 0.0636 |
| | | | $c_p$ = 0.1 | 85.30% | 61.67% | 63.21% | 0.0180 | 0.0752 | 0.0671 |
| | | | $c_p$ = 1 | 84.05% | 58.46% | 60.26% | 0.0283 | 0.0857 | 0.0773 |
| | | | $c_p$ = 10 | 83.63% | 58.00% | 59.88% | 0.0312 | 0.0884 | 0.0802 |
| | | $\mathbf{H^B + H^C}$ | $c_p$ = 0.001 | 85.09% | 69.77% | 64.60% | 0.0160 | 0.0726 | 0.0654 |
| | | | $c_p$ = 0.01 | 85.54% | 68.60% | 65.94% | 0.0141 | 0.0706 | 0.0635 |
| | | | $c_p$ = 0.1 | 85.50% | 61.87% | 63.40% | 0.0172 | 0.0745 | 0.0664 |
| | | | $c_p$ = 1 | 83.95% | 58.33% | 60.49% | 0.0253 | 0.0829 | 0.0743 |
| | | | $c_p$ = 10 | 83.72% | 57.64% | 59.98% | 0.0289 | 0.0861 | 0.0778 |
| | | $\mathbf{H^B + H^C + H^F}$ | $c_p$ = 0.001 | 84.62% | 69.67% | 64.32% | 0.0159 | 0.0726 | 0.0654 |
| | | | $c_p$ = 0.01 | 86.18% | 68.39% | 66.51% | 0.0138 | 0.0702 | 0.0631 |
| | | | $c_p$ = 0.1 | 85.66% | 60.71% | 63.12% | 0.0180 | 0.0748 | 0.0667 |
| | | | $c_p$ = 1 | 83.75% | 57.68% | 60.19% | 0.0266 | 0.0839 | 0.0752 |
| | | | $c_p$ = 10 | 83.95% | 57.11% | 59.72% | 0.0298 | 0.0872 | 0.0785 |

## APPENDIX B. SIMPLE LINEAR REGRESSION

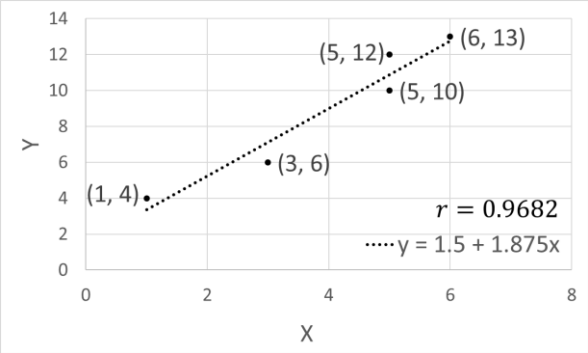



Fig. B.1. An example of simple linear regression.

Simple linear regression, described in more detail by Lane *et al*. (2014), is a model to derive how two random variables are linearly correlated. This section illustrates it by the example of two random variables $X$ and $Y$ with sample data $(x, y) \in \{(1, 4), (3, 6), (5, 10), (5, 12), (6, 13)\}$, as shown in Fig. B.1. In this paper, the Pearson's correlation coefficient $r$ is used, which is calculated by

$$r = \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})^2 \cdot \sum_{i=1}^{N}(y_i - \overline{y})^2}} \tag{B.1}$$

where $N$ is the total number of the sample data, $x_i$ and $y_i$ the $i$-th data of $x$ and $y$, and $\overline{x}$ and $\overline{y}$ the sample means of $X$ and $Y$ which are $\frac{1}{N}\sum_{i=1}^{N} x_i$ and $\frac{1}{N}\sum_{i=1}^{N} y_i$ respectively. The higher the absolute values of correlation coefficients are, the stronger the correlations between the two random variables. Correlation coefficients range from -1 to 1, and the correlation coefficient of the given example is $r = \frac{30}{\sqrt{16 \times 60}} \approx 0.9682$. A rule of thumb interprets the correlation coefficient as follows, though it depends on the purpose of the analysis and on the number of sample data. For the absolute value of $r$, $[0.9, 1]$ shows a very high correlation, $[0.7, 0.9)$ a high correlation, $[0.5, 0.7)$ a moderate correlation, $[0.3, 0.5)$ a low correlation, and $[0, 0.3)$ little if any correlation (Asuero *et al*., 2006). Thus, in the above example, $X$ and $Y$ are said to have a very highly positive correlation.

The regression line, the best-fitting straight line, is

$$y = \overline{y} + \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{N}(x_i - \overline{x})^2}(x - \overline{x}) \tag{B.2}$$

which predicts the values of $y$ given $x$. Thus, the regression line of the given example is $y = 1.5 + 1.875 \cdot x$, the dotted line shown in Fig. B.1. The sample standard error of the prediction on $Y$ is then calculated by

$$s = \sqrt{\frac{\sum_{i=1}^{N}(y_i - y_i')^2}{N - 2}} \tag{B.3}$$

where $y_i'$ is the prediction on the regression line given $x_i$. In the given example, the sample standard error is $s = \sqrt{\frac{3.75}{3}} \approx 1.1180$. A small value of sample standard error shows that the prediction can be trusted.

## 7. REFERENCES

Asuero, A. G., Sayago, A., & González, A. G. (2006). The correlation coefficient: An overview. *Critical Reviews in Analytical Chemistry, 36*(1), 41-59. doi:10.1080/10408340500526766

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning, 47*(2-3), 235-256. doi:10.1023/A:1013689704352

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., . . . Colton, S. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games, 4*(1), 1-43. doi:10.1109/TCIAIG.2012.2186810

Campbell, M., Hoane, A. J., & Hsu, F.-h. (2002). Deep Blue. *Artificial Intelligence, 134*(1), 57-83. doi:10.1016/S0004-3702(01)00129-1

Chang, H.-J., Fan, G.-Y., Chen, J.-C., Hsueh, C.-W., & Hsu, T.-s. (2017). *Validating and fine-tuning of game evaluating functions using endgame databases*. Paper presented at the 6th Workshop on Computer Games (CGW 2017), Held in Conjunction with the 26th International Conference on Artificial Intelligence (IJCAI 2017), Melbourne, Australia.

Chang, H.-J., & Hsu, T.-s. (2014). A quantitative study of 2 × 4 Chinese dark chess. In H. J. van den Herik, H. Iida, & A. Plaat (Eds.), *Computers and Games: 8th International Conference, CG 2013, Yokohama, Japan, August 13-15, 2013, Revised Selected Papers* (pp. 151-162). Cham: Springer International Publishing.

Chang, H.-J., Hsueh, C.-W., & Hsu, T.-s. (2015). *Convergence and correctness analysis of Monte-Carlo tree search algorithms: A case study of 2 by 4 Chinese dark chess*. Paper presented at the 2015 IEEE Conference on Computational Intelligence and Games (CIG), Tainan, Taiwan.

Chaslot, G. M. J. B., Winands, M. H. M., van den Herik, H. J., Uiterwijk, J. W. H. M., & Bouzy, B. (2008). Progressive strategies for Monte Carlo tree search. *New Mathematics and Natural Computation, 4*(3), 343-357. doi:10.1142/S1793005708001094

Chen, B.-N., Shen, B.-J., & Hsu, T.-s. (2010). Chinese dark chess. *ICGA Journal, 33*(2), 93-106. doi:10.3233/ICG-2010-33204

Chen, J.-C., Fan, G.-Y., Chang, H.-J., & Hsu, T.-s. (2018). Compressing Chinese Dark Chess Endgame Databases by Deep Learning. *IEEE Transactions on Games, PP*(99), 1-1. doi:10.1109/TG.2018.2802484

Chen, J.-C., Lin, T.-Y., Chen, B.-N., & Hsu, T.-s. (2015). Equivalence classes in Chinese dark chess endgames. *IEEE Transactions on Computational Intelligence and AI in Games, 7*(2), 109-122. doi:10.1109/TCIAIG.2014.2317832

Coulom, R. (2007). Computing elo tatings of move patterns in the game of Go. In H. J. van den Herik, W. Mark, U. Jos, & S. Maarten (Eds.), *Proceedings of the Computer Games Workshop 2007, Amsterdam, June 15–17, 2007* (pp. 113-124).

Gao, C., Mueller, M., & Hayward, R. B. (2017). Focused Depth-first Proof Number Search using Convolutional Neural Networks for the Game of Hex. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)* (pp. 3668-3674).

Gelly, S., & Silver, D. (2007). Combining online and offline knowledge in UCT. In *Proceedings of the 24th international conference on Machine learning* (pp. 273-280): ACM.

Henderson, P., Arneson, B., & Hayward, R. B. (2009). Solving 8 × 8 Hex. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)* (pp. 505-510): AAAI Press.

Hsueh, C.-H., & Wu, I.-C. (2015). DarkKnight wins Chinese dark chess tournament. *ICGA Journal, 38*(4), 249-251. doi:10.3233/ICG-2015-38411

Hsueh, C.-H., Wu, I.-C., Tseng, W.-J., Yen, S.-J., & Chen, J.-C. (2016). An analysis for strength improvement of an MCTS-based program playing Chinese dark chess. *Theoretical Computer Science, 644*, 63-75. doi:10.1016/j.tcs.2016.06.025

Ikeda, K., & Viennot, S. (2014). Efficiency of static knowledge bias in Monte-Carlo tree search. In H. J. van den Herik, H. Iida, & A. Plaat (Eds.), *Computers and Games: 8th International Conference, CG 2013, Yokohama, Japan, August 13-15, 2013, Revised Selected Papers* (pp. 26-38). Cham: Springer International Publishing.

Kishimoto, A., & Mueller, M. (2015). Game solvers. In R. Nakatsu, M. Rauterberg, & P. Ciancarini (Eds.), *Handbook of Digital Games and Entertainment Technologies* (pp. 1-20). Singapore: Springer Singapore.

Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In J. Fürnkranz, T. Scheffer, & M. Spiliopoulou (Eds.), *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings* (pp. 282-293). Berlin, Heidelberg: Springer Berlin Heidelberg.

Lai, M. (2015). Giraffe: using deep reinforcement learning to play chess. *CoRR, abs/1509.01549*.

Lanctot, M., Winands, M. H. M., Pepels, T., & Sturtevant, N. R. (2014). *Monte Carlo tree search with heuristic evaluations using implicit minimax backups*. Paper presented at the 2014 IEEE Conference on Computational Intelligence and Games, Dortmund, Germany.

Lane, D. M., Scott, D., Hebl, M., Guerra, R., Osherson, D., & Zimmer, H. (2014). *Introduction to statistics*. Rice University, Houston, TX.

Liskowski, P., Jaśkowski, W. M., & Krawiec, K. (2018). Learning to play Othello with deep neural networks. *IEEE Transactions on Games, PP*(99), 1-1. doi:10.1109/TG.2018.2799997

Nijssen, J. A. M., & Winands, M. H. M. (2011). Enhancements for multi-player Monte-Carlo tree search. In H. J. van den Herik, H. Iida, & A. Plaat (Eds.), *Computers and Games: 7th International Conference, CG 2010, Kanazawa, Japan, September 24-26, 2010, Revised Selected Papers* (pp. 238-249). Berlin, Heidelberg: Springer Berlin Heidelberg.

Pawlewicz, J., & Hayward, R. B. (2014). Scalable parallel DFPN search. In H. J. van den Herik, H. Iida, & A. Plaat (Eds.), *Computers and Games: 8th International Conference, CG 2013, Yokohama, Japan, August 13-15, 2013, Revised Selected Papers* (pp. 138-150). Cham: Springer International Publishing.

Pepels, T., Tak, M. J. W., Lanctot, M., & Winands, M. H. M. (2014). Quality-based rewards for Monte-Carlo tree search simulations. In T. Schaub, G. Friedrich, & B. O'Sullivan (Eds.), *Proceedings of the Twenty-first European Conference on Artificial Intelligence, ECAI 2014* (pp. 705-710): IOS Press.

Runarsson, T. P., & Lucas, S. M. (2014). Preference learning for move prediction and evaluation function approximation in Othello. *IEEE Transactions on Computational Intelligence and AI in Games, 6*(3), 300-313. doi:10.1109/TCIAIG.2014.2307272

Russell, S., & Norvig, P. (2009). *Artificail Intelligence: A modern approach* (3rd ed.): Pearson.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., . . . Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484-489. doi:10.1038/nature16961

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., . . . Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature, 550*(7676), 354-359. doi:10.1038/nature24270

Takeuchi, S., Kaneko, T., & Yamaguchi, K. (2010). Evaluation of game tree search methods by game records. *IEEE Transactions on Computational Intelligence and AI in Games, 2*(4), 288-302. doi:10.1109/TCIAIG.2010.2102022

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM, 38*(3), 58-68. doi:10.1145/203330.203343

Tian, Y., & Zhu, Y. (2016). *Better computer Go player with neural network and long-term prediction.* Paper presented at the 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico.

Tseng, W.-J., Chen, J.-C., & Wu, I.-C. (2017). DarkKnight wins Chinese dark chess tournament. *ICGA Journal, 39*(3), 163-165. doi:10.3233/ICG-170023

van den Herik, H. J., Plaat, A., & Hellemons, J. (2012). The 16th Computer Olympiad. *ICGA Journal, 35*(1), 50-50. doi:10.3233/ICG-2012-35111

van den Herik, H. J., Uiterwijk, J. W. H. M., & van Rijswijck, J. (2002). Games solved: Now and in the future. *Artificial Intelligence, 134*(1-2), 277-311. doi:10.1016/S0004-3702(01)00152-7

Yen, S.-J., Chen, J.-C., Chen, B.-N., & Tseng, W.-J. (2013). DarkKnight wins Chinese dark chess tournament. *ICGA Journal, 36*(3), 175-176. doi:10.3233/ICG-2013-36315

Yen, S.-J., Chiu, S.-Y., & Wu, I.-C. (2010). MoDark wins the Chinese dark chess tournament. *ICGA Journal, 33*(4), 230-231. doi:10.3233/ICG-2010-33410

Yen, S.-J., Chou, C.-W., Chen, J.-C., Wu, I.-C., & Kao, K.-Y. (2015). Design and implementation of Chinese dark chess programs. *IEEE Transactions on Computational Intelligence and AI in Games, 7*(1), 66-74. doi:10.1109/TCIAIG.2014.2329034