Opponent Model Selection Using Deep Learning

Hung-Jui Chang[1], Cheng Yueh[2], Gang-Yu Fan[3], Ting-Yu Lin[3], and Tsan-sheng Hsu[3]

[1] Department of Applied Mathematics, Chung Yuan Chrisitan University
[2] Department of Computer Science and Information Engineering, National Taiwan University
[3] Institute of Infomation Science, Academia Sinica

Abstract. It is observed that the strength of many game programs varies a lot in a tournament when facing opponents of different styles. In order to adapt one's playing strategy when playing against different programs to obtain the best possible outcome, it is important to estimate the strength of the opponent. We use a neural network to predict the strength or style of an opponent via the first 20 plys of a game in Chinese Dark Chess. According to the prediction result, a contempt factor is used when playing against the target opponent after the first 20 plys. Both the experiment result and the tournament results show that this approach can improve the rank of our program in a competitive tournament.

1 Introduction

In the domain of computer games, predicting the strength of the opponent player is one of the essential techniques to improve the level of the computer programs [7, 8]. For a specific opponent, the historical data can be used to estimate its strength [8]. Moreover, one can use the estimated results to implement a particular strategy when playing against the target opponent [7, 6].

One of the strategies used in computer Chess in finding a good opponent model is called the *contempt factor* [7], which is a value indicating the strength difference between two programs. When this value is non-negative, we assume the opponent has an equal or greater strength than ours. In this case, we accept a draw even if we currently have a small advantage. On the other hand, if the contempt factor is negative, the opponent is assumed to be weaker than ours. In this case, we do not accept a draw even if we currently have a small disadvantage. That is, we try to force a draw when the opponent is stronger and avoid a draw result when the opponent is weaker.

There are many participants in a computer game tournament [3]. Some participants have participated the tournament several times, but there are always some newcomers. We can use the previous results to estimate the strengths of those who have participated the tournament before. For new comers, we need to design a method to predict their strengths. Knowing that your opponent is weak, we can afford to scarify the top ranked piece and to eventually win the

game in Chinese Dark Chess whose goal is to capture all of the opponent's pieces. This is particular true for Chinese Dark Chess which is known for being able to draw easily by chasing opponent's most important piece, namely King, using the least valued piece, namely Pawn. Figure 1 shows an example of a forced draw. When the black pawn moves from B5 to C5, the red king's only way to escape is to move from C4 to B4. The black pawn can continue chasing the red king by moving from C5 to either C4 or B5. Furthermore, Chinese Dark Chess is a non-deterministic game. By flipping a dark (unrevealed) piece, a game may turn from advantage to disadvantage. For example, in Figure 2a, the red side may consider flipping the unrevealed piece in C5. If the revealed piece is a black cannon as in Figure 2b, the red king can capture one black cannon and two black guards, which is a huge advantage. On the other hand, if the revealed piece is a black pawn as in Figure 2c, the red king will be captured directly, which is a huge disadvantage. In this paper, we provide a way to estimate the opponent's program's strength by feeding the first 20 plys of a game into a deep-learning neural network model [4, 5].

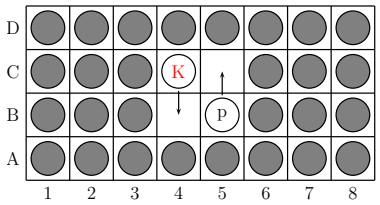

Fig. 1: An example of using black pawn to chase red king.

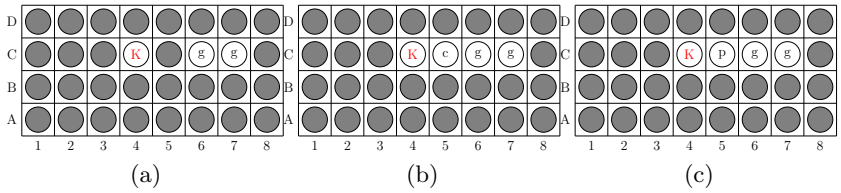

Fig. 2: An example of revealing result causes advantage and disadvantage result.

In this paper, we use a deep learning network to predict the opponent's strength base on the first 20 plys of the game. The training data are collected from the previous TAAI, TCGA and ICGA tournaments, and the self-played data from three programs with different strengths. The experiment results show

that the opponent prediction model can improve the expected score of our program in a competitive tournament.

The remains of this paper is organized as follows: In Section 2, we briefly introduce the game of Chinese Dark Chess and the data sets of the deep-learning models. In Section 3, we describe the predicting methods. In Section 4, we show the experiment results. Finally, In Section 5, we conclude this paper.

2 Background

In this section, we first introduce the terminologies used in this paper. Then we describe the game *Chinese Dark Chess* which our program plays. Finally, we describe the background of our problem and the main idea of how to use the contempt factor in solving our problem.

2.1 Terminologies

A *board position* (b) denotes the state of a game. A *legal ply* (p) is an action that can transfer the current board position into the next board position, that is $T(b, p) = b^{'}$, where $b^{'}$ is the board position after ply p is applied to board position b. The *ply sequence* of a game is represented as $p_1, p_2, \ldots, p_n$, where p_i is the i-th ply of the game. The *board position sequence* of a game is represented as $b_0, b_1, \ldots, b_n$, where b_i denotes the board position after ply p_i is applied on board position b_{i-1}. An example of the relationship between the board positions and the moves is shown in Figure 3. The *opponent ply sequence* (OPS) is the sequence of all the opponent's plys. If the opponent makes the first ply in the game, then the OPS is $\{p_1 p_3 p_5 \ldots\}$, and if we make the first ply in the game, then the OPS is $\{p_2 p_4 p_6 \ldots\}$, respectively.

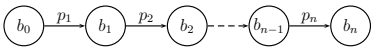

Fig. 3: An illustration of board positions and ply sequences.

2.2 Chinese Dark Chess

Chinese Dark Chess (CDC) [2], also known as Banqi, is one of the most popular variation of *Chinese Chess* (CC) in the southeast Asia. This game is played by the same pieces set on the half part of the original game board of CC. There are two colors in the CDC, that is, the red side and the black side. Each side has 16 pieces, 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). In the follows, we use the uppercase character to represent the pieces of the red

Table 1: Piece Sets of Chinese Dark Chess.

Name	King	Guard	Minister	Rook	Knight	Cannon	Pawn
Red	K	G	M	R	N	C	P
Black	k	g	m	r	n	c	p
# Pieces	1	2	2	2	2	2	5

side and the lowercase character to represent the pieces of the black side. The piece set is listed in Table 1. In this game, only a piece with higher rank can capture an opponent piece of lower rank. The order between two pieces, which is used to decide which one can capture the other, is described by three rules: 1) $K > G > M > R > N > C$, 2) $P > K$ and 3) $G, M, R, N > P$. Each piece in the CDC has two faces, the front-face and the back-face. The front-face of each piece is crafted with the name of that piece. The back-face of all the pieces are identical.

In the beginning of the game, all the pieces are shuffled and placed with the front-face facing-down. An illustration of the start position of the CDC is shown in Figure 4. In the beginning of the game, the first player chooses one facing-down piece and reverses it. In the remains of the game, the first player owns those pieces with the same color of the first revealed one, and the second player owns pieces of the other color.

In each turn, one player can either move a piece of his/her side to the adjacent square, revealed one facing-down piece or apply the special jump-move for cannon. A piece can only be moved to the adjacent square which is empty or is occupied by an opponent's piece with a lower rank, in the later case, the opponent's piece is captured by our piece and removed from the game. In Figure 5, the legal moves of red minister at C3 are C3–C2 and C3–B3, since C2 is an empty square and the black rook at B3 is lower-ranked than the red minister. Note that C3–C4 is an illegal move since C4 is occupied by the red knight which is the same as the red minister. Also, C3–D3 is an illegal move because the black king is higher-ranked than the red minister. A *jump move* is a special capture move for the cannons. When there is only one piece in between the cannon and the target piece, the cannon can use the jump move to capture that target piece. For example, in Figure 5, the red cannon in B5 can use a jump move to capture either the black pawn in B5, the black rook in B3, and the black guard in B8. Note that for each cannon, there exist at most one jump move alone the short edge, that is B5 to D5 in our example. There can be two jump moves along the long edge, that is B5 to B3 and B5 to B8 in our example. We denote these two moves as the jump moves along the long edge to the left and the jump move along the long edge to the right, respectively. The player who cannot make any legal move loses the game. As a remark that if all pieces of one side are captured, then he naturally cannot make any legal move.

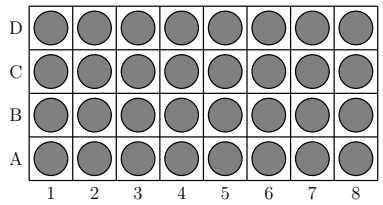

Fig. 4: The initial position of Chinese Dark Chess.

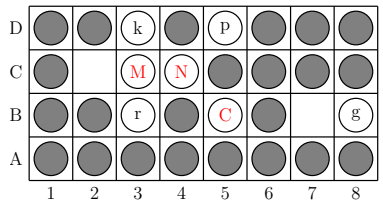

Fig. 5: The legal moves of the red minister are C3–C2 and C3–B2. And the legal jump moves of the red cannon at B5 are B5–D5, B5–B3, and B5–B8.

2.3 Game Tournament and Expected Score

In the competition of computer games, e.g, the World Champion of Computer Chess or Computer Olympics [3], the Swiss-system tournament is the most common applied playing rule. In this rule, one win gains 2 points, one draw gains 1 point. In a tournament, we can rank other participations according to their past earned scores. We defined three different ranks, S, E and W, that is, those who are *stronger* than us (S), those who are relatively *equal* to us (E), and those who are *weaker* than us (W). We have a winning probability $P_{W,S}$, $P_{W,E}$ and $P_{W,W}$ when playing against those programs with rank S, rank E, and rank W, respectively. Simultaneously, we have a losing probability $P_{L,S}$, $P_{L,E}$ and $P_{L,W}$ when playing against those programs with rank S, rank E, and rank W, respectively. Finally, the probability of having a draw result when playing against those programs with rank S, rank E, and rank W is $P_{D,S}$, $P_{D,E}$ and $P_{D,W}$, respectively. Let the numbers of participations with rank S, rank E, and rank A, be N_S, N_E, and N_W, respectively. The *expected score* (ES) is defined as

$$ES = \sum_{i \in \{E,S,W\}} N_i \times (2P_{W,i} + P_{D,i}). \tag{1}$$

In order to increase ES, we have two different approaches: 1) increasing $P_{W,W}$ by decreasing $P_{D,W}$; and 2) increasing $P_{D,S}$ by decreasing $P_{L,S}$. In the first approach, we try to avoid a draw when we still have a chance to win, usually this is what we want when we play against with weaker opponents. In the second approach, we try to avoid a lose by using a force drawing strategy.

2.4 Contempt Factor and Threefold Repetition

In Chess and many other games, a threefold repetition is often treated as a draw. Therefore many programs return a value of 0 when a threefold repetition is occurred. In Figure 6, the left tree shows one move with a value of 0 and another move with a small positive value $+\epsilon$. In the right tree shows another example that one move with a value of 0 and another move with a small negative value $-\epsilon$. The returned value of 0 will be compared with other moves' return value in the search algorithm. Traditionally, if there is one move whose value is greater than 0, no matter how small it is, we will not choose the move with value of 0. However, when we play against a strong program, the position with a small positive value may not lead us to a win result. Moreover, the risk of being beaten by a stronger opponent is high when we only have a small advantage. On the other hand, when we play against a weaker program, we will tolerant the small amount of disadvantage, because we knew we still have a chance to win that game even currently we only have a small disadvantage.

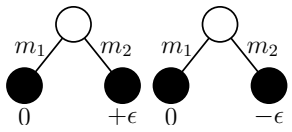

Fig. 6: An example of plys with values of 0.

The strategy described above is the main idea of using the contempt factor [7]. However, we are not sure about the strength of opponents before a tournament. For those who have participate in the tournament before, we can use its past results to estimate. For those new comer, we need a reliable method to estimate their strength during the competition.

2.5 Chinese Dark Chess Programs

In this subsection, we describe the three CDC programs involved in our experiment. The first program is "Yahari", who participated in the computer game tournament of TAAI, TCGA and Computer Olympic for several years, and achieves the first place in TAAI 2013 and TCGA 2014, the second place in 18-th and 19-th Computer Olympic, and the third place in TCGA 2019, TCGA 2020, and TAAI 2019. The second program is "PupilDarkChess" which achieves the first place in TCGA 2020, TCGA 2019 and TAAI 2019. The third program is "YanYu" who has participated from TCGA 2018, and does not win any top 3 place. It appears that "Yahari" is stronger than "YanYu" and is slightly weaker or equal to the strength of "PupilDarkChess". "Yahari" is the target program we want to improve. "Yahari" has an evaluation function with a range of -1 to 1 where -1 means loss, 1 means win, and 0 means draw.

3 Method

In order to determine the strength of an opponent, we collect the OPSs generated during the game as the input, and feed these input data into an *opponent ranking neural network* (ORNN) model to predict its rank. After obtaining the estimated strength, we set a corresponding contempt factor when playing against that program. In order to determine the proper value of the contempt factor, we calculate the average winning rate when Yahari plays against an opponent using different contempt factors.

3.1 Input Data of the ORNN

The input of our ORNN model contains one board position after the 3rd ply is played and the following seven opponent's plys. There are two reasons to use the board position after the 3rd ply is played as the initial board position. First, although the start board position of the CDC is random, there are strategies to deal with what pieces should be in the first few plys [1]. An example is when a high ranked opponent's piece is revealed, we cannot flip piece around that piece. These fixed strategies easily confuse a prediction program. Second, the predicted result is used to determine the strategy when playing with this opponent, that is, the contempt factor. And the contempt factor affects the search behavior of a program only when a threefold repetition is occurred. A three threefold repetition are rarely happened in the first 20 plys. Thus we use the first 20 plys in the OPS to predict the rank of the opponent's program without any risk.

3.2 Input Planes of the ORNN

There are two kinds of data in the input data of the ORNN: 1) the initial board position and 2) the following opponent's plys.

We use 30 planes to represent one initial board position, each plane is a 4×8 block. The first 14 planes represent the locations of all kinds of the revealed pieces, and the 15th and 16th planes represent the locations of unrevealed pieces and the empty squares, respectively. The first 16 planes are the *piece location plane* (PLP), which denote the location information about the kind or the state (unrevealed / empty).

We use 1 to denote one square is under certain condition. For example, the black king is in D-3 in Figure 5. Thus the corresponding PLP for the black king marks 1 on the square of D3. On the other hand, 0 is used when the condition is not fulfilled. Therefore all the other squares except D3 for the PLP of black king mark with 0. In Figure 5, the plane representing the black king is the left matrix in the Figure 7. The planes of the unrevealed squares and the empty squares are shown in the middle and the right of Figure 7, respectively.

The 17th to 30th planes are the numbers of unrevealed pieces, respectively for the two sides, the *unrevealed piece plane* (UPP). Each plane represents one

0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Fig. 7: The input planes of the black king, the unrevealed squares and the empty squares of the board position shown in Figure 5.

kind of the pieces according to the canonical piece order of CDC, that is "KGMRNCPkgmrncp." For example, the 17th plane denotes the number of unrevealed red king, and the 18th plane denotes the number of unrevealed red guards.

The range of each value in the UPP is -1 to 1. The value -1 means the number of unrevealed pieces in that kind is minimal, that is 0. And the value 1 means the number of unrevealed pieces in that kind is maximal, that is 1 for king, 5 for pawn, and 2 for all other kind of pieces. If one piece has neither the minimal number of pieces nor the maximal number of pieces, then the value is set to $2 \times \frac{\#\text{ unrevaled pieces}}{\text{total number of pieces}} - 1$. The mapping between the number of unrevealed pieces and the value in the UPP is shown in Table 2.

Table 2: The corresponding value of unrevealed pieces of each kind.

	K/k	G/g	M/m	R/r	N/n	C/c	P/p
0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
1	1.0	0.0	0.0	0.0	0.0	0.0	-0.6
2		1.0	1.0	1.0	1.0	1.0	-0.2
3							0.2
4							0.6
5							1.0

The basic idea to represent a move in the CDC in our ORNN is to transform the original source-destination pair into a source-direction pair. We represent a move by the source position and how it moves to the destination move. In each round, there are eight possible moves: 1) up; 2) right; 3) down; 4) left; 5) flip an unrevealed piece; 6) use a cannon to capture an opponent's piece along the short edge; 7) use a cannon to capture an opponent's piece along the long edge to the left; and 8) use a cannon to capture an opponent's piece along the long edge to the right. We use a 4×8 block to encode the above information.

For example, in the left board of Figure 8, the red king can either move up, right, down, or left. If the red king moves up, from C3 to C4, the corresponding input plane, the move-up plane, is represented as the middle matrix in Figure 8, and the rest of 7 move-planes are all zeros. If the red side plays B6–D6 to capture

the black pawn, then the corresponding jump-right plane is recorded as the right matrix in Figure 8, and the rest of the 7 move-planes are all zeros.

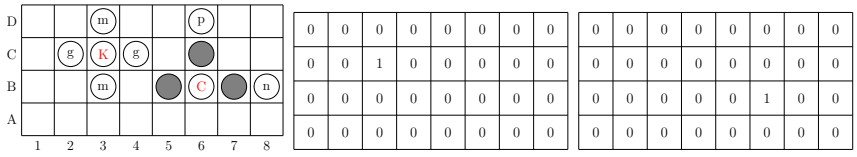

Fig. 8: The input planes of the red king's moves and the red cannon's jump-move.

We use 56 planes, called the *opponent-ply plane* (OPP) to represent the opponent's following 7 plys from the initial position. In total, we use 86 planes to input one board position and the corresponding board positions. The details of the input planes are summarized in Table 3.

Table 3: Summary of the input planes.

Type	Details	# planes
PLP	Locations of revealed pieces, unrevealed pieces and empty squares	16
UPP	Number of unrevealed pieces of each kind	14
OPP	The opponent's following 7 plys	56

3.3 Output of the ORNN

As we mentioned in Section 2.3, the strength of an opponent is estimated to be in three different levels where corresponds to the code of S, E, and W and in ORNN model as 0, 1 and 2, respectively.

3.4 Expected Score

During a tournament, a winning game, a draw game, and a losing game gains 2, 1, and 0 score, respectively.

We want to use the predictions from our ORNN to determine the contempt factor to hopely gain a higher score during a tournament.

4 Experiments

The experiment contains two parts. In the first part, we show the expected $P_{W,S}$, $P_{W,E}$ and $P_{W,L}$ when using different contempt factors. In the second part, we use the expected winning probability gathered from the first experiment to calculate the ES gain in ORNN.

4.1 Experiment Settings

In the first part of the experiment, we let our program "Yahari" to play against the other two programs with three different contempt factors, -0.6, 0 and 0.2. We play 400 games for each setting.

When we have no information about the opponent's strength, we can only assume the opponent strength is similar to our program. We use 0 as the contempt factor in this case. If our program is stronger than the opponent's program, then we will use a lower contempt factor. Since our program is stronger, then it can gain back advantage from slightly disadvantage situations in most of the case due to the facts that a weaker makes mistake easily. We will use -0.6 as the contempt factor in this case. That is our program refuses to get a draw result when it still has a chance to fight back. However, when the opponent's program's strength is stronger than our program, it is hard for our program to turn the situation from a small advantage, says board positions with evaluation value around 0.1 or 0.2, into a winning result. Therefore we accept a draw result when we only have such a small advantage when playing against a stronger opponent. Hence we use 0.2 as the contempt factor.

In order to train our ORNN, we collect the game records generated by our program, "Yahari", plays against the other two programs: "PupilDarkChess" and "YanYu". We simulate 40,000 games for each pair of programs. Each program in the pair plays as the first player in half of the games. There are a total of 80,000 games. We use 80% of the game records as the training data and the remaining 20% as the test data.

4.2 The Effect of Using the Contempt Factor

In order to maximize the effectiveness of our opponent model prediction, the selection of value of contempt factor is also important. In the following we test three different values for the contempt factor: -0.6, 0.0 and 0.2. We let our program "Yahari" with these three different contempt factor to play with the other two programs and our program itself.

Table 4: Yahari v.s. YanYu.

Total	Win	Draw	Lose	Score
-0.6	127	49	24	305
0.0	123	57	20	303
0.2	120	64	8	304

In Table 4, we use three different contempt factors: -0.6, 0.0 and 0.2. For each setting, we play 200 games against "YanYu" which is a weaker program. The experiment results show that if we play against a weaker opponent, the

setting contempt factor does not affect the expected score. However, the experiment results show that when using contempt factor as -0.6, the number of draw game decreases, and the number of winning games increase. Note that when the number of draw games decreases, both the number of winning and losing games increase. That is, when playing against a weaker opponent, we prefer to have a non-draw outcome.

Table 5: Yahari v.s. PupilDarkChess.

Total	Win	Draw	Lose	Score
-0.6	42	21	137	105
0.0	41	37	122	119
0.2	48	62	90	158

In Table 5, we use three different contempt factors: -0.6, 0.0 and 0.2. For each setting, we play 200 games against "PupilDarkChess" which is a stronger program. The experiment results show that if we play against a stronger opponent, the contempt factor helps us to gain a higher expected score. When we set the contempt factor to 0.2, we force a draw when we only have a small advantages. The experiment results show that we have a much higher expected score, 158, than the the expected score 119 as in the original setting. The experiment results also show that if one have a wrong guess and use -0.6 as the contempt factor, it causes little harm as far as the expected score is concerned.

Table 6: Yahari v.s. Yahari.

Total	Win	Draw	Lose	Score
0 v.s.- 0.6	97	23	80	217
0 v.s. 0.2	83	55	62	221
-0.6 v.s. 0.2	90	24	86	204
0 v.s. 0	77	45	78	199

In Table 6, we show the self-play result with different contempt factors. The experiment result shows that when a program with a contempt factor value 0 plays against a program with a contempt factor value 0.2, the one with a contempt factor 0 has a slight advantage. We believe the opponent forces a draw even when is has a small advantage. The experiment result also shows that when a program with a contempt factor value 0 playing against a program with a contempt factor value -0.6, the program with the contempt factor 0 also has a small advantage since the opponent tries to fight back even if they are currently

behind. Therefore, when playing against with the programs with a strength that is similar to our program, it is better to use 0 as the contempt factor.

4.3 Experiment Results

In Table 7, we show the confusion matrix of our ORNN. The experiment results show that when the opponent is weaker, equal or stronger to our program, the prediction rate is 0.877, 0.863 and 0.855, respectively. And the overall prediction accuracy in the training data and validation data are 0.8756 and 0.8717, respectively.

Table 7: Experiment results of ORNN.

	Predict		
	W	E	S
W	0.877	0.043	0.080
E	0.046	0.863	0.091
S	0.071	0.074	0.855

5 Conclusions

In this paper, we propose a novel method to predict the opponent's relative ranking in the CDC tournament by using the neural network. The experiment results show that the training accuracy is 0.8756 and the prediction accuracy is 0.8717. According to the experiment results, we also found the best contempt factor playing against a stronger opponents is 0.2 and the contempt factor for playing against a weaker opponent is -0.6. By using the above setting, the expected score increases by 0.07 for "Yahari". By applying these methods, our program "Yahari" has improved from the 3rd place in the 2019 and 2020 TCGA tournament to the 2nd place in the 2021 TCGA tournament.

References

1. Bo-Nian Chen and Tsan-sheng Hsu. Automatic generation of opening books for dark chess. In *International Conference on Computers and Games*, pages 221–232. Springer, 2013.
2. Bo-Nian Chen, Bing-Jie Shen, and Tsan-sheng Hsu. Chinese dark chess. *Icga Journal*, 33(2):93–106, 2010.
3. Jr-Chang Chen, Ting-Yu Lin, and Gang-Yu Fan. Yahari wins the chinese dark chess tournament. *ICGA Journal*, (Preprint):1–4, 2020.

4. Aaron Davidson, Darse Billings, Jonathan Schaeffer, and Duane Szafron. Improved opponent modeling in poker. In *International Conference on Artificial Intelligence, ICAI00*, pages 1467–1473, 2000.
5. He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813. PMLR, 2016.
6. Man-Je Kim and Kyung-Joong Kim. Opponent modeling based on action table for mcts-based fighting game ai. In *2017 IEEE conference on computational intelligence and games (CIG)*, pages 178–180. IEEE, 2017.
7. H Jaap van den Herik, HHLM Donkers, and Pieter HM Spronck. Opponent modelling and commercial games. *Proceedings of the IEEE*, pages 15–25, 2005.
8. Steven Walczak. Improving opening book performance through modeling of chess opponents. In *Proceedings of the 1996 ACM 24th annual conference on Computer science*, pages 53–57, 1996.