Towards master-level play of Shogi

by

JUN NAGASHIMA

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Hiroyuki Iida

School of Information Science
Japan Advanced Institute of Science and Technology

March, 2007

Abstract

This thesis presents our research that aims at defeating the human champion by computer Shogi TACOS we develop. The challenges to make a computer play games have been attempted since computers were invented. After the computer Chess machine defeated the human World Chess Champion in 1997, the interest of researches changed to more complex games such as Amazons, Shogi and Go. We have chosen Shogi for our subject because of the complexity. Moreover, we can expect big impact if a computer defeats the human champion since Shogi is the most popular board game in Japan.

In this study, we especially focus on the drawbacks of computer Shogi in the opening. Although the long-term view is required for planning a strategy in opening game, it is hard to implement such an idea on a computer. In complex games, a computer prepares an opening database (so-called *opening book*) that contains thousands moves often played by masters, and using that database a computer can play the master-level opening.

Although we can also use opening book in Shogi, it is not exhaustive. Therefore, to improve the opening play of computer Shogi, we tackle the following challenges:

- Using an opening book effectively in in-book positions.
- Playing the stable opening game even in out-of-book positions.

As the former challenge, we prepare a large opening book automatically made from thousands of master games and tune the book that a computer would select a prepared opening strategy. We propose a tuning method that tunes an opening book through lots of self-playing games. Using the opening book tuned by proposed method, TACOS has played a reasonable opening game in all tournaments and has obtained good results.

As the later challenge, we improve position evaluation in opening game by evaluating the formation that one would make in the opening stage. We enhance the piece-square tables that are commonly used in computer Shogi to make and evaluate formations. We also introduce some methods to evaluate formations more correctly. By implementing those methods on TACOS, we can improve its opening play remarkably in the out-of-book positions.

In addition to improve the playing level in opening, we also tackle other two problems that have to be improved in this paper. One is an enhancement of the use of transposition tables. To search effectively with valuable information stored during the previous search, we propose a method that uses two different transposition tables alternatively. Another one is the opportune time recognition of attacking. The first fights that start after constructing a formation are one of the weak points of computer Shogi. To tackle this problem, we deal with the edge attack, one of the attacking that a computer is poor at, while examining the possibility of attacks in a position and searching attack lines deeply when a position is regarded as an opportune timing for the attacking.

Implementing those measures and other enhancements, the playing strength of TACOS was improved awfully. These improvements lead some fruits in several tournaments such as 1st prize on the 10th Computer Olympiad and 4th place on the 16th World Computer Shogi Championship. In addition, TACOS could drive a professional Shogi player into a corner on an open game played on September 2005.

Key Words: **computer Shogi, opening book, piece development, book tuning, evaluation function, game-tree search**

Preface

First of all, I would like to thank my supervisor professor Hiroyuki Iida for his stimulating efforts to teach me scientific writing and thinking, for creating an excellent environment of my studies. I have learned a lot from his discussions and lectures. Next, I thank professor Akira Shimazu, professor Satoshi Tojo, professor Kentaro Torisawa and professor Bruno Bouzy for being the members of the assessment committee. Professor Shimazu is a famous researcher who has wide knowledge in the area of artificial intelligence, especially in the area of natural language processing. Professor Tojo has wide knowledge in natural language processing and multi-agent system. Part of this research was done under the guidance of professor Torisawa. I want to emphasize his advice and kindly supports. Professor Bouzy is famous researcher of artificial intelligence, especially in the area of computer games. I have learned a lot from their comments or the discussions with them.

I would like to thank professor Jin Yoshimura and doctor Nobusuke Sasaki for their kindly advices. I also would like to thank my colleagues at the Computer Game Research Institute in Shizuoka University and Research Unit for Computers and Games. Doctor Tsuyoshi Hashimoto, the leader of TACOS Project, always had time to give me advice. Professor Makoto Sakuta has given me many pieces of valuable advices and remarks. In addition, TACOS Project members (Yoichiro Kajihara, Masahumi Taketoshi, Takeaki Hamada, Keigo Matsubara, Akihiko Sano, Tomoki Murata and Jun'ichi Hashimoto) have helped and inspired me. I thank doctor Alessandro Cincotti for his careful and detailed reading of my English articles. His suggestions for correcting English writings were helpful. I also thank other members for their supporting my study.

Finally, I want to thank my family and friends for their supporting me in various aspects.

The research was supported by the Grant-in-Aid for Exploratory Research of the Japan Society for the Promotion of Science (#1452).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we present our research that aims at defeating the human champion by computer Shogi we develop. The challenges to make a computer play games have been attempted since computers were invented. Especially, researches on computer Chess had prosperously been tackled in the West because the game of Chess is widely regarded as a symbol of human intelligence. The subject of games has the following two advantages as compared to difficult problems that we have in daily life.

- The goal is clearly defined.
- The means for reaching the goal is also clearly defined.

Before dealing with difficult problems, we have to conquer these easier problems. In connection with this purpose, Shannon mentioned that "a satisfactory solution of this problem will act as a wedge in attacking other problems of a similar nature and of greater significance" in his paper [1]. Moreover, Kronrod mentioned that "Chess is the Drosophila of Artificial Intelligence" in 1965. With these mottos, many researchers have been working in the domain of computers and games.

1.1 Short Survey of Computers and Games

In 1949 Shannon introduced many important ideas that underlie various computer game programs [1]. The framework shown in that paper is that evaluating many positions that can be reached from a position by some plies of moves, and comparing which move leads the most advantageous position by propagating evaluation values by the minmax algorithm. After that paper, lots of enhancements and new methods have been suggested, and computer Chess programs became stronger and stronger. In 1997 Deep Blue [22], the computer Chess machine developed by IBM defeated the then-reigning World Chess Champion Garry Kimovich Kasparov in a six-game match [7]. Nowadays, computer Chess programs become stronger; programs that run on a personal computer become as strong as the human champion, whereas Deep Blue was a massively parallel system designed for only playing Chess.

In addition to Chess, many researchers have dealt with other various games while aiming at the following purposes:

- Making a strong computer program to defeat the human champion in the target domain.

Table 1.1: Complexity of game tree

game	complexity
Othello (6×6)	10^{30}
Checkers	10^{31}
Othello (8×8)	10^{58}
Go-Moku	10^{70}
Chess	10^{123}
Hex	10^{150}
Chinese Chess	10^{150}
Amazons	10^{220}
Shogi	10^{220}
Go	10^{360}

- Solving games to determine the game-theoretical values.

As the early works, LOGISTELLO [23], the computer Othello program, defeated the human champion in 1997. In the domain of Backgammon, TD-GAMMON [19, 24] became as strong as the human expert by Temporal Difference learning [4], one of the unsupervised learning methods. In the domain of Checkers [20], computers also became stronger than the human champion. As for the second purpose, some simple games such as Tic-Tac-Toe, Gomoku and 6x6 Othello have been solved [25]. In the domains of Checkers [26–28] and Hex [29–31], researchers are trying to solve these games. Other related works in this direction are presented in [32–37]. The other topic is to automatically create a strong computer program from the rules of the target game without implementing any knowledge of that game [38, 39].

When we study on some games, we have to consider the complexity of each game. In Table 1.1, we show the game-tree complexity of some games [8–11]. The numbers shown in the right column mean the order of positions in the average game tree. For example, there are about 80 legal moves in a position and a game finishes at 115 plies of moves in Shogi. Therefore, an average game tree has 80^{115} ($\approx 10^{220}$) terminal positions.

The games that have already been solved or games for which a computer became stronger than the human champion are comparatively simple. Therefore, the interest of researches changed to more complex games such as Amazons, Shogi and Go. In these games, the strongest computer programs are weaker than the human champion. For instance, the top computer Shogi programs are as strong as amateur 5th dan players while the top Go computers are as strong as amateur 1st kyu players.

Like Chess in Western, we have Shogi, a traditional Chess-like board game in Japan. We have chosen Shogi for our subject because of the following two reasons. The first one is that Shogi is the most popular board game in Japan. Since there are many amateur and professional Shogi players in Japan, we can expect big impact if a computer defeats the human champion. The second reason is related to its complexity. The characteristics of Shogi are:

- The board is larger than Chess (Shogi 9x9, Chess 8x8).

- Players can reuse captured pieces.

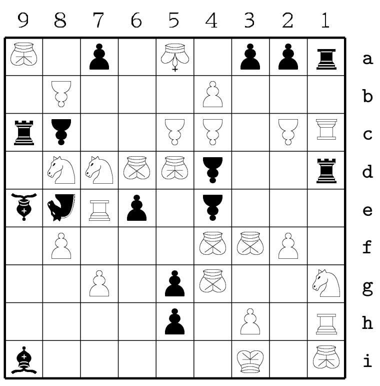


Figure 1.1: The initial position of Microcosmos

- Most Shogi pieces have less mobility than Chess pieces.
- The complexity of Shogi is higher than Chess.

Because of its complexity, some problems that are not caused in Chess are caused, and countermeasures for those problems have been proposed.

One of the most successful results in Computer Shogi is mating search for checkmate problems called *Tsume Shogi* in Japanese. Tsume Shogi is a kind of puzzles for which the goal is mating the defender's King by successive checks. There are some problems that are solved over 100 plies of moves because both players can drop captured pieces in Shogi. It is hard to solve such problems by searching with typical depth-first algorithms. Therefore, best-first search algorithms are used to solve those problems. After the depth-first algorithms that behave like as best-first algorithms [52,53] were invented, the famous problem entitled "Microcosmos" (Figure 1.1) with the longest solution sequence of 1525 plies were solved in 1997. Nowadays, these algorithms are used not only in computer Shogi but also in other domains and obtain good results.

In the field of Tsume Shogi, a computer has the ability for solving problems much better than the human champion. The playing strength of top computer Shogi is regarded as amateur 5th dan on the whole. On the other hand, there are lots of problems to improve in computer Shogi. Especially, the playing strength in the opening game is regarded only as 1st kyu. This weakness of opening play often causes some mistakes in the opening game, and it is hard to compensate such mistakes in the endgame. Therefore, the improvement of opening play becomes one of the biggest assignments for computer Shogi.

To our knowledge, there has been no extensive study on the opening play for computer Shogi because of the following three reasons. First, the task of improving the opening play was postponed until computers become sufficiently strong. This is because that the improvement in the opening game cannot work well if computers often lose games by mistaking in the middle game and endgame. Recently, computer Shogi programs become sufficiently strong both in the middle game and endgame, thus now we can tackle the problems in the opening game.

The second reason is that it is hard to evaluate the effectiveness of improvements in the opening game. Self-playing experiments, where a computer plays games against itself instead of playing against other players, are often used for performance tests. An improvement in the middle game or endgame search engine has direct influence to the results of self-playing experiments because a computer has to play some few moves after the positions where the improvement works. On the other hand, an improvement in the opening game cannot have influence to the results of self-playing experiments because a computer has to play quite a few moves after the positions where the improvement works. This means that the results of self-playing experiments can be influenced more often by the moves that are not affected by the improvement. Although playing games on the Internet Shogi server can be considered as another way for the performance test, it takes expensive costs in implementing and playing many games on the Internet. We do not have any solution to this problem, but in this study, we have to evaluate the effectiveness of improvements through self-playing games with considering the quality of play in games played on the Internet.

The third reason is that the research to improve the opening play of computer Shogi can be specific. In the study on the depth-first search algorithm used in Tsume Shogi, Shogi specific domain knowledge has not been used. Therefore, these algorithms can easily apply to other problems. On the other hand, in the study on the opening play, we have to deal with the high-level domain knowledge. It is hard to apply the improvements to other problems directly. However, we may be able to make a contribution to a methodological way if we can establish a systematic way to improve the opening play of computer Shogi.

Considering these three points, we focus on the drawbacks of computer Shogi in the opening.

1.2 Problem Statement and Research Questions

In this study, we deal with the high-level techniques of opening play in Shogi to achieve the human champion.

In general, a computer is poor at playing the opening in various games because one has to work out a long-term plan in the opening. Although the long-term view is required for planning a strategy, it is hard to implement such an idea on a computer. The following two points are thought as reasons of this problem. The one is that working out a plan for the opening cannot be accomplished by simple search. For example, one should look ahead more than 30 plies of moves to work out a plan in the opening of Shogi. However, searching over 30 plies of moves is not a practical way. Therefore, computers have to evaluate long-term strategies alongside with lookahead search. But here, the other question is arising: how can we evaluate long-term strategies? For evaluating long-term strategies one has to gain high-level domain knowledge. Therefore, implementing an evaluator for the long-term strategies is also a hard problem.

In complex games such as Chess and Othello, how do computers play the opening game? In those games, there exist standard sequences of moves that masters usually play after reaching the conclusion that those moves are the best according to their experience. For example, we show one of those sequences of moves in Othello opening game in Figure 1.2. That position is led by the Black's best moves and also the White's best moves, and is estimated as an even position by many masters. On the other hand, those sequences of moves would contain countermoves against some strange moves. Thus players can be led to disadvantageous positions when they play strange moves. For example, let us consider a position shown in Figure 1.3. Black plays **g4** as a 5th move instead of **e3**, but **g4** is not the best move. As a 6th move, White can play **e7**, a countermove against **g4**. As the

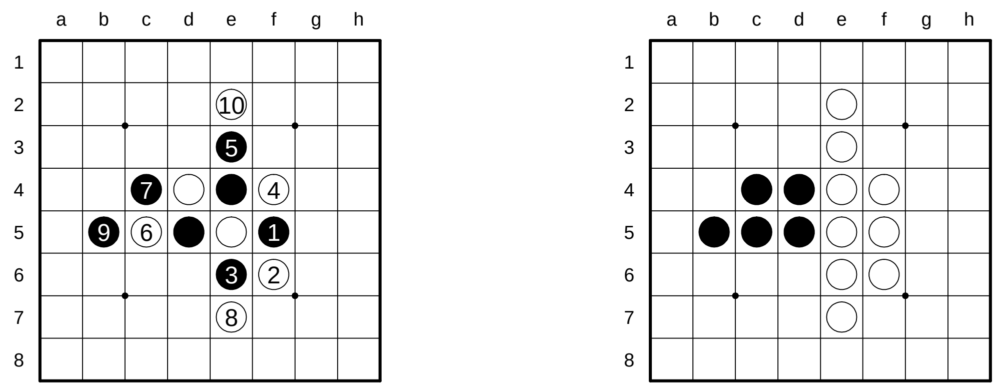


Figure 1.2: An example of opening theory in Othello (1)

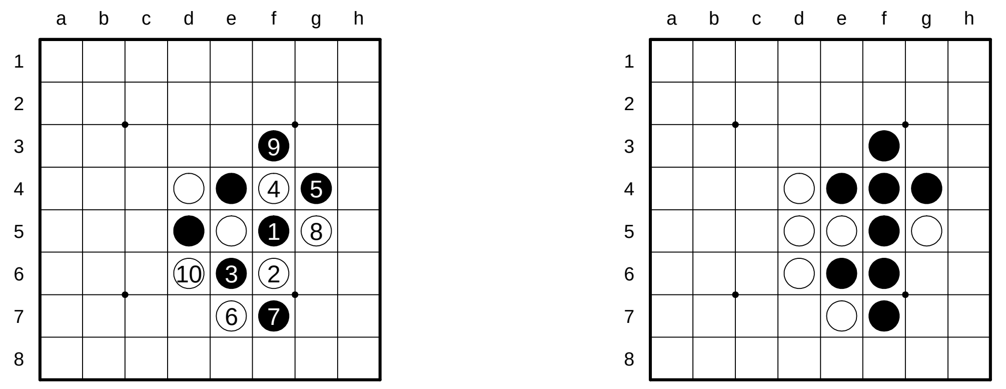
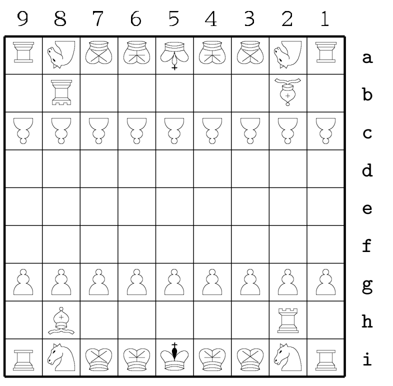
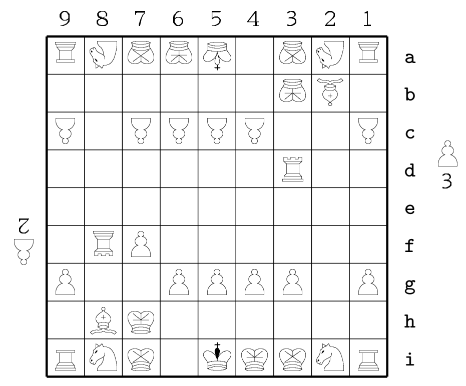


Figure 1.3: An example of opening theory in Othello (2)

result of this countermove, the position shown in Figure 1.3 is regarded disadvantageous position for Black by masters. Thus in complex games such as Chess and Othello, a computer prepares an opening database that contains standard sequences of moves, and using that database a computer can play the master-level opening. This opening database is called *opening book*.

In Shogi, there are also standard sequences of moves in the opening, and most players play the opening with considering those sequences of moves. In Figure 1.4, we show an example of standard sequences of moves in Shogi (the notation of moves and board are shown in Section 2.1). Then we can also make and use an opening book that contains standard sequences of moves in the opening for computer Shogi. In fact, computer Shogi can play the master-like opening in in-book positions. However, an opening book of Shogi is not exhaustive. Therefore, a game of Shogi is often led out-of-book positions in the early stage of the opening even if the game is played by masters. This means that a computer needs a good strategy in in-book positions as well as in out-of-book positions. As another problem, a simple use of an opening book can cause some troubles. For example, an opening book leads a computer to a strategic position where it cannot follow. In such case, a computer may play strange moves and then go into disadvantageous positions after

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-2f(2g)**	**P-8d(8c)**
3.	**P-2e(2f)**	**P-8e(8d)**	4.	**G-7h(6i)**	**G-3b(4a)**
5.	**P-2d(2e)**	**Px2d(2c)**	6.	**Rx2d(2h)**	**P-8f(8e)**
7.	**Px8f(8g)**	**Rx8f(8b)**	8.	**Rx3d(2d)**	

Figure 1.4: An example of opening theory in Shogi

an out-of-book position appear.

To improve the opening play of computer Shogi, we tackle the following challenges:

- Using an opening book effectively in in-book positions.
- Playing the stable opening game even in out-of-book positions.

In in-book positions, a computer should select proper moves from the book that it may avoid the problem mentioned above and lead the opponent into positions where a computer can play well. On the other hand, a computer has to select moves by searching and evaluating positions. To play the stable opening game, we pay attention to formations that players construct in the opening game and implement an evaluation function for the opening game.

1.3 Thesis Overview

Here we show the outline of this thesis.

In chapter 2, we explain the game of Shogi and basic ideas of computer Shogi. At first, we present the rules of Shogi and show some formations in the opening. Then we show some techniques that are commonly used in computer Shogi. At last, we describe our computer Shogi program Tacos.

In chapter 3, we introduce our research for improving the opening play of computer Shogi with focus on an opening book. At first, we present the construction and the use of an opening book in the two domains: Chess and Othello. In these games, the use of an opening book is not a copy of moves played by masters because the strongest computer is stronger than the human champion: a computer is able to find in many cases better moves by searching. On the other hand, computer Shogi programs are not so strong. Therefore, book moves taken from master games enable a computer to play a stable opening game if those moves are suitable for computers. We prepare a large opening book automatically made from thousands of game scores played by human experts and tune the book that a computer would select a prepared opening strategy. We propose a tuning method that tunes an opening book through lots of self-playing games. We actually make an opening book and tune it for Tacos. After tuning the opening book, we perform some experiments with Tacos to confirm the effectiveness of the proposed idea. The results of experiments performed show that the proposed method is able to tune the book properly for the target computer. Using the opening book tuned by this method, Tacos has played a reasonable opening game in all tournaments and has obtained good results.

In chapter 4, we focus on the opening play of computer Shogi in the out-of-book positions. It is difficult to properly evaluate the positions contained in an opening book. This is because there is no clear difference among the principal evaluation features such as material advantages and King safety in the opening stage. Therefore, a computer is unable to evaluate the positions in the opening only depending on those features and has to evaluate with other features. Since one would usually make piece formations in the opening game, evaluating those formations helps a computer understand positions properly. At first, we introduce some basic methods for evaluating formations and our proposed enhancements for extended use with a piece-square table. Then some problems in implementing and using the piece-square tables are shown with our proposed countermeasures. Using the piece-square tables, a computer can evaluate formations roughly. However, a computer cannot evaluate formations from the viewpoint of the relation between some pieces. To evaluate the relation among pieces, we introduce a notion of the combination evaluation. With this evaluation features, it became possible for a computer to play some kinds of countermoves against the opponent responses that require some disposal. These improvements mentioned above are approaches to the evaluation side. As another improvement in the searching side, we introduce a notion of the partial position opening book that enables a computer to play moves under considering book moves in the similar in-book positions. By implementing those methods on Tacos, we can improve the opening play of Tacos remarkably in the out-of-book positions.

In chapter 5, we show the two enhancements that improve some problems not in the opening. One is an enhancement of the use of transposition tables. The use of transposition tables enables a computer to efficiently search a game tree by storing the search results of sub-tree or the best moves found in the previous search. In the domain of computer Shogi the previously-stored transposition table is cleared from the memory when starting game-tree search in the current position. This means that a computer has to waste some valuable information obtained in the previous search. Although leaving and referring the entries stored in the previous search may enable a computer to search effectively, leaving old entries may cause some problems. To use valuable information stored from the previous search, we propose a method that uses two different transposition tables alternatively. This method has been implemented in Tacos and improves its playing strength. The second enhancement is the opportune time recognition of attacking. The first fights that start after constructing a formation are one of the weak points of computer Shogi because such attacking often follows a material loss. This means that a computer often overlooks the moves for starting the first fights because those moves are not well recognized in the short-term. With this weak point, a computer has to miss a good chance of starting attacks and could be damaged by overlooking the opponent's attacks. Therefore, we have to improve this weak point to make a computer much stronger. To tackle this problem, we deal with the edge attack, one of the attacking that a computer is poor at, while examining the possibility of attacks in a position and searching attack lines deeply when a position is regarded as an opportune timing for the attacking. From thousands of game scores played by human experts, we find out some features that estimate the possibility of the edge attack. Using those features, we try to develop TACOS that it may search the edge attack deeply. The results of performance test show that the routine of the edge attack enables TACOS to find the opportune moment to start the edge attack in more positions than before.

In chapter 6, we give conclusions of this study and summarize the current and future works.

Chapter 2

Computer Shogi

This chapter presents the basis of the study on computer Shogi. Section 2.1 describes the rules of Shogi and Shogi-related knowledge necessary for this study. Section 2.2 presents the basic techniques that have commonly been used in most Shogi programs. Then the overview of our Shogi program TACOS is given in section 2.3.

2.1 The Game of Shogi

This section consists of two parts. First it explains the rules of Shogi. Then it mentions about the domain knowledge in Shogi.

2.1.1 The rules of Shogi

Board and position

Shogi is a Japanese traditional Chess-like game that is played with a 9 by 9 board and 40 pieces that are classified in eight kinds. The goal of Shogi is checkmating the opponent King. Each player plays a move alternately. The first player is Black and the second player is White. Let us show in Figure 2.1 the initial position.

The pieces in the lower side of the board as shown in the position of the figure are Black's pieces and other pieces in the upper side of the board are White's pieces. We use

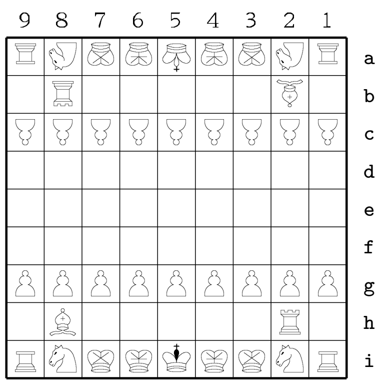


Figure 2.1: The initial position of Shogi

coordinates files as numbers (**1** to **9** from right side of the board to left) and ranks as alphabets (**a** to **i** from top of the board to bottom). The squares in rank **g** to **i** are Black's area and the squares in rank **a** to **c** is White's area.

Pieces and moves

All pieces except King and Gold can promote when they move into or move from the opponent's area.

King 王 (玉) ♔♔

King can move to the neighboring squares. The movement is the same as King of Chess.

Rook 飛 ♖♖

Rook can move along the rank and file for any number of squares, but cannot jump over any pieces. This movement is the same as Rook of Chess.

Bishop 角 ♗♗

Bishop can move along diagonal squares for any number of squares, but cannot jump over any pieces. The movement is the same as Bishop of Chess.

Gold 金 ♕♕

Gold can move one square forward, backward, sideways or diagonally forward.

Silver 銀 ♕♕

Silver can move one square forward, diagonally forward or diagonally backward.

Knight 桂 ♘♘

Knight can move one square straightforward and then one square diagonally forward, to the right or left. It is the only piece that can jump over other pieces. The movement is not the same with Knight of Chess. The Knight of Chess can move eight directions, but the Knight of Shogi can move only two forward directions.

Lance 香 ♖♖

Lance can move straightforward for any number of squares, but cannot jump over any pieces and never go back.

Pawn 歩 ♙♙

Pawn can move one square forward. The movement and capturing are different from the Pawn of Chess.

Promoted Rook 龍 ♜♜

Promoted Rook can move to squares where un-promoted Rook can move and one square diagonally in any direction.

Promoted Bishop 馬 ♝♝

Promoted Bishop can move to squares where un-promoted Bishop can move and one square forward, backward, and sideways.

Promoted Silver 成銀
Promoted Silver can move just like Gold.

Promoted Knight 成桂
Promoted Knight can move just like Gold.

Promoted Lance 成香
Promoted Lance can move just like Gold.

Promoted Pawn と
Promoted Pawn can move just like Gold.

The pieces with high mobility such as Rook, Bishop, promoted Rook and promoted Bishop are called *major pieces* while other pieces except King are called *minor pieces*.

A player can capture an opponent's piece when he/she moves own piece to the square where the opponent's piece is in. A player can drop a captured piece in hand to any empty squares in his/her turn to move instead of moving a piece on board. There are some points for caution and exceptional rules about dropping moves as follows:

- A player cannot drop Pawn in the file where his/her Pawn is in. Opponent's Pawn or promoted Pawn takes no notice of this rule.
- A piece has to be dropped as an un-promoted piece even if that piece is promoted when captured.
- A piece has to be dropped on squares where it can move. For example, Black cannot drop Pawn in **a**-rank because the Pawn cannot move no longer.
- A player cannot drop Pawn if the opponent's King is just checkmated with this move.

In this thesis, we use the notation for Shogi-moves such as **P-7f(7g)**, **Bx2b+(8h)** and **P-7f**. The first move means that Pawn in **7g** moves to **7f**. The second move means that Bishop in **8h** moves to **2b** with capturing and promoting. Writing **x** instead of **-** shows the capturing of an opponent's piece with that move. Writing + after the destination shows that the piece moved is promoting. The last move means that Pawn is dropped in **7f**. In Table 2.1 we show the notation of pieces in moves.

Terminal position and the outcome of games

A move is called *check* if one can attack the opponent's King with that move. When a player cannot avoid being captured his/her King, he/she becomes loser and that situation is called *checkmate*. For example, in the position shown in Figure 2.2, Black played the check move (**L-1f**) and White's King being in checkmate.

When one cannot play any moves, he/she also becomes loser. In the position shown in Figure 2.3, there is no White's piece that can move except King. If White moves his/her King, then Black can capture it. Thus that position is a losing position for White. In Chess, a game becomes draw when such a position appears (*stalemate*).

Table 2.1: The notation of pieces in moves

sort	un-promoted	promoted
King	**K**	
Rook	**R**	+**R**
Bishop	**B**	+**B**
Gold	**G**	
Silver	**S**	+**S**
Knight	**N**	+**N**
Lance	**L**	+**L**
Pawn	**P**	+**P**

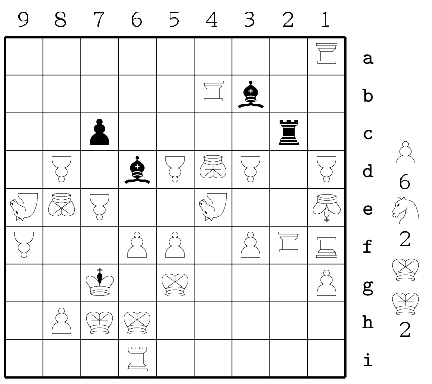


Figure 2.2: An example of terminal position (1)
Black played **L-1f** (checkmate)

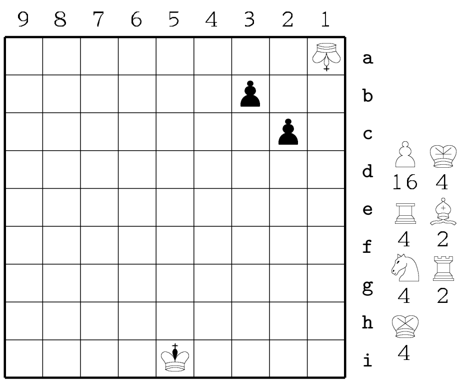


Figure 2.3: An example of terminal position (2)
Black played +**P-3b(4b)** (checkmate)

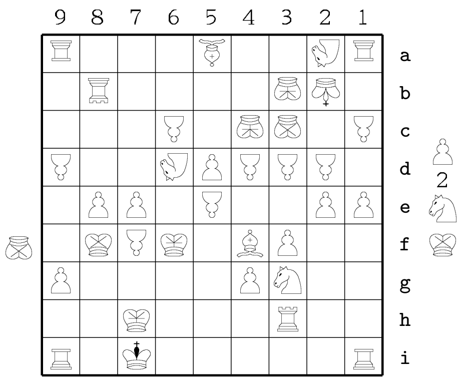


Figure 2.4: An example of perpetual
Black played **S-8f(7g)**

In Shogi, the same position may appear in many times. To avoid such situation, a game becomes (repetition) draw when a position appears in four times. For example, from the position shown in Figure 2.4, the game moves up as follows:

	S-9e
S-8g	**Sx8f(9e)**
Sx8f(8g)	**S-9e**
S-8g	**Sx8f(9e)**
Sx8f(8g)	**S-9e**
S-8g	**Sx8f(9e)**
Sx8f(8g)	

That position appears in four times; therefore this game becomes draw. In addition, there is an exceptional rule. When a position appears four times with check moves, a player who plays check moves becomes loser. For example, from the position shown in Figure 2.5, the game moves up as follows:

	K-7d(8c)
+B-5b(6a)	**K-8c(7d)**
+B-6a(5b)	**K-7d(8c)**
+B-5b(6a)	**K-8c(7d)**
+B-6a(5b)	**K-7d(8c)**
+B-5b(6a)	**K-8c(7d)**
+B-6a(5b)	

That position appears in four times with Black's check moves; therefore Black loses in this game. In Chess, there is a similar rule called *perpetual*. This rule prescribes that when a position appears in three times, the game becomes draw even if that position leads by check moves.

There is another situation that a game is finished. When both of Kings enters its opponent's area, the game will never finish by mating King. Let us show in Figure 2.6 an example. When this kind of position appears, the game will be judged after the agreement of both players. The outcome of game is judged by counting pieces. Major pieces are counted as five points while minor pieces are counted as one point. Note that King is not

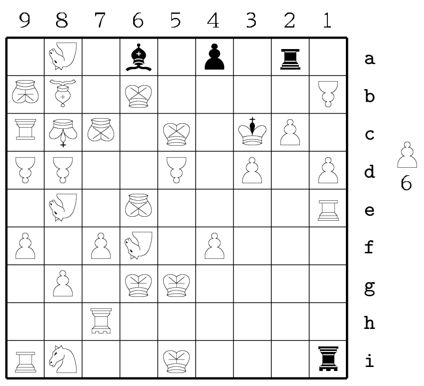


Figure 2.5: An example of perpetual check
Black played **+B-6a(5a)**

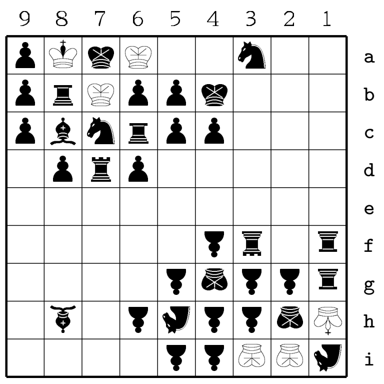


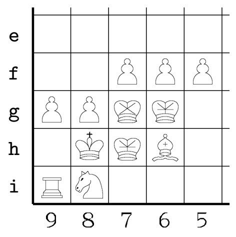
Figure 2.6: An example of King entering in the opponent's area

counted. In professional games, a game becomes draw if both of players have 24 points or more. If one has less than 24 points, then he/she becomes loser. On the other hand, in most amateur tournaments, a player who has more than 27 points becomes winner. A game becomes draw only when both of players have 27 points.

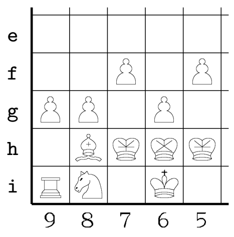
2.1.2 Domain knowledge in Shogi

There are many amateur and professional players in Shogi. Although there are some differences among players in estimating a position whether that position is advantageous or not, there are some common knowledge observed by those who are as strong as an average player. Here we focus on the castling that is one of common knowledge because castling is a very important factor of good play in Shogi.

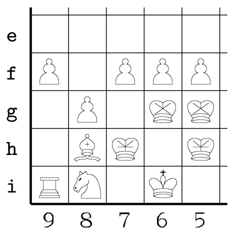
The goal of Shogi is mating the opponent's King; therefore, players have to defend own King to avoid losing. In Shogi, there are some stages such as the opening, middle game and endgame. In the opening, players usually make formations for attacking and defense. In Figure 2.7, we show fifteen formations for defense, i.e., castling. In Figure 2.8, we also show six formations for attacking. Some of those formations are for quick attack and some others are for slow games. In the opening, considering the opponent's strategy, one would try to make a formation.



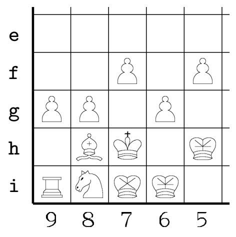
Yagura Castle



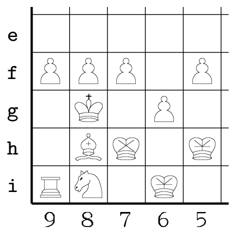
Crab Castle



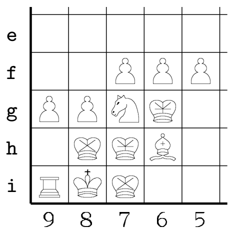
Gangi Castle



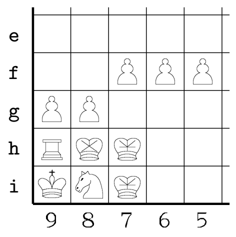
Boat Castle



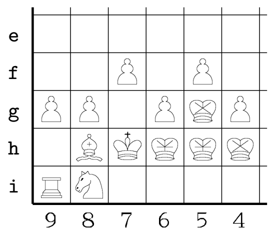
Leftside Mino



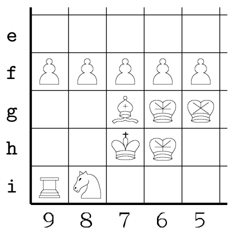
Millennium Castle



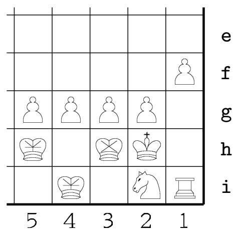
Static Rook
Bear-in-the-hole



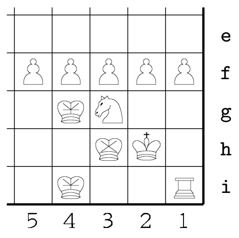
Leftside-Silver-in-**5g**



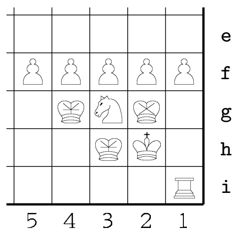
Castle against
Ranging Rook



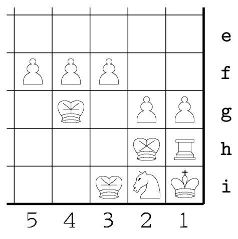
Mino Castle



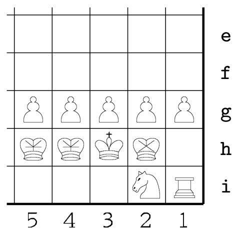
High Mino



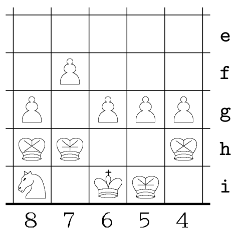
Silver Crown



Ranging Rook
Bear-in-the-hole

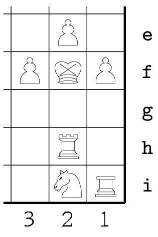


Kin-musou Castle

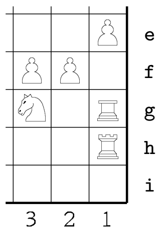


Nakahara's Castle

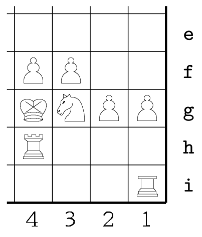
Figure 2.7: Some formations for defense (castling)



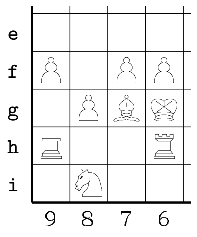
Climbing Silver



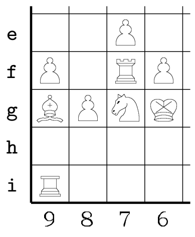
Spearing the Sparrow



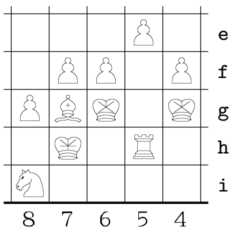
Right Side Fourth File Rook



Fourth File Rook



Third File Rook



Central Rook

Figure 2.8: Attacking formations

2.2 Basic Search Techniques

In this section, we outline some basic techniques, algorithms and structures that are used in the domain of computer Shogi.

2.2.1 Minmax search

In many computer programs that play board games such as Chess, Othello and so on, the move to be played is decided as the result of game-tree search. From a position where a computer is to move, search starts in the following procedure:

1. Play some ply of moves to make some assumed positions.
2. Evaluate each position.
3. Compare those positions to distinguish the best line.
4. Play the best move.

A game can be grasped as a graph by dealing positions with nodes and moves with edges. Although a position can be led by different path, we treat a game as a tree. A computer chooses a move after searching this game tree and evaluating leaf nodes of the tree.

In two-player zero-sum finite deterministic perfect-information games such as Chess, Othello and Shogi, many computer programs perform searching with an algorithm based on minmax. In the minmax strategy, evaluation values of each node are given as follows:

- leaf node
  The evaluation value of a leaf node is calculated by a static evaluation function that estimates the value or goodness of a position.

- internal node where the max player is to move
  The evaluation value of this node is the value of the child node that has the highest value among all child nodes. The path to that child node means the best move at this node.

- internal node where the min player is to move
  The evaluation value of this node is the value of the child node that has the lowest value among all child nodes. The path of that child node means the best move at this node.

These behaviors reflect the following three strategic aspects.

1. The max player regards that the opponent evaluates positions in the same way.

2. The max player selects the best moves in his/her view.

3. The max player regards that the opponent would select the best move form the max player's point of view.

Following the above minmax procedure, a computer (assuming the max player) finds out the best move and its evaluation value in the root node of game tree.

Searching by minmax algorithm, a computer can find accurate minmax values for all legal moves. However, it takes lots of times because using this algorithm, a computer visits all nodes in a game tree. In Shogi there are about 80 legal moves in a position on average, searching 1 ply deeper will take times about 80 times. As an example, let us assume that we have a computer that can search 200 million positions per a second like Deep Blue. Using this computer, if we perform 10-ply search by minmax, the result of the search comes out after about 1,700 years ($53,687,091,200(sec) = 80^{10}/200,000,000$). 10-ply search is not enough deep to defeat masters in Shogi. Therefore, it is required to search more efficiently.

2.2.2 Alpha-beta algorithm

Alpha-beta pruning is a search algorithm that reduces the number of nodes that need to be evaluated in the search tree by the minimax algorithm. Using a game tree shown in Figure 2.9, we explain this algorithm concretely.

Searching from the root node **A** under the depth-first manner, all child nodes of the node **D** are evaluated in Figure 2.10. Since the player to move in the node **B** is the min player, the evaluation value for the node **B** is 8 or less.

Figure 2.11 shows that the search goes on and the left node **L** is evaluated as 10. Since the player to move in the node **E** is the max player, the evaluation value of the node **E** is 10 or more. On the other hand, in the parent node **B** of the node **E**, the evaluation value is 8 or less than 8; therefore the move that leads to the node **E** from the node **B** can never be selected. For this reason, the computer can omit searching in the node **M** and **N** that are child nodes of the node **E**. This kind of cut-off that appears in the max node is called *beta-cutoff*.

The search goes on further, and the condition of game tree is shown in Figure 2.12. Since the evaluation value of the node **B** is 7, the evaluation value of the node **A** becomes 7 or more. On the other hand, the evaluation value of the node **C** is 6 or less, because

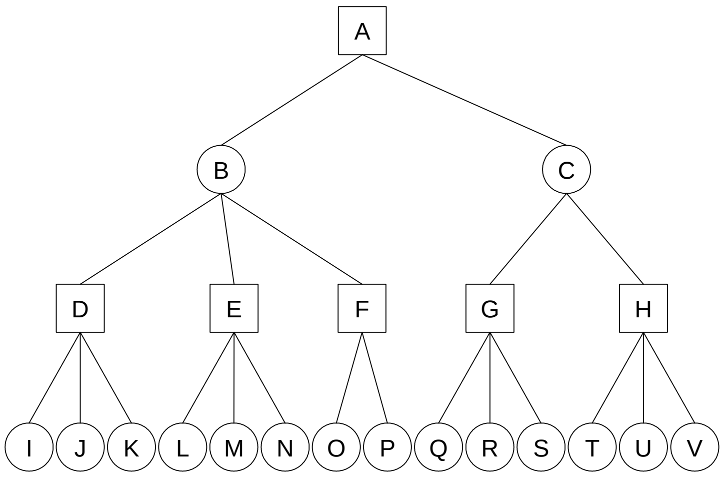


Figure 2.9: An example of minmax game tree (1)

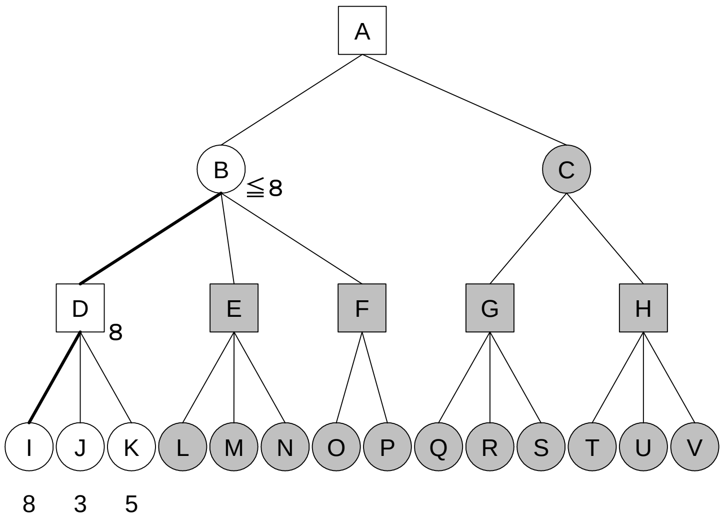


Figure 2.10: An example of minmax game tree (2)

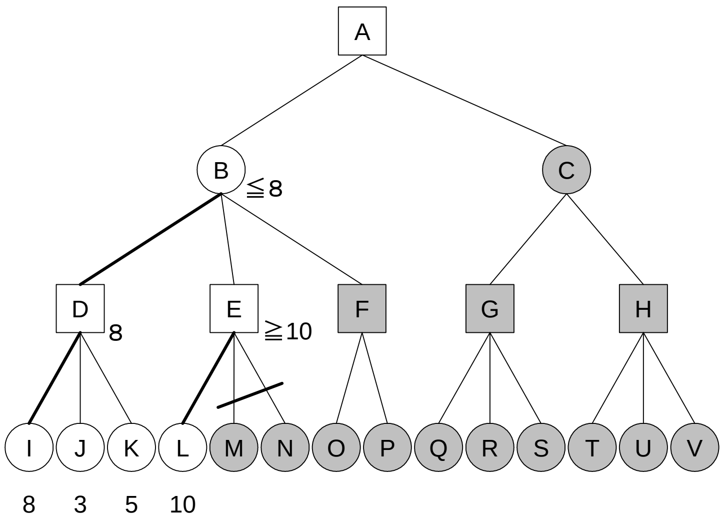


Figure 2.11: An example of beta-cutoff in alpha-beta algorithm

the evaluation value for the child node **G** of the node **C** is 6. As the result, the move that leads to the node **C** from the node **A** cannot be selected. Therefore, searching the sub-tree from node **H** can be omitted. This kind of cut-off that appears in the min node is called *alpha-cutoff*.

Omitting sub-trees that do not influence the outcome of minmax game-tree search in the framework of the alpha-beta algorithm, a computer is able to efficiently search. For more efficient pruning, it is desired to search the best move at first in every node. The notion of the move ordering is explained in section 2.2.4.

2.2.3 Transposition table

As we mentioned above, a position in a game tree can be led by different path. It is not wise to search again in a position that is visited more than one time. Thus for searching efficiently, knowing evaluation value immediately is desired instead of searching again. To realize this idea, a transposition table [2] has been used.

The basic idea of a transposition table is registering positions where searching finished and evaluation value is stored in. A computer first looks for the transposition table whether this position is stored (since once finished searching) or not. Omitting searching sub-tree from that position, a computer can know the evaluation value if that position is found in the transposition table. On the other hand, a computer is to register that position with its evaluation value after finishing searching sub-tree if that position is not found in the transposition table.

A transposition table is often implemented as a hash table. Hash keys are generated from position information such as location of pieces, player to move and so on. To register positions efficiently, Zobrist-hashing method [3] is commonly used.

2.2.4 Move ordering

To cause pruning efficiently in the alpha-beta algorithm, the best move has to be put first for searching. However, the best move can be usually found after finishing game-tree

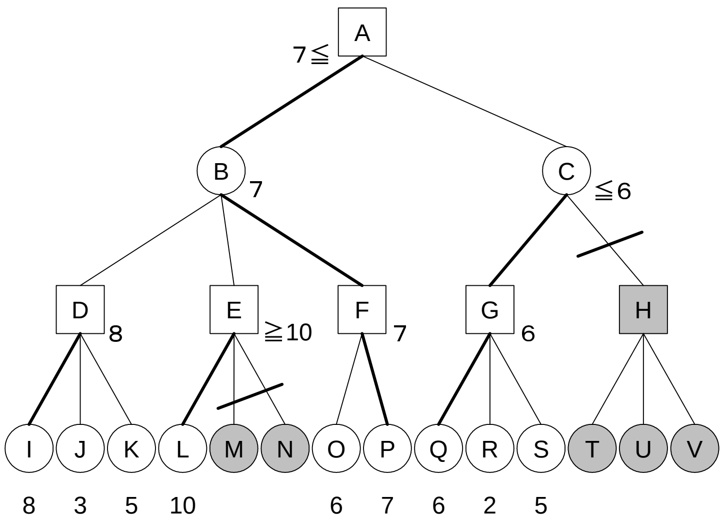


Figure 2.12: An example of alpha-cutoff in alpha-beta algorithm

search. Therefore, instead of the strictly best move, a move that seems the best is to be put first for searching in practice. Estimating the goodness of moves with small costs, a computer changes the order of moves for searching.

Then, a question arises: how do we estimate the goodness of moves? There are two approaches to change the order of moves: (1) heuristically changes with domain knowledge and (2) changes without domain knowledge. In the first approach, the moves such as capturing without any sacrifice and escaping valuable pieces from the opponent's attacks seem good in Chess-like games. On the other hand, killer move heuristics are well known in the second approach. As an example of killer move heuristics, we explain a method that uses the best move of brother nodes.

Let us consider a position shown in Figure 2.13. Black attacks White's Rook by moving **B-1a** in that position. The best moves in that position for White are escaping Rook. If White plays **S-2e(3d)** instead of escaping Rook, then Black's best move is capturing Rook by **Bx4d+(1a)**. In other cases, e.g., if White plays **P-2e(2d)** instead of escaping Rook, then Black's best move is also capturing Rook by **Bx4d+(1a)** same as the best move in the brother node position. The game tree of this situation is illustrated in Figure 2.14. Like this situation, the best move in the brother node could be the best move in a position; therefore giving high priority to the best move in the brother node would increase the efficiency in searching. In the next subsection, we explain about another method that is also used for the move ordering without using any domain knowledge.

2.2.5 Iterative deepening

In game-tree search, depth-first algorithms are often used because breadth-first or best-first search algorithms require much memory space to keep nodes that will be expanded. A computer can search deeply with depth-first algorithms that keep only the nodes visited from the root node. On the other hand, searching for mating problems by breadth-first or best-first algorithms, a computer can quickly find a solution that exists in shallow nodes of game tree. In this point, a computer cannot find that solution quickly by searching with a depth-first algorithm. To find a solution quickly by searching with a depth-first

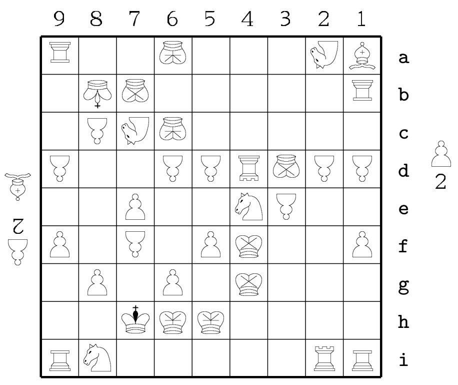


Figure 2.13: An example position for killer move heuristics
Black played **B-1a**

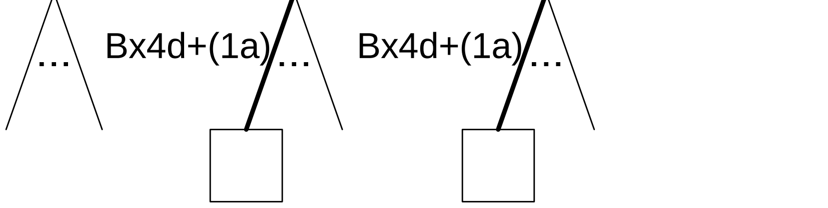


Figure 2.14: An illustration of killer move heuristics

algorithm, iterative deepening is used.

In the iterative deepening algorithm, a threshold depth where a computer stops searching is increasing little by little. Using a tree shown in Figure 2.9, we explain this algorithm.

At first, a computer performs 1-ply search from the node **A** and visits the node **B** and **C**. If the solution is not found, then it performs 2-ply search from the node **A**. During 2-ply search, it visits the nodes **B** and **C** again and new nodes **D**, **E**, **F**, **G** and **H**. If the solution is not found, then it performs 3-ply search from the node **A**. Like this, a computer can efficiently find the solution that exists in shallow nodes of a tree by increasing threshold depth. Although the nodes that exist in the shallow part of a tree are visited many times, this algorithm can be expected to reduce costs by quickly finding the solution that exists in the shallow nodes.

Iterative deepening provides the following two merits for computer programs that play a game.

Move ordering with transposition table

The best move found in n-ply search can also be the best move that will be found in $(n+1)$-ply search. Following this idea, a computer can search efficiently by searching first the best moves found during the search of the previous iteration. The implementation of the idea is as follows:

- Preparing a transposition table to store not only evaluation value of a position but also the best move in the position considered.
- A computer searches with the iterative deepening algorithm.
- When storing the search results obtained from a position into a transposition table, the best move in that position is stored with its score.
- When starting search for a position, the best move found in the previous iteration is searched with high priority.

Time control for thinking

When creating a computer program to play a game, an issue on how long it thinks for each move has to be considered. When a computer plays against human players, it is required to play a move before human players feel boring. In a tournament, a computer has to play a game within the given time for thinking. The size of game trees for positions during a game may be completely different even if a computer performs the fixed-depth search. This is because the size depends on the branching factor, i.e., the average number of possible moves at each position of the game tree for the current position considered. Therefore, the thinking time with which a computer may take would be dispersed in some degree during a game. This means that a computer may not finish searching within a limited time. Moreover, a computer may not spend sufficient time while leaving a lot of time for thinking.

Using the iterative deepening, we can cope with the time control problem. When a computer cannot finish searching for the next iteration within a limited time, the best move found in the previous iteration would be chosen. On the other hand, a computer can enter the next iteration with expectation to find a better move when there is sufficient time for it. In addition, estimating whether or not the best move will change in the next iteration, a computer can save the time to think by not entering the next iteration.

2.2.6 Search extension and forward pruning

In complex games, a computer cannot search deeply even if some search efficiency techniques are incorporated in minmax-based search. However, human players look ahead deeply only a few plausible lines in the framework of selective search. A computer can follow this idea for complex games, i.e., searching deeply promising lines and shallowly unpromising ones. Search extension is one of such ideas. It extends the search in any position where the evaluation score may change after deeper search. Forward pruning is another idea to realize more aggressive selective search. It does not expand unpromising lines.

Then, the following question is turned up: what kinds of moves are promising and what kinds of moves are unpromising? To answer the question, some ideas such as ProbCut [5], half-ply extension [41] and realization-probability search [49, 50] have been proposed. ProbCut estimates the likelihood of promising variations by performing shallower search before deeper search. If the evaluation value obtained from the shallower search is sufficiently lower or higher than some margin of evaluation score that can be the best score in the parent position, a computer omits searching deeply. This method was originally used in LOGISTELLO that defeated the human Othello champion in 1997 [23].

Although ProbCut works very effective in Othello, it does not seem effective in Shogi [92, 93]. In the domain of computer Shogi, the idea so-called half-ply extension was proposed by Yamashita and incorporated in his computer Shogi YSS, one of the strongest Shogi programs. With this idea YSS extends half-ply when a computer searches the best move in the previous iteration in each position during search. In the next subsection, we explain the realization-probability search in detail since this search algorithm is also used in TACOS.

2.2.7 Realization-probability search

Realization-probability search was proposed by Tsuruoka *et al.* and used in their computer Shogi GEKISASHI [44]. GEKISASHI is one of the strongest computer Shogi programs and won the 12th CSA World Computer Shogi Championship. After their first winning, the realization-probability search has been paid much attention and used in some other Shogi programs including TACOS. We give a short sketch of the algorithm.

- Classifying moves into some categories. For example, moves are categorized into check moves, profitable capturing moves, attacking two pieces at the same times, and so on.

- From many master games, compute the statistics of the following two frequencies in each position. The one is the number of how many times masters play the moves classified in each category. Another one is the number of how many times the moves classified in a category exist as a legal move. In case where there are some such moves classified in the same category, it is counted as only one.

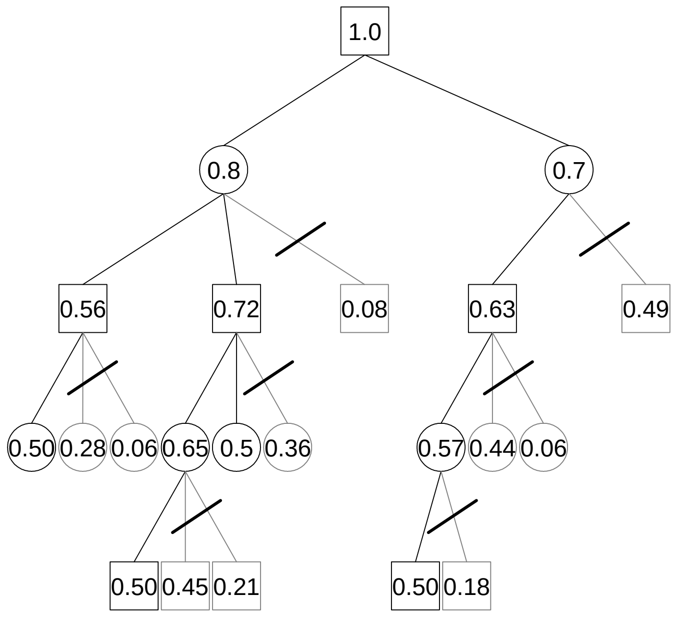


Figure 2.15: A sample game tree of realization-probability search

- Calculating transition probability for each category as follows:

$$Tp = \frac{Np}{Ne}$$

  Tp: Transition (move-category) probability.
  Np: The number of times moves of a category are played.
  Ne: The number of times moves of a category exist as legal moves.

- In every node visited during a search, realization probability of each node is calculated as follows:

$$RP_{n+1} = RP_n \times Tp$$

  RP_n: realization probability of a node in the depth n.
  The realization probability of the root node of a game tree is 1.0 because that position is already realized.

- Search stops when realization probability of a node becomes lower than a threshold probability.

Let us show, in Figure 2.15, an example of game tree searched by this algorithm with threshold probability 0.5. This figure shows that in positions where realization probability becomes lower than 0.5 the search stops even if that position exists in the shallow part of the game tree. On the other hand, the search continues in positions where realization probability is higher than 0.5 even if that position exists in the deep part of the game tree.

This algorithm can be applied with less expensive costs to any game program that uses the minmax-based search. A computer simply needs to decide whether or not it continues searching based not on the depth but on the realization probability in a position considered. With this algorithm, search extensions and selective search are systematically

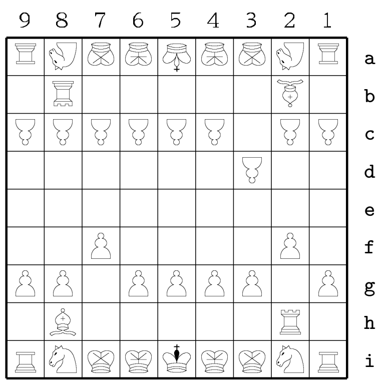


Figure 2.16: An example of quiescence search problem (1)

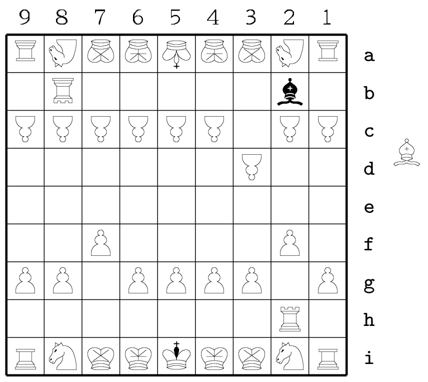


Figure 2.17: An example of quiescence search problem (2)

carried out with some good results. It is also possible to incorporate domain knowledge into this search framework.

2.2.8 Quiescence search and horizon effect

Quiescence search is an algorithm typically used to evaluate minimax game trees in game-playing computer programs. It is a remedy for the horizon effect (or horizon problem) faced by a computer player. This problem occurs because computers only search a certain number of moves ahead. When searching a large game tree (for instance using minimax or alpha-beta pruning) it is often not feasible to search the entire tree, so the tree is normally only partially searched. This results in the horizon effect where a significant change exists just over the *horizon* (slightly beyond the depth the tree has been searched) meaning that evaluating the partial tree gives a misleading result.

For example, we consider three positions as shown in Figure 2.16 to 2.18. Suppose that each appears as a leaf node when performing 3-ply search from the initial position. Let us think of how those positions are scored in the case where the only material advantage is considered for position evaluation. The position of Figure 2.16 would be evaluated as even because there is no material difference. Actually, there is no difference between both players in considering the situation. Therefore, the even score for the position is correct.

In the second position shown in Figure 2.17, it would be evaluated as advantageous for

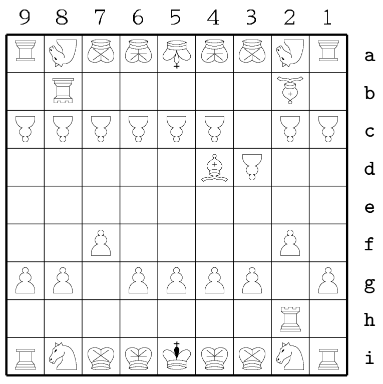


Figure 2.18: An example of quiescence search problem (3)

Black because Black just captured White's Bishop. However, White can recapture Black's promoted Bishop just after this position. Therefore, this position must be evaluated as even in material. In the third position shown in Figure 2.18, it would be evaluated as even because there is no material advantage although White can take Black's Bishop without any sacrifice. Therefore, it should be evaluated as disadvantageous for Black.

Quiescence search attempts to emulate this behavior by instructing a computer to search tactical positions to a greater depth than *quiet* ones (hence its name) to make sure there are no hidden traps and (usually equivalently) to get a better estimate of its value.

2.2.9 Position evaluation

An evaluation function, also known as heuristic evaluation function or static evaluation function by game-playing programs to estimate the value or goodness of a position in the minimax and related algorithms. The evaluation function is typically designed to be fast and accuracy is not a concern (therefore heuristic); the function looks only at the current position and does not explore possible moves (therefore static).

One standard strategy for constructing evaluation functions is as a weighted sum of various factors that are thought to influence the value of a position. For instance, an evaluation function for Shogi takes the following form.

$$\text{material advantages} \times w_1 + \text{King safety} \times w_2 + \text{mobility} \times w_3 + \text{location} \times w_4 + \ldots$$

2.2.10 Mating search [11]

A simplification of the endgame search in Shogi is a mating search called tsume or tsumeshogi in Japanese. In a mating search, the attacker is only allowed to play checking moves, until either the checks are exhausted or the defender is mated. Research in specialized mating-problem solvers has been performed concurrently and in close conjunction with the development of Shogi programs.

There are two types of mating search: (1) fixed-depth full-width search, and (2) variable depth search. Conventionally, the first type of mating search, which is based on iterative deepening depth-first search, is incorporated in most Shogi programs. It can find very short mating sequences (e.g., within 7 or 9 plies), but fails in positions requiring a deep search. Yet, this type of mating search is used in many programs for at least three reasons: (1) it is easy to implement, (2) it adequately handles time control, and (3) a lookahead of 7 to 9 plies suffices in most cases.

Variants of the proof-number search algorithm [12], such as PN*, can be used for variable-depth mating searches [13,14]. PN* is a depth-first iterative-deepening algorithm for AND/OR trees. A description of PN* is given in [14] where a tsume-Shogi-solving program based on PN* is described in detail. It solved most problems of a difficult set of human-composed problems, including the famous "Microcosmos" position with a solution sequence of 1525 plies. Subsequently, PN* was refined to use both proof- and disproofnumbers, and it is now called PDS [15, 16]. PDS (Proof-number and Disproof-number Search) behaves asymptotically the same as proof-number search.

Recently, another new algorithm, df-pn search (meaning the depth-first proof-number search) [17], has been proposed. Its behavior is the same as the proof-number search behavior in the sense that a most-proving node is expanded. The advantage of df-pn search is that it is a depth-first type of search whereas proof-number search is a best-first search. Finally we remark that Nagai's program based on both PN* and df-pn has solved nearly all the existing hard tsume-Shogi problems, including several problems that Seo's program based on PN* alone was unable to solve [18]. Even more recently some enhancements for mating search are attempted in [54–58].

2.2.11 Opening book [11]

Opening book is used to describe the database of openings given to computer game programs. Such programs are quite significantly enhanced through the provision of an electronic version of an opening book. This eliminates the need for the program to calculate the best lines during approximately the first ten moves of the game, where the positions are extremely open-ended and thus computationally expensive to evaluate. As a result it places the computer in a stronger position using considerably less resources than if it had to calculate the moves itself.

The opening phase in Shogi is generally defined as the stage from the start of a game until a player is ready to attack. Using this definition, the opening phase generally lasts 20 moves on average. An opening database, such as is used for chess, has not yet been established for Shogi. In chess playing an out-of-book move in the opening is risky; in Shogi even a grandmaster sometimes plays such a move, especially against an opening expert. Most Shogi games do not follow book moves for very long. However, games played by grandmasters show that a poor formation or poor development can rapidly lead to a disadvantageous position. Making a good formation in Shogi is very important for obtaining a positional advantage or for keeping the positional balance even.

Most Shogi programs have an opening database. Using the database, programs can build a grandmaster-like opening formation if both sides follow exactly the same opening line. As for building an opening book, it is often made from master games or textbooks written by masters.

2.2.12 Think on the opponent's time

During the time the opponent is thinking, a master would predict the opponent's move to think the next move to play. In the same way, a computer can think on the opponent's time while predicting opponent's move (so-called *predictive search*). The design and implementation for this can be many variations. Therefore, various contrives may be implemented in each program.

2.3 Overview of Tacos

In the last section of this chapter, we show the overview of our Shogi program Tacos. More details about Tacos are presented in [59–74].

First we mention shortly about the history of Tacos. Tacos has been developed by the member of Professor Iida's laboratory. Dr. Sakuta started to develop Tacos, and participated for the first time in the 9th World Computer Shogi Championship in 1999. After the tournament, the Tacos project was followed by Dr. Hashimoto. The author joined the project in 2001 and other ten students have taken part in this project till now. Tacos was amateur 8th kyu or 9th kyu level while having only 700 or 800 rating point, when the author joined the project. At that time, Tacos could understand the only piece formation (Yagura Castle) and had a small opening book. Thus the author started this study to improve the opening play of Tacos.

As the results of our various improvements, Tacos become as strong as amateur 5th dan player (2,400 rating point). Tacos has also obtained good results in some tournaments, such as the winner in the 10th Computer Olympiad in 2005 and 4th place in the 16th World Computer Shogi Championship in 2006. From these good results, Tacos can be regarded as one of the top computer Shogi programs in these days. The more detail about the tournament history of Tacos is shown in appendix A.

2.3.1 Search engine

The main search engine of Tacos is the realization-probability search [49, 50] based on alpha-beta algorithm with iterative deepening. Considering the following factors, many move categories for realization-probability search are made.

- Kind of pieces.
- Aim of the moves (capturing, double threat, escaping valuable pieces and so on).
- Check, defense, others.
- Promoting, dropping, others.
- Material advantages (large or small, positive or negative or even).

We obtain the statistics of those categories separately in the opening game, middle game, early endgame and latest endgame. Since we use the realization probability instead of depth as a threshold to stop searching, we increase the threshold probability gradually at each iteration. In section 5.2, we discuss some extensions to improve the transition probability of some moves in specific positions.

As for the move ordering, Tacos searches first the moves with higher transition probability. We show some categories that have high priority as follows:

- The best move found in the previous iteration in a position.
- A move capturing a piece which moved just before.

- The best move found in brother positions (Killer move).
- A move defending against the opponent's check moves.
- A move preventing from the opponent's threat-mate (If a player does not play defense moves in the threat-mate position, then the opponent can checkmate).
- A move preventing the opponent from capturing the most valuable piece.

In Tacos, we do not implement quiescence search, instead, potential moves which may lead to the horizon effect are not searched. Then Tacos generates and searches only some kind of moves such as capturing moves, escaping moves in positions near leaf positions. Thus Tacos can evaluate quiescent positions without quiescence search.

In a transposition table of Tacos, information such as evaluation score, best move, opponent's best move against a pass move, state of King (the number of checks or threat mate and so on) and search depth to which Tacos looks ahead are stored. We expand the use of transposition tables that is described in section 5.1.

Tacos also has mating search based on the depth-first proof-number search algorithm [53].

2.3.2 Evaluation function

In the evaluation function, Tacos considers the following factors.

- Location of pieces. Pieces in corner of a board or squares that are far from both Kings are evaluated lower. Basically promoted pieces and pieces in hand are evaluated higher than the original pieces.
- King safety. Tacos estimates King danger by counting the number of attacked squares around King, mobility of King, pieces in hand of the attacker, and so on.
- Mobility of major pieces and Lance. When those pieces can move to many squares, their evaluation value becomes higher.
- Formation evaluation. In Shogi, players make some formations in the opening game. Tacos evaluates formations by focusing on the location of pieces and combinations of a few pieces. This topic will be detailed in chapter 4.

2.3.3 Opening book

The opening book of Tacos is made from thousands of game scores played by human experts. In addition, we input some opening theories with referring some books written by professional players. After making an opening book, we tune the opening book that enables Tacos to select preferable opening lines. In chapter 3, we study the tuning method in detail.

2.3.4 Predictive search

Tacos uses the predictive search, in which it thinks on the opponent's time, assuming that the opponent plays a move. If Tacos finishes the search before the opponent plays a move, it continues the predictive search with other moves expected within the fixed time. In section 5.1, we propose an idea of improvement for predictive search by using two different transposition tables alternatively.

Chapter 3

Opening

This chapter is an updated and abridged version of

1. J. Nagashima, T. Hashimoto and H. Iida: "Opening Book Tuning", The 9th Game Programming Workshop (GPW 2004), pp.129-134. Hakone, Japan. November, 2004.

2. J. Nagashima, T. Hashimoto and H. Iida: "Finding Fruitful Positions", 8th Joint International Conference on Advanced Science and Technology (JICAST 2004), pp.161-164. Chechiang, China. December, 2004.

3. J. Nagashima, T. Hashimoto and H. Iida: "Master-like Opening Strategy in Computer Shogi", 8th Joint Conference on Information Sciences (JCIS 2005). Salt Lake City, USA. July, 2005.

4. J. Nagashima, T. Hashimoto and H. Iida: "Self-Playing-based Opening Book Tuning", New Mathematics and Natural Computation, pp.183-194, Vol.2, No.2. July, 2006.

In this chapter, we describe a measure with an opening book to improve the opening play of computer Shogi.

In complex games such as Othello, Chess and Shogi, there exist standard sequences of moves that masters usually play after reaching the conclusion that those moves are the best according to their experience. In these games, it is a reasonable opening strategy for both humans and computers to select their moves from these standard sequences of moves. The use of an opening book gives a player some practical merits, of which below we show four aspects that are currently observed in the domain of computer Shogi.

- One can play a master-like opening game while simply following an opening line from the book.
- The book would contain countermoves against some strange moves.
- One can save some time by using the book without deep search.
- By tuning opening lines stored in the book, a stupid loss in the same way could be avoided.

However, we understand that a computer Shogi needs a better opening strategy to arrive at the master level. With such an opening strategy a computer may lure the opponent to play into its own hands, just like a human master. For this purpose, our challenge in this study is to tune the book that a computer may not select its unapt lines from the book. All positions including interior and leaf nodes of the book tree should not be unfit for a computer to continue its own play. Note that the opponent may play out-of-book moves at any interior nodes of the book tree.

Section 3.1 presents a construction of opening book. Section 3.2 describes a new method that tunes an opening book properly for a target program. We generate an opening book from many game scores and tune it for TACOS. Using this opening book, TACOS can play reasonable opening games better than before. In section 3.3, we discuss about this research from the view point of Reinforcement Learning.

3.1 Opening Strategy

In this section, we discuss the basic ideas to build an opening book. We first overview several methods used for building a book in the domain of Othello and Chess.

3.1.1 LOGISTELLO

LOGISTELLO is a computer Othello program that is far stronger than the human world champion. The book was created, tuned and expanded automatically by LOGISTELLO itself [75]. The book is formed as a game tree with the initial position as the root node. The leaves of the tree have an indicator of the game outcome (win, loss or draw) for the side to move or an estimation made by deep search. Propagating these outcomes or estimations by means of minmax algorithm, LOGISTELLO can select the best book move. On the other hand, a book is extended as follows;

1. Propagating outcomes and estimations, find the leaf node where its outcome or estimation is propagated to the root node.

2. Search from that leaf node and find the best move and the second best move. These two moves are stored in the opening book.

3. Search from the parent node of that leaf node and find the best move among the out-book-moves. This best move is also stored in the opening book.

Let us show, in Figure 3.1, an example. We can find the best move orders v_1 to v_3 through v_2 by means of minmax algorithm. After searching from v_3, we get v_5 and v_6 as the best and second best moves and store them. We also get and register v_7 after searching from v_2 that is the parent node of v_3. By this expansion, the best move orders change as v_1 to v_4, and then the next expansion will be performed in v_4.

3.1.2 DEEP BLUE

We give a short sketch of the opening book of DEEP BLUE that is a computer Chess playing system that defeated the then-reigning world champion in 1997. The opening book of DEEP BLUE contains only about 4,000 positions and it is made by human grandmasters.

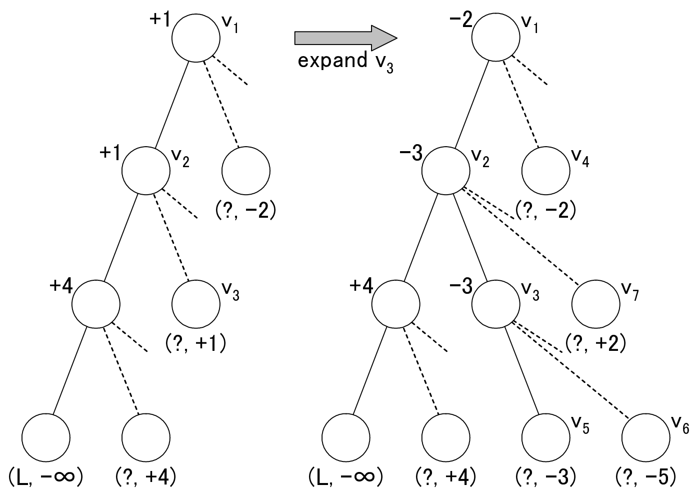


Figure 3.1: An expansion of opening book tree

Since this small opening book leads out-of-book positions very quickly, Deep Blue has to decide moves by searching. To improve quality of search, Deep Blue uses another opening database, so-called extended book, which is made from about 700,000 game scores played by masters. In section 3.2 we discuss on the extended book in more detail.

3.1.3 Crafty

We outline the opening book of Crafty [76, 77] that is one of strong computer Chess programs. In the domain of Chess, there are thousands of game scores played by grandmasters, which are electronically recorded. Programmers can easily obtain and use those game scores. Since the moves that are often played by grandmasters can be regarded as good moves, Crafty uses an opening book that is made from thousands of those game scores. These electronic game records allow a computer to automatically construct an opening book. Thus Crafty can use a large opening book that contains a lot of positions and moves. The advantages of using a large opening book are follows:

1. A computer can avoid some strategies that often lead to losing positions.
2. A computer can also avoid playing the same games repeatedly.

On the other hands, there are some weak points to use a large opening book.

1. Some stupid moves may be contained in the opening book.
2. The book management including its tuning is very heavy. Thus using such a large opening book properly is very difficult. The topic of tuning is discussed in section 3.2.

The use of game scores electronically stored is very common technique; therefore many Chess programs use opening books that are made automatically from lots of game scores.

3.1.4 Strong Shogi programs

In the previous subsections, some opening books for Othello and Chess programs were described. In the case of LOGISTELLO and DEEP BLUE that are stronger than the human champions, they may find better moves than moves played by masters. Therefore, the way generating an opening book by a computer itself or using only a small opening book can be adopted. On the other hand, computer Shogi programs are weaker than masters; therefore it is safe for computer Shogi to follow the moves played by masters. In the domain of Shogi, there are thousands of game scores electronically recorded and programmers can obtain those scores easily. Thus it is good for Shogi programmers to build a large opening book from those game scores like CRAFTY. In additional, since the opening theory of Shogi is not exhaustive, a large opening book is required if a computer plays the opening game depending on an opening book. Although a large opening book requires heavy costs for maintenance such as tuning, we consider that a large opening book is adequate to computer Shogi. In next subsection, we show the design of an opening book for TACOS. In next section, we deal with the topic of book tuning while considering how we can tune with fewer costs.

3.1.5 TACOS

Before implementation, we enumerate the requirements for the opening book of TACOS.

- The opening book can store positions and moves as many as possible.
- Searching a position from the opening book must finish quickly even if the opening book stores a lot of positions.
- The plural number of moves can be registered in a position. Turning positions of strategies contain the plural number of moves.
- A position can be stored with not only good moves but also bad moves. It is also required that a stored move can be distinguished between good moves and bad moves.
- A stored move can be distinguished between good moves for TACOS and bad moves for TACOS. There can be some moves that are good in generally but often lead TACOS to losing positions.
- The amount of memory for the opening book should be as small as possible.

Considering these requirements, we design a book as follows;

- A position stored in the opening book may have more than one book moves.
- Each book move has priority. In a position where there are more than two book moves, the move with the highest priority is to be selected. Using the indicator of priority, we see whether or not a move is good in general and a move is good for TACOS at the same time.

- Positions are stored in a hash table. Using the hash table, Tacos can search a position quickly from an opening book. Corresponding a position to the hash key, Tacos can use an opening book in case that a game returns to an in-book position after comes to an out-of-book position.

Using this design, we implement an opening book for Tacos and generate from thousands of game scores played by masters. We store the moves played in those game scores, but some of them are bad moves. The thinking time by which a move is played can show whether the move is good or not in general; thus a move that appears few times is stored and marked as a bad move or not stored.

3.2 Book Tuning Methods

In the previous section, we presented a design of the opening book for Tacos. After constructing an opening book, programmers have to tune their opening books so that a computer may avoid undesirable strategies and lead its strong strategies. Thus in this section, we discuss some methods that tune opening books.

In section 3.2.1, we describe some related works. In section 3.2.2, we explain the basic idea of our proposed method. In section 3.2.3, some types of opening books are produced to evaluate.

3.2.1 Related works

The task of tuning a large opening book is too daunting to be done by hand. Therefore, some methods that automatically create and tune the book were proposed. First, we explain the concept of so-called *extended book*, which was used by Deep Blue [22]. Second, we introduce another book tuning idea in the domain of Chess, which was developed with Crafty [76]. Finally, we summarize important issues from these related works.

Extended book of Deep Blue

Deep Blue uses a relatively small opening book that is created and tuned by hand with the aid of human grandmasters. In addition to the small-size book, Deep Blue has another opening database called *extended book* [22] that is built from about 700,000 game scores played by grandmasters. Deep Blue does not simply choose a move from the extended book. Instead it is used as an indicator to assign bonuses during search. The following factors are considered when the extended book is used:

- The number of times a move has been played
- Relative number of times a move has been played
- Strength of the players that play the moves
- Recentness of the move
- Results of the move
- Commentary on the move

- Game moves versus commentary moves

This method may be characterized by two different phases. (1) The extended book is tuned in advance by the factors above. In this way DEEP BLUE is able to understand which moves seem good in general. (2) The extended book is used during search to improve the position evaluation. As consequence, DEEP BLUE selects suitable lines using the extended book or better lines out of the book.

Book learning of CRAFTY

Hyatt introduced an idea for automatic book tuning that has been used in CRAFTY [76], a strong computer Chess program. The basic idea is that the priority of moves stored in the book is to be updated automatically after each game. Then two methods were proposed; (1) the book is tuned by the results of games (result-driven learning), and (2) the book is tuned by the results of searches from the first ten positions after the book has been left (search-driven learning). Through many games played on the Internet, the book was tuned properly for CRAFTY.

Two important points

To summarize the book tuning ideas described above, there are two important points necessary for a target computer program.

- The target computer program needs to reach the master-level strength.
- The target computer program needs to play many games on the Internet.

In the domain of Shogi that is the target in this study, computers have not yet reached the master-level strength. Therefore, the idea used by DEEP BLUE cannot be applied. On the other hand, the CRAFTY's book tuning method seems possible to apply in some way into the computer Shogi. The point is not the strength of the target computer program but the learning environment, i.e., playing many games on the Internet. It is unfortunate that we cannot simply follow CRAFTY's method in the domain of Shogi. A critical reason is that few masters play on the Internet. Moreover, professional players of Japan Shogi Federation are prohibited to play against computers.

3.2.2 Result-driven book tuning

As described in the previous section, the book tuning ideas developed in other domains are unable to simply apply into the domain of computer Shogi. We mention about another Shogi-specific issue. In Shogi, even masters often play out-of-book moves in the early opening stage. This may be caused by its large search-space complexity. Note that the search-space complexity of Shogi is 10^{220} while Othello is 10^{58} and Chess is 10^{123} [9, 11]. Therefore, the book has not yet been systematically established in Shogi. Accordingly, the book has to be tuned that a computer may play reasonably after the book has been left at any interior nodes of the book tree.

The basic idea of our proposed method

We propose a new method to tune the book through many self-playing games. We explain the basic idea of our proposed method. In the book tree each interior node has book moves with its weights. If there is the only book move in a position, then the move is to be played. If there are some book moves, then the move with the greatest weight is to be played. To tune the weight of each book move, all weights are initialized to the same value. Then a self-playing game is performed from the initial position.

When playing in an in-book position, a move is selected according to the following rules. If there exists the only book move, then that move is to be played. If there exist some book moves, then the move with the highest value v is to be played. Where v is calculated by the following formula:

$$v = w_m + r$$

Here w_m denotes the weight of the book move m, and r denotes a random number ranging from 0 to *Rand*. A self-playing game is performed until the game is over. According to the game result, the weights of book moves that are just played in the self-playing game are to be updated in the following way. For each book move played by the winner's side, the weight is incremented by Δw, whereas the weight is decremented by Δw for each book move played by the loser's side. Then other self-playing games are performed to update weights while reducing Δw until the weight of each book move becomes stable. Moreover, we prepare an option that out-of-book moves may be played. For this purpose, in each in-book-position a book move is selected with some probability. Thus, we observe whether or not the target computer program can play well after the book has been left.

Using the proposed method, the original book is tuned for the target computer program. This is because the weights of book moves leading to suitable positions become greater while the weights of book moves leading to unsuitable positions become smaller.

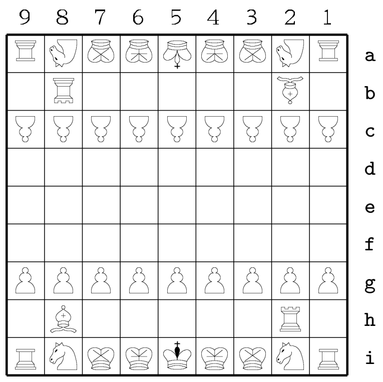
Why self-playing based tuning?

In this study, we adopted a similar way of Crafty's result-driven learning because it is easy to implement. However, there are some problems in this result-driven-based learning method as pointed by Hyatt [76]. We here repeat the important question. "Was a game lost because of a bad opening, or poor play by the program?" Our answer to this question is a statistical solution. It is to be judged whether a computer is poor to play in a book position, based on the winning rate of many self-playing games that start from the position. Namely, we use the winning rate of self-playing games as an indicator to measure the suitability of book moves for the target computer program.

The self-playing-based book tuning is a less expensive way than Crafty's result-driven learning that is based on playing many game on the Internet. It is easy to realize such a learning environment in the domain of computer Shogi. However, there might be some risk in taking such a way. It means that the book may be tuned in a biased way on account of self-playing based tuning.

3.2.3 Performance experiments with Tacos

In this section, we produce some types of opening books using the proposed self-playing-based tuning method. Experiments in the form of tournaments using Tacos are per-



P-7f(7g)	25554	**P-6f(6g)**	3
P-2f(2g)	3564	**P-3f(3g)**	3
P-5f(5g)	182	**G-7h(6i)**	3
R-5h(2h)	105	**S-7h(7i)**	2
P-1f(1g)	39	**K-6h(5i)**	1
P-9f(9g)	26	**R-7h(2h)**	1
R-6h(2h)	14	**P-8f(8g)**	1
S-4h(3i)	4		

Figure 3.2: Moves played in the initial position

formed to evaluate the proposed idea.

Original opening book

We use an opening book as the original one that was automatically built from some 29,000 master games. The games include 11,000 games played by professional players and 18,000 games played by high-scored players on the Internet. We take from those game scores [78,79] the first 45 plies of moves of each game as book moves in the form of a game tree. Although masters seldom make careless mistakes in the opening stage, mistakes may occur at any level. Figure 3.2 shows the initial Shogi position and moves played in that position with the statistics on the frequency of moves played. For example, in a game played on the Internet, the first move was **P-8f(8g)** that is rarely played because most players regard it as a bad move. Therefore, we suspect that Black intended to play **P-7f(7g)** that is one of the most popular moves in the initial position, but misplaced it. To exclude such error moves from the book, we deleted the moves that are seldom played in positions that often appear. To realize it, the moves played less than ten times in a position that appeared more than 100 times, are not to be stored in the book. However, the moves played more than ten times in a position that appeared less than 100 times, are to be left over. In this way we created the original opening book that contains some 670,000 positions and 725,000 moves.

Different types of books

For the experiments performed, we produce 24 different types of opening books using our proposed tuning method. Below we explain each type.

Type-1: book without tuning
Type-1 is the original book where all book moves have the same weight.

Type-2: book tuned by frequency
Type-2 is an opening book where the weight of each book move is represented by its frequency, i.e., the number of time that move was played.

Type-3: book of winning moves

Type-3 is an opening book of winning moves. The moves played by the loser's side are excluded from the book. All moves have the same weight. This book contains about 358,000 positions and 385,000 book moves.

Type-4: book tuned from Type-3 by frequency
Type-4 is similar to Type-3 book except for the assignment of the weight of each book move. The weight of each book move represents the frequency of that move, i.e., the number of time that move was played by the winner's side.

Type-5(a, b, c): books statistically tuned with some factors
We create three different types of opening books called Type-5a, Type-5b and Type-5c, while considering the following three factors:

- The frequency of moves
- The game result
- Whether the moves were played by professional players or not

In Type-5a, the moves played by the winner's side add 3 points and the moves played by the loser's side add 1 point to the weights of book moves. In Type-5b, the moves played by professional players add 3 points and the other moves add 1 point to the weights of book moves. In Type-5c, moves played by professional players who won add 5 points, the moves played by professional players who lost add 2 points, the moves played by not professional players who won add 3 points and the moves played by not professional players who lost add 1 point to the weights of book moves.
To summarize, Type-5a considers (i) and (ii), Type-5b considers (i) and (iii) and Type5-c considers (i), (ii) and (iii).

Type-6([1-1], [1-2], ···, [5-3]): books tuned through self-playing games
The group of Type-6 are opening books tuned from Type-1 through self-playing games. Figure 3.3 shows the whole process. First, we tune Type-1 through 1,000 self-playing games with the tuning parameter $\Delta w = 16$ to create Type-6[1-1]. In the same way, we produce Type-6[1-2], Type-6[1-3], Type-6[1-4] and Type-6[1-5] from Type-1. Then, we tune Type-6[1-1] through 1,000 self-playing games with tuning parameter $\Delta w = 8$ to create Type-6[2-1]. In the same way, we produce other books, as shown in Figure 3.3. All books are tuned in offline because we perform some games in a same time by using some machines. There is another advantage of offline method that will be mentioned in next section.
During the self-playing games, we set the probability 0.008 by which an out-of-book move may be selected. Under this condition, about 70% games reached the leaves of the book tree. We use the tuning parameter *Rand* as 512 for Type-6[x-1], 1024 for Type-6[x-2] and 1536 for Type-6[x-3]. These parameters were given that any book moves with small weights may be selected and the weight could be updated. We do not use the game scores that end in draw or continue over 300 moves[1] for the tuning process. We recognize that the use of a book does not affect sufficiently in such long games.

[1]The average length of Shogi is about 115 ply [8]. In the game scores we used for creating the book, the average length is about 113 and the standard deviation is around 27.

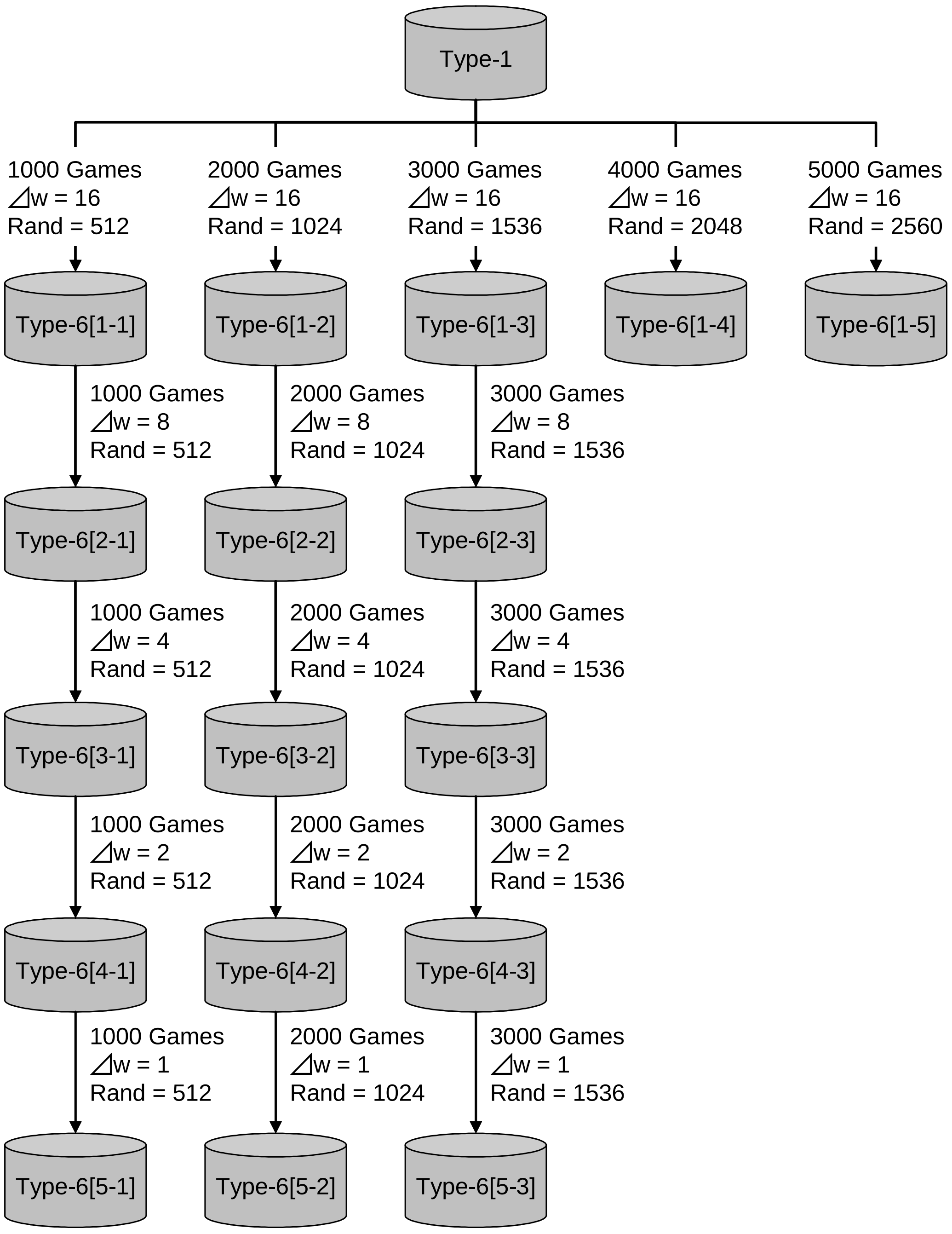

Type-1
1000 Games ⊿w = 16 Rand = 512
2000 Games ⊿w = 16 Rand = 1024
3000 Games ⊿w = 16 Rand = 1536
4000 Games ⊿w = 16 Rand = 2048
5000 Games ⊿w = 16 Rand = 2560
Type-6[1-1]
Type-6[1-2]
Type-6[1-3]
Type-6[1-4]
Type-6[1-5]
1000 Games ⊿w = 8 Rand = 512
2000 Games ⊿w = 8 Rand = 1024
3000 Games ⊿w = 8 Rand = 1536
Type-6[2-1]
Type-6[2-2]
Type-6[2-3]
1000 Games ⊿w = 4 Rand = 512
2000 Games ⊿w = 4 Rand = 1024
3000 Games ⊿w = 4 Rand = 1536
Type-6[3-1]
Type-6[3-2]
Type-6[3-3]
1000 Games ⊿w = 2 Rand = 512
2000 Games ⊿w = 2 Rand = 1024
3000 Games ⊿w = 2 Rand = 1536
Type-6[4-1]
Type-6[4-2]
Type-6[4-3]
1000 Games ⊿w = 1 Rand = 512
2000 Games ⊿w = 1 Rand = 1024
3000 Games ⊿w = 1 Rand = 1536
Type-6[5-1]
Type-6[5-2]
Type-6[5-3]

Figure 3.3: Process to create Type-6 opening books

Table 3.1: The results of the tournament

Type-1 - Type-2	39 - 61	Type-1 - Type-6[2-2]	39 - 61
Type-1 - Type-3	44 - 56	Type-1 - Type-6[2-3]	39 - 61
Type-1 - Type-4	50 - 50	Type-1 - Type-6[3-1]	45 - 55
Type-1 - Type-5a	53 - 47	Type-1 - Type-6[3-2]	38.5 - 61.5
Type-1 - Type-5b	46 - 54	Type-1 - Type-6[3-3]	40 - 60
Type-1 - Type-5c	47 - 53	Type-1 - Type-6[4-1]	29.5 - 70.5
Type-1 - Type-6[1-1]	43 - 57	Type-1 - Type-6[4-2]	37 - 63
Type-1 - Type-6[1-2]	39.5 - 60.5	Type-1 - Type-6[4-3]	27 - 73
Type-1 - Type-6[1-3]	43.5 - 56.5	Type-1 - Type-6[5-1]	37.5 - 62.5
Type-1 - Type-6[1-4]	38 - 62	Type-1 - Type-6[5-2]	35 - 65
Type-1 - Type-6[1-5]	32.5 - 67.5	Type-1 - Type-6[5-3]	38.5 - 61.5
Type-1 - Type-6[2-1]	49 - 51		

Experimental design and results

To confirm the effectiveness of the book tuning, we performed a tournament between Tacos with Type-1 book and Tacos with other types. In case where both players use the same game scores to create the book, their games always arrive at the leaves of the book tree. However, in actual games one may sometimes play out-of-book moves. Therefore, in this tournament we set probability 0.008 by which out-of-book moves may be played. In this tournament, each game starts from the initial position, and Tacos with some types of books plays 50 games as Black and 50 games as White against Tacos with the Type-1 book. A game becomes draw by repetition or taking over 300 moves, then the score is given as 0.5 win and 0.5 loss.

We show, in Table 3.1 and Figure 3.4, the results of the tournament. The result of Type-2 seems good. The results of Type-3, Type-4 and Type-5 show that those books are not suitable for Tacos. This means that some opening lines in the book are not suitable for Tacos even if those opening lines are often played by masters. The point is that the book has to be well tuned for the target computer program. On the other hand, Tacos with Type-6 won Tacos with Type-1 in all cases with some margin. It shows that a suitable opening book can be created for the target computer program by self-playing based tuning.

However, only by the self-playing games against Type-1, we cannot fairly evaluate the effectiveness of the proposed method. Type-6 books might be tuned in a biased way because of self-playing based tuning. Therefore, we perform further experiments.

We perform the second tournament among Tacos with Type-2, Type-6[4-3] and Type-6[5-3]. We pick up these three books since Type-2 and Type-6[4-3] show very good results against Type-1, and Type-6[5-3] was tuned through the greatest deal of self-playing games. To play various opening lines, a random number is assigned to each book move. The gap of weights between moves with the greatest weight and the smallest one is over 10,000 in Type-2 while about 1,500 in Type-6[4-3] and Type-6[5-3]. It is not fair to assign random numbers ranging fixed range to weights of moves in each book. Therefore, we set the range of random numbers according to the gap between the greatest weight and the smallest weight. The ranges of random numbers are the same with the gap or its 1.5 times. In this tournament, we set the probability 0.008 by which out-of-book moves may be played.

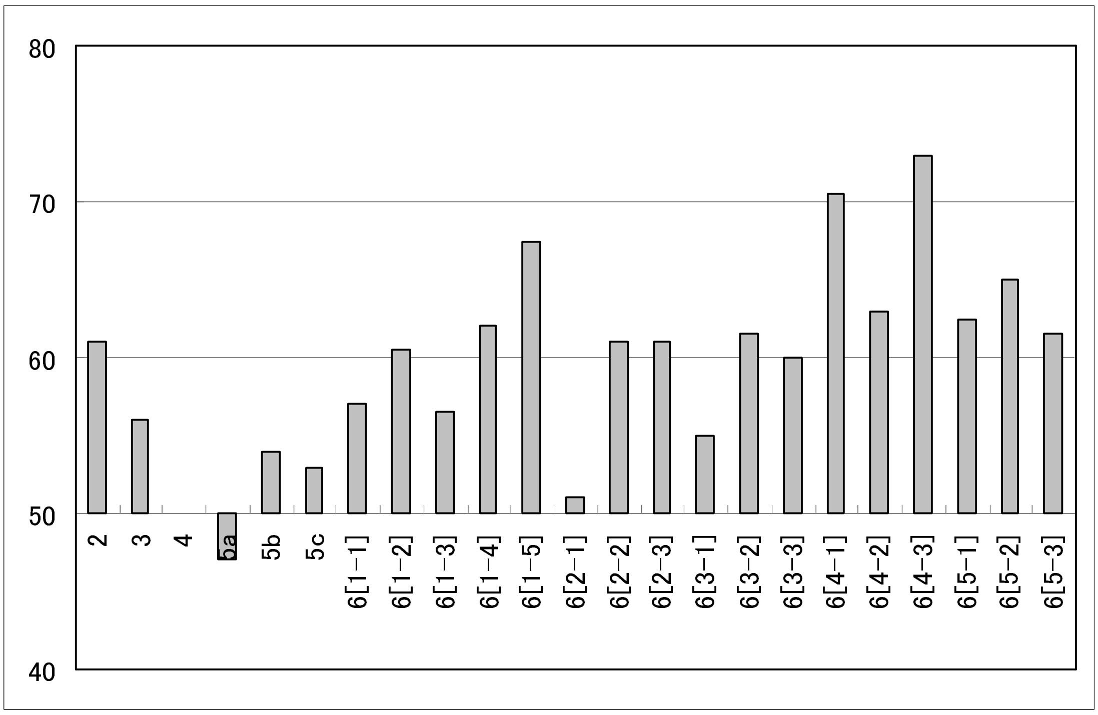


Figure 3.4: The results of the tournament

Both computers play 250 games as Black and 250 games as White. A game becomes draw by repetition or taking over 300 moves, then the score is given as 0.5 win and 0.5 loss. Table 3.2 shows the results of the second tournament. Both Type-6[4-3] and Type-6[5-3] show good performance.

Table 3.2: The results of Type-2 vs. Type-6[4-3] and Type-6[5-3]

	$Rand = gap$	$Rand = gap \times 1.5$
Type-2 - Type-6[4-3]	202.5 - 297.5	249 - 251
Type-2 - Type-6[5-3]	226 - 274	232.5 - 267.5

In addition to these tournaments, we perform another experiment in which Tacos with some books plays against "Todai Shogi 6", a strongest commercial program which won the World Computer Shogi Championship four times in the past eight years [11]. In this experiment, no random number is assigned to the weight of each book move since "Todai Shogi 6" selects different opening lines. As for the previous experiments, Tacos played the half of games as Black and the other games as White. The probability of playing out-of-book moves was set to 0.008. A game becomes draw by repetition or taking over 300 moves, then the score is given as 0.5 win and 0.5 loss. The strength of "Todai Shogi 6" is set as the "Elder" mode, where it plays with the second longest thinking time. Table 3.3 shows the results of the experiment. The results when using Type-6[5-3] and Type-2 show that the proposed book tuning method can strengthen the target computer program by some 6% up of wining percentages.

From these experiments described in this section, we can clearly observe that the

Table 3.3: The results of TODAI Shogi vs. TACOS with four different books

			score			wining rate		
Type-1	-	TODAI SHOGI	219	-	781	21.9%	-	78.1%
Type-2	-	TODAI SHOGI	83	-	217	27.7%	-	72.3%
Type-6[4-3]	-	TODAI SHOGI	68	-	232	22.7%	-	77.3%
Type-6[5-3]	-	TODAI SHOGI	84	-	216	28.0%	-	72.0%

opening book was successfully tuned for the target computer program through self-playing based tuning. We also observe that the proposed method is released from the concern, i.e., biased way tuning, on account of self-playing based tuning.

3.3 Book Tuning Method Proposed and Reinforcement Learning Compared

In this section, we compare the proposed book tuning method with reinforcement learning. We first give the short sketch of reinforcement learning. Then the relation between the opening book tuning and n-armed bandit problem will be discussed. Moreover, we consider the relation between our proposed method for book tuning and reinforcement learning. To observe it, self-playing experiments with reinforcement learning for book tuning are performed.

3.3.1 Reinforcement learning [103]

Reinforcement learning is a kind of machine learning. In machine learning, a learner learns how to select actions to maximize a numerical reward signal. In supervised learning, one of a kind of machine learning, many pairs of sample input and its correct output are given. Using those training data, a learner revises their actions to coincide with correct output. On the other hand, pairs of sample input and its correct output are not given in reinforcement learning. Therefore, a learner has to find which actions yield the most reward by trying them. From the viewpoint of reinforcement learning [103], we discuss on our proposed method for book tuning.

3.3.2 Opening book tuning and n-armed bandit problem

In the domain of reinforcement learning, there is a problem called n-armed bandit problem. In this problem, a learner receives a numerical reward chosen from a stationary probability distribution that depends on the action the learner selected. The objective of a learner is to maximize the expected total reward over some time period.

The problem of tuning an opening book can be regarded as a kind of n-armed bandit problem. A computer that is the target for tuning would select a book move from possible n-book moves and receive a numerical reward that means the result of a game such as win, loss or draw. If a book move selected is suitable for the computer, it brings a high numerical reward, this means that the computer won a game with relatively high probability. On the other hand, if a book move selected is unsuitable for the computer, it receives a low numerical reward, this means that the computer lost a game with relatively high probability.

Let $Q^*(a)$ be the true (actual) reward value of book move a, and let $Q_t(a)$ be the estimated value after selecting this move t-times. When a computer uses a tuned opening book, it can play a move a^* whose estimated value is the greatest at the time by the following formula:

$$Q_t(a^*) = max_a Q_t(a)$$

This move selection is called a *greedy* action. On the other hand, a question arises: "how does a computer select a move from all book moves stored in a position during the process of book tuning?" To answer this question, we consider the following three points.

How to decide $Q_t(a)$?

Let us consider the case that move a is selected k_a-times and the reward values are $r_1, r_2, \cdots, r_{k_a}$. In such case, some methods to decide $Q_t(a)$ were proposed. Here we explain *sample-average* method and *recency-weighted average* method [103].

In the sample-average method, $Q_t(a)$ is estimated by the following formula:

$$Q_t(a) = \frac{(r_1 + r_2 + \cdots + r_{k_a})}{k_a}$$

And $Q_t(a)$ is updated by the following formula:

$$Q_{k+1} = Q_k + \frac{(r_{k+1} - Q_k)}{(k+1)}$$

The advantage of this method is that $Q_t(a)$ is guaranteed to converge to $Q^*(a)$ after the move is selected infinite number of times in stationary environments. In the case of opening book tuning, stationary environments mean that a computer search engine is stationary and not updated.

In exponential, recency-weighted average method, $Q_t(a)$ is estimated by the following formula:

$$Q_k = (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} r_i$$

And $Q_t(a)$ is updated by the following formula:

$$Q_{k+1} = Q_k + \alpha(r_{k+1} - Q_k)$$

The step-size parameter, α, $0 < \alpha \leq 1$ is constant. Different from the sample-average method, $Q_t(a)$ does not converge to $Q^*(a)$ but continues to change in exponential, recency-weighted average method. Thus this method is suitable in nonstationary environments. In the case of opening book tuning for a computer that is often updated, this method can work effectively.

How often exploit and explore?

When a computer selects a book move, the well-balanced selection between *exploit* to receive high rewards and *explore* to find a better move is required. Note that in the case of opening book tuning, a computer always has to explore because a computer does not need to improve the winning rate during the book tuning. However, when a computer tunes a large opening book, it is impossible to select each move enough times. Thus the moves that are generally regarded as good moves and are often played have to be tuned intensively. Therefore, the well-balanced selection between exploit and explore is important even in the case of opening book tuning. Some methods were proposed for settling this problem.

Here we explain about *ϵ-greedy* method and *softmax action selection rule* [103]. In ϵ-greedy method, a learner selects greedy actions in most of the time, but every once in a while, with small probability ϵ, instead selects an action at random, uniformly, independently of the action-value estimates. Since ϵ has only simple meaning, we can deal with this parameter easily. On the other hand, this method selects non-greedy actions with the same probability. This cause a problem, i.e., a move that is clearly bad is selected more often than the move to be selected in actual.

In the softmax action selection rule, probabilities for selecting an action are assigned to improve the weak point of ϵ-greedy method by the following formula.

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$$

By this formula, the action with higher reward estimate can be selected more often than the action with lower reward estimate. On the other hand, the parameter τ has more complex influence than ϵ. Therefore, τ has to be dealt with carefully.

How to initialize Q_0?

There are some ways to initialize each Q_0. One is to initialize as some suitable value such as zero. Another way is to assign optimistic initial values. Since optimistic initial values are bigger than expectation of reward values, an agent will not try to exploit before all moves are explored.

3.3.3 Our proposed method and reinforcement learning

We compare our proposed method for book tuning with reinforcement learning.

How to decide $Q_t(a)$?

In the proposed method, a weight corresponding to $Q_t(a)$ is calculated by the following formula.

$$Q_t(a) = (tw - tl) \times \Delta w$$

Here *tw* means the number of times a computer won by playing the move and *tl* means the number of times a computer lost. Thus in our method, the weight means the sum whereas weights means the mean value in reinforcement learning such as the sample-average method.

This can cause a question: “Which is better, a move with score 3-0 or another move with score 50-45?” The score 50-45 is better than the score 3-0 in the sense of the winning margin. However, the score 3-0 is better than the score 50-45 in the sense of the winning rate. It is clear that the winning rate must be considered by the reinforcement learning. On the other hand, our proposed method prefers the score 50-45 since our method uses

Table 3.4: How book moves are played for tuning Type-6[1-1]

Book move	(1)	(2)	(3)	Winning rate (mean value)	$Wins - Loses$ (sum value)	(4)	(5)
P-7f(7g)	145	14.50%	79	54.48%	13	1	1
P-2f(2g)	164	16.40%	80	48.78%	-4	4	4
P-5f(5g)	148	14.80%	71	47.97%	-6	6	6
R-5h(2h)	137	13.70%	69	50.36%	1	2	2
P-1f(1g)	141	14.10%	68	48.23%	-5	5	5
P-9f(9g)	132	13.20%	66	50.00%	0	3	3
R-6h(2h)	133	13.30%	62	46.62%	-9	7	7

(1) How many times the book move was played. (4) Order of mean value.
(2) The rate of the book move was played. (5) Order of sum value.
(3) How many wins led by the book move.

Table 3.5: How book moves are played for tuning Type-6[1-3]

Book move	(1)	(2)	(3)	Winning rate (mean value)	$Wins - Loses$ (sum value)	(4)	(5)
P-7f(7g)	462	15.40%	213	46.10%	-36	6	6
P-2f(2g)	454	15.13%	242	53.30%	30	1	1
P-5f(5g)	443	14.77%	197	44.47%	-49	7	7
R-5h(2h)	424	14.13%	216	50.94%	8	2	2
P-1f(1g)	427	14.23%	210	49.18%	-7	4	4
P-9f(9g)	387	12.90%	195	50.39%	3	3	3
R-6h(2h)	401	13.37%	194	48.38%	-13	5	5

(1) How many times the book move was played. (4) Order of mean value.
(2) The rate of the book move was played. (5) Order of sum value.
(3) How many wins led by the book move.

the sum values (winning margin) as move selection priority. We understand that it is better to select the move with score 3-0 in this case. However, we seldom meet such a case since our proposed method is offline learning.

In the previous section, we tuned Type-6 opening books by the offline method and we perform thousands self-playing games in each iteration (see Figure 3.3 again). By doing so, each move may be played nearly the same number of times in the first iteration. Table 3.4, 3.5 and 3.6 shows how book moves in the initial position are played for tuning Type-6[1-1], Type-6[1-3] and Type-6[1-5]. These tables clearly show that actually each move is played nearly the same number of times in the first iteration, and the order of weights calculated as a difference between winning times and losing times is the same with the order of winning rate. In addition, we use relatively large Δw in early iterations. This helps to assign high weight values to good moves and low weight values to bad moves. Thus good moves are often played. Therefore, we consider that the problem mentioned above can seldom happen.

Table 3.6: How book moves are played for tuning Type-6[1-5]

Book move	(1)	(2)	(3)	Winning rate (mean value)	$Wins - Loses$ (sum value)	(4)	(5)
P-7f(7g)	781	15.62%	385	49.30%	-11	4	4
P-2f(2g)	713	14.26%	368	51.61%	23	1	1
P-5f(5g)	731	14.62%	327	44.73%	-77	7	7
R-5h(2h)	719	14.38%	348	48.40%	-23	6	6
P-1f(1g)	687	13.74%	341	49.64%	-5	3	3
P-9f(9g)	676	13.52%	338	50.00%	0	2	2
R-6h(2h)	691	13.82%	335	48.48%	-21	5	5

(1) How many times the book move was played.
(2) The rate of the book move was played.
(3) How many wins led by the book move.
(4) Order of mean value.
(5) Order of sum value.

How often exploit and explore?

Concerning the balance between exploiting and exploring, random values generated with the parameter *Rand* are used to keep a proper balance in our proposed method. This corresponds to the use of relatively large τ in the softmax action selection rule. On the other hand, we have to assign different *Rand* to Type-6[x-1], Type-6[x-2], Type-6[x-3], Type-6[x-4] and Type-6[x-5] while only one τ is required in the softmax action selection rule or only one ϵ in the ϵ-greedy method. We decided each *Rand* by considering the difference between the highest weight and the lowest weight in a position. It is useful to prepare fewer numbers of parameters to realize the same effect. Therefore, we can enhance our method in the sense of decreasing the number of learning parameters, based on the ϵ-greedy method and softmax action selection rule.

How to initialize Q_0 ?

In our method, all weights are initialized as zero and this is not a problem. Although weights mean sum values in our method, we cannot apply the method to the case where all weights are initialized as an optimistic initial value. Therefore, our method cannot be flexible in this point. On the other hand, we can assign relatively high weights to moves regarded as good moves and low weights to moves regarded as bad moves by counting the number of moves played in master games. Actually, we tune the opening book for the tournament version of TACOS in this way.

3.3.4 Self-playing experiment with reinforcement learning

We tune two opening books by reinforcement learning through self-playing games, and perform self-playing experiments. Here we call Type-7[1-1] opening book and Type-7[2-1] opening book. We tuned Type-7[1-1] opening book in offline through 1000 self-playing games with sample-averaging method and ϵ-greedy ($\epsilon = 0.1$) method. Then Type-7[2-1] opening book is tuned in offline through another 1000 self-playing games in the same way. Using Type-1, Type-2, Type-6[1-1], Type-6[2-1], Type-7[1-1] and Type-7[2-1] opening books, then we perform a similar tournament as performed in the previous section. Since

Table 3.7: The results of the additional tournament

	result
Type-1 - Type-2	51 - 49
Type-1 - Type-6[1-1]	49 - 51
Type-1 - Type-6[2-1]	45 - 55
Type-1 - Type-7[1-1]	36 - 64
Type-1 - Type-7[2-1]	43 - 57

the machines we used in the previous tournament and this tournament are different, we perform self-playing games played by Type-1 and Type-2 for comparison. In Table 3.7, we show the results of this tournament.

From the result of this tournament, we can confirm that Tacos with opening books tuned by reinforcement learning won more games than that with opening books tuned by our proposed method. It is hard to analyze the results in more detail due to the small number of games. Although we do not perform more games in this time, we realize that we may able to enhance our method based on reinforcement learning.

3.4 Conclusions

In this chapter, we studied the use of an opening book to improve the opening play of computer Shogi. Considering the strength of computer Shogi and the complexity of Shogi, it is better for computer Shogi to have a large opening book that contains lots of positions and moves. Thus we generated a large opening book from master games and tuned that opening book for a target program. Although some tuning methods as used in Deep Blue or Crafty have been proposed, it is hard to apply these methods in the domain of Shogi. Therefore, we proposed a new method to tune an opening book through many self-playing games. The results of some experiments show that using our method an opening book can be well tuned for a target program. Tuning with this method, it is possible to create a good opening book for Tacos with small implementation costs.

Based on the proposed method, we created an opening book for Tacos to participate the 15th World Computer Shogi Championship and updated the book for the 10th Computer Olympiad. It was clear that the well tuned book led Tacos to promising opening positions in most tournament games, then won the Computer Olympiad.

Chapter 4

Piece Development

This chapter is an updated and abridged version of

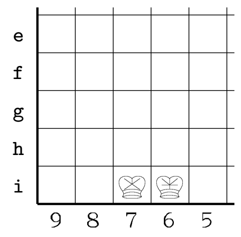
1. J. Nagashima, M. Taketoshi, Y. Kajihara, T. Hashimoto and H. Iida: "An Efficient Use of Piece-Square Tables in Computer Shogi", IPSJ SIG Technical Reports 2002-GI-8, pp.29-36. Edmonton, Canada. July, 2002.

2. J. Nagashima, M. Taketoshi, Y. Kajihara, T. Hashimoto and H. Iida: "Several Enhancements for Better Development in Computer Shogi", 7th Joint International Conference on Advanced Science and Technology (JICAST 2002), pp.306-309. Hamamatsu, Japan. October, 2002.

In this chapter we describe some measures to improve the piece development of computer Shogi. In the previous chapter we discussed the opening play using an opening book. Although the use of opening book is very effective, an opening book is able to assist a computer only in the in-book positions. Since in Shogi games often go out-of-book positions in the early stage even in master games, computer Shogi is required to play a reasonable opening in such out-of-book positions.

In the opening, a computer cannot grasp the situation by evaluating some simple features such as material and King safety, because those features are usually even. One would make a formation, i.e., piece development in the opening stage. Then a computer has to observe the situation by evaluating the formation in the opening. In addition one has to understand the possibility of the opponent's early attacking in the opening stage. Since the move sequences of attacking are relatively not short, it is hard for a computer to search those moves completely within the practical time. Therefore, computer Shogi sometimes overlooks the opponent's early attacking and falls into a disadvantageous position. To avoid this, a computer has to notice the opponent's early attacking with the use of an evaluation function.

This chapter contains the following sections. Section 4.1 presents the methods used in many computer Shogi programs to construct formations. However, with these methods a computer cannot evaluate some formations properly. Therefore in section 4.2, we describe another method that can evaluate those formations. In section 4.3, we discuss the use of an opening book for piece development in the out-of-book positions.



```
// Table for Gold                  // Table for Silver
{// 9  8  7  6  5  ···             {// 9  8  7  6  5  ···
   ···                                ···
   { 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, 0,14, 0, 0, ··· },
   { 0, 0, 4, 0, 0, ··· },            { 0, 0, 8, 0, 0, ··· },
   { 0, 0, 0, 0, 0, ··· },            { 0, 0, 0, 0, 0, ··· },
},                                 },
```

Figure 4.1: An example of otoshi-ana method

4.1 Hill Climbing with Piece-Square Table

To make a formation in the opening, one has to know the following two points. The first one is the shape of the target formation to be made. The second one is the order of moves. To realize this idea in the domain of computer Shogi, so-called "otoshi-ana method", i.e., hill climbing, has been used. We first explain the otoshi-ana method in section 4.1.1. Then we describe two methods that are natural extensions of the otoshi-ana method in section 4.1.2 and 4.1.3. In section 4.1.4, we discuss some problems that have to be solved in implementing the otoshi-ana method.

4.1.1 The basic idea of otoshi-ana method

Otoshi-ana method has been proposed by Yamashita and used in his Shogi program YSS, one of the strongest Shogi programs [41, 46]. This idea is incorporated in many other Shogi programs for a master-like formation in the opening. Otoshi-ana method uses a set of tables that have evaluation scores if there is a specific kind of piece in the tables. A position with the highest weight means the destination for a piece that refers to the table. A margin between a position and next position shows the order of moving. A move having the largest margin is to be selected.

For example, assuming an otoshi-ana table and a position of Figure 4.1, the destination of Gold is **7h**, and the destination of Silver is **7g**. Moves to complete a formation are **S-7h(7i)**, **S-7g(7h)** and **G-7h(6i)** by margin (+8, +6, +4). It means that **S-7h(7i)** takes $8 - 0 = +8$ point, **S-7g(7h)** $14 - 8 = +6$ point and **G-7h(6i)** $4 - 0 = +4$ point. Here, the point is given by the difference between the score for the destination and the score for the present square.

A computer is able to select various strategies to use these different piece-square tables and choose one in the actual game. Although this idea can be implemented without much difficult, the weight tuning for the tables are required that a computer may be able to choose the collect move orders and evaluate formations properly.

4.1.2 Attack table and castling table

As stated in section 2.1.2, there are many possible formations in Shogi. Those formations are classified roughly into two groups; formations for attacking and formations for castling.

Thus one would make a formation that consists of attacking and castling formations. For example, the formations of Climbing Silver and Right Side Fourth File Rook are used with the formation of Yagura Castle. Those formations for attacking are also used with other castling formations. Since there are many such combinations, otoshi-ana tables for all combinations would occupy too much memory space. In order to make a computer construct formations in a flexible way with smaller memory requirement, a method dividing otoshi-ana tables into attack tables and castling tables has been proposed [83].

There are two merits in dividing the otoshi-ana tables. The first, the total numbers of weights in the whole tables can be reduced. This means that the costs of tuning these weights can be also reduced. The second, a computer is able to construct various formations with smaller memory space. Although this method has these merits, we do not apply it to TACOS due to the following problems:

- The priority of the attack tables.
- How to decide the combination of the attack and the castling tables.

The problem on the priority of attacking tables is caused by some attack tables used in the strategies for the early fight as well as the slow game. One would give precedence to the attacking moves over the castling moves in an early fight game, whereas one might give precedence to the castling moves over the attacking moves in a slow game. Moreover, the degree of giving precedence depends on the castling and attacking formations of both players. Therefore, it is very hard to tune weights to let a computer construct formations in the correct move order.

The second problem is how to decide a combination of the attacking and castling tables. Masters would choose their target formations while considering own current formations and the opponent's formation. It seems that as players become stronger, the players consider the more detailed part of positions to decide the formations. Thus to select a proper combination of the attacking and castling formations requires high-level domain knowledge of Shogi. Since it is hard to implement such high-level knowledge for selecting better formations, this method does not work sufficiently for the top master-level at present.

4.1.3 Using formation tables for opening and middle games

Piece-square tables have been used in two ways: one is for developing a formation; another one is for evaluating a formation. Thus the weights in tables have different three meanings.

1. Evaluation for the complete formations.
2. Orders of move sequences to complete formations.
3. Evaluation for formations missing some part.

Containing these different types of weights, piece-square tables cause two problems. The first one can be caused by the above (2). To develop a formation, intermediate squares have weights classified in (2) even if the location is worthless in evaluating a formation. When a computer tries to defend own King, it may not be a problem that a computer moves a piece into squares of (1) or (3), but it can be a problem that a computer moves a piece into squares of (2). The second problem is caused by (3). Squares observed in

```
 // for Yagura Castle of King
{ // 9   8   7   6   5  ···
  ···
  {  0,  0,  0,  0,  0, ··· }, // 5
  {  0,  0,  0,  0,  0, ··· }, // 6
  {-30,-30,-30,-30,-50, ··· }, // 7
  { 80,100, 60,  5,-50, ··· }, // 8
  {  0,  0, 67, 31,-23, ··· }, // 9
},
```

Figure 4.2: An example of piece-square table for King in opening

```
 // for Yagura Castle of King
{ // 9   8   7   6    5  ···
  ···
  {  0,  0,  0,  0,   0, ··· }, // 5
  {  0,  0,  0,  0,   0, ··· }, // 6
  {-50,  0,-50,-50,-100, ··· }, // 7
  {-50,125,-20,-50,-100, ··· }, // 8
  {  0,  0,-20,-50,-100, ··· }, // 9
},
```

Figure 4.3: An example of piece-square table for King in middle game

```
 // for Yagura Castle of Gold
{ // 9   8   7   6   5  ···
  ···
  {  0,  0,  0,  0,  0, ··· }, // 5
  {  0,  0,  0,  0,  0, ··· }, // 6
  {  0,  0, 20, 44,  0, ··· }, // 7
  {  0,  0, 46,  0, -7, ··· }, // 8
  {  0,  0,-20,-11,-40, ··· }, // 9
},
```

Figure 4.4: An example of piece-square table for Gold in opening

```
 // for Yagura Castle of Gold
{ // 9   8   7   6   5  ···
  ···
  {  0,  0,  0,  0,  0, ··· }, // 5
  {  0,  5,  5, 10,  5, ··· }, // 6
  {  0, 10, 20, 40,  5, ··· }, // 7
  {  0,  0, 50, 25,  0, ··· }, // 8
  {  0, 15, 10,  0,  0, ··· }, // 9
},
```

Figure 4.5: An example of piece-square table for Gold in middle game

(3) are located around squares considered in (1). Thus a computer may be disturbed from constructing a formation in the correct order by moving the necessary pieces into squares as considered in (3). For these problems, we propose that piece-square tables must be divided into two parts: one has values for making a formation and the other has the weights for the evaluation of formations. A computer uses the tables for making formations in the opening and the tables for the evaluation of formations when a formation completes. Let us show, in Figure 4.2 - 4.5, some examples of these tables.

4.1.4 Problems of implementation and its solutions

We found some problems in the process of implementation for piece-square tables. Here we discuss those problems and solutions.

Problems on the reverse orders of move sequences

While evaluating formations using piece-square tables in the leaf nodes of a game tree, a computer selects candidates and completes a formation. During a search, a position may appear after different paths. For example, we consider a case where a computer looks ahead by 3-ply in the position of Figure 4.1. In order that a computer may play the move sequence of **S-7h(7i)**, **S-7g(7h)** and **G-6h(6i)** in that position, weights in the tables should be tuned. Choosing the move order results in 18 points of the score, while gaining the same 18 points by other orders such as **S-6h(7i)**, **G-7h(6i)** and **S-7g(6h)**. Since both move orders lead the same position as shown in Figure 4.6 due to the same score, a computer may choose an unexpected sequence of moves.

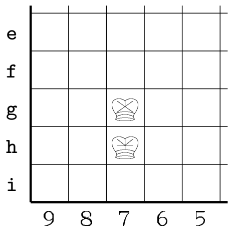


Figure 4.6: A leaf position of 3-ply search

Then, a question arises: "why does this problem happen? Both of the move orders seem good because those scores are even." This question is natural, but in the following case the reverse orders of move sequences can lead disadvantageous positions;

1. A computer plays the reverse orders of a move sequence.
2. Taking advantage of this reverse order of a move sequence, the opponent attacks with long steps of moves.
3. Before playing the reverse order of a move sequence, a computer cannot grasp the whole order of countermoves of the opponent; therefore a computer cannot understand in advance the risk of the reverse order.

To prevent the opponent's attacking, the weights in piece-square tables should be tuned. Therefore, a computer has to play in the correct orders of moves even if the reverse order provides the equal scores.

To let a computer make a formation in the correct order, we have to distinguish whether a position is led in the correct orders or not in the leaf positions of game tree. To settle this, we obtain some scores that are proportion to the margin of evaluation of formations caused by moves played in the first ply during search. Gaining this score, the correct order of moves and reverse order of moves can have different scores. For example, the evaluation of a position by the move sequence **S-7h(7i)**, **S-7g(7h)** and **G-6h(6i)** obtains 18 points and some extra points that is proportional to 8 points (margin of **S-7h(7i)**), whereas the move sequence **S-6h(7i)**, **G-7h(6i)** and **S-7g(6h)** has 18 points and 0 points that is proportional to 0 points (margin of **S-6h(7i)**). Using this method, TACOS can play moves to make a formation in the correct order.

No stability under variable-depth search

When using not-fixed depth search such as realization-probability search [49, 50] or selective deepening like search extension, the move recommended with the highest priority by piece-square tables cannot be sometimes chosen. Figure 4.7 shows the progress of the changes of position evaluation in function of search depth. Here, ● is higher than ■ when positions are evaluated at the same ply. However, ■ is higher than ● when positions are evaluated at the different ply.

Searching more deeply can solve this problem. But if the depth is not enough, the problem still remains. Gaining the bonus values that are given for avoiding the reverse orders of moves can also avoid this problem.

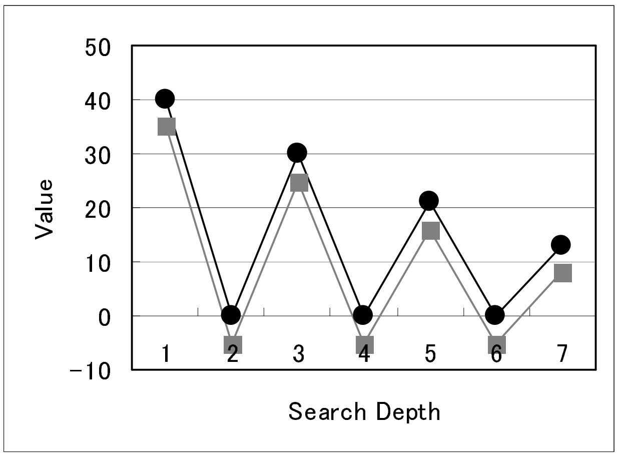


Figure 4.7: The changes of evaluation values

Problem with prior moves independent of piece-square table

There are some cases that prior moves in a piece-square table cannot make a suitable formation. For example, let us consider a position shown in Figure 4.8. In this position, the following two problems are caused: (1) a computer cannot play **P-9f(9g)** when playing Black side, and (2) a computer cannot play **P-9e(9d)** when playing White side.

In that position, there are some other moves including **S-5g(4h)** (for Black) or **S-7b(7a)** (for White) that has the higher priority according to the piece-square table. Since **P-9f(9g)** has the lower priority, a computer may play **S-5g(4h)** while considering that the opponent will play **S-7b(7a)**. In addition in case where the opponent plays **P-9e(9d)**, a computer becomes a little happy because White gains a smaller score by playing **P-9e(9d)** than playing **S-7b(7a)**. After a few moves, a computer notices that Black lost the opportunity to play **P-9f(9g)** no longer and becomes a little disadvantage that is caused by the margin between **P-9g** and **P-9f**. This margin is small in the square table because they express just the order of moves in the opening, but actually the impossibility of playing **P-9g** causes a big disadvantage after the opening in this strategy. In brief, there are some moves that have to be played immediately as a countermove of the opponent's move. It is hard to realize the disadvantage of not playing such a countermove beforehand by searching with simple piece-square tables. Therefore, a framework that enables a computer to notice that playing a countermove is required.

In this framework, we evaluate countermoves directory [43] to take precedence of those moves over the moves that have higher priority in the piece-square tables. In TACOS, a position is explored whether there are any countermoves or not before searching. If there are some moves that have to be played immediately, some bonus scores are given if those moves are searched in the first ply of the game tree. In the same way, some moves that must not be played in a position are also examined, and some scores are reduced if those moves are searched in the first ply of the game tree.

How to determine an appropriate piece-square table

To play the opening game properly, considering the opponent's moves, one would decide a long-term strategy. Therefore, a computer has to prepare some piece-square tables and decide a proper table to use. Here we consider a way to decide a proper table in some

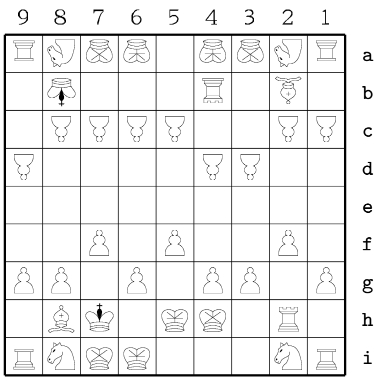


Figure 4.8: Prior moves and counter moves

cases.

The first case is that an opening book leads a computer to the position in which a formation is complete. In this case, the only thing to realize for a computer is considering what formations a computer and the opponent made during the opening line. An opening book can make formations that a computer cannot understand, but as mentioned in chapter 3, by tuning an opening book properly, this problem can be avoided in most cases. In case where the opponent makes a formation that a computer cannot understand, it may evaluate that formation by using tables of the most similar formation.

In another case is that an opening book does not lead a computer to the position where a formation is complete. In this case, how does a computer choose the piece-square tables for evaluating formations? In [83], a computer decides the piece-square tables by which it evaluates a position as the highest. We consider that this method can evaluate unsuitably in some cases. For example, all pieces are located in correct squares except the piece that is very important for constructing a formation. In such case, the most similar formation cannot be chosen properly although the evaluation value is the highest among all square tables.

To avoid this problem, we prepare many blueprints of various formations. The size of these blueprints is 5×5, and the set of the location of pieces and indication of tables for evaluation is contained in it. While looking at a position, TACOS is able to find the closest blueprint and decide the tables for evaluating formations. There are some positions where a computer can construct more than two formations. Each blueprint for those positions indicates only one formation where we consider that TACOS can play the best among all formation candidates. For example, from the position shown in Figure 4.9, both Yagura Castle (Figure 4.10) and Gangi Castle (Figure 4.11) can be constructed. In the blueprint for the position shown in Figure 4.9, we intend to make Yagura Castle because TACOS seems to play better with Yagura Castle than Gangi Castle.

In case where there are no close blueprints, TACOS tries to recognize the types of strategies such as Static Rook, Ranging Rook or Floating Rook of both sides by considering the location of Kings, Rooks and some other key pieces. After recognizing strategies, TACOS decides provisional tables by applying some rules that are given by hand.

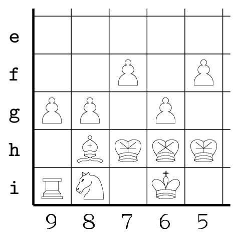


Figure 4.9: Crab Castle

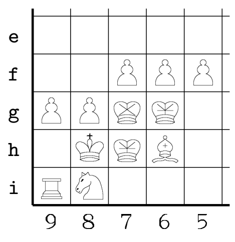


Figure 4.10: Yagura Castle

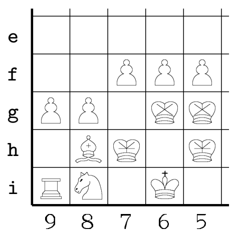


Figure 4.11: Gangi Castle

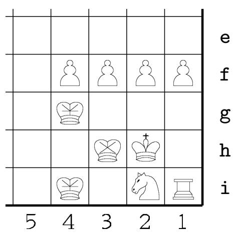


Figure 4.12: High-Mino Castle (Pos-A)

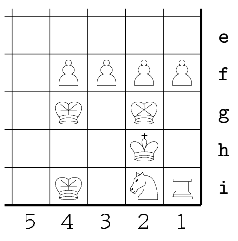


Figure 4.13: The Halfway position (Pos-B)

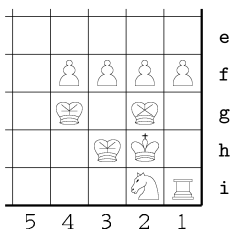


Figure 4.14: Silver Crown (Pos-C)

4.2 Combination Evaluation

Using piece-square tables, a computer can evaluate a formation at some level. However, a computer evaluates only one piece separately at the same time. Therefore, a computer cannot evaluate formations from the viewpoint of the relation between some pieces. For example, let us consider the positions shown in Figure 4.12 (Pos-A), 4.13 (Pos-B) and 4.14 (Pos-C). These positions appear when one tries to develop a formation Silver Crown from High-Mino Castle. Since the destination of Silver is **2g**, the evaluation of Silver in **2g** is higher than in **3h**. In the same way, the evaluation of Gold in **3g** is higher than in **4i**. Therefore, the evaluation of formation in the position of Pos-C is the highest. The position Pos-B is the second highest and the position Pos-A is the smallest. Although a computer estimates that the position Pos-B is better than the position Pos-A, the position Pos-B is worse than the position Pos-A in actual games because the Gold locates in **4i** is unstable. To let a computer understand this instability, we have to evaluate the combination of plural pieces.

For this evaluation, we implement an evaluator called combination evaluation to Tacos. Using this evaluation function, Tacos reduces some evaluation value when it finds some unstable combination of pieces in a position and also gains when it finds some desirable combination. For example, Tacos reduces some evaluation value in the position Pos-B by applying the rules that indicate subtraction if King locates in **2h**, Silver locates in **2g**, Gold locates in **4g** and **4i**.

Using this evaluation function, Tacos can evaluate more correctly. On the other hand, this evaluation function can cause the decline of search speed because it has to examine whether some combinations that have to be evaluated are seen in a position. Therefore, we have to confirm whether this evaluation function improve the playing strength of

Table 4.1: The result of experiment with combination evaluation

	Tac-A	-	Tac-B
10 sec / move	106	-	94

Tacos or not. To confirm this, we perform a self-playing experiment. We prepare two types of Tacos: Tac-A is with a combination evaluation function and Tac-B is without the combination evaluation function. Except from the combination evaluation function, both programs are same. We also prepare 100 even positions from an opening book. From these positions, both programs play 100 games as Black and 100 games as White. Each move is played within 10 seconds. The results of this experiment are shown in Table 4.1. From the results, Tacos seems to be improved its playing strength with the combination evaluation function.

At the present, the combinations of pieces that are evaluated are given by hand. As future works, we would try to extract those combinations automatically from master games.

4.3 Partial Position Opening Book

In an out-of-book position, one would select a move while considering book moves in some in-book positions that look like that position. In the same way, we expect that Tacos may consider book moves even in the out-of-book positions to improve the opening play.

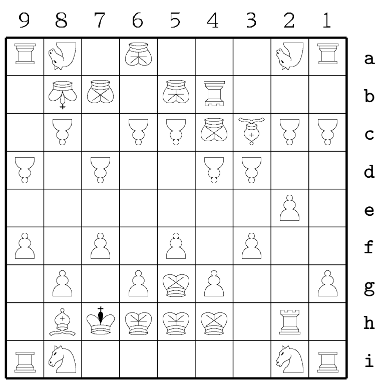
At first, we have to consider the similarity of positions. It is required that some squares that have influence on the book move orders have to coincide between an in-book position and an out-of-book position to play book moves in an out-of-book position. In other words, other squares that do not have any influence on the book move orders do not have to coincide. However, squares that have influence on the book move orders depend on book positions and book move orders. Therefore, it is very hard to judge whether an out-of-book position is similar to an in-book position. Therefore, we make the following steps:

1. Judging roughly similarity between an in-book position and an out-of-book position.
2. If an out-of-book position is judged as similar to an in-book position, book moves are used for the move ordering or direct move evaluating.

Since the judgments of similarity are rough, a computer cannot play book moves immediately without searching. Using book moves during search, a computer can avoid playing book moves that should not be played in a position immediately.

To improve the quality of search, it is required for a computer to refer the book moves not only in the root node but also in the internal nodes of game tree. In addition, as mentioned in chapter 3, a large opening book is used for computer Shogi. For these reasons, the judgment of similarity between an in-book position and an out-of-book position has to be done as quickly as possible.

How can we judge the similarity of positions? In making an opening book, we can store positions in a hash table where we use a hash key made from a position to retrieve



P-4f(4g)	103	**P-1f(1g)**	39
S-3g(4h)	81	**P-2d(2e)**	11
P-4e(4f)	76	**R-3h(2h)**	10
S-4f(5g)	63	**S-4g(4h)**	7
N-3g(2i)	44	**S-5f(5g)**	5

Figure 4.15: An example of out-of-book positions

quickly. In this case, we also store positions in a hash table with which we use a hash key made from a position. Considering partial squares we make a hash key in this case, while looking all squares in the opening book construction. To consider fixed squares or pieces in any positions allows making a hash key quickly. Thus we make a hash key from the following information:

- Belonging and location of King, Rook, Bishop, Gold and Silver.
- Location of Pawn that exists in front of own Rook.

From the positions stored in the opening book, we make a partial position opening book in the same way of an opening book construction. Some different positions may have the same hash key. This is because in those positions pieces that are described above are to be located in the same squares. Therefore, those positions are regarded as the same position in this database. Figure 4.3 shows an example. This position is led by White's out-of-book move **P-7d(7c)**. In the database, we can see book moves such as **P-1f(1g)**, **S-4F(5g)**, **P-4f(4g)** and **S-3g(4h)** that are stored in positions led by White's book moves of **P-6d(6c)** or **L-1b(1a)** in an opening book.

In every node of a game tree, a computer retrieves the current position from the partial position opening book. After searching very important moves such as killer moves, capturing valuable pieces and so on, book moves stored in the partial position opening book are searched if the current position is found. Generating book moves after very important moves allows reducing costs to retrieve the current position from a partial position of opening book if these important moves cause the beta-cutoff.

To confirm the effectiveness of this proposed method, we perform a self-playing experiment. We prepare two types of Tacos: Tac-C is with a partial position opening book and Tac-D is without the partial position opening book. Except from the partial position opening book, both programs are same. In this experiment, 100 even positions from an opening book are prepared, and from these positions, both programs play 100 games as Black and 100 games as White. Each move is played within 10 seconds. The result of this experiment is shown in Table 4.2. From this result, we cannot confirm any effect of a partial position opening book. Although the order of move sequences played by Tac-C seem to be more natural than Tac-D, the changes of move sequences do not have any influence on the result of games. Since it is hard for a computer to take advantage

Table 4.2: The result of experiment with partial position opening book

	Tac-C	-	Tac-D
10 sec / move	100	-	100

of opponent's strange move orders, it may be hard to confirm the effect of this method through self-playing games. However, considering the games played on the Internet, we confirmed that this method improved the opening play of Tacos.

In this study, squares and pieces that are used to make a hash key is given by hand. As a future work, deciding squares and pieces used in a hash key automatically by analyzing master games is worth trying.

4.4 Conclusions

In this chapter, we introduced some methods that improve the opening play for computer Shogi. In the implementation of piece-square tables for making formations, we pointed out some problems and countermeasures for them. Since piece-square tables cannot evaluate combinations of pieces, we proposed a function of combination evaluation. In addition, we implemented a partial position opening book to consult an opening book in out-of-book positions. Experimental results show that the implementation improves the playing strength of Tacos by stabilizing the opening play.

In this chapter, we sometimes used domain knowledge. To acquire such knowledge automatically from game scores is a future work. Acquiring knowledge automatically will enable a computer to play more stable opening while evaluating more combinations of pieces and judging similarity of positions more correctly.

Chapter 5

Various Enhancements

This chapter is an updated and abridged version of

1. J. Nagashima, T. Hashimoto and H. Iida: "Alternative use of two transposition tables for game tree search", IPSJ SIG Technical Reports 2006-GI-15, pp.49-55. Tokyo, Japan. March, 2006.

2. J. Nagashima, J. Hashimoto and T. Hashimoto: "Opportune Time Recognition of Edge Attacks in Computer Shogi", The 11th Game Programming Workshop (GPW 2006), pp.1-8. Hakone, Japan. November, 2006.

In the previous chapters we have presented our new ideas to improve the performance of computer Shogi with focus on the opening which includes the book and piece development. Towards the master-level play we need more improvements in the opening as well as in other stages. Some of these should be applied in most practical circumstances, since added performance outweighs the additional implementation effort. Several enhancements exist.

In this chapter we focus on two important enhancements we applied in Tacos. In section 5.1 we focus on the use of transposition tables. Section 5.2 deals with the opportune moment to attack on the edge file.

5.1 Alternative Use of Two Transposition Tables

The use of transposition tables enables a computer to efficiently search a game tree. A transposition table has been used during search for the present position [2]. We propose an idea to use alternatively two different transposition tables. One is for game-tree search in a position where a player is to move, whereas another one is for game-tree search in the position where the player is to move next. We implement it on Tacos to evaluate. The results of experiments performed show that the proposed idea improves the search efficiency as well as the playing strength.

5.1.1 Introduction

A computer often reaches the same positions after different move sequences during a game-tree search. Let us show, in Figure 5.1, an example of such a case. In the figure, position **F** appears after two distinct move sequences: (**A** $\rightarrow$ **B** $\rightarrow$ **F**) and (**A** $\rightarrow$ **C** $\rightarrow$ **F**). Suppose a case where a computer first meets the position **F** with the move sequence (**A**

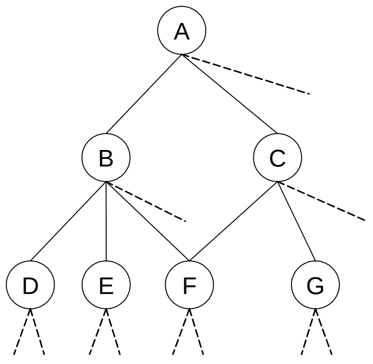


Figure 5.1: An example of a game tree that leads the same position in the different path

→ **B** → **F**) and meets it again with the other move sequence (**A** → **C** → **F**). If a computer can use, in the second visit, the search results obtained in the position **F**, then the search in the second visit can be omitted. For this purpose one reasonable way is the use of a transposition table. It is implemented as a hash table. In the table search information such as an evaluation score for a position is stored together with a hash key to express the position code. In addition to these, some other information such as the best move of a position may be stored. There are some related works on the use of transposition tables with the following research questions:

1. What kind of information would work the best?
2. What kind of positions should be stored in a transposition table?

In the domain of computer Shogi the previously-stored transposition table is cleared (or released) from the memory when starting game-tree search in the current position. This means that computers have to waste some valuable information obtained in the previous search. Although leaving and referring the entries stored in the previous search may enable a computer to search effectively, leaving old entries may cause the following problems: If a computer does not overwrite, too many entries let a computer speed down in searching and adding entries in transposition tables. Even in a case where a computer overwrites entries, the same problem will also happen in overwriting many times.

In this section we propose a method that uses two different transposition tables alternatively. With this method a computer may be able to use valuable information stored from the previous search and then its performance can be improved.

5.1.2 Related works

If the number of entries to be newly stored from the present search exceeds the assigned number of entries in a transposition table, it is no longer possible to leave all entries. Therefore, it is required to leave only useful entries and delete useless entries. To tackle the problem, some replacement schemes were suggested.

In the thesis [84] [85], the following seven replacement schemes are examined.

- *deep* : a position is replaced when another position that is searched deeper than the position comes into collision.

- *new* : a position is replaced when another position comes into collision.
- *old* : a position is not replaced even if any other position comes into collision.
- *big1, bigAll* : a position is replaced when another position that is searched more nodes than the position comes into collision.
- *twoDeep, towBig* : these schemes are combination of *new* and *deep*/*big1*. All entries in transposition tables can store two positions and one position is used for *new* and the other is used for *big1* or *deep*.

In Crafty a transposition table is implemented in the twoDeep way [77]. Each entry in a transposition table can store three positions. One position is used for the *deep* scheme, and the other two positions are used for the *new* scheme. In many domains, this kind of replacement scheme is applied.

5.1.3 Alternative use of two transposition tables

Most Shogi programs clear information once stored in transposition tables before starting game-tree search in the current position. For example, a transposition table is implemented for Yss, one of the strongest computer Shogi, in the following ways [41].

- Yss allocates the hash address 16 times in the maximum case by open addressing.
- Yss does not register new positions under the condition that positions are already registered 0.75 times as many as maximum positions that the transposition table can contain.

Although there is no description, it is obvious that Yss has to clear the transposition table before staring the search in the position to come.

According to [44], another strongest computer Shogi Gekisashi thinks on the opponent time while predicting the opponent next move. If the opponent plays the move expected, Gekisashi starts the search without clearing the transposition tables. We then understand that Gekisashi clears the transposition table when starting the search under the normal condition. To clear the transposition table means that any information obtained in the previous search is never used.

Figure 5.2 illustrates the previous and current search trees. During the game-tree search some information such as position's score and the best move at each position visited can be stored in a transposition table. When visiting the same positions, the results are already known by looking the entries stored instead of searching again.

There are some reasons to clear entries from transposition tables before starting the search in the position to come. Leaving all entries of the positions searched in the previous search causes the lack of available spaces in the transposition table and may often come into collisions. Therefore, when using a transposition table without any replacement scheme, a computer must try rehashing over and over. When using a transposition table with some replacement schemes, a computer must replace more times than when clearing the transposition table before starting the next search.

We consider another way to leave and clear positions in a transposition table. All positions stored in a transposition table during the previous search are to be considered whether each position is useful or not for the search in the position to come. However, it

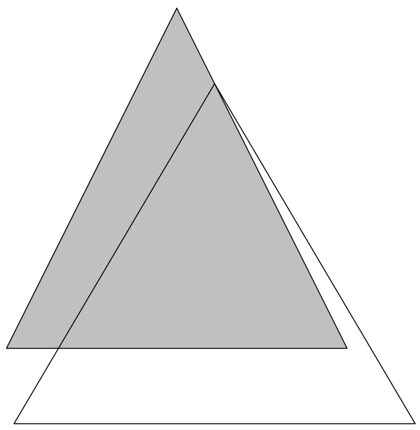


Figure 5.2: The image of search space in a game tree

takes expensive costs to check all positions stored in the transposition table. Therefore, this way is not useful.

In this section, we propose a method to use two different transposition tables alternatively. Let us show the outline of this method.

- Prepare two same transposition tables, say transposition table tA and tB.
- Prepare two pointers (say pP and pC). The pointer pP means that the transposition table referred by pP has positions searched in the previous search. On the other hand, the pointer pC means that the transposition table referred by pC will store positions searched in the current search.
- Both transposition tables are cleared and each transposition table is referred by different pointer when a new game starts. For example, transposition table tA is referred by pointer pP and transposition table tB is referred by pointer pC.
- Before searching, the transposition table referred by pP is cleared. This table stores information on the positions searched in the previous time.
- After clearing, swap pointers. For example, pP referred tA and pC referred tB. After swapping pP refers tB and pC refers tA.
- When a computer checks transposition tables during search, first the transposition table referred by pC is checked. Only when a computer could not find the target position in first transposition table, the transposition table referred by pP is checked.
- A computer stores a position in the transposition table referred by pC.

Figure 5.3 shows the source code written like C++.

There are only two difference between common transposition tables and this implementation. The first one is swapping tables before searching and the second one is that the previous table is checked when the target position cannot be found in the current table. Therefore, this method can implement easily. On the other hand, this method has to prepare two transposition tables, thus this method requires much memory. We consider that when a computer can allocate enough memory, this method would work well.

5.1.4 Experiments

In this section, we implement our proposed method on Tacos. The transposition table of Tacos is implemented as a hash table that uses open addressing for the experiments performed in this section. When collisions happen in transposition tables, Tacos does not replace and try rehashing. In the case where a transposition table does not have enough space in order to add new positions, Tacos stops searching. This transposition table uses 64 bit code that is made from position information as a hash key and holds evaluation value, best move in a position and so on. Tacos usually runs under the condition that Tacos allocate about 192 MB memory for transposition tables. In this case, transposition tables can hold 2^{22} positions and Tacos can search for about 35 seconds on Athlon 64 X2 4800+ computer. More details of Tacos is shown in section 2.3.

We prepare two versions of Tacos for the experiments. On *Tac-1*, we do not implement our method. On the other hand, we implement our method on *Tac-2*. *Tac-1* and *Tac-2* are the same programs except for transposition tables. Using these programs, we perform two kinds of experiments. Both experiments are performed on Athlon 64 X2 4800+ computer.

Performance test with many positions

First, we perform an experiment that compares the number of nodes visited in the fixed time or times that programs take during search and so on. We prepare 100 positions that appear experts' games played on the internet Shogi server [78] [79]. Each position appears between 30th moves to 80th moves from the initial position. We perform this test under the following six conditions.

- Fixed time. The programs search for 10 seconds or 25 seconds.
- Fixed iteration. The programs search with 10 or 12 iterations.
- Fixed number of node. The programs stop when they finish searching 1.5 or 3 million nodes.

Under these conditions, we check the maximum depth that programs reach, the number of iteration, the number of nodes visited, time that programs spent for searching. *Tac-1* directly searches the test positions. On the other hand, *Tac-2* searches the test positions after searching positions that appears two moves before the test positions.

Under the condition where the time for search is fixed, it seems to be not so big difference except for the number of nodes visited. The reason why the number of visited nodes of *Tac-2* is less than *Tac-1* is the decrease of search speed of *Tac-2*. *Tac-2* tries to find a target position from the previous transposition table when the target position could not be found in the current transposition table. This task takes extra costs, thus *Tac-2* reduces its search speed. On the other hand, the number of visited nodes of *Tac-2* is also less than *Tac-1* under the condition of the fixed iteration. We consider that finding useful information from the previous transposition table, *Tac-2* can perform the search effectively. Therefore, while searching fewer nodes, *Tac-2* can finish searching. The result under the fixed node condition shows that the number of iteration of *Tac-2* is bigger than *Tac-1*. We observe that this result also means that *Tac-2* can search more effectively than *Tac-1*. In other cases, the maximum depth and number of iteration do not show any

Table 5.1: The results of searching experiment

Search condition	Depth	Iteration	Node	Time
Tac-1 (10 sec / move)	18.5	12.27	184.7	-
Tac-2 (10 sec / move)	18.6	12.24	179.8	-
Tac-1 (25 sec / move)	20.2	13.14	445.8	-
Tac-2 (25 sec / move)	20.1	13.24	425.7	-
Tac-1 (10 iteration)	15.3	-	52.0	9.27
Tac-2 (10 iteration)	15.3	-	46.7	8.70
Tac-1 (12 iteration)	18.6	-	275.8	25.46
Tac-2 (12 iteration)	18.4	-	273.3	25.52
Tac-1 (1.5 million node)	18.2	12.03	-	18.25
Tac-2 (1.5 million node)	18.5	12.10	-	18.20
Tac-1 (3 million node)	19.5	12.76	-	27.89
Tac-2 (3 million node)	19.4	12.82	-	28.23

Depth : maximum search depth	Iteration : number of iteration
Node : number of search nodes (million)	Time : search time (second)

big difference between *Tac-1* and *Tac-2*. This is because *Tac-2* can search effectively but reduce its search speed. Therefore, a good effect and bad effect of the proposed method deny each other.

Self-playing experiment

Next, using *Tac-1* and *Tac-2*, we perform a self-playing experiment. TACOS does not use any random factors in the opening, thus self-playing games when starting from the initial position become almost the same games. Avoiding this problem, we prepare 100 positions that appeared in the games played actually by grandmasters in the opening phase or in the Shogi books mentioning on the opening strategy. These positions can be regarded as even. From these positions, each program plays as a Black and White under the condition that the programs can search 10 or 25 seconds per move. The cases where the same position appears four times in a game or a game continuing over 300 moves are treated as a draw. We count draw games as 0.5 win and 0.5 lose. Table 5.2 shows the results of this experiment.

Table 5.2: The results of self-playing games

	Tac-1	-	Tac-2
10 sec / move	83	-	117
25 sec / move	86.5	-	113.5

Considering error testing, in 200 games, over 112 wins indicate that a winner is stronger than a loser with over 95% probability. In this experiment, *Tac-2* wins over 112 games against *Tac-1* in both conditions. Therefore, we see that *Tac-2* becomes stronger than *Tac-1* with over 95% probability. Thus our method can improve the strength of TACOS. The other point shown in this experiment is that *Tac-2* wins more times under the condition of 10 seconds per move than 20 seconds. We see that our method works more effectively in short search time.

5.1.5 Implementation to think on opponent's time

Using our proposed method, we implement the routine for thinking on the opponent's time (so-called predictive search). As mentioned in section 2.3.4, Tacos performs predictive search for the opponent moves. In case where a transposition table is used without clearing, the transposition table contains lots of positions that are searched in the sub-tree from a brother position. In this situation, when lots of positions contained in the sub-tree from a brother position and in the sub-tree from the current position overlap each other (see Figure 5.4), we can expect to search effectively. On the other hand, when few positions overlap each other (see Figure 5.5), we cannot expect to search effectively. To search effectively in any case, we apply alternative use of two different transposition tables to the predictive search. In our implement, one transposition table is used for holding positions searched in a sub-tree from the parent position while one transposition table is used for storing positions searched in a sub-tree from the current position. In most cases, positions in a sub-tree from the parent position and sub-tree from a child position can be expected to overlap in some extent. Therefore, we can expect to search effectively in the case shown in Figure 5.5.

We implement alternative use of two transposition tables for the predictive search performed in Tacos. Although leaving the positions visited during a search from a brother position may cause a problem concerning the effectiveness in some case, that problem would disappear if most positions in a sub-tree from the parent position and sub-tree from a child position overlap in many cases. If so, we can expect sufficiently effective to search with holding positions searched in a sub-tree from the parent position. Here we present a new idea with its implementation but leave the verification as a future work.

5.1.6 Conclusions and future works

In this section, we proposed a method that uses two different transposition tables alternatively. This method has been implemented in Tacos and improves playing strength of Tacos. Under the condition of short search time, this method particularly shows better performance using information stored in the previous search.

The weak point of this method is that this requires memory twice as much as general implements. On the other hand, this method can be implemented in a simple way. Therefore, in case where a program can allocate enough memory for transposition table.

The results of the self-playing experiment show that our method improves search efficiency, but also causes the decrease of search speed. The speed down is caused by checking the previous transposition table every time the target position could not be found in the current transposition tables. The positions that appear in the deep part of the current search do not seem to exist in the previous transposition tables. Therefore, a computer does not check the previous transposition tables when it searches the deep part of game tree may settle this problem. This enhancement would be a future work.

5.2 Opportune time recognition of attacks

After constructing a formation, both players usually start the first attacks that often follow some material loss. Although this loss can be recovered afterwards, a computer often fails to search by enough depth. This means that a computer often overlooks the moves for starting the first attacks because those moves seem bad in the short-term view. Therefore, a computer is poor at starting the first attacks. In this section, we try to improve this weak point while recognizing an opportune time for attacks. At present, we focus on the edge attacks.

5.2.1 Introduction

Nowadays, computer Shogi becomes as strong as armature 5-dan level. In some area, solving checkmate problems for example, computers perform much better than human masters. On the other hands, there are several seriously weak points. Of which, one is the poor strategy for the edge attacks.

The edge attack often appears in many strategies. These attacks aim at the 1st file or 9th file that locates the edge of the board and have various tactical purposes. Figure 5.6 is an example of the edge attack [80]. From this position, the sequence of moves: **P-9e(9f)**, **Px9e(9d)**, **P-9b**, **Lx9b(9a)**, **P-9c**, **Lx9c(9b)**, **P-9d** and **Lx9d(9c)** might be played. After the move sequence, Black tries to capture White's Lance on **9d** by dropping Knight on **8f**.

In many cases including the previous example, the edge attack starts with sacrificing some pieces, mainly pawns. The attack will stop with getting some compensation after some moves. This compensation means that the attacker tries to make the opponent's King dangerous or capture the opponent's valuable pieces. Although the human players who are as strong as computers can play the edge attack properly, computers are poor at playing the edge attack for some reasons. The first, the edge attack starts with sacrificing moves that often become the target of forward pruning. The second, a computer cannot understand that they can get some compensation for sacrificing their pieces because it is hard for a computer to search by sufficient depth. With this weak point, a computer misses good chance for starting the edge attack and could be cornered by overlooking the opponent's edge attack. Therefore, we have to improve this weak point to make our program much stronger.

A professional player describes the edge attack in his book [80] as follows: "How can we perform the edge attack in practice? Unexpectedly, the way for the edge attack is simple. The move orders we use are very limited and formations for attacking do not have much difference. The only problem is when one would start the edge attack." If there are limited move orders, it is not difficult to search those move orders deeply. On the other hand, a computer cannot search other moves deeply. Thus we need the framework that concentrates search costs only when a position is adequate for starting the edge attack.

In this section, we focus on the following two points:

1. Ascertaining the possibility of the edge attack against a target position.

2. Search the edge attack moves deeply when a position is regarded as an opportune position for the edge attack.

Analyzing thousands of master games, we try to understand the condition on when one could start the edge attack and find some factors. Using those factors, we implement the routine that enables Tacos to search the edge attack moves deeply only at opportune time. Using this routine, Tacos could perform the edge attack in more positions than before.

In section 5.2.2, we outline some related works. In section 5.2.3, we present an analysis of master games and some factors of the edge attack. In section 5.2.4, we implement a routine for the edge attack on Tacos and perform some experiments. In section 5.2.5 we give some conclusions and future works.

5.2.2 Related works

There are some related works that conjecture whether a position is in checkmate from some information of piece location, pieces in hand and so on. There are also some related works that aim at developing a computer that can lookahead deeply among tactical moves. Here we give a short sketch of such related works.

Researched for estimating positions and moves

Using a perceptron with 161 inputs that consider pieces and effects around King, Grimbergen estimates King danger [86]. The perceptron is trained 500 safe middle game positions and 500 dangerous positions that are for solving checkmate problems. Under the condition that uses enough training data, the perceptron estimates King danger with over 90% possibility.

Using Support Vector Machine (SVM) with 129 inputs that consider pieces and effects around King, Miwa estimates whether a King is in checkmate or not for a position [87]. The SVM is trained with 40,000 positions. Those positions are made as follows:

1. Prepare 4,000 professional game records.
2. For each game score, extract last 10 positions.
3. For each 40,000 position, perform search.
4. Pick out a position that appears during search at random.

After training, the SVM estimate mating positions with about 86% probabilities. Using this estimator, their Shogi program Gekisashi reduced the search time by about 62% without losing the accuracy of search. Miwa also estimates mating positions with the perceptron that uses input factors generated automatically [89].

Using Maximum Entropy Method, Tsuruoka estimates which moves are likely to be played [88]. The inputs contain the square of the moved piece, kinds of the moved piece, the condition around squares where the piece moves from and to, and so on. The estimator is trained with Ohyama's, the 15th Lifetime Meijin, and 600 game scores. After training, this estimator estimates Ohyama's moves with 35% probabilities with a candidate and 70% probabilities with five candidates in positions unknown to that program.

Researches for searching tactical moves

Tanase proposed the notion of so-called macro moves [6] that treat three orders of moves as if it were one move to search tactical moves deeply in his program Is Shogi [42] . In that method, tactical moves are given by hand. In Shogi, there are hundred of such tactical moves; therefore it is impossible to implement all tactical moves by hand. On the other hands, Ohtsuki extracts force moves from master games by using n-gram method [90]. This method can collect many tactical moves automatically. Although a computer can use many tactical moves with this method, it would supply lots of useless moves for a target position. To supply only proper tactical moves, we consider that extraction has to perform while considering a placement of pieces and progress of a game. In this topic, there are some works that extend Otsuki's work [74, 91].

5.2.3 Game score analysis

As stated above, good timing is very important for the edge attack. Thus we try to estimate a position whether an edge attack move is effective for a player.

The edge attack appears in many different ways in the actual master games. Therefore, each edge attack may have a different aim and move order. Since it is hard to estimate all kind of edge attacks, we treat only with one strategy in this study. We expect to find characteristic features by this limited study. The strategy we focused on is the edge attack by the side of Static Rook against Ranging Rook. We exclude the strategies that one side or both side of players make Bear-in-the-Hole castle.

At first, we explain about positions we analyzed. Second, we explain about features that we use for estimation. Then we analyze many master games and show the features that have effects on the edge attack.

Target positions for analysis

At first, we prepare many positions for analysis. We collect about 17,000 professional game scores and about 84,000 game scores played by amateur strong players who have over 2,000 Elo-rating points in Shogi Club 24 [78,79,82], the greatest Internet Shogi server. The positions that appear in those games are selected by the following rules. Regarding coordinates, we look them from the viewpoint of the Static Rook side.

- Exclude the positions where the Ranging Rook side is to move.
- Exclude the positions where the King of Static Rook side does not stand on the square **7h**, **7i**, **8g**, **8h** or **8i**. Exclude also the Bear-in-the-Hole line for the Static Rook side.
- Exclude the positions where the King of Ranging Rook side does not locate on **8b**. Exclude also the Bear-in-the-Hole lines for the Ranging Rook side.
- Pawn of the Ranging Rook side must exist on **9d** and Pawn of the Static Rook side must exist on **9f** and the square **9e** must be empty.
- The positions that regarded as opening or middle game stage are excluded.
- The positions that regarded as book positions are excluded.

- The positions in which there exits some forced moves are excluded.
- The player who is to move must have more than one Pawn in hand.

Using the condition about forced moves, we exclude checked or attacked positions where a player has to defend. We judge this condition using Tacos; if Tacos scores an edge attack move less than 800 points, as valuable as getting Lance without any compensation, lower than a move played by human after shallow search, we regard that position has forced moves. The condition excludes the opening and middle game positions also used in Tacos. Tacos estimates the progress of games by a factor named *Shinkodo*. We exclude positions where Tacos sets *Shinkodo* less than 50 points, players start fighting around there in many games. We also exclude positions registered in the opening book for Tacos. We add last condition about Pawn in hand because most edge attacks start with sacrificing some Pawns. Using these conditions, we extract about 192,000 positions containing about 4,000 positions where human players start the edge attack.

Features for estimation

Below we list the features used for estimating positions where a computer has to start the edge attack.

- The condition of a square. We treat Gold and promoted minor pieces as a same condition while Black's pieces and White's same kind of pieces as a different condition. A square can be 21 conditions of pieces.
- How many pieces effect to a square. For both side, we count effects as 0, 1, 2, 3, more than 4. A square can be 25 conditions of effect.
- Pieces in hand. For both side, we count Pawns in hand as 0, 1, 2, 3, 4, more than 5. In the same way, we count other minor pieces as 0, 1, more than 2 and Bishops as 0, 1, 2.
- The condition of Rooks. Since Rooks can move horizontally, the file of square where a Rook is sometimes does not have any importance. Thus we consider only horizontal location instead of coordinates. A Rook has following eight condition; (1) be on 1st lank, (2) be on 2nd rank, (3) be on 3rd rank, (4) be between 4th rank and 9th rank, (5) be in hand, (6) be between 7th rank and 9th rank, (7)be between 1st rank and 6th rank, and (8) be in hand. The conditions (1) to (5) are for Static Rook side and (6) to (8) are for Ranging Rook side. This condition can be 36 conditions.

In this study, we estimate the edge attack for the Static Rook player against the Ranging Rook player; therefore we pay attention to the area where Kings of both sides make castling formation. To put it precisely, we consider the condition of a square to 27 squares, **5a** to **9c** and **7d** to **9g**. On the other hands, we count the effects to 24 squares, **5a** to **9c** and **7d** to **9f**.

Analysis of features

Then we analyze target positions. We count how many times a feature appears in positive positions where human players start the edge attack and in negative positions where human players do not start the edge attack. We suspect that the features that seldom appear do not represent accurate frequency. Therefore, we deal with the features that appear more than 50 times in this study. We also exclude some features that appear in all the target positions. Pawn belonging to the Ranging Rook side stands in **9d** for example.

Analysis with one feature

At first, we count frequency for each feature. The number of features that appear more than 50 times is 622. To these features, we calculate the edge attack probability (EP) as follows:

$$EP = \frac{fp}{fa}$$

fp means how many positive positions the feature appears and fa means how many positions the feature appears. Table 5.3 shows some features that have high or low edge attack probabilities. Table 5.4 shows the distribution of the edge attack probabilities. The results of analysis clearly show that the edge attack probabilities of many features are as same as 2.07%, the edge attack probability for all the target positions. Since these edge attack probabilities are very low, we consider that it is hard to estimate with these features.

Analysis with combinations of some features

We cannot find out features that have enough high probability of edge attacks. Then we consider new features that are made by combining some features. An edge attack probability for these new features is calculated as same way. For example, the edge attack probability EP_c for the new feature F_c made by combining F_a and F_b is calculated as follows:

$$EP_c = \frac{fp_c}{fp_{ab}}$$

fp_c means how many positive positions where both F_a and F_b appear and fp_{ab} means how many positions where both F_a and F_b appear. We change the number of combined features and perform estimations. The results of estimations are given in Table 5.5. The number of valid features means that how many features have higher probability than threshold. The number of valid positions means that how many positions where more than one valid feature appears. The number of invalid positions means that how many positions where any valid feature does not appear.

By combining features, we can find out some features that have enough high edge attack probabilities. On the other hands, under the some condition of number of combining or threshold, we cannot estimate some positive positions because any valid features do not appear in those positions. Since it is hard to find out proper set of features that we can estimate with in this way, using the set that we find in this subsection we try to implement a routine for edge attack in next subsection.

Table 5.3: THe edge attack probability for each features

	(1)	(2)	(3)		(1)	(2)	(3)
6b Black Lance	79	14	17.72%	**8d** White +Bishop	510	0	0.00%
9d Effects (2, 1)	106	13	12.26%	**7f** White +Bishop	366	0	0.00%
9c White Knight	149	17	11.41%	**9e** Effects (2, 4)	279	0	0.00%
6a Effects (2, 3)	161	18	11.18%	**8e** White Rook	206	0	0.00%
9b White Silver	150	15	10.00%	**7g** White Pawn	203	0	0.00%

(1) The times each feature appears in all positions.
(2) The times each feature appears in positive positions.
(3) Edge attack probabilities.
Black means the player in Static Rook side.
White means the player in Ranging Rook side.
Effects (x, y) means that there are x effects of Static Rook side and y effects of Ranging Rook side in the square.
+Bishop means promoted Bishop.

Table 5.4: Distribution of edge attack probabilities for each features

$EP = 0\%$	32
$0\% < EP < 1\%$	69
$1\% \leq EP < 2\%$	196
$2\% \leq EP < 3\%$	188
$3\% \leq EP < 5\%$	95
$5\% \leq EP < 10\%$	36
$10\% \leq EP$	6

EP : edge attack probability

Table 5.5: The results of analysis for combinating features

(1)	(2)	(3)	(4)	(5)
2	5%	3528	3607	370
	10%	407	1179	2798
3	10%	15333	2813	1164
	15%	1562	1315	2662
	20%	299	341	3636
4	10%	315761	3774	203
	15%	38691	2697	1280
	20%	3851	1292	2685
	25%	383	423	3554
5	15%	614806	3628	349
	20%	75505	2370	1607
	25%	6963	1000	2977
	30%	267	334	3643

(1) The number of combination
(2) Threshold probability
(3) The number of features
(4) The number of positions where more than one features appears
(5) The number of positions where no features appear

5.2.4 Implementation to the edge attack

In this subsection, we implement a routine for edge attack to Tacos using the features that are got in previous subsection. Before we implement, we have to consider about the edge attack probabilities of those features.

Although about 20% probabilities do not seem enough high, we think we can use those features because human players consider edge attacks not only in the position they actually stack but also in some position they do not attack. In other words, human players sometimes give up edge attacks although they spend times on thinking edge attacks. Thus it is enough for a program to estimate whether it is valuable to search edge attack in a target position. The features we got will make Tacos searching in five times as many as positions; we are not sure that it is best, but we think it is not unsuitable.

Implementation

Below we outline the edge attack routine that we implement to Tacos.

- Before searching, the root position is estimated whether attacking edge is effective or not. If edge attack is effective in that position, *Position Flag* is set as *on*. If edge attack is ineffective, *Position Flag* is set as *off* and perform searching normally.
- Another flag, *Attacking Flag* is prepared. Attacking Flag is also set as *off* before searching.
- In every position visited during searching, *Attacking Flag* can become *on* if following three conditions happens at the same time. (1) *Position Flag* is *on*. (2)

Tacos searches enough depth from the position. (3) Tacos plays the moves that Tacos starts edge attack with in the position. After finishing searching the edge attack move, *Attacking Flag* become *off*.

- In every position visited during searching, search is extended with the moves that are played on 8th or 9th file and affect edge if *Attacking Flag* is *on*.

In this study, we estimate whether the edge attack is effective or not only in the root position of search. This is because that we cannot estimate in small costs by the following two reasons; (1) we have to combine some features for high edge attack probabilities, and (2) we have to check many features to estimate as many positions as possible. The target of search extension is given as follows:

- Gold, Silver, Knight, Lance, Pawn and those promoted pieces are moved or dropped on **9a**, **9b**, **9c** or **9d**.
- Gold, Silver, Knight, Lance, Pawn and those promoted pieces are moved or dropped on **8a**, **8b**, **8c**, **8d**, **8e** or **8f**.

Tacos uses realization-probability search; therefore the search extensions are performed by increasing translate probability of moves. Tacos orders moves by controlling move generation. The moves that are regards as important moves are generated beforehand, and moves that are regards as stupid moves are generated later. In this time, Tacos generates the moves that start edge attacks after searching important moves such as capturing valuable opponent pieces. To do this, in positions where there are some important moves exclude edge attacks, Tacos can avoid searching edge attacks deeply.

Evaluation

Then we perform some experiments for evaluating edge attack routine. The routine we implement works only in some strategies; therefore the result of self-playing experiments or games against other programs may not show the effect clearly. Thus we try to evaluate under the form of next-move problem. We check that how many positions Tacos can start edge attacks in the positions where human players start edge attacks.

We prepare following five programs.

- Type-A: without edge attack routine.
- Type-B: with edge attack routine. Position estimator uses the set of 4-15 in Table 5.5.
- Type-C: with edge attack routine. Position estimator uses the set of 5-20 in Table 5.5.
- Type-D: with edge attack routine. Position estimator uses the set of 5-25 in Table 5.5.
- Type-E: with edge attack routine. Position estimator regards all positions where edge attack is effective.

Table 5.6: The results of searching experiment

	Type-A	Type-B	Type-C	Type-D	Type-E
(1)	4	15	15	15	15
(2)	115.2	115.5	114.4	115.3	114.5
(3)	109.0	109.4	108.7	109.3	108.7

(1) Number of positions where TACOS can start edge attack.
(2) Search speed in positions where human players start edge attacks.
(3) Search speed in positions where edge attacks are not started.
all the number of search speed means thousands node per a second.

Table 5.7: The results of self-playing experiments with edge attack routine

	(1)	(2)	(3)
Type-A - Type-B	51 - 49	8 - 8	50 - 50
Type-A - Type-C	49 - 51	7 - 12	49 - 51

(1) Games started from edge attack positions.
(2) How many times the programs started edge attack.
(3) Games started from non-edge attack positions.

We prepare 200 positions where human players start edge attacks from 4,000 positions we prepared in 5.2.3. For comparing search speed, we also prepare 200 positions where human players do not consider edge attacks from game records played in Shogi Club 24 and book of next-move problem [81].

There are no differences except the routine of edge attack among those five programs. Each programs searches 20 seconds per a position. We use the computer that has Ahtlon64 3200+ CPU for these experiments. Table 5.6 shows the result of these experiments. Figure 5.7 shows the position where TACOS become to be able to start edge attacks with edge attack routine. In the position shown in Figure 5.7, players as strong as amateur higher grades player may regard edge attacks as a quite good move; therefore the routine of edge attack we implemented has good influence on TACOS.

The programs with the edge attack routine tend to perform the edge attack in general. Although we are worried about the decline of search speed, the experimental results do not indicate it. On the other hand, we have to consider the influence to the quality of search. The quality can be declined in the case where the search concerning the edge attack takes too much time even in the positions where it is not so effective.

We perform additional experiments. Type-B and Type-C play some self-playing games against Type-A in the experiments. We prepare 100 initial positions, where 50 positions in which human players performed the edge attack and other 50 positions in which they did not perform it. These 100 positions are taken from the positions used in the previous experiment. From each position, a program plays once as Black and White respectively. Each program searches 10 seconds per a position. Table 5.7 shows the results of the experiments. From the results we observe that the inclusion of the edge attack routine does not change the playing strength of the computer. On the other hand, Type-C tends to initiate the edge attack as desired to play master-like tactics.

5.2.5 Summary and future works

In this subsection, we estimate the possibility of edge attacks and implement edge attack routine to Tacos using that estimation. From thousands of game records played by human experts, we find out some features that estimates the possibility of edge attacks. Using those features, we make Tacos searching edge attack moves deeply. The result of performance test shows that the routine of edge attacks enable to Tacos to start edge attacks in more positions.

The set of features we used in this time contains over 10 thousand conditions; therefore the cost to estimate a position is not small. In addition to this, that set of features cannot estimate properly in some positions. Thus refining features to decrease number of conditions and to estimate in more positions is one of future works.

In this time, we deal with the only some strategies. There are many strategies that we did not deal with and where other types of edge attacks appear. To estimate those edge attacks is another future works.

In this time, we pay attention to estimation mainly and do not take notice of move orders of edge attacks. There are some characteristic set of moves in edge attacks; therefore using those set of moves will enable Tacos to search more effective. Improving evaluation function will also enable Tacos to search more effective. These are also future works.

5.3 Conclusions

In this chapter, we presented two enhancements to improve the play level of computer Shogi in the middle game and endgame.

In section 5.1, we proposed alternative use of two transposition tables. With this method, a computer can use information that are stored in the previous search and wasted up to the present. We implemented this method in Tacos and performed some experiments. The results of these experiments showed that our proposed method improved the playing strength of Tacos especially in short-time games.

In section 5.2, we dealt with the problem concerning the start of attacking. In Shogi, the order of moves for starting to attack contains continuation of sacrifices. In addition, it takes long plies to take those sacrifices back. Since computers tend to stop searching disadvantageous lines, they cannot find positions where they can take those sacrifices back. Therefore, starting to attack is to be a problem. We focused on the edge attack with respect to this problem. Using some factors that are found during analyzing many master games, we implemented an estimator that judges whether a position can be started the edge attack. Tacos searches deeply some moves to initiate the edge attack when the estimator judges an opportune timing for edge attack. With this implement, Tacos can start edge attacks in more positions than before.

```
TranspositionTable tTable[2]; // Transposition table [0]: tA, [1]:tB
TranspositionTable *pPrevious; // pP
TranspositionTable *pCurrent; // pC

void initGame() // initialize before new game
{
    tTable[0].crear();
    tTable[1].crear();
    pPrevious = &( tTable [0]);
    pCurrent = &( tTable [1]);
}

void initSearch() // initialize before search
{
    pPrevious →crear();
    TranspositionTable *temp = pPrevious;    // swap
    pPrevious = pCurrent;
    pCurrent = temp;
}

int search(Board *pBoard)
{
    TableElem *pElem; // TableElem: element of transposition table
    pElem = find(pCurrent, pBoard);    // first, check current table.
                                       // if failed NULL is set in pElem
    if(pElem == NULL) // target position could not be found in current table
    {
        pElem = find(pPrevious, pBoard); // check previous table
    }
    // if the target position is found
    if(pElem ≠ NULL)
    ...

    // call evaluation in leaf nodes
    // in other nodes, call search() and find best move in this node
    ...

    // after finding the best move, this position is set in current table
    update((pCurrent, pElem); // (the result of this node is in pElem)
```

Figure 5.3: Pseudo-code of the proposed method

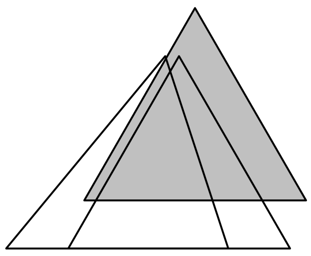


Figure 5.4: Trees with lots of common nodes

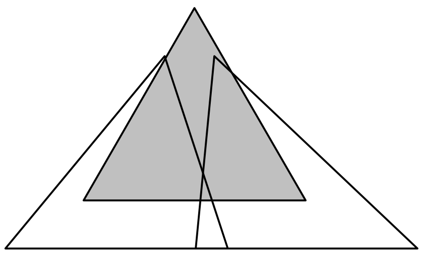


Figure 5.5: Trees with few common nodes

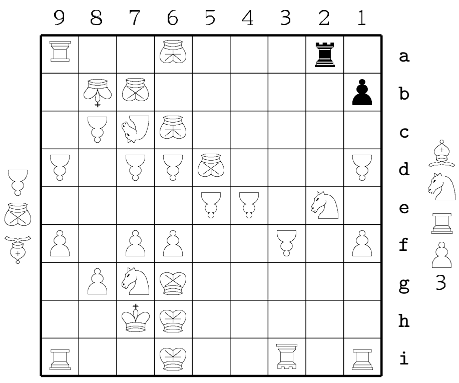


Figure 5.6: An example of the staring position of an edge attack

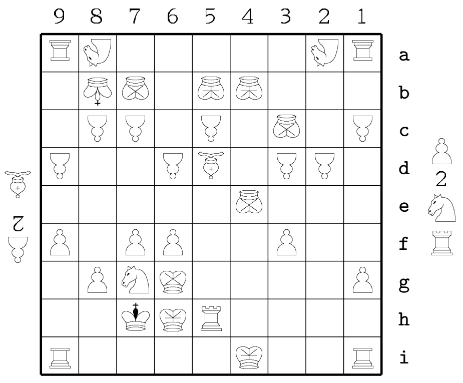


Figure 5.7: An example of a position where TACOS become to be able to start edge attacks

Chapter 6

Conclusions and future works

In this thesis, we have presented our research that aims at defeating the world human Shogi champion by creating strong computer Shogi. Although top computer Shogi programs are as strong as amateur 5th-dan players at present, the playing strength of them in the opening game has been regarded only as 1st-kyu. Therefore, improving the opening play of computer Shogi was strongly required to achieve that great goal.

In games such as Chess, Othello and Shogi, there are also standard sequences of moves in the opening game and most players play the opening game with considering those sequences of moves. Although out-of-book moves are seldom played in Chess and Othello, those moves can sometimes be played in Shogi because the opening book is not exhaustive.

In this thesis, we have descried some techniques that improve the opening play of computer Shogi. To improve the opening play of computer Shogi, we dealt with the following subjects: using an opening book effectively in in-book positions, and playing a stable opening game in out-of-book positions. In those subjects, we have pointed out some problems to improve and proposed its measures. Implementing those measures and other enhancements, the playing strength of Tacos was improved awfully. These improvements lead some fruits in several tournaments such as 1st prize on the 10th Computer Olympiad and 4th place on the 16th World Computer Shogi Championship. In addition, Tacos could drive a professional Shogi player into a corner on an open game played on September 2005. (The results of those tournaments and the open game are shown in appendix A minutely.)

In section 6.1, we summarize the enhancements in in-book positions while enhancements in out-of-book positions in section 6.2. In section 6.3, we summarize other enhancements described in chapter 5. Finally we mention about future works of Tacos Project in section 6.4.

6.1 Improving Opening Play in In-book Positions

In chapter 3, we presented a measure with an opening book to improve the opening play of computer Shogi. To arrive at the master level, we aim to tune the book that a computer may not select its unapt lines from the book. Considering the characteristics of Shogi, we at first design a large opening book for computer Shogi from thousands of game scores played by masters. Then we consider about the way of tuning the book. Since computer Shogi programs are not as strong as master players, we propose the tuning method with result-driven learning through thousands of self-playing games. We apply this method to Tacos and perform experiments. The results show that our method successfully tunes the opening book for the target program better than other tuning methods that may be commonly used in computer Shogi.

As the future works, we list the following topics: applying search-driven learning [76]; and applying the extended book [22] to computer Shogi. Although we consider that computer Shogi is not enough strong to apply the extended book currently, we can apply this idea by emphasizing book moves in an early stage and reducing a degree of referring gradually.

6.2 Improving Opening Play in Out-of-book Positions

In chapter 4, we have presented some measures to improve the opening play of computer Shogi in out-of-book positions. The problems caused in the opening game are that a computer cannot grasp the situation by evaluating some simple features such as material and King safety, because those features are usually even. To tackle these problems, we supplemented search with an evaluation function by evaluating formations that are constructed in the opening game. We then proposed an extension of otoshi-ana method that has been commonly used in the domain of computer Shogi and pointed out some problems on implementing otoshi-ana method and countermeasures for them. We also implemented combination evaluation function for a proper evaluation of combination of pieces that otoshi-ana method cannot evaluate. In addition, we implemented partial position opening book to refer the book moves stored for the in-book positions that look like the current position. Using this sub opening book enables Tacos to search promising candidates quickly. After implementing those enhancements, Tacos can play the opening game more stable than before. Such stable opening performance plays one of the important roles of activities in various tournaments and also putting up a good fight in the open game against Hashimoto 5-dan at that time, a strong professional Shogi player.

In this thesis, we sometimes used domain knowledge in the topic concerned. Up to the present, some researches that try to acquire such knowledge automatically from game scores have been proposed in Shogi [94–100], but they do not seem effective awfully. However, Bonanza, the champion of the 16th WCSC, succeeded in building a good evaluation function automatically from thousands of game scores [47,48]. As future works in this direction, we also aim to acquire a good evaluation function automatically.

6.3 Various Enhancements

Since there are also various problems, we need more improvements in the opening as well as in other stages towards the real master-level or more. Thus we presented two enhancements we applied in Tacos in chapter 5.

As the first enhancement, we focused on the use of transposition tables. In the domain of computer Shogi the previously-stored transposition table is often cleared from the memory when starting game-tree search in the current position. This means that computers have to waste some valuable information obtained in the previous search. Therefore, we proposed a method that uses two different transposition tables alternatively to use such valuable information. We have implemented this method in TACOS and performed some experiments. The results show that this method improves the playing strength of TACOS especially under the condition of quick play (very short thinking time).

As the second enhancement, we tackled the problem on the poorness of starting the first attacks. Although material loss for starting attacks can be recovered afterwards, a computer often fails to search by enough depth because those moves seem bad in the short-term view of computers. Dealing with the edge attacks, a kind of the first attacks, we tried to estimate the possibility of edge attacks and implemented an edge attack routine in TACOS. From thousands of master games, we found out some features that can estimate the possibility of the edge attack and made an edge attack routine that enables TACOS to search edge attack moves deeply. The experimental results show that TACOS is now able to start the edge attack in more positions with that routine.

6.4 Future Works of TACOSProject

At the end of this thesis, here we enumerate the current and future works of TACOS project.

- The use of opening book. We already mentioned about this in section 6.1.
- Learning an evaluation function automatically and properly. We also already mentioned about this in section 6.2.
- Increasing the efficiency of selective search. Although BONANZA succeed brute-force-based search, we consider that it is important to search deeply in promising nodes while shallowly in non-promising nodes. We expect that enhancing the methods proposed in section 5.2 or on-going work [74] might enable TACOS to search more effectively and properly.
- Improvement for position evaluation in the case where King is trying to enter into the opponent's area. Since King entering into the opponent's area is one of the weak points of computer Shogi, this improvement enable a computer to play more correctly in such positions.
- Making Shogi processors and parallelizing computer for increasing search speed explosively. Although a processor was proposed in [101], this attempt has not been adopted in any strong computer Shogi yet.

Appendix A

Tournament History of TACOS

Here we show the tournament history of TACOS after the author joined the TACOS Project. In Table A.1 we show the results of some computer Shogi tournaments. In May 2006, TACOS arrived at amateur 5th dan level (2,400 rating point) on the Internet while professional players have over 2,800 rating point. Some of game records played in the major tournaments and on the Internet are listed in Appendix B.

Table A.1: The tournament history of TACOS

Tournanment		Month / Year		Ranking	Rating
6th	CO	August	2001	3	800
12th	WCSC	May	2002	30	1600
7th	CO	July	2002	4	
13th	WCSC	May	2003	10	1800
8th	CO	November	2003	3	
14th	WCSC	May	2004	7	2000
15th	WCSC	May	2005	6	2200
10th	CO	September	2005	1	
3rd	CSGP	October	2005	7	
16th	WCSC	May	2006	4	2400
11th	CO	May	2006	3	

WCSC	:	World Computer Shogi Championship
CO	:	Computer Olympiad
CSGP	:	Computer Shogi Grand Prix

A.1 World Computer Shogi Championship

The World Computer Shogi Championship (WCSC) is organized by Computer Shogi Association and is held every year. The participants can use any machines. Therefore, participants tried to use the most powerful machines at that time.

In every year, the lower division contest took place at first. Then top eight programs go to the upper division contest with 16 seeded programs. The top five programs can get through to the final standings with 3 seeded programs.

The 12th World Computer Shogi Championship

TACOS took part of the lower division contest and got 4 points of 7 games (4x win, 3x loss) - finished on 11th place of 32 participants of this contest. In that time, TACOS could not pass to the upper division contest.

The 13th World Computer Shogi Championship

TACOS took part of the lower division contest and got 6 points of 7 games (6x win, 1x loss) - finished on 2nd place of 26 participants of this contest. Then TACOS took part of the upper division contest and got 5 points of 9 games (5x win, 4x loss) - finished on 7th place of 24 participants of this contest and got a seed for the upper division contest of next year.

The 14th World Computer Shogi Championship

TACOS took part of the upper division contest and got 7 points of 9 games (7x win, 2x loss) - finished on 2nd place of 24 participants of this contest. Then TACOS took part to the final and got 2 points of 7 games (2x win, 5x loss) - finished on 7th place.

Table A.2: The results of 14th World Computer Shogi Championship

	program name	1	2	3	4	5	6	7	8	win	loss	draw	ranking
1	IS SHOGI		o	x	x	o	o	x	o	4	3	0	3
2	YSS	x		o	o	o	o	o	o	6	1	0	1
3	GEKISASHI	o	x		o	o	x	o	o	5	2	0	2
4	KCC	o	x	x		o	o	x	o	4	3	0	4
5	TACOS	x	x	x	x		o	o	x	2	5	0	7
6	EISEI MEIJIN	x	x	o	x	x		o	o	3	4	0	6
7	KAKINOKI SHOGI	o	x	x	o	x	x		o	3	4	0	5
8	KANAZAWA SHOGI	x	x	x	x	o	x	x		1	6	0	8

The 15th World Computer Shogi Championship

TACOS took part of the upper division contest and got 7 points of 9 games (7x win, 2x loss) - finished on 1st place of 24 participants of this contest. Then TACOS took part to the final and got 2 points of 7 games (2x win, 5x loss) - finished on 6th place.

The 16th World Computer Shogi Championship

TACOS took part of the upper division contest and got 7 points of 9 games (7x win, 2x loss) - finished on 3rd place of 24 participants of this contest. Then TACOS took part to the final and got 4 points of 7 games (4x win, 3x loss) - finished on 4th place.

Table A.3: The results of 15th World Computer Shogi Championship

	program name	1	2	3	4	5	6	7	8	win	loss	draw	ranking
1	YSS		x	x	o	x	x	o	o	3	4	0	4
2	GEKISASHI	o		o	o	o	o	o	o	7	0	0	1
3	IS SHOGI	o	x		o	o	x	o	o	5	2	0	3
4	TACOS	x	x	x		o	x	o	x	2	5	0	6
5	GPS SHOGI	o	x	x	x		x	x	x	1	6	0	8
6	KCC	o	x	o	o	o		=	o	5	1	1	2
7	RYUU NO TAMAGO	x	x	x	x	o	=		x	1	5	1	7
8	BINGO SHOGI	x	x	x	o	o	x	o		3	4	0	5

Table A.4: The results of 16th World Computer Shogi Championship

	program name	1	2	3	4	5	6	7	8	win	loss	draw	ranking
1	GEKISASHI		x	x	o	=	o	x	o	3	3	1	5
2	KCC	o		=	o	o	o	x	o	5	1	1	3
3	YSS	o	=		o	o	x	o	o	5	1	1	2
4	OTSUKI SHOGI	x	x	x		x	x	x	x	0	7	0	8
5	KAKINOKI SHOGI	=	x	x	o		x	x	o	2	4	1	6
6	TACOS	x	x	o	o	o		x	o	4	3	0	4
7	BONANZA	o	o	x	o	o	o		o	6	1	0	1
8	RYUU NO TAMAGO	x	x	x	o	x	x	x		1	6	0	7

A.2 Computer Olympiad

In Computer Olympiad, there is a tournament for Shogi. The time to think is longer than other tournaments such as WCSC. Each has 60 minutes in this tournament while 25 minutes for each in WCSC.

The 6th Computer Olympiad

TACOS got 1 point of 4 games (1x win, 3x loss) - finished 3rd place of 3 participants.

Table A.5: The results of 6th Computer Olympiad

program name	score
SHOTEST	8
SPEAR	3
TACOS	1

The 7th Computer Olympiad

Tacos got 4 points of 8 games (4x win, 4x loss) - finished 4th place of 5 participants as the result of the playoff.

Table A.6: The results of 7th Computer Olympiad

program name	score
Is Shogi	6
kanazawa Shogi	6
Shotest	4
Tacos	4
Spear	0

The 8th Computer Olympiad

Tacos got 1 point of 8 games (1x win, 7x loss) - finished 3rd place of 3 participants.

Table A.7: The results of 8th Computer Olympiad

program name	score
Yss	6
Is Shogi	5
Tacos	1

The 10th Computer Olympiad

Tacos got 6 points of 6 games (6x win, 0x loss) - finished 1st place of 4 participants.

Table A.8: The results of 10th Computer Olympiad

program name	score
Tacos	6
Yss	4
Spear	2
Mattari Yuchan	0

The 11th Computer Olympiad

Tacos got 2 points of 4 games (2x win, 2x loss) - finished 3rd place of 3 participants as the result of the playoff.

Table A.9: The results of 11th Computer Olympiad

program name	score
YSS	2
BONANZA	2
TACOS	2

A.3 Computer Shogi Grand Prix

The Computer Shogi Grand Prix (CSGP) is organized by Japan Shogi Association and is held every three year. Top eight programs from the WCSC are invited to participate. In this tournament each program runs on the same machine.

The 3rd Computer Shogi Grand Prix

TACOS got 1 point of 7 games (1x win, 6x loss) - finished 7th place.

Table A.10: The results of 3rd Computer Shogi Grand Prix

	program name	1	2	3	4	5	6	7	8	win	loss	draw	ranking
1	GEKISASHI		x	o	x	o	o	o	x	4	3	0	3
2	KCC	o		o	x	o	o	o	o	6	1	0	2
3	IS SHOGI	x	x		x	o	o	x	o	3	4	0	4
4	YSS	o	o	o		o	o	o	o	7	0	0	1
5	BINGO SHOGI	x	x	x	x		o	o	o	3	4	0	4
6	TACOS	x	x	x	x	x		o	x	1	6	0	7
7	RYUU NO TAMAGO	x	x	o	x	x	x		x	1	6	0	7
8	GPS SHOGI	o	x	x	x	x	o	o		3	4	0	4

A.4 Man-Machine Match on September, 2005

On September, 2005, we had a special opportunity to play against Hashimoto 5th dan, one of the young grandmasters. Although TACOS lost the game in the end, TACOS was in the advantageous position during the game. Here we explain the detail of that game.

TACOS (Black) vs. Hashimoto (White)

Table A.11 shows the game record of this match.

First four moves played by White contain some kind of threaten: he seems trying to confuse TACOS. Since White's first move **P-1d(1c)** (Figure A.1) is unusual move, the position where TACOS played **P-2f(2g)** is out-of-book position. After White plays **P-3d(3c)** (Figure A.2), the position re-enters in in-book position. However, the position soon become out-of-book position by White's move **Bx8h(2b)** (Figure A.3). In this

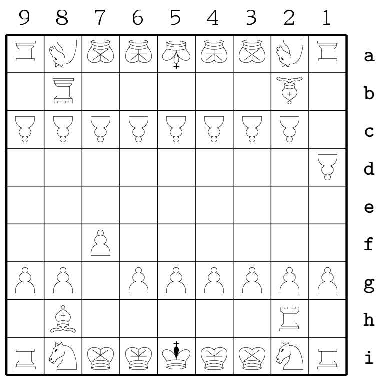


Figure A.1: Position A
White played **P-1d(1c)**

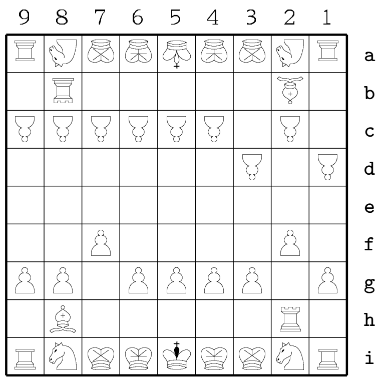


Figure A.2: Position B
White played **P-3d(3c)**

position, the move **Bx8h+(2b)** is natural. Although White tries to confuse Tacos by playing strange moves, Tacos plays stable moves. Therefore White seems giving up threatening. After playing **S-2b(3a)** (Figure A.4), a game goes along book moves till White plays **P-9d(9c)** (Figure A.5).

Black's move of **G-4g(5h)** (Figure A.6)has quite important meaning. The purpose of this move is to defense Knight, and this move is very natural. However, Tacos poor at this move because the location value of Gold is higher in **5i** than in **4g**. Although that sense is not wrong, Gold in **4g** is necessary when Knight in **3g** is attacked. To make Tacos understand this sense, we use combination evaluation. After inputting some combinations, Tacos can play **G-4g(5h)** to defense Knight.

After defending 3rd file, Tacos starts counterattack in 6th file. In the position shown in Figure A.7, Tacos claim exchanging Rooks each other by playing **S-7g(6f)**, but White cannot agree with exchanging Rooks because White's formation is not strong against horizontal attacking. Thus White plays **P-6e** to avoid exchanging Rooks. Then Tacos continues attacking by playing **P-6c** (Figure A.8). Although White can capture this Pawn without any sacrifice, White's formation is muddled. In the position shown in Figure A.9, Tacos's attack is settled temporarily, and Tacos has got some advantage.

In the position shown in Figure A.10, Tacos is attacking White's Rook. Although Rook is valuable piece, it seems useless to drop Silver and Gold for capturing Rook driven into a corner. After playing **G-8d**, White recovers disadvantageous position and is taking advantage gradually. The move **L-1c** (Figure A.11) played by Tacos is bad, and Tacos understand its disadvantage after playing this move. After trying some desperate attacking moves, Tacos resign in the position shown in Figure A.12.

Figure A.4 shows the transition of evaluation value. Although we consider that Tacos spoiled its advantage around position J, Tacos feels good in a while after there. This difference can mean incorrectness of position evaluation. Therefore we have to improve position evaluation more and more. On the other hand, Tacos can play very well in opening game. We consider that this is one of the best cases, and Tacos may play poorly in other cases. To make Tacos play opening game as well as this case in all time, we want to try improving Tacos's opening strategy.

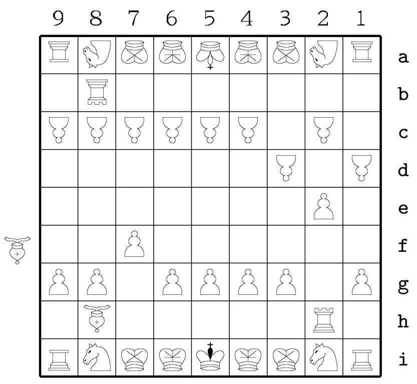


Figure A.3: Position C
White played **Bx8h(2b)**

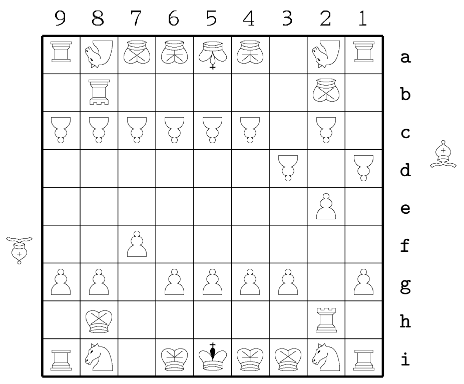


Figure A.4: Position D
White played **S-2b(3a)**

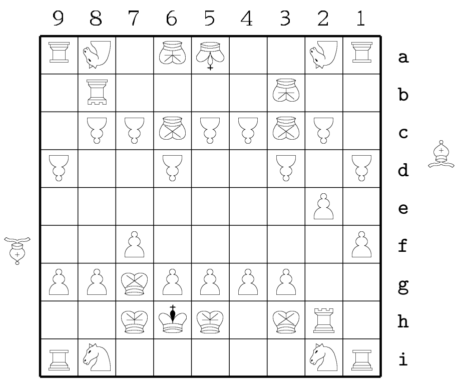


Figure A.5: Position E
White played **P-9d(9c)**

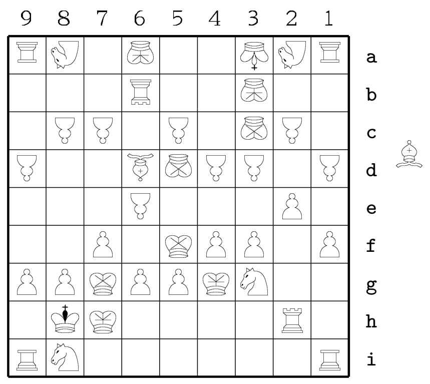


Figure A.6: Position F
Black played **G-4g(5h)**

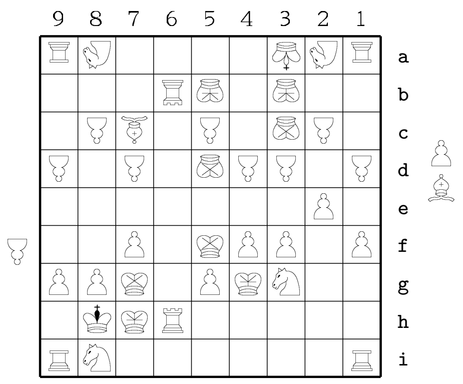


Figure A.7: Position G
Black played **S-7g(6f)**

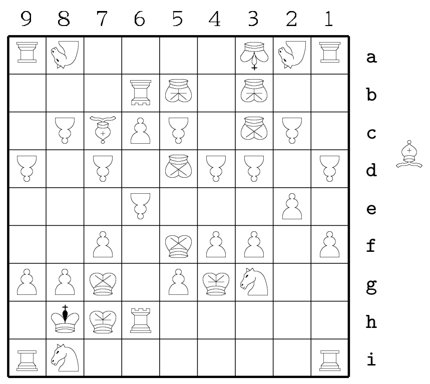


Figure A.8: Position H
Black played **P-6c**

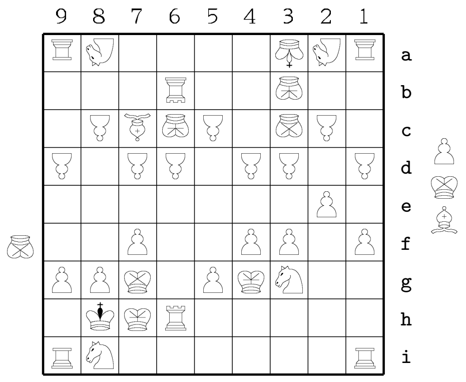


Figure A.9: Position I
Black played **R-6h(6e)**

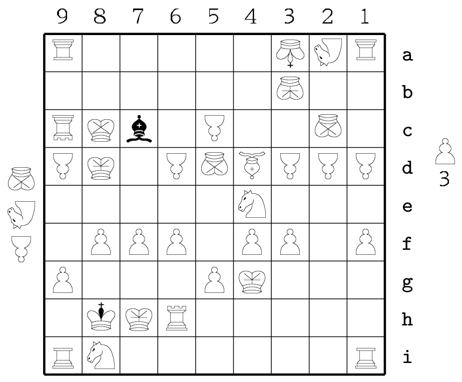


Figure A.10: Position J
Black played **G-8d**

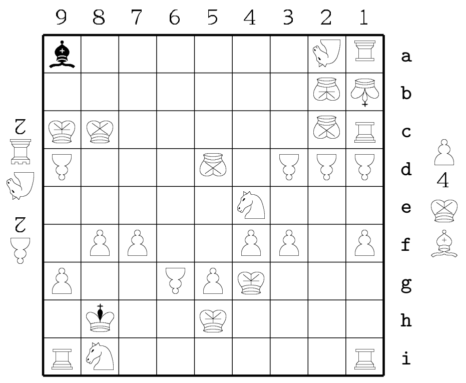


Figure A.11: Position K
Black played **L-1c**

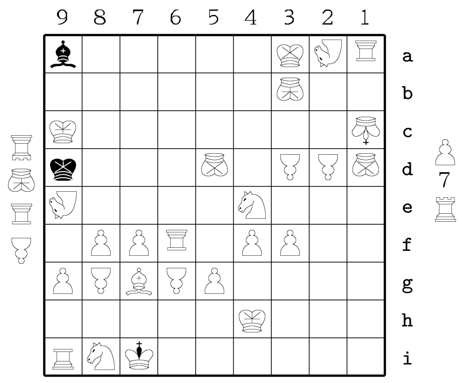


Figure A.12: Position L
Black resigned

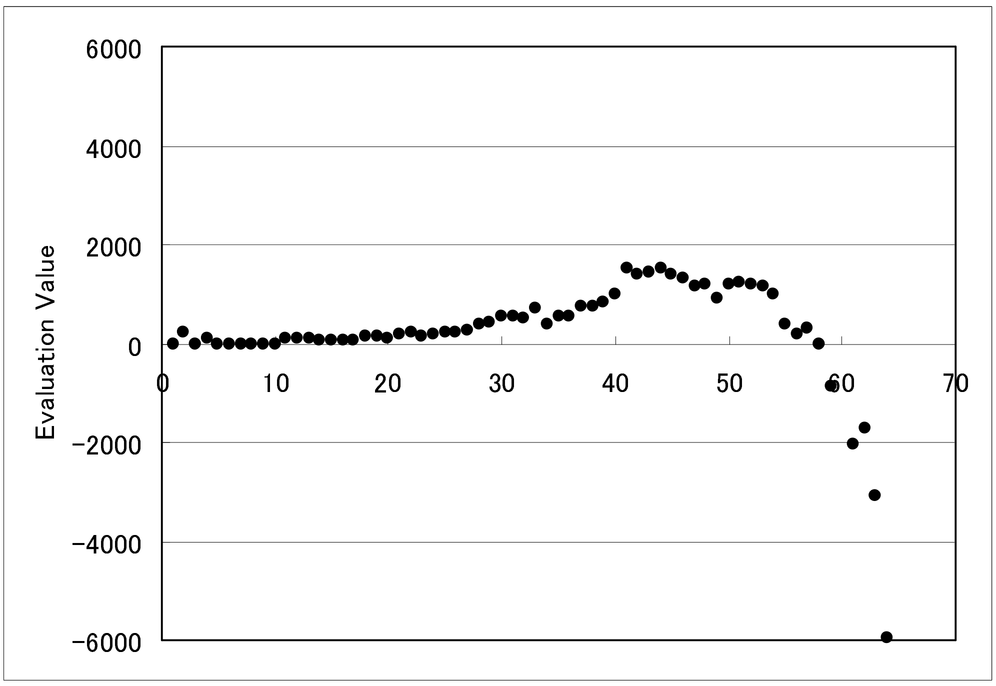


Figure A.13: Transition of evaluation value

Table A.11: Game record of the man-machine match

1.	**P-7f(7g)**	**P-1d(1c)** *A	2.	**P-2f(2g)**	**P-3d(3c)** *B
3.	**P-2e(2f)**	**Bx8h(2b)** *C	4.	**Sx8h(7i)**	**S-2b(3a)** *D
5.	**S-7g(8h)**	**S-3c(2b)**	6.	**P-1f(1g)**	**S-7b(7a)**
7.	**S-3h(3i)**	**P-6d(6c)**	8.	**G-7h(6i)**	**S-6c(7b)**
9.	**K-6h(5i)**	**G-3b(4a)**	10.	**G-5h(4i)**	**P-9d(9c)** *E
11.	**K-7i(6h)**	**P-4d(4c)**	12.	**P-4f(4g)**	**S-5d(6c)**
13.	**K-8h(7i)**	**K-4b(5a)**	14.	**S-4g(3h)**	**R-6b(8b)**
15.	**P-3f(3g)**	**P-6e(6d)**	16.	**S-5f(4g)**	**K-3a(4b)**
17.	**N-3g(2i)**	**B-6d** *F	18.	**G-4g(5h)**	**G-5b(6a)**
19.	**P-6f(6g)**	**Px6f(6e)**	20.	**R-6h(2h)**	**P-7d(7c)**
21.	**Sx6f(7g)**	**B-7c(6d)**	22.	**S-7g(6f)** *G	**P-6e**
23.	**P-6c** *H	**Gx6c(5b)**	24.	**Sx6e(5f)**	**Sx6e(5d)**
25.	**Rx6e(6h)**	**P-6d**	26.	**R-6h(6e)** *I	**N-9c(8a)**
27.	**P-4e(4f)**	**Px4e(4d)**	28.	**Nx4e(3g)**	**S-4d(3c)**
29.	**P-4f**	**N-8e(9c)**	30.	**S-8f(7g)**	**P-8d(8c)**
31.	**P-2d(2e)**	**Px2d(2c)**	32.	**B-5a**	**R-9b(6b)**
33.	**Bx7c+(5a)**	**Gx7c(6c)**	34.	**B-5a**	**G-6c(7c)**
35.	**Bx8d+(5a)**	**Sx4e(4d)**	36.	**Sx8e(8f)**	**B-3c**
37.	**P-6f**	**S-5d(4e)**	38.	**N-4e**	**B-4d(3c)**
39.	**Sx7d(8e)**	**P-8f**	40.	**Px8f(8g)**	**S-2c**
41.	**Sx6c+(7d)**	**Sx6c(5d)**	42.	**+B-7c(8d)**	**S-5d(6c)**
43.	**S-8c**	**R-9c(9b)**	44.	**G-8d** *J	**P-6e(6d)**
45.	**Gx9c(8d)**	**Px6f(6e)**	46.	**R-5a**	**K-2b(3a)**
47.	**Rx5c+(5a)**	**P-6g+(6f)**	48.	**+Rx4d(5c)**	**+Px6h(6g)**
49.	**Gx6h(7h)**	**P-6g**	50.	**G-5h(6h)**	**S-4c**
51.	**+Rx5d(4d)**	**Sx5d(4c)**	52.	**+Bx9a(7c)**	**K-1b(2b)**
53.	**P-2b**	**Gx2b(3b)**	54.	**L-1c** *K	**Kx1c(1b)**
55.	**P-1e(1f)**	**Px1e(1d)**	56.	**Lx1e(1i)**	**P-1d**
57.	**S-3a**	**G-3b(2b)**	58.	**B-7g**	**N-9e**
59.	**Sx9d+(8c)**	**P-8g**	60.	**K-7i(8h)**	**R-1h**
61.	**G-4h(4g)**	**Rx4h+(1h)**	62.	**Gx4h(5h)**	**L-6f**
63.	**Lx1d(1e)**	**Sx1d(2c)**	64.	**Resign** *L	

Appendix B

Games Scores Selected

Here we show some games played on the tournaments mentioned in Appendix A and Internet Shogi server.

B.1 The 14th WCSC (2004)

TACOS (Black) v.s. IS SHOGI (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-6f(6g)**	**P-8d(8c)**	3.	**R-6h(2h)**	**S-6b(7a)**	4.	**P-1f(1g)**	**P-1d(1c)**
5.	**S-3h(3i)**	**P-5d(5c)**	6.	**S-7h(7i)**	**K-4b(5a)**	7.	**K-4h(5i)**	**P-8e(8d)**	8.	**B-7g(8h)**	**K-3b(4b)**
9.	**K-3i(4h)**	**P-7d(7c)**	10.	**G-5h(6i)**	**S-4b(3a)**	11.	**K-2h(3i)**	**S-5c(4b)**	12.	**P-5f(5g)**	**S-6d(5c)**
13.	**P-4f(4g)**	**G-5b(6a)**	14.	**P-3f(3g)**	**P-7e(7d)**	15.	**S-6g(7h)**	**R-7b(8b)**	16.	**Px7e(7f)**	**Sx7e(6d)**
17.	**P-7f**	**Sx7f(7e)**	18.	**Sx7f(6g)**	**Rx7f(7b)**	19.	**S-8b**	**P-8f(8e)**	20.	**Px8f(8g)**	**R-7b(7f)**
21.	**Sx8a(8b)**	**R-7a(7b)**	22.	**P-6e(6f)**	**Bx7g+(2b)**	23.	**Nx7g(8i)**	**P-7f**	24.	**R-6i(6h)**	**Px7g+(7f)**
25.	**N-7d**	**Rx7d(7a)**	26.	**P-6d(6e)**	**N-2d**	27.	**G-4g(5h)**	**+P-6h(7g)**	28.	**Px6c+(6d)**	**Sx6c(6b)**
29.	**Rx6h(6i)**	**R-7i+(7d)**	30.	**P-6i**	**N-4d**	31.	**S-3g(3h)**	**B-7g**	32.	**R-3h(6h)**	**Bx9i+(7g)**
33.	**B-8b**	**Nx3f(4d)**	34.	**Sx3f(3g)**	**+Rx6i(7i)**	35.	**G-4h(4i)**	**Nx3f(2d)**	36.	**Gx3f(4g)**	**L-3e**
37.	**Gx3e(3f)**	**Px3e(3d)**	38.	**Rx3e(3h)**	**P-3c**	39.	**L-4i**	**+B-4d(9i)**	40.	**R-3i(3e)**	**+R-6h(6i)**
41.	**G-3h(4h)**	**P-1e(1d)**	42.	**Px1e(1f)**	**P-1h**	43.	**Lx1h(1i)**	**P-1g**	44.	**Lx1g(1h)**	**P-1f**
45.	**Lx1f(1g)**	**S-2e**	46.	**P-4e(4f)**	**+B-3e(4d)**	47.	**G-4h(3h)**	**S-3f**	48.	**N-1i**	**Sx1f(2e)**
49.	**Rx3f(3i)**	**+Bx3f(3e)**	50.	**S-5g**	**G-1g**	51.	**Nx1g(2i)**	**Sx1g+(1f)**	52.	**K-2i(2h)**	**L-2h**
53.	**Bx2h+(8b)**	**+Sx2h(1g)**	54.	**Kx2h(2i)**	**B-3g**	55.	**K-2i(2h)**	**R-2h**	56.	**K-3i(2i)**	**N-4g**
57.	**Gx4g(4h)**	**+R-3h(6h)**	58.	**Resign**							

TACOS (Black) v.s. GEKISASHI (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-6f(6g)**	**P-8d(8c)**	3.	**R-6h(2h)**	**S-6b(7a)**	4.	**P-1f(1g)**	**P-1d(1c)**
5.	**S-3h(3i)**	**K-4b(5a)**	6.	**S-7h(7i)**	**K-3b(4b)**	7.	**S-6g(7h)**	**P-5d(5c)**	8.	**K-4h(5i)**	**G-5b(6a)**
9.	**K-3i(4h)**	**S-4b(3a)**	10.	**G-5h(6i)**	**S-5c(4b)**	11.	**K-2h(3i)**	**S-5a(6b)**	12.	**B-7g(8h)**	**B-3c(2b)**
13.	**P-4f(4g)**	**K-2b(3b)**	14.	**S-5f(6g)**	**P-4d(4c)**	15.	**P-3f(3g)**	**G-3b(4a)**	16.	**P-2f(2g)**	**G-4c(5b)**
17.	**G-4g(5h)**	**S-4b(5a)**	18.	**S-2g(3h)**	**P-7d(7c)**	19.	**G-3h(4i)**	**R-7b(8b)**	20.	**S-6g(5f)**	**P-7e(7d)**
21.	**Px7e(7f)**	**Rx7e(7b)**	22.	**P-5f(5g)**	**P-9d(9c)**	23.	**P-7f**	**R-7a(7e)**	24.	**P-9f(9g)**	**R-7d(7a)**
25.	**R-6i(6h)**	**P-8e(8d)**	26.	**R-6h(6i)**	**P-6d(6c)**	27.	**R-8h(6h)**	**R-8d(7d)**	28.	**R-5h(8h)**	**N-7c(8a)**
29.	**B-5i(7g)**	**P-6e(6d)**	30.	**B-7g(5i)**	**R-6d(8d)**	31.	**R-6h(5h)**	**Px6f(6e)**	32.	**Sx6f(6g)**	**B-2d(3c)**
33.	**N-3g(2i)**	**R-6a(6d)**	34.	**P-2e(2f)**	**B-3c(2d)**	35.	**L-9h(9i)**	**R-6b(6a)**	36.	**G-4h(4g)**	**R-6d(6b)**
37.	**G-4g(4h)**	**S-3a(4b)**	38.	**R-6i(6h)**	**R-6a(6d)**	39.	**P-7e(7f)**	**R-6d(6a)**	40.	**G-4h(3h)**	**S-4b(3a)**
41.	**G-3h(4h)**	**L-9b(9a)**	42.	**P-6e**	**R-8d(6d)**	43.	**S-5g(6f)**	**P-7f**	44.	**B-5i(7g)**	**P-4e(4d)**
45.	**S-6f(5g)**	**Px4f(4e)**	46.	**Gx4f(4g)**	**P-8f(8e)**	47.	**Px8f(8g)**	**P-6h**	48.	**Rx6h(6i)**	**Rx8f(8d)**
49.	**N-4e(3g)**	**Rx8i+(8f)**	50.	**Nx3c+(4e)**	**Nx3c(2a)**	51.	**B-4h(5i)**	**P-4e**	52.	**G-4g(4f)**	**N-4f**
53.	**G-3i(3h)**	**N-8e(7c)**	54.	**P-4d**	**Sx4d(5c)**	55.	**B-2f(4h)**	**P-7g+(7f)**	56.	**R-4h(6h)**	**+P-6g(7g)**
57.	**G-4i(3i)**	**+Px6f(6g)**	58.	**P-2d(2e)**	**Px2d(2c)**	59.	**B-7a**	**S-5g**	60.	**Bx4d+(7a)**	**Gx4d(4c)**
61.	**P-2c**	**Kx2c(2b)**	62.	**P-2e**	**Nx2e(3c)**	63.	**P-1e(1f)**	**Sx4h+(5g)**	64.	**Gx4h(4g)**	**+Rx4i(8i)**
65.	**S-3i**	**N-1f**	66.	**Sx1f(2g)**	**G-3h**	67.	**Sx3h(3i)**	**B-3i**	68.	**K-2i(2h)**	**R-2h**
69.	**Resign**										

KCC (Black) v.s. TACOS (White)

1.**P-7f(7g)**	**P-3d(3c)**	2.**P-2f(2g)**	**P-4d(4c)**	3.**S-4h(3i)**	**R-4b(8b)**	4.**K-6h(5i)**	**K-6b(5a)**
5.**K-7h(6h)**	**K-7b(6b)**	6.**P-5f(5g)**	**K-8b(7b)**	7.**G-5h(4i)**	**S-3b(3a)**	8.**P-9f(9g)**	**L-9b(9a)**
9.**B-7g(8h)**	**S-4c(3b)**	10.**K-8h(7h)**	**K-9a(8b)**	11.**S-7h(7i)**	**S-8b(7a)**	12.**S-5g(4h)**	**G-5b(4a)**
13.**P-8f(8g)**	**G-7a(6a)**	14.**S-8g(7h)**	**P-6d(6c)**	15.**G-7h(6i)**	**G-6c(5b)**	16.**P-2e(2f)**	**B-3c(2b)**
17.**P-6f(6g)**	**P-4e(4d)**	18.**P-8e(8f)**	**S-4d(4c)**	19.**G-6g(5h)**	**P-5d(5c)**	20.**R-4h(2h)**	**S-5c(4d)**
21.**P-9e(9f)**	**P-7d(7c)**	22.**P-4f(4g)**	**Px4f(4e)**	23.**Sx4f(5g)**	**P-6e(6d)**	24.**P-4e**	**Px6f(6e)**
25.**Gx6f(6g)**	**P-6e**	26.**G-6g(6f)**	**P-4g**	27.**R-5h(4h)**	**Bx7g+(3c)**	28.**Nx7g(8i)**	**N-3c(2a)**
29.**Nx6e(7g)**	**S-6d(5c)**	30.**P-6f**	**B-3i**	31.**B-5a**	**R-4c(4b)**	32.**R-3h(5h)**	**P-4h+(4g)**
33.**Rx3i(3h)**	**+Px3i(4h)**	34.**B-5b**	**R-3a**	35.**Bx4c+(5b)**	**Rx5a(3a)**	36.**+Bx3c(4c)**	**R-6a(5a)**
37.**+Bx3d(3c)**	**R-5a(6a)**	38.**R-4c**	**G-6b(7a)**	39.**R-4b+(4c)**	**R-7a(5a)**	40.**P-4d(4e)**	**+Px2i(3i)**
41.**P-4c+(4d)**	**B-1e**	42.**+R-2b(4b)**	**B-2f(1e)**	43.**N-3h**	**B-1e(2f)**	44.**+P-5b(4c)**	**Gx5b(6b)**
45.**+Rx5b(2b)**	**P-6b**	46.**G-6a**	**R-7b(7a)**	47.**P-1f(1g)**	**P-7e(7d)**	48.**Px7e(7f)**	**+P-3i(2i)**
49.**Px1e(1f)**	**+Px3h(3i)**	50.**+B-4d(3d)**	**Sx7e(6d)**	51.**P-7c**	**Gx7c(6c)**	52.**Nx7c+(6e)**	**Rx7c(7b)**
53.**+Rx6b(5b)**	**P-7b**	54.**+R-7a(6b)**	**N-7f**	55.**Sx7f(8g)**	**N-9f**	56.**Lx9f(9i)**	**Sx7a(8b)**
57.**+Bx7a(4d)**	**R-8f**	58.**K-7i(8h)**	**R-8i+(8f)**	59.**Kx8i(7i)**	**N-9g**	60.**K-7i(8i)**	**R-4c(7c)**
61.**S-8b**	**Resign**						

Kakinoki Shogi (Black) v.s. Tacos (White)

1.**P-7f(7g)**	**P-3d(3c)**	2.**P-6f(6g)**	**P-8d(8c)**	3.**R-7h(2h)**	**P-8e(8d)**	4.**B-7g(8h)**	**S-6b(7a)**
5.**P-9f(9g)**	**K-4b(5a)**	6.**K-4h(5i)**	**K-3b(4b)**	7.**K-3h(4h)**	**G-5b(6a)**	8.**S-6h(7i)**	**P-1d(1c)**
9.**P-1f(1g)**	**P-5d(5c)**	10.**G-5h(6i)**	**S-4b(3a)**	11.**P-5f(5g)**	**S-5c(4b)**	12.**K-2h(3h)**	**P-7d(7c)**
13.**L-1h(1i)**	**R-7b(8b)**	14.**K-1i(2h)**	**P-7e(7d)**	15.**Px7e(7f)**	**Rx7e(7b)**	16.**S-6g(6h)**	**G-4b(4a)**
17.**B-8h(7g)**	**Rx7h+(7e)**	18.**Sx7h(6g)**	**R-8d**	19.**S-2h(3i)**	**P-8f(8e)**	20.**Px8f(8g)**	**Rx8f(8d)**
21.**B-9g(8h)**	**R-8b(8f)**	22.**P-8g**	**Bx6f(2b)**	23.**P-7g**	**L-9b(9a)**	24.**P-4f(4g)**	**P-6d(6c)**
25.**P-3f(3g)**	**P-9d(9c)**	26.**B-7i(9g)**	**P-6e(6d)**	27.**P-4e(4f)**	**N-7c(8a)**	28.**B-8h(7i)**	**P-7a**
29.**G-3h(4i)**	**P-8f**	30.**Px8f(8g)**	**Rx8f(8b)**	31.**P-8g**	**R-8d(8f)**	32.**R-9a**	**R-8b(8d)**
33.**P-9e(9f)**	**Px9e(9d)**	34.**Lx9e(9i)**	**P-9d**	35.**P-6g**	**B-8d(6f)**	36.**P-7f(7g)**	**G-3c(4b)**
37.**P-3e(3f)**	**P-4d(4c)**	38.**Lx9d(9e)**	**P-9c**	39.**Px4d(4e)**	**Sx4d(5c)**	40.**P-7e(7f)**	**Px3e(3d)**
41.**P-7d(7e)**	**Px9d(9c)**	42.**P-9c**	**P-3f(3e)**	43.**Px9b+(9c)**	**L-3g**	44.**G-4h(3h)**	**P-4g**
45.**Gx4g(4h)**	**R-8c(8b)**	46.**P-4e**	**Sx4e(4d)**	47.**+P-9c(9b)**	**Rx9c(8c)**	48.**Rx9c+(9a)**	**Bx9c(8d)**
49.**Px7c+(7d)**	**L-3h+(3g)**	50.**Bx3c+(8h)**	**Nx3c(2a)**	51.**N-4d**	**K-4c(3b)**	52.**R-4a**	**G-4b(5b)**
53.**Rx4b+(4a)**	**Kx4b(4c)**	54.**N-5b+(4d)**	**Kx5b(4b)**	55.**+Px6b(7c)**	**Kx6b(5b)**	56.**G-4h**	**+Lx2h(3h)**
57.**Kx2h(1i)**	**B-6d**	58.**P-5e(5f)**	**Bx5e(6d)**	59.**P-4f**	**Sx4f(4e)**	60.**L-6d**	**Bx6d(5e)**
61.**G-6a**	**Kx6a(6b)**	62.**L-6c**	**K-5b(6a)**	63.**S-6a**	**Kx6c(5b)**	64.**L-1g(1h)**	**Sx4g(4f)**
65.**N-3g(2i)**	**R-2i**	66.**Kx2i(2h)**	**R-3i**	67.**Kx3i(2i)**	**G-3h**	68.**Resign**	

Yss (Black) v.s. Tacos (White)

1.**P-7f(7g)**	**P-3d(3c)**	2.**P-6f(6g)**	**P-8d(8c)**	3.**R-6h(2h)**	**S-6b(7a)**	4.**S-7h(7i)**	**K-4b(5a)**
5.**K-4h(5i)**	**K-3b(4b)**	6.**S-3h(3i)**	**P-5d(5c)**	7.**K-3i(4h)**	**G-5b(6a)**	8.**B-7g(8h)**	**S-5c(6b)**
9.**P-1f(1g)**	**B-3c(2b)**	10.**P-1e(1f)**	**K-2b(3b)**	11.**K-2h(3i)**	**L-1b(1a)**	12.**P-6e(6f)**	**P-4d(4c)**
13.**P-7e(7f)**	**K-1a(2b)**	14.**R-6f(6h)**	**S-2b(3a)**	15.**R-7f(6f)**	**G-3a(4a)**	16.**P-7d(7e)**	**Px7d(7c)**
17.**Rx7d(7f)**	**P-7c**	18.**R-7f(7d)**	**G-4c(5b)**	19.**S-6g(7h)**	**P-4e(4d)**	20.**S-6f(6g)**	**B-4d(3c)**
21.**G-5h(6i)**	**G-3c(4c)**	22.**P-5f(5g)**	**P-9d(9c)**	23.**P-5e(5f)**	**Px5e(5d)**	24.**Sx5e(6f)**	**B-3e(4d)**
25.**P-5d**	**S-4b(5c)**	26.**P-6d(6e)**	**Px6d(6c)**	27.**Sx6d(5e)**	**S-5a(4b)**	28.**P-5c+(5d)**	**P-5b**
29.**Sx7c+(6d)**	**Nx7c(8a)**	30.**Rx7c+(7f)**	**R-9b(8b)**	31.**+P-4c(5c)**	**G-2d(3c)**	32.**N-3f**	**P-7f**
33.**Bx2b+(7g)**	**Gx2b(3a)**	34.**Nx2d(3f)**	**Px2d(2c)**	35.**S-3a**	**P-5c(5b)**	36.**P-5b**	**G-2c(2b)**
37.**Px5a+(5b)**	**S-2b**	38.**S-3b**	**B-5e**	39.**S-4b+(3a)**	**N-3f**	40.**K-1h(2h)**	**B-1g+(3e)**
41.**Nx1g(2i)**	**N-2h+(3f)**	42.**Kx2h(1h)**	**Bx3g+(5e)**	43.**Sx3g(3h)**	**P-1d(1c)**	44.**Sx2a+(3b)**	**Kx2a(1a)**
45.**+P-3b(4c)**	**K-1a(2a)**	46.**G-2a**	**Resign**				

Tacos (Black) v.s. kanazawa Shogi (White)

1.**P-7f(7g)**	**P-3d(3c)**	2.**P-6f(6g)**	**P-4d(4c)**	3.**S-7h(7i)**	**B-3c(2b)**	4.**G-5h(4i)**	**S-3b(3a)**
5.**P-2f(2g)**	**S-4c(3b)**	6.**S-4h(3i)**	**R-2b(8b)**	7.**K-6h(5i)**	**K-6b(5a)**	8.**P-5f(5g)**	**G-5b(4a)**
9.**P-9f(9g)**	**K-7b(6b)**	10.**P-8f(8g)**	**K-8b(7b)**	11.**S-5g(4h)**	**P-4e(4d)**	12.**K-7i(6h)**	**S-5d(4c)**
13.**S-8g(7h)**	**R-4b(2b)**	14.**P-9e(9f)**	**L-9b(9a)**	15.**G-7h(6i)**	**K-9a(8b)**	16.**G-6g(5h)**	**S-8b(7a)**
17.**P-1f(1g)**	**G-7a(6a)**	18.**P-1e(1f)**	**G-6b(5b)**	19.**B-7g(8h)**	**G-7b(6b)**	20.**K-8h(7i)**	**P-6d(6c)**
21.**B-6h(7g)**	**P-6e(6d)**	22.**B-7g(6h)**	**Px6f(6e)**	23.**Bx6f(7g)**	**Bx6f(3c)**	24.**Sx6f(5g)**	**P-4f(4e)**
25.**B-6d**	**B-3e**	26.**Bx4f(6d)**	**Bx4f(3e)**	27.**Px4f(4g)**	**Rx4f(4b)**	28.**P-5e(5f)**	**S-6c(5d)**
29.**P-6d**	**Sx6d(6c)**	30.**P-6e**	**B-3i**	31.**R-6h(2h)**	**Bx6f+(3i)**	32.**Gx6f(6g)**	**S-5g**
33.**R-6i(6h)**	**P-6h**	34.**Px6d(6e)**	**Px6i+(6h)**	35.**G-6g(6f)**	**R-4h+(4f)**	36.**P-6c+(6d)**	**Gx6c(7b)**
37.**B-3f**	**P-5d(5c)**	38.**Gx5g(6g)**	**+P-6h(6i)**	39.**B-9f**	**P-7d(7c)**	40.**S-5h**	**+Px5h(6h)**
41.**Gx5h(5g)**	**+R-4i(4h)**	42.**Px5d(5e)**	**R-3i**	43.**S-7i**	**Rx2i+(3i)**	44.**P-6d**	**Gx6d(6c)**
45.**P-6e**	**N-6f**	46.**Px6d(6e)**	**Nx7h+(6f)**	47.**Sx7h(7i)**	**+R-7i(4i)**	48.**K-7g(8h)**	**G-6e**
49.**P-8e(8f)**	**S-6f**	50.**K-8f(7g)**	**+R-8h(7i)**	51.**N-6g**	**+Rx8i(2i)**	52.**G-6h**	**+Rx7h(8i)**
53.**Gx7h(6h)**	**S-7g**	54.**Gx7g(7h)**	**Sx7g(6f)**	55.**Resign**			

Tacos (Black) v.s. Eisei Meijin (White)

1. **P-7f(7g)**	**P-3d(3c)**	2. **P-2f(2g)**	**P-4d(4c)**	3. **S-4h(3i)**	**S-4b(3a)**	4. **P-5f(5g)**	**G-3b(4a)**
5. **G-5h(4i)**	**G-5b(6a)**	6. **G-7h(6i)**	**S-3c(4b)**	7. **P-6f(6g)**	**K-4a(5a)**	8. **B-7g(8h)**	**B-3a(2b)**
9. **S-8h(7i)**	**B-4b(3a)**	10. **K-6i(5i)**	**K-3a(4a)**	11. **G-6g(5h)**	**K-2b(3a)**	12. **P-3f(3g)**	**P-8d(8c)**
13. **B-6h(7g)**	**P-8e(8d)**	14. **S-7g(8h)**	**S-6b(7a)**	15. **P-2e(2f)**	**P-6d(6c)**	16. **K-7i(6i)**	**G-4c(5b)**
17. **K-8h(7i)**	**S-6c(6b)**	18. **P-1f(1g)**	**P-7d(7c)**	19. **S-3g(4h)**	**B-5a(4b)**	20. **S-2f(3g)**	**B-8d(5a)**
21. **S-1e(2f)**	**P-1d(1c)**	22. **Sx1d(1e)**	**B-7c(8d)**	23. **B-4f(6h)**	**R-6b(8b)**	24. **N-3g(2i)**	**P-8f(8e)**
25. **Sx8f(7g)**	**P-3e(3d)**	26. **Px3e(3f)**	**Lx1d(1a)**	27. **P-1e(1f)**	**Lx1e(1d)**	28. **Lx1e(1i)**	**P-4e(4d)**
29. **Nx4e(3g)**	**S-4d(3c)**	30. **P-1c**	**P-1a**	31. **P-2d(2e)**	**Px2d(2c)**	32. **Rx2d(2h)**	**S-2c**
33. **Rx4d(2d)**	**Gx4d(4c)**	34. **Nx5c+(4e)**	**R-6a(6b)**	35. **L-6b**	**Bx6b(7c)**	36. **+Nx6b(5c)**	**Rx6b(6a)**
37. **B-5c**	**R-4b**	38. **S-5a**	**P-8e**	39. **Bx6b+(5c)**	**Rx6b(4b)**	40. **Sx6b+(5a)**	**Px8f(8e)**
41. **Px8f(8g)**	**L-8g**	42. **Gx8g(7h)**	**N-9e**	43. **+Sx6c(6b)**	**G-4e(4d)**	44. **L-2f**	**Nx8g+(9e)**
45. **Kx8g(8h)**	**B-6i**	46. **S-7h**	**Gx4f(4e)**	47. **N-3d**	**K-3c(2b)**	48. **R-5c**	**K-4d(3c)**
49. **R-5e+(5c)**	**K-3c(4d)**	50. **+R-5c(5e)**	**S-4c**	51. **+Rx4c(5c)**	**Gx4c(3b)**	52. **Lx2c+(2f)**	**K-4d(3c)**
53. **S-5e**	**K-4e(4d)**	54. **Sx4f(5e)**	**K-5d(4e)**	55. **R-5b**	**Kx6c(5d)**	56. **G-6b**	**K-7c(6c)**
57. **G-7b(6b)**	**K-8c(7c)**	58. **G-8b(7b)**	**K-8d(8c)**	59. **S-8e**	**K-9e(8d)**	60. **P-9f(9g)**	**Resign**

B.2 The 15th WCSC (2005)

Tacos (Black) v.s. Kcc (White)

1. **P-7f(7g)**	**P-3d(3c)**	2. **P-2f(2g)**	**P-4d(4c)**	3. **S-4h(3i)**	**S-3b(3a)**	4. **P-5f(5g)**	**R-4b(8b)**
5. **K-6h(5i)**	**K-6b(5a)**	6. **K-7h(6h)**	**K-7b(6b)**	7. **G-5h(4i)**	**K-8b(7b)**	8. **P-9f(9g)**	**P-9d(9c)**
9. **P-3f(3g)**	**S-7b(7a)**	10. **S-6h(7i)**	**G-5b(4a)**	11. **S-5g(6h)**	**B-3c(2b)**	12. **P-2e(2f)**	**P-5d(5c)**
13. **G-6h(6i)**	**S-4c(3b)**	14. **P-4f(4g)**	**P-6d(6c)**	15. **P-4e(4f)**	**G-6c(5b)**	16. **N-3g(2i)**	**P-7d(7c)**
17. **P-2d(2e)**	**Px2d(2c)**	18. **Px4d(4e)**	**Sx4d(4c)**	19. **P-4e**	**Sx4e(4d)**	20. **Bx3c+(8h)**	**Nx3c(2a)**
21. **B-8h**	**P-5e(5d)**	22. **Bx5e(8h)**	**R-4c(4b)**	23. **Rx2d(2h)**	**S-5d(4e)**	24. **B-8h(5e)**	**P-3e(3d)**
25. **R-2c+(2d)**	**P-4d**	26. **Px3e(3f)**	**B-1d**	27. **+R-2b(2c)**	**P-3f**	28. **P-4f**	**Px3g+(3f)**
29. **Sx3g(4h)**	**N-8d**	30. **G-6i(6h)**	**P-4e(4d)**	31. **S-4h(3g)**	**Nx7f(8d)**	32. **B-7g(8h)**	**Px4f(4e)**
33. **P-4d**	**R-5c(4c)**	34. **Sx4f(5g)**	**N-4e(3c)**	35. **P-5e(5f)**	**P-5g**	36. **G-5i(5h)**	**S-6e(5d)**
37. **Sx4e(4f)**	**R-2c(5c)**	38. **+Rx1a(2b)**	**R-2h+(2c)**	39. **P-2i**	**+Rx1i(2h)**	40. **Sx5g(4h)**	**+R-1h(1i)**
41. **L-5h**	**P-5f**	42. **Sx5f(4e)**	**Sx5f(6e)**	43. **Sx5f(5g)**	**P-5g**	44. **S-4h**	**Px5h+(5g)**
45. **Gx5h(6i)**	**Bx5h+(1d)**	46. **Gx5h(5i)**	**L-5g**	47. **Gx5g(5h)**	**+Rx4h(1h)**	48. **L-5h**	**G-7i**
49. **Kx7i(7h)**	**S-8h**	50. **Bx8h(7g)**	**+R-5i(4h)**	51. **K-7h(7i)**	**+R-6h(5i)**	52. **Resign**	

Tacos (Black) v.s. Bingo Shogi (White)

1. **P-7f(7g)**	**P-3d(3c)**	2. **P-2f(2g)**	**P-8d(8c)**	3. **P-2e(2f)**	**P-8e(8d)**	4. **G-7h(6i)**	**G-3b(4a)**
5. **P-2d(2e)**	**Px2d(2c)**	6. **Rx2d(2h)**	**P-8f(8e)**	7. **Px8f(8g)**	**Rx8f(8b)**	8. **R-2f(2d)**	**P-2c**
9. **P-8g**	**R-8b(8f)**	10. **P-1f(1g)**	**Bx8h+(2b)**	11. **Sx8h(7i)**	**B-3e**	12. **R-2e(2f)**	**Bx5g+(3e)**
13. **S-4h(3i)**	**+B-3e(5g)**	14. **Rx3e(2e)**	**Px3e(3d)**	15. **P-2h**	**S-2b(3a)**	16. **P-1e(1f)**	**S-6b(7a)**
17. **S-7g(8h)**	**S-3c(2b)**	18. **S-5g(4h)**	**P-1d(1c)**	19. **S-4f(5g)**	**Px1e(1d)**	20. **K-6h(5i)**	**P-1f(1e)**
21. **K-7i(6h)**	**P-1g+(1f)**	22. **Lx1g(1i)**	**Lx1g(1a)**	23. **Nx1g(2i)**	**R-1i**	24. **L-3i**	**Rx1g+(1i)**
25. **P-9f(9g)**	**L-4d**	26. **Sx3e(4f)**	**P-3d**	27. **Sx4d(3e)**	**Sx4d(3c)**	28. **L-4f**	**P-8f**
29. **Px8f(8g)**	**N-6e**	30. **S-6f(7g)**	**P-6d(6c)**	31. **Lx4d(4f)**	**Px4d(4c)**	32. **G-3h(4i)**	**L-8c**
33. **S-8g**	**G-4b(3b)**	34. **B-6h**	**+R-1i(1g)**	35. **B-2b**	**S-4i**	36. **K-8h(7i)**	**Sx3h+(4i)**
37. **Lx3h(3i)**	**P-8e**	38. **Px8e(8f)**	**G-5h**	39. **B-4f(6h)**	**Lx8e(8c)**	40. **P-8f**	**Lx8f(8e)**
41. **Sx8f(8g)**	**Rx8f(8b)**	42. **L-8g**	**Rx7f(8f)**	43. **P-7g**	**R-7d(7f)**	44. **B-3a+(2b)**	**P-8f**
45. **Lx8f(8g)**	**P-4e(4d)**	46. **B-5e(4f)**	**N-5g+(6e)**	47. **Sx5g(6f)**	**P-8g**	48. **K-9h(8h)**	**S-8h**
49. **Kx8g(9h)**	**Sx8i(8h)**	50. **G-8h(7h)**	**Gx5g(5h)**	51. **N-2f**	**+R-7i(1i)**	52. **+Bx4b(3a)**	**Kx4b(5a)**
53. **Nx3d(2f)**	**K-5b(4b)**	54. **N-4b+(3d)**	**Kx4b(5b)**	55. **S-3a**	**K-5b(4b)**	56. **G-4c**	**Kx4c(5b)**
57. **B-3c+(5e)**	**Nx3c(2a)**	58. **S-4b+(3a)**	**K-3d(4c)**	59. **K-9g(8g)**	**+Rx8h(7i)**	60. **Kx8h(9g)**	**G-7h**
61. **K-9g(8h)**	**G-8g**	62. **Kx8g(9g)**	**Rx7g+(7d)**	63. **Resign**			

Gps Shogi (Black) v.s. Tacos (White)

1. **P-7f(7g)**	**P-8d(8c)**	2. **P-6f(6g)**	**P-3d(3c)**	3. **R-6h(2h)**	**S-6b(7a)**	4. **S-7h(7i)**	**K-4b(5a)**
5. **K-4h(5i)**	**K-3b(4b)**	6. **K-3h(4h)**	**P-5d(5c)**	7. **K-2h(3h)**	**G-5b(6a)**	8. **L-1h(1i)**	**P-7d(7c)**
9. **K-1i(2h)**	**S-4b(3a)**	10. **S-2h(3i)**	**P-1d(1c)**	11. **S-6g(7h)**	**P-8e(8d)**	12. **B-7g(8h)**	**R-7b(8b)**
13. **G-3i(4i)**	**S-5c(4b)**	14. **G-5h(6i)**	**P-7e(7d)**	15. **Px7e(7f)**	**Rx7e(7b)**	16. **R-7h(6h)**	**G-4b(4a)**
17. **B-8h(7g)**	**Rx7h+(7e)**	18. **Sx7h(6g)**	**R-7b**	19. **B-7g(8h)**	**R-8b(7b)**	20. **P-7d**	**P-7f**
21. **B-8h(7g)**	**P-1e(1d)**	22. **G-6g(5h)**	**P-8f(8e)**	23. **Px8f(8g)**	**Rx8f(8b)**	24. **P-8g**	**R-8d(8f)**
25. **P-4f(4g)**	**P-9d(9c)**	26. **Gx7f(6g)**	**Rx7d(8d)**	27. **P-7e**	**R-8d(7d)**	28. **P-8f(8g)**	**P-6d(6c)**
29. **B-7g(8h)**	**B-5e(2b)**	30. **R-3f**	**G-3c(4b)**	31. **G-8e(7f)**	**R-8b(8d)**	32. **P-5f(5g)**	**B-4d(5e)**
33. **P-7d(7e)**	**P-6e(6d)**	34. **P-5e(5f)**	**Bx5e(4d)**	35. **S-6g(7h)**	**Px6f(6e)**	36. **S-7f(6g)**	**P-7h**
37. **P-1f(1g)**	**Px1f(1e)**	38. **Lx1f(1h)**	**P-1e**	39. **P-5f**	**Px1f(1e)**	40. **Px5e(5f)**	**L-1g**
41. **Nx1g(2i)**	**Px1g+(1f)**	42. **Sx1g(2h)**	**Lx1g+(1a)**	43. **P-1h**	**+Lx2g(1g)**	44. **G-2h(3i)**	**N-1e**
45. **L-2i**	**+Lx2h(2g)**	46. **Lx2h(2i)**	**S-3h**	47. **L-2i**	**G-3i**	48. **Lx2c+(2h)**	**K-3a(3b)**
49. **R-2f(3f)**	**P-2g**	50. **B-1c**	**Nx1c(2a)**	51. **+L-2b(2c)**	**K-4b(3a)**	52. **+L-3b(2b)**	**Gx3b(3c)**
53. **Resign**							

Gekisashi (Black) v.s. Tacos (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-2f(2g)**	**P-4d(4c)**	3.	**S-4h(3i)**	**S-4b(3a)**	4.	**P-5f(5g)**	**P-5d(5c)**
5.	**P-2e(2f)**	**S-3c(4b)**	6.	**S-6h(7i)**	**G-5b(6a)**	7.	**G-7h(6i)**	**G-4c(5b)**	8.	**K-6i(5i)**	**S-6b(7a)**
9.	**P-3f(3g)**	**P-8d(8c)**	10.	**S-5g(4h)**	**P-7d(7c)**	11.	**P-5e(5f)**	**G-3b(4a)**	12.	**Px5d(5e)**	**K-4a(5a)**
13.	**S-4f(5g)**	**Gx5d(4c)**	14.	**P-3e(3f)**	**Px3e(3d)**	15.	**Sx3e(4f)**	**P-3d**	16.	**P-2d(2e)**	**Px2d(2c)**
17.	**Sx2d(3e)**	**Sx2d(3c)**	18.	**Rx2d(2h)**	**S-3c**	19.	**R-2f(2d)**	**P-2d**	20.	**N-3g(2i)**	**P-8e(8d)**
21.	**P-2e**	**Px2e(2d)**	22.	**Nx2e(3g)**	**S-2d(3c)**	23.	**P-7e(7f)**	**Px7e(7d)**	24.	**P-3c**	**Nx3c(2a)**
25.	**Nx3c(2e)**	**Bx3c(2b)**	26.	**N-7d**	**R-7b(8b)**	27.	**S-8c**	**R-7c(7b)**	28.	**Nx6b+(7d)**	**Rx8c(7c)**
29.	**S-7b**	**R-8d(8c)**	30.	**B-6f(8h)**	**G-6e(5d)**	31.	**P-3e**	**N-2b**	32.	**P-2c**	**Gx2c(3b)**
33.	**N-7g(8i)**	**G-7f(6e)**	34.	**Px3d(3e)**	**Nx3d(2b)**	35.	**R-5f(2f)**	**P-5d**	36.	**P-3e**	**P-8f(8e)**
37.	**Px8f(8g)**	**Rx8f(8d)**	38.	**P-8g**	**Gx8g(7f)**	39.	**Rx5d(5f)**	**Gx7h(8g)**	40.	**Kx7h(6i)**	**S-8i**
41.	**K-6i(7h)**	**G-7h**	42.	**K-5h(6i)**	**Gx6h(7h)**	43.	**K-4h(5h)**	**S-4c**	44.	**Bx7e(6f)**	**K-3b(4a)**
45.	**Bx8f(7e)**	**P-7e**	46.	**Bx7e(8f)**	**G-2b(2c)**	47.	**R-5c+(5d)**	**Sx3e(2d)**	48.	**R-4a**	**K-2c(3b)**
49.	**+Rx4c(5c)**	**P-3a**	50.	**Rx3a+(4a)**	**G-5h(6h)**	51.	**Gx5h(4i)**	**P-2a**	52.	**+Rx3c(3a)**	**Gx3c(2b)**
53.	**+Rx3c(4c)**	**Kx3c(2c)**	54.	**S-4b**	**K-4c(3c)**	55.	**B-5c+(7e)**	**K-3b(4c)**	56.	**G-3c**	**Resign**

Ryuu no Tamago (Black) v.s. Tacos (White)

1.	**P-7f(7g)**	**P-8d(8c)**	2.	**G-6h(6i)**	**G-5b(6a)**	3.	**S-4h(3i)**	**P-3d(3c)**	4.	**Bx2b+(8h)**	**Sx2b(3a)**
5.	**S-8h(7i)**	**G-3b(4a)**	6.	**S-7g(8h)**	**S-6b(7a)**	7.	**G-7h(6h)**	**K-4b(5a)**	8.	**K-6i(5i)**	**S-3c(2b)**
9.	**P-2f(2g)**	**K-3a(4b)**	10.	**P-2e(2f)**	**K-2b(3a)**	11.	**G-5h(4i)**	**P-7d(7c)**	12.	**G-6h(5h)**	**N-7c(8a)**
13.	**K-7i(6i)**	**P-5d(5c)**	14.	**S-8h(7g)**	**S-5c(6b)**	15.	**S-5i(4h)**	**P-8e(8d)**	16.	**S-7g(8h)**	**P-6d(6c)**
17.	**P-6f(6g)**	**P-6e(6d)**	18.	**G-6g(6h)**	**Px6f(6e)**	19.	**Sx6f(7g)**	**P-8f(8e)**	20.	**Px8f(8g)**	**Rx8f(8b)**
21.	**P-8g**	**R-8b(8f)**	22.	**P-1f(1g)**	**P-4d(4c)**	23.	**P-9f(9g)**	**P-6e**	24.	**S-7g(6f)**	**S-6d(5c)**
25.	**S-5h(5i)**	**G-5c(5b)**	26.	**P-1e(1f)**	**P-4e(4d)**	27.	**S-8f(7g)**	**P-9d(9c)**	28.	**S-7g(8f)**	**P-3e(3d)**
29.	**B-7a**	**R-4b(8b)**	30.	**P-6c**	**N-8e(7c)**	31.	**S-8f(7g)**	**B-5e**	32.	**N-7g(8i)**	**Nx7g+(8e)**
33.	**Sx7g(8f)**	**P-3f(3e)**	34.	**G-5f(6g)**	**Px3g+(3f)**	35.	**R-1h(2h)**	**Bx7g+(5e)**	36.	**Gx7g(7h)**	**P-5e(5d)**
37.	**Gx6e(5f)**	**N-8e**	38.	**G-7h(7g)**	**Sx6e(6d)**	39.	**Bx5c+(7a)**	**S-7g**	40.	**G-8h**	**Sx7f(6e)**
41.	**N-8i**	**Sx7h+(7g)**	42.	**Gx7h(8h)**	**G-7g**	43.	**S-6h**	**Gx7h(7g)**	44.	**Kx7h(7i)**	**+Px4g(3g)**
45.	**Sx4g(5h)**	**P-6g**	46.	**+Bx4b(5c)**	**Px6h+(6g)**	47.	**Kx6h(7h)**	**Sx4b(3c)**	48.	**G-8f**	**G-6g**
49.	**K-5i(6h)**	**B-2f**	50.	**P-3g**	**P-4f(4e)**	51.	**Sx4f(4g)**	**G-4g**	52.	**B-6i**	**Gx4f(4g)**
53.	**Gx7f(8f)**	**P-4g**	54.	**S-5h**	**S-4h**	55.	**Rx4h(1h)**	**Px4h+(4g)**	56.	**Kx4h(5i)**	**R-2h**
57.	**K-3i(4h)**	**Gx3g(4f)**	58.	**Sx6g(5h)**	**S-3h**	59.	**Resign**				

Tacos (Black) v.s. Yss (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-2f(2g)**	**P-4d(4c)**	3.	**P-2e(2f)**	**B-3c(2b)**	4.	**S-4h(3i)**	**S-2b(3a)**
5.	**P-4f(4g)**	**G-3b(4a)**	6.	**S-4g(4h)**	**G-5b(6a)**	7.	**S-5f(4g)**	**G-4c(5b)**	8.	**K-6h(5i)**	**S-6b(7a)**
9.	**G-5h(4i)**	**P-5d(5c)**	10.	**P-3f(3g)**	**K-4a(5a)**	11.	**N-3g(2i)**	**K-3a(4a)**	12.	**S-7h(7i)**	**P-8d(8c)**
13.	**S-4g(5f)**	**S-5c(6b)**	14.	**P-1f(1g)**	**S-6d(5c)**	15.	**B-7i(8h)**	**P-4e(4d)**	16.	**S-7g(7h)**	**Px4f(4e)**
17.	**Sx4f(4g)**	**P-7d(7c)**	18.	**K-7h(6h)**	**P-7e(7d)**	19.	**Px7e(7f)**	**P-7f**	20.	**S-6f(7g)**	**P-8e(8d)**
21.	**B-6h(7i)**	**R-7b(8b)**	22.	**N-4e(3g)**	**B-4d(3c)**	23.	**P-5f(5g)**	**Sx7e(6d)**	24.	**Sx7e(6f)**	**Bx9i+(4d)**
25.	**S-8h**	**+B-9h(9i)**	26.	**P-4d**	**G-4b(4c)**	27.	**P-7d**	**P-4c**	28.	**Px4c+(4d)**	**Gx4c(3b)**
29.	**P-4d**	**Gx4d(4c)**	30.	**B-9e(6h)**	**G-4c(4d)**	31.	**B-5a+(9e)**	**P-4d**	32.	**+B-6a(5a)**	**R-9b(7b)**
33.	**P-6f(6g)**	**P-9d(9c)**	34.	**G-7i(6i)**	**P-9e(9d)**	35.	**+B-8c(6a)**	**Px4e(4d)**	36.	**+Bx9b(8c)**	**Lx9b(9a)**
37.	**R-6a**	**K-3b(3a)**	38.	**Sx4e(4f)**	**B-3g**	39.	**R-2i(2h)**	**P-4d**	40.	**Sx4d(4e)**	**Gx4d(4c)**
41.	**Rx8a+(6a)**	**B-4f+(3g)**	42.	**N-6h**	**+Bx3f(4f)**	43.	**G-6g(5h)**	**L-9d(9b)**	44.	**P-7c+(7d)**	**S-5h**
45.	**+Px6c(7c)**	**N-4f**	46.	**G-5g(6g)**	**P-9f(9e)**	47.	**Px9f(9g)**	**N-3h+(4f)**	48.	**P-3g**	**S-4g+(5h)**
49.	**Gx4g(5g)**	**+Bx4g(3f)**	50.	**S-5c**	**G-4c**	51.	**Sx4b+(5c)**	**Gx4b(4c)**	52.	**G-5b**	**P-4a**
53.	**Gx4b(5b)**	**Px4b(4a)**	54.	**+P-5b(6c)**	**G-3a**	55.	**R-2g(2i)**	**+Nx3g(3h)**	56.	**R-1g(2g)**	**S-5h**
57.	**+Rx8e(8a)**	**L-8a**	58.	**+Rx7f(8e)**	**P-6g**	59.	**P-6e(6f)**	**Px6h+(6g)**	60.	**Gx6h(7i)**	**S-5i(5h)**
61.	**G-6g(6h)**	**+B-5h(4g)**	62.	**P-4c**	**S-6h+(5i)**	63.	**Gx6h(6g)**	**+Bx7f(5h)**	64.	**Rx3g(1g)**	**+Bx7e(7f)**
65.	**G-9i**	**N-6f**	66.	**K-6g(7h)**	**R-4h**	67.	**Px4b+(4c)**	**K-3c(3b)**	68.	**+P-4c(4b)**	**Kx4c(3c)**
69.	**N-5e**	**Px5e(5d)**	70.	**S-5d**	**Kx5d(4c)**	71.	**+P-5c(5b)**	**Kx5c(5d)**	72.	**G-7g(6h)**	**R-5h+(4h)**
73.	**Resign**										

Tacos (Black) v.s. Is Shogi (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-2f(2g)**	**P-4d(4c)**	3.	**S-4h(3i)**	**S-4b(3a)**	4.	**P-5f(5g)**	**P-5d(5c)**
5.	**G-5h(4i)**	**S-6b(7a)**	6.	**S-7h(7i)**	**G-5b(6a)**	7.	**P-6f(6g)**	**G-3b(4a)**	8.	**B-7i(8h)**	**K-4a(5a)**
9.	**S-7g(7h)**	**S-3c(4b)**	10.	**G-6g(5h)**	**B-3a(2b)**	11.	**K-6h(5i)**	**P-7d(7c)**	12.	**K-7h(6h)**	**G-4c(5b)**
13.	**K-8h(7h)**	**S-7c(6b)**	14.	**G-7h(6i)**	**P-7e(7d)**	15.	**Px7e(7f)**	**Bx7e(3a)**	16.	**B-4f(7i)**	**B-6d(7e)**
17.	**S-5g(4h)**	**S-7d(7c)**	18.	**Bx6d(4f)**	**Px6d(6c)**	19.	**P-3f(3g)**	**N-7c(8a)**	20.	**P-7e**	**Sx7e(7d)**
21.	**B-6c**	**B-5b**	22.	**Bx5b+(6c)**	**Kx5b(4a)**	23.	**B-9f**	**K-5c(5b)**	24.	**B-4a+(9f)**	**S-8d(7e)**
25.	**N-3g(2i)**	**B-6c**	26.	**+Bx6c(4a)**	**Kx6c(5c)**	27.	**B-4a**	**K-5c(6c)**	28.	**P-1f(1g)**	**N-8e(7c)**

29. S-8f(7g)	G-3a(3b)	30. Bx2c+(4a)	P-2b	31. +Bx3c(2c)	Nx3c(2a)	32. S-7f	P-3e(3d)
33. Px3e(3f)	P-3f	34. N-2e(3g)	Nx2e(3c)	35. Sx8e(8f)	Sx8e(8d)	36. Sx8e(7f)	P-3g+(3f)
37. R-1h(2h)	+Px4g(3g)	38. Px2e(2f)	N-7e	39. G-7f(6g)	+Px5g(4g)	40. Gx7e(7f)	B-2i
41. S-7c	R-9b(8b)	42. Sx6d+(7c)	K-4b(5c)	43. G-6e(7e)	Bx1h+(2i)	44. Lx1h(1i)	S-6g
45. P-7i	Sx7h+(6g)	46. Px7h(7i)	R-6i	47. S-5c	K-3b(4b)	48. +Sx5d(6d)	R-7i+(6i)
49. Kx7i(8h)	S-6h	50. K-8h(7i)	B-7i	51. K-9h(8h)	G-8h	52. Resign	

B.3 The 10th CO (2005)

Tacos (Black) v.s. Yss (White)

1. P-7f(7g)	P-3d(3c)	2. P-2f(2g)	P-4d(4c)	3. S-4h(3i)	R-4b(8b)	4. P-5f(5g)	S-7b(7a)
5. K-6h(5i)	K-6b(5a)	6. K-7h(6h)	G-5b(4a)	7. P-9f(9g)	P-9d(9c)	8. G-5h(4i)	K-7a(6b)
9. P-2e(2f)	B-3c(2b)	10. P-3f(3g)	K-8b(7a)	11. S-6h(7i)	P-6d(6c)	12. S-5g(6h)	P-7d(7c)
13. R-3h(2h)	R-3b(4b)	14. G-6h(5h)	S-4b(3a)	15. P-3e(3f)	Px3e(3d)	16. Rx3e(3h)	S-4c(4b)
17. G-5h(6i)	P-5d(5c)	18. R-3i(3e)	B-2b(3c)	19. S-3g(4h)	P-7e(7d)	20. Px7e(7f)	R-3e(3b)
21. S-6f(5g)	Rx2e(3e)	22. P-2f	R-2d(2e)	23. S-5g(6f)	N-3c(2a)	24. P-4f(4g)	P-3f
25. Sx3f(3g)	Rx2f(2d)	26. P-2g	R-2d(2f)	27. R-3g(3i)	P-3d	28. P-3e	G-6c(5b)
29. Px3d(3e)	Sx3d(4c)	30. Bx4d(8h)	P-3e	31. S-4g(3f)	P-6e(6d)	32. L-1h(1i)	P-7f
33. B-8h(4d)	G-6d(6c)	34. P-3b	P-5e(5d)	35. S-3h(4g)	Px5f(5e)	36. Sx5f(5g)	P-4d
37. S-5e(5f)	P-7g+(7f)	38. Bx7g(8h)	Gx7e(6d)	39. P-5d	P-5b	40. B-8h(7g)	P-4e(4d)
41. R-5g(3g)	Px4f(4e)	42. Sx4f(5e)	P-6f(6e)	43. Px6f(6g)	G-6b(6a)	44. G-6g(5h)	P-6e
45. Px6e(6f)	Gx6e(7e)	46. R-3g(5g)	P-6f	47. Gx6f(6g)	Gx6f(6e)	48. Bx6f(8h)	G-5f
49. B-4d(6f)	G-7c(6b)	50. G-2a	N-2e(3c)	51. Gx2b(2a)	Nx3g+(2e)	52. Sx3g(4f)	R-3i
53. N-4h	Rx3h+(3i)	54. Nx5f(4h)	K-9b(8b)	55. P-9e(9f)	S-5i	56. P-4h	Sx6h+(5i)
57. Kx6h(7h)	Rx2g+(2d)	58. Px9d(9e)	G-6g	59. Kx6g(6h)	+Rx3g(3h)	60. Nx3g(2i)	+Rx3g(2g)
61. G-4g	+Rx4g(3g)	62. Px4g(4h)	N-7e	63. K-7f(6g)	S-8e	64. Kx8e(7f)	P-8d(8c)
65. Kx7e(8e)	G-7d(7c)	66. Kx7d(7e)	S-8c(7b)	67. K-6d(7d)	P-6c	68. K-5e(6d)	G-4e
69. K-6e(5e)	N-7c(8a)	70. K-7f(6e)	Sx9d(8c)	71. G-9c	Kx9c(9b)	72. B-7a+(4d)	G-8b
73. G-8c	Kx8c(9c)	74. B-7d	Kx7d(8c)	75. P-7e	K-8c(7d)	76. S-7d	K-9b(8c)
77. R-9c	Kx9c(9b)	78. R-8c	Resign				

Yss (Black) v.s. Tacos (White)

1. P-7f(7g)	P-8d(8c)	2. R-6h(2h)	P-3d(3c)	3. P-6f(6g)	S-6b(7a)	4. S-3h(3i)	P-5d(5c)
5. S-7h(7i)	K-4b(5a)	6. K-4h(5i)	K-3b(4b)	7. K-3i(4h)	G-5b(6a)	8. K-2h(3i)	P-1d(1c)
9. P-1f(1g)	P-7d(7c)	10. S-6g(7h)	S-4b(3a)	11. G-5h(6i)	S-5c(4b)	12. P-4f(4g)	P-8e(8d)
13. B-7g(8h)	G-4b(4a)	14. P-3f(3g)	R-7b(8b)	15. R-7h(6h)	S-6d(5c)	16. P-2f(2g)	P-7e(7d)
17. Px7e(7f)	Sx7e(6d)	18. G-4g(5h)	P-7f	19. B-5i(7g)	Sx6f(7e)	20. Sx6f(6g)	Bx6f(2b)
21. S-6a	R-7e(7b)	22. Sx5b+(6a)	Gx5b(4b)	23. R-9h(7h)	B-2b(6f)	24. P-5f(5g)	S-8h
25. P-5e(5f)	Sx8i+(8h)	26. B-4h(5i)	R-7b(7e)	27. R-6h(9h)	+Sx9i(8i)	28. R-6e(6h)	S-7d
29. R-6i(6e)	Bx5e(2b)	30. G-6f	B-2b(5e)	31. P-7e	S-8c(7d)	32. Rx9i(6i)	S-5c(6b)
33. S-6a	R-9b(7b)	34. Sx5b+(6a)	Rx5b(9b)	35. R-7i(9i)	L-6d	36. G-5f	N-8d
37. S-2g(3h)	S-7g	38. G-5g(4g)	S-6h(7g)	39. R-8i(7i)	P-7g+(7f)	40. G-5h(4i)	Sx5g+(6h)
41. Gx5g(5h)	+P-7f(7g)	42. G-6e(6f)	Lx6e(6d)	43. Gx6e(5f)	B-7g+(2b)	44. N-3g(2i)	+B-7h(7g)
45. R-9i(8i)	N-7c(8a)	46. G-7d(6e)	Sx7d(8c)	47. Px7d(7e)	N-6e(7c)	48. G-4g(5g)	G-5f
49. Gx5f(4g)	+Bx5f(7h)	50. Bx8d(4h)	G-4g	51. S-3i	N-5g+(6e)	52. N-4i	+N-4h(5g)
53. Sx4h(3i)	G-3h	54. K-1g(2h)	Gx4h(4g)	55. G-1h	Gx4i(3h)	56. P-7c+(7d)	P-1e(1d)
57. Px1e(1f)	Lx1e(1a)	58. P-1f	Lx1f(1e)	59. Sx1f(2g)	P-1e	60. Sx1e(1f)	P-1d
61. N-2d	Px2d(2c)	62. Sx2d(1e)	N-1b	63. S-2c	K-3a(3b)	64. Sx1b+(2c)	Rx1b(5b)
65. P-1f	+B-3h(5f)	66. Bx4h(8d)	Gx4h(4i)	67. G-2g	N-1e	68. L-3c	Nx3c(2a)
69. N-2c	K-2b(3a)	70. Sx3c(2d)	Kx3c(2b)	71. N-2e(3g)	Kx2c(3c)	72. N-3c+(2e)	Kx3c(2c)
73. N-4e	K-2c(3c)	74. L-2e	P-2d	75. Px1e(1f)	S-1f	76. Gx1f(2g)	+Bx1f(3h)
77. Kx1f(1g)	Px1e(1d)	78. K-1g(1f)	S-1f	79. Resign			

Tacos (Black) v.s. Spear (White)

1. P-7f(7g)	P-3d(3c)	2. P-2f(2g)	P-4d(4c)	3. S-4h(3i)	R-4b(8b)	4. P-5f(5g)	S-3b(3a)
5. K-6h(5i)	K-6b(5a)	6. K-7h(6h)	K-7b(6b)	7. G-5h(4i)	K-8b(7b)	8. P-9f(9g)	P-9d(9c)
9. P-3f(3g)	S-4c(3b)	10. S-6h(7i)	S-5d(4c)	11. P-2e(2f)	B-3c(2b)	12. R-3h(2h)	S-6e(5d)
13. P-3e(3f)	Px3e(3d)	14. Rx3e(3h)	P-6d(6c)	15. P-7e(7f)	B-2b(3c)	16. B-5e(8h)	R-6b(4b)
17. S-5g(6h)	P-5d(5c)	18. B-8h(5e)	S-7b(7a)	19. G-6h(6i)	R-3b(6b)	20. P-3d	G-4b(4a)
21. G-7g(6h)	P-5e(5d)	22. Px5e(5f)	P-5f	23. S-6h(5g)	P-4e(4d)	24. Rx4e(3e)	G-5b(6a)
25. P-6f(6g)	Rx3d(3b)	26. P-3g	P-3f	27. Px6e(6f)	Px6e(6d)	28. P-5d(5e)	Px3g+(3f)
29. Sx3g(4h)	Rx5d(3d)	30. Rx6e(4e)	P-3h	31. S-4e	R-1d(5d)	32. S-4h(3g)	P-6d
33. R-6g(6e)	P-2d(2c)	34. N-3g(2i)	G-6c(5b)	35. P-1f(1g)	Px2e(2d)	36. P-2c	B-3a(2b)
37. G-8f(7g)	P-4d	38. Sx4d(4e)	L-1b(1a)	39. S-3e(4d)	G-3c(4b)	40. P-3d	G-2d(3c)
41. P-1e(1f)	Gx1e(2d)	42. P-1f	Gx1f(1e)	43. N-4e(3g)	P-4d	44. Bx4d(8h)	G-5d(6c)
45. B-6b+(4d)	Gx4e(5d)	46. Lx1f(1i)	Rx1f(1d)	47. G-7a	S-6c(7b)	48. +Bx6c(6b)	Kx7a(8b)
49. S-7b	K-8b(7a)	50. Sx8a+(7b)	K-9c(8b)	51. N-8e	K-9b(9c)	52. +Bx4e(6c)	G-7b

53. +Bx7b(4e)	L-7f	54. Gx7f(8f)	N-8f	55. Gx8f(7f)	R-1i+(1f)	56. L-9c	Resign

Spear (Black) v.s. Tacos (White)

1. P-7f(7g)	P-8d(8c)	2. S-6h(7i)	P-3d(3c)	3. S-7g(6h)	S-6b(7a)	4. S-4h(3i)	P-5d(5c)
5. P-5f(5g)	S-4b(3a)	6. G-7h(6i)	G-3b(4a)	7. K-6i(5i)	K-4a(5a)	8. G-5h(4i)	G-5b(6a)
9. P-6f(6g)	S-3c(4b)	10. G-6g(5h)	B-3a(2b)	11. B-7i(8h)	P-4d(4c)	12. P-3f(3g)	G-4c(5b)
13. P-2f(2g)	P-7d(7c)	14. P-6e(6f)	S-7c(6b)	15. B-4f(7i)	B-4b(3a)	16. N-3g(2i)	K-3a(4a)
17. K-7i(6i)	P-8e(8d)	18. K-8h(7i)	K-2b(3a)	19. P-1f(1g)	P-4e(4d)	20. B-5g(4f)	S-4d(3c)
21. L-1g(1i)	P-9d(9c)	22. B-6f(5g)	P-6d(6c)	23. Px6d(6e)	Bx6d(4b)	24. P-6e	B-4b(6d)
25. R-2i(2h)	S-6b(7c)	26. Nx4e(3g)	N-7c(8a)	27. P-4f(4g)	Nx6e(7c)	28. S-6h(7g)	S-7c(6b)
29. P-2e(2f)	P-5e(5d)	30. Px5e(5f)	S-6d(7c)	31. G-5f(6g)	P-5d	32. Px5d(5e)	Gx5d(4c)
33. S-7g(6h)	S-5e(4d)	34. Gx5e(5f)	Sx5e(6d)	35. R-5i(2i)	Sx6f(5e)	36. Sx6f(7g)	P-8f(8e)
37. S-7c	Px8g+(8f)	38. Gx8g(7h)	Rx8g+(8b)	39. Kx8g(8h)	G-6g	40. R-8b	P-8f
41. Rx8f+(8b)	Bx8f(4b)	42. P-8h	N-7g+(6e)	43. Sx7g(6f)	Bx7g+(8f)	44. Nx7g(8i)	B-7h
45. K-9h(8g)	S-8i	46. Rx8i(5i)	Bx8i+(7h)	47. Kx8i(9h)	G-7h	48. K-9h(8i)	Gx8h(7h)
49. Kx8h(9h)	R-7h	50. K-8g(8h)	R-8h	51. K-9f(8g)	P-9e(9d)	52. Resign	

Tacos (Black) v.s. Mattari Yuchan (White)

1. P-7f(7g)	P-3d(3c)	2. P-2f(2g)	P-4d(4c)	3. S-4h(3i)	S-4b(3a)	4. P-5f(5g)	R-5b(8b)
5. K-6h(5i)	K-6b(5a)	6. K-7h(6h)	S-4c(4b)	7. G-5h(4i)	P-5d(5c)	8. S-6h(7i)	K-7b(6b)
9. P-9f(9g)	K-8b(7b)	10. P-2e(2f)	B-3c(2b)	11. P-3f(3g)	P-9d(9c)	12. P-4f(4g)	S-7b(7a)
13. P-2d(2e)	Px2d(2c)	14. P-4e(4f)	K-7a(8b)	15. Px4d(4e)	Sx4d(4c)	16. P-4e	S-5c(4d)
17. Bx3c+(8h)	Nx3c(2a)	18. Rx2d(2h)	P-2b	19. B-8h	G-3b(4a)	20. N-3g(2i)	G-2c(3b)
21. R-2i(2d)	P-5e(5d)	22. P-2d	G-1d(2c)	23. P-2c+(2d)	Px2c(2b)	24. Rx2c+(2i)	S-4b(5c)
25. Bx5e(8h)	P-3e(3d)	26. S-4g(4h)	K-8b(7a)	27. +R-2b(2c)	P-4f	28. Sx4f(4g)	P-5d
29. B-6f(5e)	Px3f(3e)	30. P-3d	Px3g+(3f)	31. Px3c+(3d)	Sx3c(4b)	32. +Rx3c(2b)	+P-2h(3g)
33. +R-2c(3c)	+Px1i(2h)	34. Bx1a+(6f)	B-6d	35. S-5g(4f)	P-3f	36. L-6e	B-5c(6d)
37. S-4c	R-5a(5b)	38. +B-3c(1a)	Bx1g+(5c)	39. S-4b(4c)	R-5b(5a)	40. N-4d	R-6b(5b)
41. S-5c+(4b)	L-2b	42. +Bx2b(3c)	Rx2b(6b)	43. +Rx2b(2c)	N-7a	44. R-1a	+B-1f(1g)
45. L-6b	P-2a	46. +Rx2a(2b)	P-4a	47. +Rx4a(2a)	B-2c	48. Lx6a+(6b)	Bx4a(2c)
49. +Lx7a(6a)	Kx7a(8b)	50. Rx4a+(1a)	S-6a(7b)	51. N-5b+(4d)	L-5a	52. +Rx5a(4a)	+Bx5b(1f)
53. +Sx5b(5c)	N-8f	54. Px8f(8g)	R-8h	55. Kx8h(7h)	L-9b(9a)	56. +Rx6a(5a)	K-8b(7a)
57. B-7a	K-9a(8b)	58. G-8b	Resign				

Mattari Yuchan (Black) v.s. Tacos (White)

1. P-7f(7g)	P-8d(8c)	2. P-2f(2g)	P-8e(8d)	3. P-2e(2f)	G-3b(4a)	4. G-7h(6i)	P-8f(8e)
5. Px8f(8g)	Rx8f(8b)	6. P-2d(2e)	Px2d(2c)	7. Rx2d(2h)	P-2c	8. R-2f(2d)	S-7b(7a)
9. P-9f(9g)	P-1d(1c)	10. N-7g(8i)	S-4b(3a)	11. K-4h(5i)	P-9d(9c)	12. P-1f(1g)	K-4a(5a)
13. S-3h(3i)	R-8b(8f)	14. P-7e(7f)	P-6d(6c)	15. P-8e	S-6c(7b)	16. R-7f(2f)	P-3d(3c)
17. K-3i(4h)	K-3a(4a)	18. S-6h(7i)	B-3c(2b)	19. K-2h(3i)	K-2b(3a)	20. N-1g(2i)	P-6e(6d)
21. N-2e(1g)	B-4d(3c)	22. Nx6e(7g)	Bx8h+(4d)	23. Gx8h(7h)	S-5d(6c)	24. B-6f	B-4d
25. Bx4d(6f)	Px4d(4c)	26. R-6f(7f)	P-6d	27. Nx7c(6e)	Nx7c(8a)	28. Rx6d(6f)	G-5a(6a)
29. P-8d(8e)	B-5e	30. P-8c+(8d)	Rx8c(8b)	31. P-8d	R-8b(8c)	32. P-8c+(8d)	Rx8c(8b)
33. P-8d	R-8b(8c)	34. P-8c+(8d)	Rx8c(8b)	35. P-8d	R-8b(8c)	36. P-8c+(8d)	Rx8c(8b)
37. P-8d	R-8b(8c)	38. R-7d(6d)	S-6e(5d)	39. P-5f(5g)	Sx7d(6e)	40. Px5e(5f)	Sx7e(7d)
41. B-3f	Rx8d(8b)	42. G-9h(8h)	R-8i+(8d)	43. B-2f	G-4c(3b)	44. B-7b+(3f)	R-6i
45. G-5i(4i)	Rx5i+(6i)	46. Sx5i(6h)	+Rx5i(8i)	47. +Bx7c(7b)	S-3i	48. K-1g(2h)	S-4h+(3i)
49. R-2i	G-3i	50. Rx3i(2i)	+Rx3i(5i)	51. G-2i	+Rx3h(3i)	52. Gx3h(2i)	+Sx3h(4h)
53. +Bx9a(7c)	S-2h	54. K-2g(1g)	+Sx3g(3h)	55. Bx3g(2f)	Sx3g+(2h)	56. Kx3g(2g)	N-4e
57. K-4f(3g)	B-5g	58. K-5f(4f)	G-6f	59. Px6f(6g)	Bx6f+(5g)	60. K-4f(5f)	R-5f
61. Resign							

B.4 The 3rd CSGP (2005)

Tacos (Black) v.s. Is Shogi (White)

1. P-7f(7g)	P-8d(8c)	2. S-6h(7i)	P-3d(3c)	3. P-6f(6g)	S-6b(7a)	4. P-5f(5g)	P-5d(5c)
5. G-5h(4i)	S-4b(3a)	6. S-4h(3i)	G-3b(4a)	7. G-6g(5h)	K-4a(5a)	8. G-7h(6i)	G-5b(6a)
9. K-6i(5i)	P-7d(7c)	10. S-7g(6h)	S-3c(4b)	11. B-7i(8h)	B-3a(2b)	12. P-3f(3g)	P-4d(4c)
13. B-6h(7i)	S-7c(6b)	14. K-7i(6i)	P-7e(7d)	15. Px7e(7f)	Bx7e(3a)	16. P-6e(6f)	G-4c(5b)
17. K-8h(7i)	B-4b(7e)	18. B-4f(6h)	P-7d	19. S-5g(4h)	K-3a(4a)	20. P-2f(2g)	R-6b(8b)
21. S-6f(5g)	P-9d(9c)	22. N-3g(2i)	P-6d(6c)	23. Px6d(6e)	P-4e(4d)	24. B-6h(4f)	Sx6d(7c)
25. P-2e(2f)	N-7c(8a)	26. Nx4e(3g)	S-2b(3c)	27. P-2d(2e)	P-6e	28. Px2c+(2d)	Sx2c(2b)
29. P-6c	R-7b(6b)	30. P-2d	S-1b(2c)	31. S-5g(6f)	P-4d	32. P-3e(3f)	Px4e(4d)

33. Px3d(3e)	Gx3d(4c)	34. R-3h(2h)	Gx2d(3d)	35. S-8f(7g)	P-2f	36. P-6b+(6c)	Rx6b(7b)
37. R-2h(3h)	N-6f	38. G-7i(7h)	P-8e(8d)	39. S-7g(8f)	P-2g+(2f)	40. Rx2g(2h)	N-5h+(6f)
41. R-2h(2g)	+Nx6h(5h)	42. Gx6h(6g)	B-3g	43. R-2i(2h)	B-1e+(3g)	44. P-2e	Gx2e(2d)
45. P-1f(1g)	+B-2d(1e)	46. N-1g	P-2h	47. Rx2h(2i)	P-2g	48. Rx2g(2h)	P-2f
49. R-3g(2g)	Gx1f(2e)	50. P-2h	G-1e(1f)	51. P-9f(9g)	S-7e(6d)	52. P-7f	S-8d(7e)
53. G-7h(7i)	B-6d(4b)	54. P-4f(4g)	Px4f(4e)	55. G-5h(6h)	P-3f	56. R-3h(3g)	P-4g+(4f)
57. Gx4g(5h)	P-3g+(3f)	58. Rx3g(3h)	Bx3g+(6d)	59. Gx3g(4g)	+Bx5g(2d)	60. B-5c	R-4b(6b)
61. Bx4b+(5c)	Gx4b(3b)	62. R-8a	K-2b(3a)	63. P-4c	B-6i	64. P-6h	+Bx5f(5g)
65. P-6g(6h)	R-4h	66. R-3a+(8a)	K-2c(2b)	67. +Rx2a(3a)	Sx2a(1b)	68. N-3e	K-2b(2c)
69. K-9g(8h)	Rx7h+(4h)	70. N-2c+(3e)	+Bx2c(5f)	71. S-8h(7g)	+Rx8h(7h)	72. Kx8h(9g)	R-7h
73. K-9g(8h)	S-8f	74. Px8f(8g)	R-8h+(7h)	75. Kx8h(9g)	G-8g	76. K-7i(8h)	G-7h(8g)
77. Resign							

YSS (Black) v.s. TACOS (White)

1. P-7f(7g)	P-8d(8c)	2. S-6h(7i)	P-3d(3c)	3. S-7g(6h)	S-6b(7a)	4. P-5f(5g)	P-5d(5c)
5. S-4h(3i)	S-4b(3a)	6. G-5h(4i)	G-3b(4a)	7. P-6f(6g)	K-4a(5a)	8. G-7h(6i)	P-7d(7c)
9. B-7i(8h)	B-3a(2b)	10. K-6i(5i)	S-3c(4b)	11. P-2f(2g)	P-4d(4c)	12. B-6h(7i)	G-5b(6a)
13. K-7i(6i)	B-6d(3a)	14. K-8h(7i)	K-3a(4a)	15. G-6g(5h)	G-4c(5b)	16. S-5g(4h)	N-7c(8a)
17. S-4f(5g)	N-8e(7c)	18. S-8f(7g)	K-2b(3a)	19. P-2e(2f)	P-9d(9c)	20. S-5g(4f)	S-5c(6b)
21. P-6e(6f)	B-7c(6d)	22. P-7e(7f)	P-6d(6c)	23. Px7d(7e)	B-5a(7c)	24. S-7e(8f)	Px6e(6d)
25. P-8f(8g)	Nx9g+(8e)	26. Kx9g(8h)	P-9e(9d)	27. K-8h(9g)	P-9f(9e)	28. G-8g(7h)	P-9g+(9f)
29. Lx9g(9i)	Lx9g(9a)	30. Gx9g(8g)	P-8e(8d)	31. Px8e(8f)	Rx8e(8b)	32. G-8f(9g)	R-8a(8e)
33. P-8e	B-4b(5a)	34. P-7c+(7d)	S-6d(5c)	35. +P-7d(7c)	Sx7e(6d)	36. +Px7e(7d)	L-6f
37. Sx6f(5g)	Px6f(6e)	38. Gx6f(6g)	P-6e	39. Gx6e(6f)	S-6g	40. B-4f(6h)	Sx5f(6g)
41. G-6f(6e)	S-5e	42. Gx5f(6f)	Sx5f(5e)	43. P-2d(2e)	Px2d(2c)	44. P-2c	Kx2c(2b)
45. N-1e	K-2b(2c)	46. S-5b	G-5c(4c)	47. S-4a	G-3a	48. Sx3b+(4a)	Gx3b(3a)
49. L-4c	Gx5b(5c)	50. Lx4b+(4c)	Gx4b(5b)	51. P-2c	K-3a(2b)	52. B-9b	L-2e
53. R-6h(2h)	P-6g	54. R-7h(6h)	P-8g	55. Kx8g(8h)	S-6i	56. R-4h(7h)	S-7h
57. K-9f(8g)	R-5a(8a)	58. Bx5f+(9b)	Sx8i(7h)	59. K-9e(9f)	P-1d(1c)	60. B-7c+(4f)	R-5c(5a)
61. G-6c	R-4c(5c)	62. S-5b	Px1e(1d)	63. Sx4c+(5b)	Gx4c(3b)	64. K-9d(9e)	K-3b(3a)
65. R-5a	S-5i	66. R-3h(4h)	P-7b	67. Gx7b(6c)	S-7h(6i)	68. K-8c(9d)	Lx2i+(2e)
69. P-5c	Gx5c(4c)	70. +P-6d(7e)	G-5b(5c)	71. R-6a+(5a)	G-4a(4b)	72. +B-7d(5f)	+Lx1i(2i)
73. +Rx4a(6a)	Kx2c(3b)	74. +Rx2a(4a)	N-2b	75. +Bx5b(7d)	N-9a	76. +Bx9a(7c)	L-3b
77. +B-4c(5b)	P-8b	78. Kx8b(8c)	P-8a	79. +Bx8a(9a)	P-1f(1e)	80. +Bx3b(4c)	K-1c(2c)
81. G-2c	K-1d(1c)	82. +Rx1a(2a)	N-1b	83. +Rx1b(1a)	K-2e(1d)	84. G-3f	Resign

GEKISASHI (Black) v.s. TACOS (White)

1. P-7f(7g)	P-8d(8c)	2. R-7h(2h)	P-8e(8d)	3. B-7g(8h)	P-3d(3c)	4. P-6f(6g)	S-6b(7a)
5. S-6h(7i)	K-4b(5a)	6. K-4h(5i)	K-3b(4b)	7. K-3h(4h)	P-5d(5c)	8. K-2h(3h)	G-5b(6a)
9. P-5f(5g)	P-7d(7c)	10. G-5h(6i)	S-4b(3a)	11. S-3h(3i)	S-5c(4b)	12. P-1f(1g)	P-1d(1c)
13. P-4f(4g)	G-4b(4a)	14. R-8h(7h)	S-6d(5c)	15. S-5g(6h)	P-9d(9c)	16. G-4g(5h)	P-7e(7d)
17. P-6e(6f)	Sx6e(6d)	18. Bx2b+(7g)	Kx2b(3b)	19. Px7e(7f)	B-4d	20. R-6h(8h)	Bx9i+(4d)
21. Rx6e(6h)	L-6d	22. R-2e(6e)	N-3c(2a)	23. S-3a	Kx3a(2b)	24. Rx2c+(2e)	S-2b
25. +Rx3d(2c)	S-5c(6b)	26. P-1e(1f)	Px1e(1d)	27. P-1b	Lx1b(1a)	28. P-1c	Lx1c(1b)
29. P-1d	Lx1d(1c)	30. +Rx1d(3d)	G-3b(4b)	31. L-2f	P-2e	32. Lx2e(2f)	Nx2e(3c)
33. +Rx2e(1d)	L-6g+(6d)	34. S-4h(5g)	+Bx8i(9i)	35. N-2d	G-3c(3b)	36. N-1b+(2d)	S-2c(2b)
37. B-2b	K-4b(3a)	38. Bx3c+(2b)	Kx3c(4b)	39. +N-1c(1b)	P-2b	40. +Nx2c(1c)	Px2c(2b)
41. S-3d	K-4b(3c)	42. +Rx2c(2e)	+B-8h(8i)	43. G-3c	K-5a(4b)	44. Gx4c(3c)	P-2b
45. Gx5b(4c)	Rx5b(8b)	46. +Rx5c(2c)	Rx5c(5b)	47. G-4c	K-6b(5a)	48. Gx5c(4c)	Kx5c(6b)
49. R-8b	N-1f	50. Lx1f(1i)	Px1f(1e)	51. P-1h	R-1a	52. S-5g(4h)	P-1g+(1f)
53. Px1g(1h)	Rx1g+(1a)	54. Kx1g(2h)	L-1b	55. P-1f	Lx1f(1b)	56. Kx1f(1g)	L-1a
57. P-1c	G-2e	58. Kx2e(1f)	G-1e	59. Kx1e(2e)	B-4b	60. Rx4b+(8b)	K-6d(5c)
61. G-7d	K-6e(6d)	62. N-7g	K-7f(6e)	63. B-6e	Kx7g(7f)	64. R-7f	Kx8g(7g)
65. R-7i(7f)	Kx9g(8g)	66. L-9i	+Bx9i(8h)	67. Rx9i(7i)	K-8h(9g)	68. G-8i	K-7g(8h)
69. S-8h	K-8f(7g)	70. S-8g(8h)	K-7g(8f)	71. B-8h	Resign		

In this game, TACOS mistook that the move 52. **P-1g+(1f)** leads wining position because of a kind of software bug.

TACOS (Black) v.s. KCC (White)

1. P-7f(7g)	P-3d(3c)	2. P-2f(2g)	P-4d(4c)	3. S-4h(3i)	S-3b(3a)	4. P-5f(5g)	R-4b(8b)
5. K-6h(5i)	K-6b(5a)	6. K-7h(6h)	K-7b(6b)	7. G-5h(4i)	K-8b(7b)	8. P-9f(9g)	P-9d(9c)
9. P-3f(3g)	S-7b(7a)	10. S-6h(7i)	S-4c(3b)	11. S-5g(6h)	P-5d(5c)	12. P-2e(2f)	B-3c(2b)
13. G-6h(6i)	P-6d(6c)	14. P-4f(4g)	G-3b(4a)	15. S-3g(4h)	P-7d(7c)	16. S-2f(3g)	S-6c(7b)
17. P-3e(3f)	R-7b(4b)	18. S-6f(5g)	P-6e(6d)	19. S-7g(6f)	S-6d(6c)	20. Px3d(3e)	Sx3d(4c)
21. R-3h(2h)	S-4c(3d)	22. B-9g(8h)	P-7e(7d)	23. Px7e(7f)	P-4e(4d)	24. Px4e(4f)	P-9e(9d)
25. B-8f(9g)	P-8d(8c)	26. P-4d(4e)	Sx4d(4c)	27. S-7f(7g)	S-5c(4d)	28. N-7g(8i)	R-4b(7b)
29. P-4d	Rx4d(4b)	30. P-4g	R-4b(4d)	31. S-3e(2f)	Px9f(9e)	32. P-7d(7e)	P-7e

33. Sx7e(7f)	Sx7e(6d)	34. Bx7e(8f)	S-6d	35. S-7c	Nx7c(8a)	36. Px7c+(7d)	Kx7c(8b)
37. Nx6e(7g)	K-8c(7c)	38. Nx5c+(6e)	R-7b(4b)	39. P-7f	Sx7e(6d)	40. +N-6c(5c)	Sx7f(7e)
41. +Nx7b(6c)	Gx7b(6a)	42. R-8a	S-8b	43. Rx2a+(8a)	Bx9i+(3c)	44. +Rx3b(2a)	N-7g
45. S-7d	K-9b(8c)	46. G-5g(6h)	P-6b	47. P-9c	Kx9c(9b)	48. +Rx2c(3b)	P-7c
49. G-6f(5g)	P-2b	50. +R-4c(2c)	P-4b	51. +R-5c(4c)	B-9g	52. N-7e	N-8i+(7g)
53. K-6h(7h)	+Bx6f(9i)	54. Px6f(6g)	Bx7e+(9g)	55. N-6d	G-7a(7b)	56. Sx7c+(7d)	L-6c
57. G-7d	+Bx6f(7e)	58. +Sx8b(7c)	Gx8b(7a)	59. K-5i(6h)	S-5g	60. Gx5g(5h)	+Bx5g(6f)
61. P-9d	Kx9d(9c)	62. P-6h	S-7g+(7f)	63. S-7h	N-3f	64. P-9e	K-8e(9d)
65. B-5h	Resign						

Tacos (Black) v.s. Bingo Shogi (White)

1. P-7f(7g)	P-3d(3c)	2. P-2f(2g)	P-4d(4c)	3. S-4h(3i)	R-3b(8b)	4. P-2e(2f)	B-3c(2b)
5. K-6h(5i)	K-6b(5a)	6. K-7h(6h)	K-7b(6b)	7. P-5f(5g)	S-4b(3a)	8. G-5h(4i)	P-9d(9c)
9. P-9f(9g)	K-8b(7b)	10. P-3f(3g)	S-7b(7a)	11. P-4f(4g)	S-4c(4b)	12. S-6h(7i)	G-5a(4a)
13. P-1f(1g)	G-5b(5a)	14. S-5g(6h)	P-6d(6c)	15. R-3h(2h)	R-2b(3b)	16. P-3e(3f)	Px3e(3d)
17. Rx3e(3h)	P-3d	18. R-3f(3e)	P-2d(2c)	19. P-4e(4f)	Px4e(4d)	20. Bx3c+(8h)	Nx3c(2a)
21. B-5e	G-4b(5b)	22. Bx6d(5e)	Px2e(2d)	23. P-3b	P-2f(2e)	24. P-3a+(3b)	P-2g+(2f)
25. +P-4a(3a)	Gx4a(4b)	26. Bx5c+(6d)	G-5b(6a)	27. +B-6d(5c)	B-2h	28. +Bx2h(6d)	+Px2h(2g)
29. P-4d	S-3b(4c)	30. Rx3d(3f)	+Px2i(2h)	31. B-1e	N-2a	32. R-2d(3d)	Rx2d(2b)
33. Bx2d(1e)	+Px1i(2i)	34. P-3d	L-8h	35. Kx8h(7h)	R-2i	36. G-6h(5h)	Rx2d+(2i)
37. Px3c+(3d)	+Rx3c(2d)	38. N-6d	G-6c(5b)	39. Nx7b+(6d)	Kx7b(8b)	40. S-5i(4h)	+Rx4d(3c)
41. S-5e	+R-5c(4d)	42. R-2b	N-8d	43. K-7h(8h)	Nx7f(8d)	44. G-7g(6h)	B-8h
45. Gx7f(7g)	Bx9i+(8h)	46. L-6f	P-6e	47. Lx6e(6f)	P-6d	48. N-7e	Px6e(6d)
49. P-3c	+Rx3c(5c)	50. Nx6c+(7e)	+Rx6c(3c)	51. R-2h+(2b)	N-8d	52. G-8f(7f)	P-5d
53. G-6d	Px5e(5d)	54. Gx6c(6d)	Kx6c(7b)	55. +R-2d(2h)	Px5f(5e)	56. R-6d	K-7b(6c)
57. Sx5f(5g)	B-4f	58. G-7i(6i)	Bx2d(4f)	59. Rx2d(6d)	R-9h	60. K-6i(7h)	+Bx8i(9i)
61. B-5d	K-6a(7b)	62. P-7h	+Bx7i(8i)	63. Kx7i(6i)	R-9i+(9h)	64. B-8i	G-5g
65. B-4c+(5d)	Sx4c(3b)	66. R-6d(2d)	K-7a(6a)	67. R-6a+(6d)	Kx6a(7a)	68. P-7g(7h)	G-8h
69. K-6i(7i)	+Rx8i(9i)	70. Resign					

Gps Shogi (Black) v.s. Tacos (White)

1. P-7f(7g)	P-8d(8c)	2. P-6f(6g)	P-3d(3c)	3. R-6h(2h)	S-6b(7a)	4. S-7h(7i)	K-4b(5a)
5. K-4h(5i)	K-3b(4b)	6. K-3h(4h)	P-5d(5c)	7. K-2h(3h)	G-5b(6a)	8. L-1h(1i)	P-7d(7c)
9. K-1i(2h)	S-4b(3a)	10. S-2h(3i)	P-1d(1c)	11. G-3i(4i)	S-5c(4b)	12. S-6g(7h)	G-4b(4a)
13. S-5f(6g)	P-8e(8d)	14. B-7g(8h)	P-7e(7d)	15. Px7e(7f)	R-7b(8b)	16. S-6g(5f)	Rx7e(7b)
17. G-5h(6i)	P-9d(9c)	18. P-7f	R-7d(7e)	19. G-4h(5h)	S-6d(5c)	20. B-8h(7g)	P-7e
21. Px7e(7f)	Sx7e(6d)	22. P-6e(6f)	Bx8h+(2b)	23. Rx8h(6h)	B-2b	24. R-9h(8h)	P-8f(8e)
25. Px8f(8g)	P-8h	26. B-8b	Px8i+(8h)	27. Bx9a+(8b)	+P-8h(8i)	28. +B-9b(9a)	R-7a(7d)
29. L-8b	+Px9h(8h)	30. Lx8a+(8b)	R-3a(7a)	31. Lx9h(9i)	B-7g+(2b)	32. S-5h(6g)	R-8h
33. N-7d	S-5c(6b)	34. P-7c	N-6g	35. P-4f(4g)	N-5i+(6g)	36. S-4g(5h)	S-7f(7e)
37. P-7b+(7c)	S-6g+(7f)	38. N-6b+(7d)	Sx6b(5c)	39. +Px6b(7b)	Gx6b(5b)	40. S-5f	+N-5h(5i)
41. Gx5h(4h)	+Sx5h(6g)	42. Sx5h(4g)	Rx5h+(8h)	43. N-7d	G-5b(6b)	44. P-7c	+Rx9h(5h)
45. S-4i	+B-6f(7g)	46. S-4g(5f)	N-5e	47. S-5h(4g)	G-5i	48. P-7b+(7c)	Gx5h(5i)
49. Sx5h(4i)	+Rx5h(9h)	50. G-4h	+R-6i(5h)	51. +P-6b(7b)	G-5c(5b)	52. P-6d(6e)	L-4g
53. +Px6c(6b)	Lx4h+(4g)	54. Gx4h(3i)	+Bx5g(6f)	55. +Px5c(6c)	+Bx4h(5g)	56. +Px4b(5c)	Kx4b(3b)
57. G-5c	Resign (crashed)						

Tacos lost this game by crashing although Tacos was almost in winning position.

Tacos (Black) v.s. Ryuu no Tamago (White)

1. P-7f(7g)	P-3d(3c)	2. P-2f(2g)	Bx8h+(2b)	3. Sx8h(7i)	S-6b(7a)	4. S-3h(3i)	S-2b(3a)
5. G-5h(4i)	K-4b(5a)	6. S-2g(3h)	G-5b(6a)	7. G-7h(6i)	K-3b(4b)	8. S-3f(2g)	G-4b(4a)
9. S-7g(8h)	P-8d(8c)	10. K-6h(5i)	P-9d(9c)	11. K-7i(6h)	P-8e(8d)	12. K-8h(7i)	P-9e(9d)
13. P-2e(2f)	S-3c(2b)	14. P-1f(1g)	P-4d(4c)	15. P-4f(4g)	P-1d(1c)	16. P-4e(4f)	Px4e(4d)
17. Sx4e(3f)	P-4d	18. S-5f(4e)	P-3e(3d)	19. B-4f	S-3d(3c)	20. P-3f(3g)	Px3f(3e)
21. P-2d(2e)	Px2d(2c)	22. G-4g(5h)	G-3c(4b)	23. Bx2d(4f)	Gx2d(3c)	24. Rx2d(2h)	S-2c(3d)
25. G-3a	Kx3a(3b)	26. Rx2c+(2d)	G-3b	27. +R-2f(2c)	P-2b	28. S-2d	B-1b
29. P-1e(1f)	P-2c(2b)	30. S-3e(2d)	G-2b(3b)	31. Px1d(1e)	R-8d(8b)	32. P-1c+(1d)	Nx1c(2a)
33. P-1d	B-2a(1b)	34. Px1c+(1d)	Lx1c(1a)	35. Lx1c+(1i)	Gx1c(2b)	36. L-3d	K-4b(3a)
37. N-2e	G-2d(1c)	38. Sx2d(3e)	Px2d(2c)	39. N-3c+(2e)	K-5a(4b)	40. +Rx2d(2f)	B-5d(2a)
41. +R-2b(2d)	K-6a(5a)	42. P-4c	P-2a	43. +R-3a(2b)	K-7b(6a)	44. P-4b+(4c)	Gx4b(5b)
45. +Nx4b(3c)	S-2b	46. +R-4a(3a)	K-8b(7b)	47. +N-5b(4b)	B-2c	48. +Nx6b(5b)	Bx4a(2c)
49. S-7e	R-6d(8d)	50. G-7b	K-9c(8b)	51. Sx6d(7e)	Px6d(6c)	52. R-7a	S-8c
53. Rx8a+(7a)	B-7d(4a)	54. N-6f	Bx7b(5d)	55. +Nx7b(6b)	R-4a	56. +R-8b(8a)	K-9d(9c)
57. Nx7d(6f)	G-9b	58. +Nx7c(7b)	L-8a	59. +Nx8c(7c)	Gx8c(9b)	60. +Rx9a(8b)	N-9c
61. L-3b+(3d)	Gx7d(8c)	62. B-7b	N-8c	63. +Lx4a(3b)	G-7c(7d)	64. +Rx9c(9a)	Kx9c(9d)
65. R-9a	R-9b	66. Rx9b+(9a)	Kx9b(9c)	67. L-9d	R-9c	68. G-9a	Kx9a(9b)
69. Lx9c(9d)	G-9b	70. Lx9b+(9c)	Kx9b(9a)	71. R-9d	L-9c	72. Bx8a+(7b)	Kx8a(9b)
73. R-9a	Kx9a(8a)	74. Rx9c+(9d)	B-9b	75. S-8b	Resign		

B.5 The 16th WCSC (2006)

OTSUKI SHOGI (Black) v.s. TACOS (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-1f(1g)**	**P-8d(8c)**	3.	**Bx2b+(8h)**	**Sx2b(3a)**	4.	**P-1e(1f)**	**S-6b(7a)**
5.	**S-7h(7i)**	**B-6e**	6.	**S-4h(3i)**	**Bx7f(6e)**	7.	**P-9f(9g)**	**K-4b(5a)**	8.	**P-9e(9f)**	**K-3b(4b)**
9.	**G-6h(6i)**	**B-5d(7f)**	10.	**P-5f(5g)**	**P-8e(8d)**	11.	**G-7g(6h)**	**P-8f(8e)**	12.	**Px8f(8g)**	**P-8h**
13.	**N-9g(8i)**	**P-9d(9c)**	14.	**P-7d**	**Px9e(9d)**	15.	**N-8e(9g)**	**P-9f(9e)**	16.	**Nx7c+(8e)**	**Nx7c(8a)**
17.	**Px7c+(7d)**	**Sx7c(6b)**	18.	**P-5e(5f)**	**B-6e(5d)**	19.	**N-5g**	**B-7d(6e)**	20.	**P-7e**	**B-5f(7d)**
21.	**G-6f(7g)**	**B-8c(5f)**	22.	**N-6e(5g)**	**S-6b(7c)**	23.	**B-4f**	**N-6d**	24.	**P-8e(8f)**	**P-9g+(9f)**
25.	**P-8d(8e)**	**Bx6e(8c)**	26.	**Gx6e(6f)**	**Rx8d(8b)**	27.	**G-7d(6e)**	**R-8e(8d)**	28.	**Lx9g(9i)**	**Lx9g+(9a)**
29.	**B-9d**	**R-8a(8e)**	30.	**R-1h(2h)**	**P-8i+(8h)**	31.	**P-8b**	**R-9a(8a)**	32.	**Bx6a+(9d)**	**Rx6a(9a)**
33.	**G-7b**	**L-5f**	34.	**S-5g(4h)**	**B-9e**	35.	**K-4h(5i)**	**R-5a(6a)**	36.	**Gx6b(7b)**	**Bx6b(9e)**
37.	**Sx8i(7h)**	**Lx5g+(5f)**	38.	**Bx5g(4f)**	**B-9e(6b)**	39.	**Gx6c(7d)**	**B-7g+(9e)**	40.	**Gx6d(6c)**	**S-5f**
41.	**B-4f(5g)**	**G-4e**	42.	**L-5i**	**+Bx6g(7g)**	43.	**Lx5f(5i)**	**Gx4f(4e)**	44.	**Px4f(4g)**	**B-6f**
45.	**K-3h(4h)**	**B-5g+(6f)**	46.	**S-5h**	**+Bx5f(6g)**	47.	**S-4g**	**N-3e**	48.	**Sx5f(4g)**	**+Bx5f(5g)**
49.	**N-4g**	**L-2d**	50.	**G-2h**	**S-5g**	51.	**P-3f(3g)**	**Sx5h(5g)**	52.	**Gx5h(4i)**	**S-5g**
53.	**S-4i**	**Sx5h(5g)**	54.	**Sx5h(4i)**	**G-5i**	55.	**Px3e(3f)**	**Gx5h(5i)**	56.	**G-3g(2h)**	**S-4h**
57.	**N-4d**	**K-3a(3b)**	58.	**B-8e**	**Px4d(4c)**	59.	**Bx5h(8e)**	**S-5g+(4h)**	60.	**G-5i**	**Px3e(3d)**
61.	**S-4c**	**N-2e**	62.	**G-2f(3g)**	**+Sx5h(5g)**	63.	**Gx5h(5i)**	**B-7f**	64.	**P-3b**	**Gx3b(4a)**
65.	**Sx3b(4c)**	**Kx3b(3a)**	66.	**G-5g**	**+Bx8i(5f)**	67.	**P-5d(5e)**	**S-3c(2b)**	68.	**S-6b**	**R-4a(5a)**
69.	**K-3i(3h)**	**N-3g+(2e)**	70.	**Nx3g(2i)**	**Lx2f(2d)**	71.	**Px2f(2g)**	**G-2g**	72.	**N-2e**	**Gx1h(2g)**
73.	**Lx1h(1i)**	**R-6i**	74.	**L-5i**	**P-3f(3e)**	75.	**Nx3c+(2e)**	**Nx3c(2a)**	76.	**G-3h**	**Px3g+(3f)**
77.	**Gx3g(3h)**	**P-3f**	78.	**Resign**							

TACOS (Black) v.s. GEKISASHI (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-2f(2g)**	**P-8d(8c)**	3.	**P-2e(2f)**	**P-8e(8d)**	4.	**G-7h(6i)**	**G-3b(4a)**
5.	**P-2d(2e)**	**Px2d(2c)**	6.	**Rx2d(2h)**	**P-8f(8e)**	7.	**Px8f(8g)**	**Rx8f(8b)**	8.	**R-2f(2d)**	**R-8b(8f)**
9.	**P-8g**	**P-2c**	10.	**P-1f(1g)**	**P-1d(1c)**	11.	**S-3h(3i)**	**S-6b(7a)**	12.	**P-3f(3g)**	**P-5d(5c)**
13.	**P-4f(4g)**	**P-4d(4c)**	14.	**S-6h(7i)**	**P-4e(4d)**	15.	**Px4e(4f)**	**Bx8h+(2b)**	16.	**Gx8h(7h)**	**B-5e**
17.	**B-6f**	**Bx1i+(5e)**	18.	**Bx1a+(6f)**	**N-3c(2a)**	19.	**P-2d**	**L-2e**	20.	**Px2c+(2d)**	**Gx2c(3b)**
21.	**L-2d**	**Lx2f(2e)**	22.	**Lx2c+(2d)**	**Nx4e(3c)**	23.	**+B-3c(1a)**	**P-4b**	24.	**G-2a**	**S-5c(6b)**
25.	**P-2g**	**Lx2g+(2f)**	26.	**Sx2g(3h)**	**+Bx2i(1i)**	27.	**Gx3a(2a)**	**N-3g**	28.	**G-4h(4i)**	**+B-4g(2i)**
29.	**G-4a(3a)**	**K-6b(5a)**	30.	**S-5h**	**R-3i**	31.	**P-4i**	**Nx4i+(3g)**	32.	**K-6i(5i)**	**+Nx4h(4i)**
33.	**K-7h(6i)**	**+Bx5h(4g)**	34.	**G-9h(8h)**	**R-6i+(3i)**	35.	**K-7g(7h)**	**+Rx6h(6i)**	36.	**K-6f(7g)**	**G-5e**
37.	**+Bx5e(3c)**	**+Bx5g(5h)**	38.	**K-6e(6f)**	**S-7d**	39.	**Resign**				

KCC (Black) v.s. TACOS (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-2f(2g)**	**P-4d(4c)**	3.	**S-4h(3i)**	**S-4b(3a)**	4.	**P-5f(5g)**	**P-5d(5c)**
5.	**G-5h(4i)**	**S-6b(7a)**	6.	**S-6h(7i)**	**G-5b(6a)**	7.	**G-7h(6i)**	**S-3c(4b)**	8.	**K-6i(5i)**	**G-4c(5b)**
9.	**P-3f(3g)**	**P-7d(7c)**	10.	**P-4f(4g)**	**G-3b(4a)**	11.	**P-2e(2f)**	**P-8d(8c)**	12.	**N-3g(2i)**	**K-4a(5a)**
13.	**P-4e(4f)**	**Px4e(4d)**	14.	**Nx4e(3g)**	**S-4b(3c)**	15.	**Bx2b+(8h)**	**Gx2b(3b)**	16.	**P-2d(2e)**	**Px2d(2c)**
17.	**P-2e**	**Px2e(2d)**	18.	**Rx2e(2h)**	**P-2c**	19.	**B-4f**	**S-7c(6b)**	20.	**P-2d**	**Px2d(2c)**
21.	**Bx2d(4f)**	**P-2c**	22.	**P-4d**	**Gx4d(4c)**	23.	**Bx4b+(2d)**	**Kx4b(4a)**	24.	**S-5c**	**K-4c(4b)**
25.	**L-9h(9i)**	**B-4f**	26.	**Sx4d+(5c)**	**Kx4d(4c)**	27.	**R-2i(2e)**	**S-3h**	28.	**S-5g(6h)**	**B-6d(4f)**
29.	**G-6e**	**Sx2i(3h)**	30.	**Gx6d(6e)**	**Kx4e(4d)**	31.	**Gx7c(6d)**	**Nx7c(8a)**	32.	**B-7a**	**R-5b(8b)**
33.	**S-5c**	**R-5a(5b)**	34.	**G-4g(5h)**	**R-4i**	35.	**K-5h(6i)**	**Rx5c(5a)**	36.	**Kx4i(5h)**	**K-4d(4e)**
37.	**R-4f**	**K-3c(4d)**	38.	**Bx5c+(7a)**	**B-3h**	39.	**K-5h(4i)**	**K-2d(3c)**	40.	**R-4b+(4f)**	**S-4i**
41.	**K-6h(5h)**	**G-3b**	42.	**R-2f**	**K-1d(2d)**	43.	**+Rx3b(4b)**	**S-5h+(4i)**	44.	**K-7g(6h)**	**N-2e**
45.	**+Rx3d(3b)**	**P-2d(2c)**	46.	**+Rx2d(3d)**	**Kx2d(1d)**	47.	**Rx2e(2f)**	**Kx2e(2d)**	48.	**+B-3e(5c)**	**K-1d(2e)**
49.	**G-2d**	**K-1e(1d)**	50.	**G-2e(2d)**	**Resign**						

TACOS (Black) v.s. YSS (White)

1.	**P-7f(7g)**	**P-8d(8c)**	2.	**S-6h(7i)**	**P-3d(3c)**	3.	**S-7g(6h)**	**S-6b(7a)**	4.	**S-4h(3i)**	**S-4b(3a)**
5.	**P-5f(5g)**	**G-3b(4a)**	6.	**G-5h(4i)**	**P-5d(5c)**	7.	**P-6f(6g)**	**K-4a(5a)**	8.	**G-6g(5h)**	**G-5b(6a)**
9.	**B-7i(8h)**	**S-3c(4b)**	10.	**P-3f(3g)**	**B-3a(2b)**	11.	**K-6h(5i)**	**P-4d(4c)**	12.	**K-7h(6h)**	**P-8e(8d)**
13.	**S-3g(4h)**	**B-4b(3a)**	14.	**K-8h(7h)**	**K-3a(4a)**	15.	**P-2f(2g)**	**K-2b(3a)**	16.	**G-7h(6i)**	**G-4c(5b)**
17.	**S-4f(3g)**	**P-9d(9c)**	18.	**P-1f(1g)**	**P-9e(9d)**	19.	**N-3g(2i)**	**S-5c(6b)**	20.	**N-2e(3g)**	**S-2d(3c)**
21.	**P-1e(1f)**	**P-7d(7c)**	22.	**P-3e(3f)**	**P-4e(4d)**	23.	**Sx4e(4f)**	**Px3e(3d)**	24.	**B-4f(7i)**	**P-6d(6c)**
25.	**P-5e(5f)**	**Px5e(5d)**	26.	**Bx5e(4f)**	**P-4d**	27.	**P-5d**	**Sx5d(5c)**	28.	**Bx4d(5e)**	**Gx4d(4c)**
29.	**Sx4d(4e)**	**P-4c**	30.	**P-3c**	**Px4d(4c)**	31.	**Px3b+(3c)**	**Kx3b(2b)**	32.	**G-3d**	**S-4c**
33.	**Gx2d(3d)**	**Px2d(2c)**	34.	**Nx1c+(2e)**	**Lx1c(1a)**	35.	**P-1d(1e)**	**Lx1d(1c)**	36.	**Lx1d(1i)**	**N-5e**
37.	**L-1b+(1d)**	**Nx6g+(5e)**	38.	**G-2b**	**K-3c(3b)**	39.	**Gx6g(7h)**	**P-8f(8e)**	40.	**Sx8f(7g)**	**B-3d**
41.	**P-5f**	**G-3a**	42.	**Gx3a(2b)**	**Bx1b(3d)**	43.	**P-3d**	**Kx3d(3c)**	44.	**N-4f**	**K-3c(3d)**
45.	**P-3d**	**Bx3d(1b)**	46.	**Gx2a(3a)**	**B-4e(3d)**	47.	**S-2b**	**K-3b(3c)**	48.	**Nx5d(4f)**	**Bx5d(4e)**
49.	**L-5e**	**G-2c**	50.	**G-3a**	**Bx3a(4b)**	51.	**Sx3a+(2b)**	**K-3c(3b)**	52.	**Lx5d(5e)**	**Sx5d(4c)**
53.	**N-4f**	**S-4c(5d)**	54.	**B-5a**	**L-4b**	55.	**B-1a**	**N-2b**	56.	**Gx2b(2a)**	**Gx2b(2c)**

57.	N-1d	G-3b	58.	Nx2b+(1d)	Gx2b(3b)	59.	G-3b	K-2c(3c)	60.	Bx2b+(1a)	K-1d(2c)
61.	S-2e	Px2e(2d)	62.	R-1h(2h)	Resign						

Tacos (Black) v.s. Ryuu no Tamago (White)

1.	P-7f(7g)	P-3d(3c)	2.	P-2f(2g)	Bx8h+(2b)	3.	Sx8h(7i)	S-3b(3a)	4.	S-7g(8h)	P-8d(8c)
5.	B-4e	S-7b(7a)	6.	Bx3d(4e)	P-8e(8d)	7.	B-5f(3d)	R-8d(8b)	8.	S-4h(3i)	G-4b(4a)
9.	P-3f(3g)	P-5d(5c)	10.	P-6f(6g)	K-6b(5a)	11.	G-5h(4i)	P-5e(5d)	12.	B-6g(5f)	G-5b(6a)
13.	S-3g(4h)	G-5c(4b)	14.	S-4f(3g)	G-5d(5c)	15.	S-3e(4f)	G-5c(5d)	16.	S-3d(3e)	B-4a
17.	P-2e(2f)	P-5f(5e)	18.	Bx5f(6g)	G-4d(5c)	19.	S-4e(3d)	G-5e(4d)	20.	B-6g(5f)	K-7a(6b)
21.	S-3d(4e)	P-1d(1c)	22.	P-2d(2e)	Px2d(2c)	23.	Rx2d(2h)	P-3c	24.	R-2b+(2d)	Px3d(3c)
25.	+Rx1a(2b)	S-2h	26.	P-2b	Sx2i(2h)	27.	Px2a+(2b)	G-5a(5b)	28.	+P-3a(2a)	R-5d(8d)
29.	+Px4a(3a)	Sx4a(3b)	30.	N-4f	R-5b(5d)	31.	L-8d	N-8c	32.	Bx3d(6g)	S-4b(4a)
33.	P-5f(5g)	Gx4f(5e)	34.	Px4f(4g)	S-3h+(2i)	35.	K-6h(5i)	+S-4i(3h)	36.	B-2g	P-3h
37.	G-3b	K-8b(7a)	38.	Bx3h(2g)	+S-3i(4i)	39.	Gx4b(3b)	Gx4b(5a)	40.	B-4g(3h)	N-5a
41.	P-5e(5f)	P-7d(7c)	42.	B-2c+(3d)	+S-4i(3i)	43.	Lx8c+(8d)	Sx8c(7b)	44.	N-5d	G-6a
45.	Nx4b+(5d)	Rx4b(5b)	46.	+B-3c(2c)	R-7b(4b)	47.	S-4b	P-8f(8e)	48.	Sx5a+(4b)	L-3b
49.	+Bx4c(3c)	Gx5a(6a)	50.	+Rx5a(1a)	S-6b	51.	+R-6a(5a)	Px8g+(8f)	52.	P-8d	+Px7g(8g)
53.	Nx7g(8i)	S-9d	54.	Px8c+(8d)	Sx8c(9d)	55.	N-9e	N-5f	56.	Bx5f(4g)	Lx3f(3b)
57.	Nx8c+(9e)	Kx8c(8b)	58.	N-9e	K-8d(8c)	59.	G-8c	Kx9e(8d)	60.	G-8e	Resign

Bonanza (Black) v.s. Tacos (White)

1.	P-7f(7g)	P-3d(3c)	2.	P-2f(2g)	P-4d(4c)	3.	S-4h(3i)	S-4b(3a)	4.	P-5f(5g)	P-5d(5c)
5.	G-5h(4i)	S-6b(7a)	6.	S-6h(7i)	G-5b(6a)	7.	G-7h(6i)	S-3c(4b)	8.	K-6i(5i)	G-4c(5b)
9.	S-7g(6h)	B-3a(2b)	10.	P-3f(3g)	K-4b(5a)	11.	N-3g(2i)	P-8d(8c)	12.	B-7i(8h)	K-3b(4b)
13.	B-4f(7i)	S-2d(3c)	14.	K-7i(6i)	K-2b(3b)	15.	K-8h(7i)	G-3b(4a)	16.	Bx2d(4f)	Px2d(2c)
17.	S-5g(4h)	G-2c(3b)	18.	P-2e(2f)	Px2e(2d)	19.	Rx2e(2h)	P-2d	20.	R-2h(2e)	P-1d(1c)
21.	P-6f(6g)	S-5a(6b)	22.	L-9h(9i)	P-7d(7c)	23.	K-9i(8h)	N-7c(8a)	24.	P-4f(4g)	B-6d(3a)
25.	S-8h(7g)	P-9d(9c)	26.	G-7i(7h)	S-4b(5a)	27.	P-2e	Px2e(2d)	28.	P-7e(7f)	Bx7e(6d)
29.	Rx2e(2h)	P-2d	30.	Rx7e(2e)	Px7e(7d)	31.	B-4a	R-2h	32.	G-6h(5h)	S-3a(4b)
33.	P-7d	N-8e(7c)	34.	Bx6c+(4a)	R-2i+(2h)	35.	P-7c+(7d)	R-3b(8b)	36.	P-4e(4f)	+Rx1i(2i)
37.	Px4d(4e)	Gx4d(4c)	38.	+B-5c(6c)	L-4b	39.	P-2e	P-6g	40.	Px2d(2e)	Gx2d(2c)
41.	Gx6g(6h)	R-3c(3b)	42.	+B-6d(5c)	B-4i	43.	G-6h(6g)	B-7f+(4i)	44.	N-4e(3g)	+Rx7i(1i)
45.	Nx3c+(4e)	Nx3c(2a)	46.	Sx7i(8h)	N-9e	47.	S-8h	Nx8g+(9e)	48.	Sx8g(8h)	+Bx8g(7f)
49.	P-8h	G-6i	50.	R-2i	+Bx9h(8g)	51.	Kx9h(9i)	Nx9g+(8e)	52.	Nx9g(8i)	L-2f
53.	Rx6i(2i)	P-6g	54.	G-7h(6h)	P-9e(9d)	55.	R-4a	P-9f(9e)	56.	Rx9a+(4a)	Px9g+(9f)
57.	Kx9g(9h)	S-8e	58.	N-1f	P-9f	59.	K-8g(9g)	P-6h+(6g)	60.	Sx6h(7i)	P-6a
61.	Nx2d(1f)	P-9g+(9f)	62.	Kx9g(8g)	N-4e(3c)	63.	+Rx6a(9a)	P-9f	64.	K-8g(9g)	N-9e
65.	K-7g(8g)	K-2c(2b)	66.	B-4a	S-3b(3a)	67.	Bx3b+(4a)	Kx2d(2c)	68.	+R-2a(6a)	N-2c
69.	+Rx2c(2a)	K-1e(2d)	70.	L-1f	Resign						

Tacos (Black) v.s. Kakinoki Shogi (White)

1.	P-7f(7g)	P-3d(3c)	2.	P-2f(2g)	P-4d(4c)	3.	S-4h(3i)	S-4b(3a)	4.	P-5f(5g)	P-9d(9c)
5.	K-6h(5i)	S-4c(4b)	6.	P-9f(9g)	R-4b(8b)	7.	K-7h(6h)	K-6b(5a)	8.	G-5h(4i)	K-7b(6b)
9.	P-2e(2f)	B-3c(2b)	10.	S-6h(7i)	G-5b(4a)	11.	P-3f(3g)	K-8b(7b)	12.	S-5g(6h)	L-9b(9a)
13.	G-6h(6i)	K-9a(8b)	14.	P-1f(1g)	S-8b(7a)	15.	S-4f(5g)	R-3b(4b)	16.	S-5g(4h)	G-7a(6a)
17.	P-1e(1f)	G-4b(5b)	18.	P-3e(3f)	L-1b(1a)	19.	Px3d(3e)	Sx3d(4c)	20.	S-5e(4f)	G-4c(4b)
21.	P-4f(4g)	P-5d(5c)	22.	S-6f(5e)	P-6d(6c)	23.	P-5e(5f)	R-5b(3b)	24.	Px5d(5e)	Rx5d(5b)
25.	S-5e(6f)	R-5a(5d)	26.	Sx6d(5e)	P-6b	27.	S-5e(6d)	P-4e(4d)	28.	P-5f	P-5d
29.	S-6d(5e)	Bx8h+(3c)	30.	Kx8h(7h)	B-4b	31.	S-7e(6d)	Px4f(4e)	32.	Sx4f(5g)	B-3c(4b)
33.	S-6f(7e)	B-4d(3c)	34.	P-2d(2e)	Px2d(2c)	35.	Rx2d(2h)	B-3c(4d)	36.	R-2g(2d)	P-2d
37.	N-3g(2i)	P-8d(8c)	38.	S-7g(6f)	R-4a(5a)	39.	N-4e(3g)	B-4d(3c)	40.	Rx2d(2g)	N-3c(2a)
41.	B-5b	R-5a(4a)	42.	Bx4c+(5b)	Sx4c(3d)	43.	Nx3c+(4e)	Bx3c(4d)	44.	R-2c+(2d)	R-3a(5a)
45.	P-3b	Sx3b(4c)	46.	+Rx1b(2c)	B-2c	47.	+Rx1c(1b)	P-1b	48.	+Rx2c(1c)	Bx7g+(3c)
49.	Gx7g(6h)	Sx2c(3b)	50.	L-3e	R-4a(3a)	51.	P-4b	R-5a(4a)	52.	P-5e(5f)	R-2h
53.	G-6i	Px5e(5d)	54.	P-5b	R-1a(5a)	55.	L-3c+(3e)	S-4g	56.	G-6h(5h)	N-5f
57.	G-7h(6h)	P-4e	58.	Sx4e(4f)	S-5h+(4g)	59.	Gx5h(6i)	Rx5h+(2h)	60.	+Lx2c(3c)	N-6h+(5f)
61.	B-5d	G-6c	62.	Gx6h(7h)	+Rx6h(5h)	63.	G-7h(7g)	+R-2h(6h)	64.	B-3b+(5d)	G-5g
65.	B-4f	+R-4h(2h)	66.	Bx5g(4f)	+Rx5g(4h)	67.	N-7e	G-7b(7a)	68.	S-8c	Gx8c(7b)
69.	S-7b	S-7d	70.	+L-2b(2c)	R-7a(1a)	71.	Nx8c+(7e)	Sx8c(7d)	72.	Sx7a+(7b)	Sx7a(8b)
73.	R-2a	S-8b	74.	N-7e	B-7d	75.	P-5a+(5b)	S-7b(7a)	76.	+P-5b(5a)	+R-4h(5g)
77.	+P-5c(5b)	Gx5c(6c)	78.	Nx8c+(7e)	Bx8c(7d)	79.	R-5a+(2a)	N-6a	80.	S-5d(4e)	Gx5d(5c)
81.	+Bx5d(3b)	S-7a	82.	+Bx5e(5d)	P-5f	83.	+B-6f(5e)	P-5g+(5f)	84.	+Bx5g(6f)	+R-5i(4h)
85.	P-4a+(4b)	B-3h+(8c)	86.	+B-4f(5g)	+R-4h(5i)	87.	P-4g	+B-2h(3h)	88.	P-1d(1e)	+Bx4f(2h)
89.	Px4f(4g)	B-3c	90.	B-6f	Bx6f(3c)	91.	Px6f(6g)	B-3c	92.	B-1e	Bx1e(3c)
93.	Lx1e(1i)	B-3c	94.	+R-5b(5a)	Bx6f(3c)	95.	S-7g	Bx7g+(6f)	96.	Kx7g(8h)	N-8e
97.	K-6g(7g)	S-6c(7b)	98.	+Rx6a(5b)	P-5f	99.	G-5h	S-6f	00.	Kx6f(6g)	+Rx5h(4h)
01.	G-6h	+R-5i(5h)	02.	G-6i	G-6e	03.	Kx6e(6f)	S-7b(7a)	04.	N-8c	Sx8c(7b)

05.	Gx5i(6i)	P-5g+(5f)	06.	S-7a	Sx7a(8b)	07.	+Rx7a(6a)	S-7d(8c)	08.	K-5e(6e)	S-8b
09.	R-6a	Sx7a(8b)	10.	Rx7a+(6a)	P-5d	11.	K-4e(5e)	R-7b	12.	+Rx7b(7a)	Sx7b(6c)
13.	S-7a	R-8b	14.	Gx5g(6h)	S-6e(7d)	15.	K-3d(4e)	L-9c(9b)	16.	K-3c(3d)	K-9b(9a)
17.	+L-2a(2b)	N-6f	18.	G-6g(7h)	P-7d(7c)	19.	Sx8b+(7a)	Kx8b(9b)	20.	B-6d	S-7c
21.	Bx7c+(6d)	Sx7c(7b)	22.	G-8c	Kx8c(8b)	23.	B-6a	G-7b	24.	Bx7b+(6a)	Kx7b(8c)
25.	S-6a	K-7a(7b)	26.	G-7b	Resign						

B.6 The 11th CO (2006)

Bonanza (Black) v.s. Tacos (White)

1.	P-7f(7g)	P-3d(3c)	2.	P-6f(6g)	P-8d(8c)	3.	S-7h(7i)	S-6b(7a)	4.	R-6h(2h)	K-4b(5a)
5.	K-4h(5i)	K-3b(4b)	6.	K-3h(4h)	P-5d(5c)	7.	K-2h(3h)	G-5b(6a)	8.	L-1h(1i)	P-7d(7c)
9.	K-1i(2h)	S-4b(3a)	10.	S-2h(3i)	P-1d(1c)	11.	G-3i(4i)	S-5c(4b)	12.	P-3f(3g)	P-8e(8d)
13.	B-7g(8h)	R-7b(8b)	14.	P-6e(6f)	Bx7g+(2b)	15.	Sx7g(7h)	B-4e	16.	G-7h(6i)	Bx3f(4e)
17.	G-3h(3i)	P-5e(5d)	18.	S-6f(7g)	S-4d(5c)	19.	B-8h	P-5f(5e)	20.	Px5f(5g)	P-3e(3d)
21.	G-6g(7h)	S-5c(6b)	22.	G-5g(6g)	R-8b(7b)	23.	B-7g(8h)	P-5d	24.	G-4f(5g)	B-2e(3f)
25.	G-3i(3h)	P-9d(9c)	26.	R-3h(6h)	G-4b(4a)	27.	P-2f(2g)	B-3d(2e)	28.	S-5g(6f)	G-3c(4b)
29.	Gx3e(4f)	Sx3e(4d)	30.	Rx3e(3h)	S-4d(5c)	31.	R-3g(3e)	G-3e	32.	S-2g	P-2d(2c)
33.	P-4f(4g)	P-2e(2d)	34.	Px2e(2f)	Gx2e(3e)	35.	R-3h(3g)	N-7c(8a)	36.	N-3g(2i)	Nx6e(7c)
37.	Nx2e(3g)	Bx2e(3d)	38.	P-2b	Kx2b(3b)	39.	Bx4d(7g)	Px4d(4c)	40.	P-2c	Kx2c(2b)
41.	P-2d	K-3b(2c)	42.	G-2c	K-4b(3b)	43.	Gx3c(2c)	Nx3c(2a)	44.	P-2c+(2d)	P-3d
45.	S-6f(5g)	B-4g+(2e)	46.	Rx3d(3h)	G-4c(5b)	47.	+Px3c(2c)	Gx3c(4c)	48.	G-5c	K-3b(4b)
49.	S-4c	K-2b(3b)	50.	Rx3c+(3d)	Kx3c(2b)	51.	G-3d	K-2b(3c)	52.	P-2d	B-1b
53.	N-3e	Bx3d(1b)	54.	Sx3d(4c)	K-3a(2b)	55.	N-4c(3e)	K-2a(3a)	56.	P-2c+(2d)	G-4a
57.	Sx6e(6f)	N-3c	58.	Sx3c+(3d)	G-2i	59.	Gx2i(3i)	+Bx2i(4g)	60.	Kx2i(1i)	R-6i
61.	G-3i	L-1b(1a)	62.	B-3b	Gx3b(4a)	63.	+Sx3b(3c)	Rx3b(8b)	64.	N-3c	K-1a(2a)
65.	G-2a	Resign									

Tacos (Black) v.s. Bonanza (White)

1.	P-7f(7g)	P-3d(3c)	2.	P-2f(2g)	P-4d(4c)	3.	S-4h(3i)	R-4b(8b)	4.	K-6h(5i)	P-9d(9c)
5.	K-7h(6h)	S-7b(7a)	6.	P-5f(5g)	S-3b(3a)	7.	G-5h(4i)	G-5b(4a)	8.	P-9f(9g)	K-6b(5a)
9.	P-3f(3g)	K-7a(6b)	10.	S-6h(7i)	K-8b(7a)	11.	S-5g(6h)	S-4c(3b)	12.	P-2e(2f)	B-3c(2b)
13.	G-6h(6i)	P-5d(5c)	14.	P-4f(4g)	P-6d(6c)	15.	P-4e(4f)	G-6c(5b)	16.	N-3g(2i)	P-7d(7c)
17.	P-2d(2e)	Bx2d(3c)	18.	Px4d(4e)	Sx4d(4c)	19.	P-4e	S-3c(4d)	20.	S-4g(4h)	P-1d(1c)
21.	S-4f(5g)	L-1b(1a)	22.	P-1f(1g)	N-7c(8a)	23.	R-2f(2h)	P-8d(8c)	24.	B-6f(8h)	S-8c(7b)
25.	P-1e(1f)	Px1e(1d)	26.	P-3e(3f)	Px3e(3d)	27.	P-5e(5f)	Px5e(5d)	28.	Sx5e(4f)	P-1f(1e)
29.	P-2e	B-1e(2d)	30.	R-2g(2f)	S-3d(3c)	31.	Lx1f(1i)	B-3c(1e)	32.	P-1c	Lx1c(1b)
33.	Lx1c+(1f)	P-5g	34.	Gx5g(5h)	Nx1c(2a)	35.	S-4d(5e)	Bx4d(3c)	36.	Bx4d(6f)	L-5a
37.	L-5f	P-5e	38.	Lx5e(5f)	Lx5e(5a)	39.	Bx5e(4d)	L-5a	40.	B-3c+(5e)	Lx5g+(5a)
41.	Gx5g(6h)	R-5b(4b)	42.	L-5f	P-5e	43.	Lx5e(5f)	P-5d	44.	+Bx3d(3c)	N-6e(7c)
45.	G-5h(5g)	Px5e(5d)	46.	B-4c	G-5c(6c)	47.	Bx5b+(4c)	Gx5b(5c)	48.	R-3c	L-7g
49.	Nx7g(8i)	Nx7g+(6e)	50.	Kx7g(7h)	N-6e	51.	K-6h(7g)	B-7g	52.	K-6i(6h)	N-5g(6e)
53.	K-7h(6i)	Bx9i+(7g)	54.	S-7c	K-9b(8b)	55.	K-6h(7h)	L-5f	56.	Gx5g(5h)	S-4h
57.	+Bx3e(3d)	P-3f	58.	N-8e	Px8e(8d)	59.	L-8d	Sx8d(8c)	60.	Sx8d(7c)	Lx5g+(5f)
61.	+Bx5g(3e)	Sx5g+(4h)	62.	Kx5g(6h)	P-5f(5e)	63.	Sx5f(4g)	B-7i	64.	L-6h	G-4f
65.	K-5h(5g)	P-5g	66.	K-4i(5h)	Resign						

Tacos (Black) v.s. Yss (White)

1.	P-7f(7g)	P-3d(3c)	2.	P-2f(2g)	P-4d(4c)	3.	S-4h(3i)	R-4b(8b)	4.	K-6h(5i)	K-6b(5a)
5.	K-7h(6h)	K-7b(6b)	6.	P-5f(5g)	S-3b(3a)	7.	G-5h(4i)	K-8b(7b)	8.	P-9f(9g)	P-9d(9c)
9.	P-3f(3g)	S-7b(7a)	10.	S-6h(7i)	S-4c(3b)	11.	S-5g(6h)	G-5b(4a)	12.	P-2e(2f)	B-3c(2b)
13.	G-6h(6i)	P-6d(6c)	14.	P-1f(1g)	P-7d(7c)	15.	S-4f(5g)	P-8d(8c)	16.	P-3e(3f)	G-6c(5b)
17.	R-3h(2h)	Px3e(3d)	18.	Sx3e(4f)	R-4a(4b)	19.	P-4f(4g)	G-5d(6c)	20.	S-5g(4h)	P-1d(1c)
21.	S-6f(5g)	P-6e(6d)	22.	S-5e(6f)	Gx5e(5d)	23.	Bx5e(8h)	N-7c(8a)	24.	S-3d(3e)	Sx3d(4c)
25.	G-3b	S-4i	26.	Rx3d(3h)	Sx5h+(4i)	27.	Gx5h(6h)	R-4c(4a)	28.	Gx2a(3b)	B-5a(3c)
29.	N-6d	S-6c(7b)	30.	R-3b+(3d)	R-4b(4c)	31.	S-8c	Kx8c(8b)	32.	+Rx4b(3b)	Bx4b(5a)
33.	R-8a	S-8b	34.	Rx6a+(8a)	R-2h	35.	S-6i	S-5d	36.	G-6b	G-5a
37.	Gx5a(6b)	Sx5e(5d)	38.	Px5e(5f)	B-2g	39.	G-5b(5a)	B-3f+(2g)	40.	Gx4b(5b)	P-6f(6e)
41.	S-7a	Px6g+(6f)	42.	Kx6g(7h)	+Bx5h(3f)	43.	Sx5h(6i)	P-6f	44.	Kx6f(6g)	P-6e
45.	K-7g(6f)	Sx7a(8b)	46.	+Rx7a(6a)	S-6f	47.	K-8f(7g)	G-8e	48.	K-9g(8f)	Gx9f(8e)
49.	K-8h(9g)	Rx5h+(2h)	50.	G-7h	P-7e(7d)	51.	B-8b	S-7g	52.	Nx7g(8i)	Sx7g+(6f)
53.	Kx7g(8h)	Px7f(7e)	54.	K-8h(7g)	P-7g+(7f)	55.	Kx7g(8h)	N-8e	56.	K-8h(7g)	Sx6d(6c)
57.	P-7f	+Rx7h(5h)	58.	Kx7h(8h)	Gx8g(9f)	59.	K-6i(7h)	N-7g(8e)	60.	K-5h(6i)	N-6f
61.	K-5g(5h)	G-7b	62.	R-9c	Lx9c(9a)	63.	Bx7c+(8b)	Gx7c(7b)	64.	S-7b	Gx7b(7c)
65.	S-7d	Kx7d(8c)	66.	+Rx7b(7a)	Resign						

YSS (Black) v.s. TACOS (White)

1. P-7f(7g)	P-3d(3c)	2. P-7e(7f)	K-4b(5a)	3. Bx2b+(8h)	Sx2b(3a)	4. G-7h(6i)	P-8d(8c)
5. S-4h(3i)	S-6b(7a)	6. K-6i(5i)	S-3c(2b)	7. S-8h(7i)	G-5b(6a)	8. G-5h(4i)	K-3b(4b)
9. S-7g(8h)	K-2b(3b)	10. K-7i(6i)	P-8e(8d)	11. K-8h(7i)	B-6d	12. S-6f(7g)	P-8f(8e)
13. Px8f(8g)	Rx8f(8b)	14. P-8g	R-8d(8f)	15. P-2f(2g)	P-8f	16. Px8f(8g)	Rx8f(8d)
17. P-8g	R-8b(8f)	18. P-2e(2f)	P-4d(4c)	19. P-5f(5g)	G-3a(4a)	20. P-5e(5f)	G-4b(3a)
21. S-5g(4h)	P-5d(5c)	22. Px5d(5e)	P-1d(1c)	23. G-6h(5h)	G-4c(4b)	24. S-4f(5g)	B-3a(6d)
25. R-5h(2h)	P-6d(6c)	26. G-7g(7h)	S-6c(6b)	27. S-5e(4f)	P-7d(7c)	28. G-7f(7g)	Px7e(7d)
29. Sx7e(6f)	P-9d(9c)	30. P-1f(1g)	P-7d	31. S-6f(7e)	N-7c(8a)	32. R-5f(5h)	P-6e(6d)
33. S-5g(6f)	P-7e(7d)	34. G-7g(7f)	N-8e(7c)	35. G-7h(7g)	S-6d(6c)	36. P-8f(8g)	Sx5e(6d)
37. Rx5e(5f)	Nx9g+(8e)	38. Kx9g(8h)	P-9e(9d)	39. P-9d	Lx9d(9a)	40. P-2d(2e)	Sx2d(3c)
41. K-8h(9g)	Rx8f(8b)	42. S-8g	R-2f(8f)	43. P-2g	Rx2g+(2f)	44. B-6a	G-4b(5b)
45. N-1g(2i)	+R-2i(2g)	46. Bx9d+(6a)	+Rx1i(2i)	47. +Bx9e(9d)	S-6d	48. R-5f(5e)	L-5e
49. R-8f(5f)	Lx5g+(5e)	50. Gx5g(6h)	P-9h	51. Kx9h(8h)	P-9d	52. +Bx9d(9e)	+R-5i(1i)
53. G-4f(5g)	P-9g	54. K-8h(9h)	P-4e(4d)	55. Gx4e(4f)	N-3c(2a)	56. L-4d	Nx4e(3c)
57. Lx4c+(4d)	Gx4c(4b)	58. G-4a	L-7f	59. P-7g	Lx7g+(7f)	60. Gx7g(7h)	P-9h+(9g)
61. Sx9h(8g)	+R-6h(5i)	62. P-7h	P-8e	63. +Bx8e(9d)	P-8d	64. +B-6c(8e)	P-7f(7e)
65. Gx7f(7g)	S-7e(6d)	66. Gx3a(4a)	Sx8f(7e)	67. Gx8f(7f)	P-7g	68. B-5e	Kx3a(2b)
69. Bx7g(5e)	S-7i	70. K-8g(8h)	+Rx6g(6h)	71. P-5c+(5d)	G-8h	72. K-9f(8g)	R-9c
73. P-9d	Rx6c(9c)	74. +Px4c(5c)	Rx4c(6c)	75. G-2b	K-4b(3a)	76. L-4d	Rx4d(4c)
77. Bx4d(7g)	P-5c	78. N-5d	K-4c(4b)	79. R-4b	Kx5d(4c)	80. P-5e	K-6d(5d)
81. S-7e	K-6c(6d)	82. P-6d	K-7c(6c)	83. L-7d	K-8c(7c)	84. R-7b+(4b)	Kx9d(8c)
85. P-9e	K-9c(9d)	86. P-9d(9e)	Kx9d(9c)	87. P-9e	K-9c(9d)	88. P-9d(9e)	Kx9d(9c)
89. P-9e	K-9c(9d)	90. P-9d(9e)	Kx9d(9c)	91. G-9e(8f)	K-9c(9d)	92. Sx8d(7e)	Resign

Playoff TACOS (Black) v.s. YSS (White)

1. P-7f(7g)	P-8d(8c)	2. S-6h(7i)	P-3d(3c)	3. S-7g(6h)	S-6b(7a)	4. S-4h(3i)	P-5d(5c)
5. P-5f(5g)	S-4b(3a)	6. G-5h(4i)	G-3b(4a)	7. P-6f(6g)	K-4a(5a)	8. G-6g(5h)	G-5b(6a)
9. B-7i(8h)	P-4d(4c)	10. K-6h(5i)	S-3c(4b)	11. K-7h(6h)	B-3a(2b)	12. P-3f(3g)	P-8e(8d)
13. S-3g(4h)	B-4b(3a)	14. K-8h(7h)	K-3a(4a)	15. G-7h(6i)	K-2b(3a)	16. S-4f(3g)	G-4c(5b)
17. P-2f(2g)	P-9d(9c)	18. N-3g(2i)	P-9e(9d)	19. N-2e(3g)	S-2d(3c)	20. P-1f(1g)	S-5c(6b)
21. R-3h(2h)	P-1d(1c)	22. P-3e(3f)	P-4e(4d)	23. Sx4e(4f)	Px3e(3d)	24. Bx3e(7i)	Sx3e(2d)
25. Rx3e(3h)	P-3c	26. P-3d	Px3d(3c)	27. Sx3d(4e)	Gx3d(4c)	28. Rx3d(3e)	P-3c
29. R-3i(3d)	P-2d(2c)	30. G-4a	S-2c	31. Gx4b(4a)	Sx4b(5c)	32. B-7a	R-9b(8b)
33. S-8c	R-5b(9b)	34. B-8b+(7a)	Rx8b(5b)	35. Sx8b+(8c)	B-4h	36. R-3h(3i)	Bx2f+(4h)
37. Nx3c+(2e)	Sx3c(4b)	38. +Sx9a(8b)	B-4i	39. R-6h(3h)	G-5h	40. L-3i	P-3f
41. P-3d	Sx3d(3c)	42. R-4a	Gx6h(5h)	43. Gx6h(6g)	N-9c(8a)	44. P-3c	Nx3c(2a)
45. G-2a	K-1c(2b)	46. R-5a+(4a)	R-2i	47. G-3a(2a)	G-4c(3b)	48. G-2a(3a)	Rx3i+(2i)
49. +R-5b(5a)	P-4b	50. Gx1a(2a)	N-6d	51. +Rx6c(5b)	Nx7f(6d)	52. Sx7f(7g)	Bx7f+(4i)
53. L-7g	+B-4i(7f)	54. N-4f	P-6g	55. G-6i(6h)	+B-4h(4i)	56. G-7i(7h)	+B-5g(4h)
57. P-9f(9g)	Px9f(9e)	58. +R-5b(6c)	+Bx7i(5g)	59. Gx7i(6i)	+Rx7i(3i)	60. Kx7i(8h)	G-6h
61. K-8h(7i)	G-7h	62. K-9h(8h)	L-9g	63. Nx9g(8i)	S-8i	64. Resign	

Playoff BONANZA (Black) v.s. TACOS (White)

1. P-7f(7g)	P-3d(3c)	2. P-6f(6g)	P-8d(8c)	3. S-7h(7i)	S-6b(7a)	4. R-6h(2h)	K-4b(5a)
5. K-4h(5i)	K-3b(4b)	6. K-3h(4h)	P-5d(5c)	7. K-2h(3h)	G-5b(6a)	8. L-1h(1i)	P-7d(7c)
9. K-1i(2h)	S-4b(3a)	10. S-2h(3i)	P-1d(1c)	11. G-3i(4i)	S-5c(4b)	12. P-3f(3g)	P-8e(8d)
13. B-7g(8h)	R-7b(8b)	14. P-6e(6f)	Bx7g+(2b)	15. Sx7g(7h)	B-4e	16. G-7h(6i)	Bx3f(4e)
17. G-3h(3i)	P-3e(3d)	18. G-6g(7h)	S-4d(5c)	19. G-5f(6g)	S-5c(6b)	20. S-6f(7g)	P-9d(9c)
21. N-7g(8i)	G-4b(4a)	22. B-4f	R-9b(7b)	23. Nx8e(7g)	S-6b(5c)	24. P-7e(7f)	P-8d
25. P-3g	B-2e(3f)	26. N-9c+(8e)	Rx9c(9b)	27. B-8b+(4f)	Px7e(7d)	28. +Bx8a(8b)	P-7f(7e)
29. +B-8b(8a)	N-8e	30. N-4f	P-7g+(7f)	31. R-6i(6h)	+P-7f(7g)	32. Nx5d(4f)	+Px6f(7f)
33. Rx6f(6i)	G-4a(4b)	34. Nx6b+(5d)	Gx6b(5b)	35. R-7f(6f)	P-7b	36. +Bx7b(8b)	Gx7b(6b)
37. Rx7b+(7f)	S-5b	38. S-6a	B-8c	39. +Rx5b(7b)	Gx5b(4a)	40. Sx5b+(6a)	R-8h
41. G-4b	K-3c(3b)	42. G-3b	K-2d(3c)	43. P-2f(2g)	B-3d(2e)	44. S-2e	Bx2e(3d)
45. Px2e(2f)	Kx2e(2d)	46. Gx4c(4b)	Bx6e(8c)	47. Gx6e(5f)	N-2f	48. Gx4d(4c)	S-3c
49. S-1f	K-2d(2e)	50. P-2e	K-1c(2d)	51. B-3a	K-1b(1c)	52. Gx3c(4d)	Nx1h+(2f)
53. Kx1h(1i)	L-2b	54. Bx2b+(3a)	Resign				

B.7 Shogi Club 24 (Internet Shogi Server)

TACOS (Black) v.s. Anonymous Player (Rating:2196) (White)

1. P-7f(7g)	P-3d(3c)	2. P-2f(2g)	P-8d(8c)	3. P-2e(2f)	P-8e(8d)	4. G-7h(6i)	G-3b(4a)
5. P-2d(2e)	Px2d(2c)	6. Rx2d(2h)	P-8f(8e)	7. Px8f(8g)	Rx8f(8b)	8. R-2f(2d)	P-2c
9. P-8g	R-8d(8f)	10. S-4h(3i)	S-7b(7a)	11. R-2h(2f)	P-9d(9c)	12. P-5f(5g)	K-6b(5a)
13. P-9f(9g)	P-3e(3d)	14. S-5g(4h)	K-7a(6b)	15. Bx2b+(8h)	Sx2b(3a)	16. B-7e	R-5d(8d)

17. **P-4f(4g)**	**S-3c(2b)**	18. **G-5h(4i)**	**S-4b(3c)**	19. **B-6f(7e)**	**N-3c(2a)**	20. **K-6i(5i)**	**R-2d(5d)**
21. **P-2f**	**R-3d(2d)**	22. **S-6h(7i)**	**P-3f(3e)**	23. **Px3f(3g)**	**Rx3f(3d)**	24. **S-4h(5g)**	**R-3d(3f)**
25. **S-7g(6h)**	**P-5d(5c)**	26. **G-6h(7h)**	**S-5c(4b)**	27. **K-7h(6i)**	**S-6d(5c)**	28. **P-1f(1g)**	**K-8b(7a)**
29. **P-1e(1f)**	**P-4d(4c)**	30. **P-2e(2f)**	**P-4e(4d)**	31. **Px4e(4f)**	**P-5e(5d)**	32. **Px5e(5f)**	**Nx4e(3c)**
33. **P-3g**	**P-5f**	34. **P-5d(5e)**	**P-5g+(5f)**	35. **Gx5g(5h)**	**Nx5g+(4e)**	36. **Sx5g(4h)**	**Rx5d(3d)**
37. **Bx1a+(6f)**	**B-3i**	38. **R-5h(2h)**	**G-4g**	39. **P-4h**	**Gx5h(4g)**	40. **Gx5h(6h)**	**P-5f**
41. **S-6f(5g)**	**P-4g**	42. **Gx4g(5h)**	**P-4f**	43. **L-8e**	**K-7a(8b)**	44. **P-5e**	**R-5a(5d)**
45. **N-5d**	**Rx5d(5a)**	46. **Px5d(5e)**	**R-4a**	47. **R-1b**	**N-2b**	48. **+Bx2b(1a)**	**Gx2b(3b)**
49. **G-8b**	**Resign**						

Tacos (Black) v.s. Anonymous Player (Rating:2188) (White)

1. **P-7f(7g)**	**P-3d(3c)**	2. **P-2f(2g)**	**P-3e(3d)**	3. **P-5f(5g)**	**R-4b(8b)**	4. **S-4h(3i)**	**K-6b(5a)**
5. **K-6h(5i)**	**K-7b(6b)**	6. **K-7h(6h)**	**R-3b(4b)**	7. **P-2e(2f)**	**R-3d(3b)**	8. **Bx2b+(8h)**	**Sx2b(3a)**
9. **S-6h(7i)**	**K-8b(7b)**	10. **B-4e**	**R-6d(3d)**	11. **G-5h(4i)**	**S-7b(7a)**	12. **P-9f(9g)**	**P-9d(9c)**
13. **S-5g(4h)**	**G-3b(4a)**	14. **S-6f(5g)**	**S-3c(2b)**	15. **P-1f(1g)**	**P-1d(1c)**	16. **S-5g(6h)**	**S-3d(3c)**
17. **B-1h(4e)**	**P-1e(1d)**	18. **Px1e(1f)**	**Lx1e(1a)**	19. **P-1g**	**Lx1g+(1e)**	20. **Nx1g(2i)**	**P-1f**
21. **B-2i(1h)**	**Px1g+(1f)**	22. **Lx1g(1i)**	**P-3f(3e)**	23. **Px3f(3g)**	**B-3g**	24. **R-2g(2h)**	**B-1i+(3g)**
25. **P-4f(4g)**	**P-3h**	26. **S-7e(6f)**	**R-4d(6d)**	27. **Bx3h(2i)**	**+B-1h(1i)**	28. **L-6f**	**+Bx2g(1h)**
29. **Bx2g(3h)**	**R-2h**	30. **B-4i**	**Sx2e(3d)**	31. **L-1a+(1g)**	**N-3c(2a)**	32. **P-3e(3f)**	**P-3f**
33. **+L-2a(1a)**	**N-1e**	34. **P-2i**	**Rx2i+(2h)**	35. **B-3h(2g)**	**+R-1i(2i)**	36. **G-4g(5h)**	**R-2d(4d)**
37. **S-4h(5g)**	**S-2f(2e)**	38. **G-5i(6i)**	**N-2g+(1e)**	39. **Bx2g(3h)**	**Sx2g+(2f)**	40. **P-2e**	**Rx2e(2d)**
41. **Gx3f(4g)**	**B-3h**	42. **B-5h(4i)**	**R-1e(2e)**	43. **P-1f**	**Rx1f(1e)**	44. **P-3d(3e)**	**R-1h+(1f)**
45. **Px3c+(3d)**	**B-4g+(3h)**	46. **P-2h**	**+Rx2h(1h)**	47. **Sx4g(4h)**	**+Rx5i(1i)**	48. **+Px3b(3c)**	**+Rx5h(2h)**
49. **Sx5h(4g)**	**+Rx5h(5i)**	50. **N-6h**	**B-6i**	51. **K-7g(7h)**	**+Rx6h(5h)**	52. **K-8f(7g)**	**+R-8h(6h)**
53. **Resign**							

Anonymous Player (Rating:2361) (Black) v.s. Tacos (White)

1. **P-5f(5g)**	**P-3d(3c)**	2. **R-5h(2h)**	**P-5d(5c)**	3. **P-7f(7g)**	**S-6b(7a)**	4. **K-4h(5i)**	**K-4b(5a)**
5. **Bx2b+(8h)**	**Sx2b(3a)**	6. **S-7h(7i)**	**S-5c(6b)**	7. **K-3h(4h)**	**S-3c(2b)**	8. **P-6f(6g)**	**P-8d(8c)**
9. **P-5e(5f)**	**Px5e(5d)**	10. **Rx5e(5h)**	**S-4d(5c)**	11. **R-5i(5e)**	**K-3a(4b)**	12. **K-2h(3h)**	**P-8e(8d)**
13. **S-3h(3i)**	**K-2b(3a)**	14. **P-4f(4g)**	**P-8f(8e)**	15. **Px8f(8g)**	**P-8h**	16. **N-7g(8i)**	**P-8i+(8h)**
17. **Sx8i(7h)**	**B-6g**	18. **S-7h(8i)**	**Bx7f+(6g)**	19. **B-6e**	**+Bx6f(7f)**	20. **Bx4c+(6e)**	**G-5b(6a)**
21. **+B-6e(4c)**	**+Bx6e(6f)**	22. **Nx6e(7g)**	**B-7f**	23. **P-6f**	**P-6d(6c)**	24. **P-4e(4f)**	**Sx4e(4d)**
25. **P-7g**	**Bx4i+(7f)**	26. **Rx4i(5i)**	**P-4f**	27. **P-5c**	**G-5a(5b)**	28. **P-4c**	**Px6e(6d)**
29. **B-6c**	**S-5f(4e)**	30. **Rx4f(4i)**	**S-5g(5f)**	31. **R-5f(4f)**	**S-4h+(5g)**	32. **P-2f(2g)**	**G-3i**
33. **S-2g(3h)**	**N-3e**	34. **P-1f(1g)**	**Nx2g+(3e)**	35. **Bx2g+(6c)**	**S-3h**	36. **+B-5d(2g)**	**Gx2i(3i)**
37. **K-1g(2h)**	**Gx1i(2i)**	38. **N-2e**	**Px6f(6e)**	39. **Nx3c+(2e)**	**Nx3c(2a)**	40. **S-4d**	**L-3b**
41. **B-7f**	**N-6b**	42. **+B-5e(5d)**	**P-5d**	43. **+Bx6f(5e)**	**+S-4g(4h)**	44. **P-5b+(5c)**	**+Sx3g(4g)**
45. **Resign**							

Anonymous Player (Rating:2601) (Black) v.s. Tacos (White)

1. **P-7f(7g)**	**P-3d(3c)**	2. **P-6f(6g)**	**P-8d(8c)**	3. **S-6h(7i)**	**S-6b(7a)**	4. **P-5f(5g)**	**P-5d(5c)**
5. **S-4h(3i)**	**S-4b(3a)**	6. **G-5h(4i)**	**G-3b(4a)**	7. **G-7h(6i)**	**K-4a(5a)**	8. **K-6i(5i)**	**G-5b(6a)**
9. **S-7g(6h)**	**S-3c(4b)**	10. **B-7i(8h)**	**B-3a(2b)**	11. **P-3f(3g)**	**P-4d(4c)**	12. **G-6g(5h)**	**G-4c(5b)**
13. **S-3g(4h)**	**P-8e(8d)**	14. **P-3e(3f)**	**Px3e(3d)**	15. **Bx3e(7i)**	**P-4e(4d)**	16. **S-3f(3g)**	**B-6d(3a)**
17. **R-1h(2h)**	**S-4d(3c)**	18. **B-6h(3e)**	**P-7d(7c)**	19. **P-6e(6f)**	**B-7c(6d)**	20. **P-7e(7f)**	**Px7e(7d)**
21. **P-7d**	**B-8d(7c)**	22. **P-4f(4g)**	**Px4f(4e)**	23. **Bx4f(6h)**	**R-9b(8b)**	24. **P-4e**	**S-3c(4d)**
25. **R-3h(1h)**	**P-9d(9c)**	26. **S-6f(7g)**	**K-3a(4a)**	27. **B-5g(4f)**	**B-9c(8d)**	28. **Sx7e(6f)**	**B-8b(9c)**
29. **P-6d(6e)**	**Px6d(6c)**	30. **S-8d(7e)**	**P-6e(6d)**	31. **P-7c+(7d)**	**Bx7c(8b)**	32. **Sx7c+(8d)**	**Sx7c(6b)**
33. **B-4f(5g)**	**G-5c(4c)**	34. **N-7g(8i)**	**S-6d(7c)**	35. **B-8c**	**R-6b(9b)**	36. **B-3e(4f)**	**S-2d(3c)**
37. **B-2f(3e)**	**P-3e**	38. **Sx3e(3f)**	**S-4g**	39. **R-3g(3h)**	**Sx3e(2d)**	40. **Bx3e(2f)**	**S-5h**
41. **K-7i(6i)**	**S-4h(4g)**	42. **G-6h(6g)**	**Sx3g(4h)**	43. **Nx6e(7g)**	**R-5i**	44. **P-6i**	**P-6g**
45. **Gx5h(6h)**	**Rx5h+(5i)**	46. **S-4i**	**+Rx4i(5h)**	47. **Nx5c+(6e)**	**P-7g**	48. **Gx7g(7h)**	**S-6h**
49. **Resign**							

Anonymous Player (Rating:2424) (Black) v.s. Tacos (White)

1. **P-7f(7g)**	**P-3d(3c)**	2. **P-6f(6g)**	**P-8d(8c)**	3. **S-6h(7i)**	**S-6b(7a)**	4. **P-5f(5g)**	**P-5d(5c)**
5. **S-5g(6h)**	**G-5b(6a)**	6. **R-6h(2h)**	**P-8e(8d)**	7. **B-7g(8h)**	**K-4b(5a)**	8. **K-4h(5i)**	**K-3b(4b)**
9. **K-3h(4h)**	**P-1d(1c)**	10. **P-1f(1g)**	**S-4b(3a)**	11. **G-5h(6i)**	**P-7d(7c)**	12. **K-2h(3h)**	**R-7b(8b)**
13. **R-7h(6h)**	**P-7e(7d)**	14. **Px7e(7f)**	**Rx7e(7b)**	15. **G-6h(5h)**	**P-9d(9c)**	16. **S-3h(3i)**	**G-5a(5b)**
17. **B-8h(7g)**	**Rx7h+(7e)**	18. **Gx7h(6h)**	**R-6i**	19. **P-7i**	**B-1c(2b)**	20. **P-4f(4g)**	**P-5e(5d)**
21. **P-6e(6f)**	**Px5f(5e)**	22. **Sx5f(5g)**	**S-3c(4b)**	23. **P-6d(6e)**	**Bx4f(1c)**	24. **R-8c**	**B-5g+(4f)**
25. **S-6g(5f)**	**Rx4i+(6i)**	26. **Sx4i(3h)**	**G-3i**	27. **S-5h(4i)**	**+B-4h(5g)**	28. **Rx8a+(8c)**	**+B-3h(4h)**
29. **K-1g(2h)**	**Gx2i(3i)**	30. **K-2f(1g)**	**N-3e**	31. **K-3f(2f)**	**Gx1i(2i)**	32. **K-4f(3f)**	**Nx2g+(3e)**
33. **K-5g(4f)**	**+Nx3g(2g)**	34. **+Rx9a(8a)**	**L-5b**	35. **P-5f**	**+N-4h(3g)**	36. **Px6c+(6d)**	**+Nx5h(4h)**
37. **Kx5h(5g)**	**Sx6c(6b)**	38. **N-5e**	**P-7g**	39. **Bx7g(8h)**	**S-4g**	40. **K-6h(5h)**	**Sx5f+(4g)**

41. **Sx5f(6g)**	**+Bx5f(3h)**	42. **L-5i**	**+B-4f(5f)**	43. **S-5g**	**+Bx5e(4f)**	44. **Bx5e(7g)**	**Lx5e(5b)**
45. **P-5f**	**B-9e**	46. **P-8f(8g)**	**N-6e**	47. **S-6f(5g)**	**Bx8f(9e)**	48. **N-7g**	**Nx7g+(6e)**
49. **Gx7g(7h)**	**N-7e**	50. **Gx8f(7g)**	**S-6g**	51. **K-7g(6h)**	**Px8f(8e)**	52. **Kx8f(7g)**	**P-8d**
53. **R-8b**	**G-5b(4a)**	54. **Rx8d+(8b)**	**G-7d**	55. **+Rx9d(8d)**	**P-5h**	56. **Sx7e(6f)**	**P-8e**
57. **K-9e(8f)**	**Gx7e(7d)**	58. **K-8d(9e)**	**Px5i+(5h)**	59. **P-8c**	**Resign**		

Anonymous Player (Rating:2382) (Black) v.s. Tacos (White)

1. **P-7f(7g)**	**P-3d(3c)**	2. **P-6f(6g)**	**P-8d(8c)**	3. **P-1f(1g)**	**S-6b(7a)**	4. **B-7g(8h)**	**K-4b(5a)**
5. **R-8h(2h)**	**P-7d(7c)**	6. **S-6h(7i)**	**R-7b(8b)**	7. **S-6g(6h)**	**G-5b(6a)**	8. **P-8f(8g)**	**S-7c(6b)**
9. **K-4h(5i)**	**P-7e(7d)**	10. **Px7e(7f)**	**S-6d(7c)**	11. **P-8e(8f)**	**Sx7e(6d)**	12. **P-7c**	**Rx7c(7b)**
13. **Px8d(8e)**	**P-7f**	14. **B-6h(7g)**	**Sx6f(7e)**	15. **Sx6f(6g)**	**Bx6f(2b)**	16. **R-8e(8h)**	**P-7g+(7f)**
17. **Nx7g(8i)**	**Bx7g+(6f)**	18. **P-7d**	**Rx7d(7c)**	19. **S-7e**	**N-7c(8a)**	20. **Bx7g(6h)**	**Nx8e(7c)**
21. **B-6f(7g)**	**Rx7e(7d)**	22. **Bx7e(6f)**	**R-6e**	23. **B-6f**	**P-7d**	24. **P-6g**	**Px7e(7d)**
25. **Bx1a+(6f)**	**Rx6g+(6e)**	26. **G-5h(6i)**	**+R-7h(6g)**	27. **K-3h(4h)**	**N-4e**	28. **R-8h**	**+R-7i(7h)**
29. **Rx8e(8h)**	**Nx5g+(4e)**	30. **L-5i**	**+Nx5h(5g)**	31. **Gx5h(4i)**	**S-6g**	32. **Gx6g(5h)**	**+Rx5i(7i)**
33. **S-4i**	**S-5h**	34. **S-4h(4i)**	**B-4i**	35. **K-2h(3h)**	**G-3h**	36. **K-1g(2h)**	**Gx4h(3h)**
37. **Sx4h(3i)**	**L-2d**	38. **Resign**					

Tacos (Black) v.s. Anonymous Player (Rating:2285) (White)

1. **P-7f(7g)**	**P-3d(3c)**	2. **P-2f(2g)**	**P-4d(4c)**	3. **S-4h(3i)**	**R-4b(8b)**	4. **K-6h(5i)**	**K-6b(5a)**
5. **K-7h(6h)**	**S-7b(7a)**	6. **P-5f(5g)**	**K-7a(6b)**	7. **G-5h(4i)**	**G-5b(4a)**	8. **S-6h(7i)**	**S-3b(3a)**
9. **P-3f(3g)**	**S-4c(3b)**	10. **P-2e(2f)**	**B-3c(2b)**	11. **S-5g(6h)**	**K-8b(7a)**	12. **G-6h(6i)**	**P-9d(9c)**
13. **P-9f(9g)**	**L-1b(1a)**	14. **P-1f(1g)**	**P-1d(1c)**	15. **S-3g(4h)**	**P-5d(5c)**	16. **S-2f(3g)**	**R-3b(4b)**
17. **P-4f(4g)**	**B-4b(3c)**	18. **R-3h(2h)**	**B-5c(4b)**	19. **P-3e(3f)**	**G-4b(5b)**	20. **Px3d(3e)**	**Sx3d(4c)**
21. **P-3e**	**S-4c(3d)**	22. **R-3f(3h)**	**G-5b(4b)**	23. **P-1e(1f)**	**Px1e(1d)**	24. **Lx1e(1i)**	**Lx1e(1b)**
25. **Sx1e(2f)**	**P-4e(4d)**	26. **S-2f(1e)**	**S-4d(4c)**	27. **N-3g(2i)**	**Sx3e(4d)**	28. **Sx3e(2f)**	**Rx3e(3b)**
29. **Rx3e(3f)**	**Bx3e(5c)**	30. **Nx4e(3g)**	**R-2i**	31. **R-3a**	**B-1g+(3e)**	32. **Rx2a+(3a)**	**L-8d**
33. **N-6i**	**P-9e(9d)**	34. **Px9e(9f)**	**P-9h**	35. **Lx9h(9i)**	**P-9g**	36. **Lx9g(9h)**	**S-9h**
37. **S-9f**	**Lx9e(9a)**	38. **L-9i**	**Sx9i(9h)**	39. **Sx9e(9f)**	**Sx8h+(9i)**	40. **Kx8h(7h)**	**+B-4d(1g)**
41. **L-6f**	**L-8e**	42. **Sx8d(9e)**	**Px8d(8c)**	43. **+R-4a(2a)**	**B-9i**	44. **K-7h(8h)**	**S-9h**
45. **S-9f**	**P-9e**	46. **Sx8e(9f)**	**Px8e(8d)**	47. **L-8h**	**P-8f(8e)**	48. **L-8e**	**K-7a(8b)**
49. **Lx8a+(8e)**	**Kx8a(7a)**	50. **N-7i**	**L-8c**	51. **Px8f(8g)**	**P-8g**	52. **Nx8g(7i)**	**Lx8f(8c)**
53. **P-8b**	**Kx8b(8a)**	54. **P-8c**	**Sx8c(7b)**	55. **G-7g(6h)**	**Sx8i+(9h)**	56. **Kx8i(7h)**	**Rx6i+(2i)**
57. **S-7i**	**N-8e**	58. **+Rx5b(4a)**	**Gx5b(6a)**	59. **G-7h**	**S-9h**	60. **Kx9h(8i)**	**Nx9g+(8e)**
61. **Resign**							

Tacos (Black) v.s. Anonymous Player (Rating:2125) (White)

1. **P-7f(7g)**	**P-8d(8c)**	2. **S-6h(7i)**	**P-3d(3c)**	3. **P-6f(6g)**	**S-6b(7a)**	4. **P-5f(5g)**	**P-5d(5c)**
5. **S-4h(3i)**	**S-4b(3a)**	6. **G-5h(4i)**	**G-3b(4a)**	7. **G-7h(6i)**	**K-4a(5a)**	8. **K-6i(5i)**	**G-5b(6a)**
9. **S-7g(6h)**	**S-3c(4b)**	10. **B-7i(8h)**	**B-3a(2b)**	11. **P-3f(3g)**	**P-4d(4c)**	12. **G-6g(5h)**	**P-7d(7c)**
13. **B-6h(7i)**	**B-6d(3a)**	14. **N-3g(2i)**	**G-4c(5b)**	15. **K-7i(6i)**	**K-3a(4a)**	16. **K-8h(7i)**	**B-7c(6d)**
17. **P-4f(4g)**	**S-5c(6b)**	18. **S-4g(4h)**	**P-8e(8d)**	19. **P-1f(1g)**	**P-1d(1c)**	20. **P-2f(2g)**	**S-4b(5c)**
21. **P-9f(9g)**	**P-9d(9c)**	22. **S-5h(4g)**	**P-6d(6c)**	23. **R-4h(2h)**	**B-8d(7c)**	24. **R-4i(4h)**	**N-7c(8a)**
25. **N-2e(3g)**	**S-2d(3c)**	26. **P-4e(4f)**	**Px4e(4d)**	27. **Rx4e(4i)**	**P-4d**	28. **R-4i(4e)**	**K-2b(3a)**
29. **B-4f(6h)**	**S-5c(4b)**	30. **S-6i(5h)**	**R-8a(8b)**	31. **P-5e(5f)**	**Px5e(5d)**	32. **Bx5e(4f)**	**P-5d**
33. **B-4f(5e)**	**L-9b(9a)**	34. **B-6h(4f)**	**R-9a(8a)**	35. **G-7i(7h)**	**P-9e(9d)**	36. **Px9e(9f)**	**Lx9e(9b)**
37. **P-9g**	**R-9b(9a)**	38. **P-9f(9g)**	**Lx9f(9e)**	39. **Lx9f(9i)**	**P-9e**	40. **Lx9e(9f)**	**Rx9e(9b)**
41. **P-7e(7f)**	**P-9g**	42. **Nx9g(8i)**	**Bx7e(8d)**	43. **P-9f**	**Rx9f(9e)**	44. **L-9h**	**R-9a(9f)**
45. **S-7f(7g)**	**B-8d(7e)**	46. **P-9b**	**Rx9b(9a)**	47. **P-9c**	**R-4b(9b)**	48. **Nx8e(9g)**	**Nx8e(7c)**
49. **Sx8e(7f)**	**B-5a(8d)**	50. **S-7h(6i)**	**P-7e(7d)**	51. **P-7f**	**Px7f(7e)**	52. **Sx7f(8e)**	**P-9g**
53. **Lx9g(9h)**	**P-7g**	54. **Bx7g(6h)**	**P-7e**	55. **Sx7e(7f)**	**N-8e**	56. **B-6h(7g)**	**Nx9g+(8e)**
57. **Kx9g(8h)**	**L-9d**	58. **K-8h(9g)**	**P-9f**	59. **N-8i**	**L-9e**	60. **K-7g(8h)**	**P-9g+(9f)**
61. **K-7f(7g)**	**+P-9h(9g)**	62. **N-6c**	**B-6b(5a)**	63. **K-8e(7f)**	**+P-8h(9h)**	64. **Gx8h(7i)**	**L-9h+(9e)**
65. **G-7g(8h)**	**+L-8h(9h)**	66. **Kx9d(8e)**	**P-6e(6d)**	67. **Px6e(6f)**	**B-7c(6b)**	68. **P-4f**	**R-6b(4b)**
69. **S-7d(7e)**	**B-5e(7c)**	70. **P-5f**	**+Lx7h(8h)**	71. **Gx7h(7g)**	**B-9i+(5e)**	72. **S-7c+(7d)**	**R-4b(6b)**
73. **N-7a+(6c)**	**S-5h**	74. **R-2i(4i)**	**Sx6g+(5h)**	75. **Gx6g(7h)**	**+B-8h(9i)**	76. **+S-7b(7c)**	**+B-7h(8h)**
77. **S-7f**	**G-7e**	78. **B-9e(6h)**	**Gx7f(7e)**	79. **Gx7f(6g)**	**+B-6g(7h)**	80. **G-6f**	**+B-5g(6g)**
81. **P-9b+(9c)**	**S-6g**	82. **B-5a+(9e)**	**+B-5h(5g)**	83. **G-7g(7f)**	**Sx5f+(6g)**	84. **K-8c(9d)**	**+B-4g(5h)**
85. **R-5i(2i)**	**+Sx4f(5f)**	86. **L-3i**	**+B-4h(4g)**	87. **R-7i(5i)**	**+B-5g(4h)**	88. **R-4i(7i)**	**+S-4g(4f)**
89. **G-6g(7g)**	**+B-5h(5g)**	90. **R-7i(4i)**	**+S-5g(4g)**	91. **R-7g(7i)**	**+Sx6g(5g)**	92. **Gx6g(6f)**	**P-7f**
93. **Gx7f(6g)**	**G-6g**	94. **+P-9a(9b)**	**Gx7g(6g)**	95. **Gx7g(7f)**	**R-7i**	96. **P-5i**	**Rx5i+(7i)**
97. **S-6g**	**+B-4i(5h)**	98. **G-4a**	**+Bx3i(4i)**	99. **P-9c**	**+Rx8i(5i)**	00. **Gx4b(4a)**	**Sx4b(5c)**
01. **+B-4a(5a)**	**+R-8h(8i)**	02. **S-6f(6g)**	**P-7f**	03. **+Bx3b(4a)**	**Kx3b(2b)**	04. **Gx7f(7g)**	**+Rx8g(8h)**
05. **G-8f**	**+R-9g(8g)**	06. **P-9b+(9c)**	**N-7d**	07. **Kx7d(8c)**	**+R-9d(9g)**	08. **L-8d**	**G-8c**
09. **K-6c(7d)**	**+Rx8d(9d)**	10. **+S-6b(7b)**	**P-6a**	11. **K-7b(6c)**	**Px6b(6a)**	12. **K-8a(7b)**	**B-6c**
13. **R-7b**	**S-6a**	14. **+Nx6a(7a)**	**G-7c(8c)**	15. **S-8b**	**Gx7b(7c)**	16. **Resign**	

Tacos (Black) v.s. Anonymous Player (Rating:2338) (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-2f(2g)**	**P-5d(5c)**	3.	**P-2e(2f)**	**R-5b(8b)**	4.	**G-5h(4i)**	**K-6b(5a)**
5.	**S-4h(3i)**	**K-7b(6b)**	6.	**K-6h(5i)**	**K-8b(7b)**	7.	**K-7h(6h)**	**S-7b(7a)**	8.	**P-4f(4g)**	**G-3b(4a)**
9.	**S-4g(4h)**	**P-5e(5d)**	10.	**P-3f(3g)**	**R-5a(5b)**	11.	**P-4e(4f)**	**S-4b(3a)**	12.	**S-4f(4g)**	**P-5f(5e)**
13.	**Bx2b+(8h)**	**Gx2b(3b)**	14.	**Px5f(5g)**	**Rx5f(5a)**	15.	**G-4g(5h)**	**Rx7f(5f)**	16.	**P-7g**	**R-7d(7f)**
17.	**S-5e(4f)**	**G-3b(2b)**	18.	**P-2d(2e)**	**Px2d(2c)**	19.	**Rx2d(2h)**	**P-2c**	20.	**R-2f(2d)**	**R-7e(7d)**
21.	**P-5f**	**P-9d(9c)**	22.	**N-3g(2i)**	**B-3h**	23.	**B-5h**	**Rx5e(7e)**	24.	**Px5e(5f)**	**P-5g**
25.	**Gx5g(4g)**	**P-5f**	26.	**G-4g(5g)**	**S-5g**	27.	**P-3e(3f)**	**Px3e(3d)**	28.	**P-5d(5e)**	**Sx5h(5g)**
29.	**Gx5h(6i)**	**Bx4g+(3h)**	30.	**Gx4g(5h)**	**G-3f**	31.	**Gx3f(4g)**	**Px3f(3e)**	32.	**Rx3f(2f)**	**P-5g+(5f)**
33.	**Rx3b+(3f)**	**B-4f**	34.	**R-2i**	**G-6h**	35.	**Sx6h(7i)**	**+Px6h(5g)**	36.	**K-8h(7h)**	**S-7i**
37.	**K-9h(8h)**	**+P-7h(6h)**	38.	**P-9f(9g)**	**P-9e(9d)**	39.	**K-9g(9h)**	**+Px8i(7h)**	40.	**Px9e(9f)**	**S-8h(7i)**
41.	**K-8f(9g)**	**+Px9i(8i)**	42.	**Rx2c+(2i)**	**B-6h+(4f)**	43.	**B-5e**	**L-7d**	44.	**S-6f**	**+Bx6g(6h)**
45.	**P-6h**	**+Bx6h(6g)**	46.	**+Rx2a(2c)**	**+P-9h(9i)**	47.	**N-9d**	**K-9c(8b)**	48.	**G-7e**	**Lx7e(7d)**
49.	**Sx7e(6f)**	**G-6e**	50.	**G-8d**	**Px8d(8c)**	51.	**Sx8d(7e)**	**Kx8d(9c)**	52.	**L-8e**	**K-9c(8d)**
53.	**G-8d**	**K-9b(9c)**	54.	**G-8c(8d)**	**Sx8c(7b)**	55.	**+Rx4b(3b)**	**Resign**			

Tacos (Black) v.s. Anonymous Player (Rating:1383) (White)

1.	**P-7f(7g)**	**P-3d(3c)**	2.	**P-2f(2g)**	**Bx8h+(2b)**	3.	**Sx8h(7i)**	**S-3b(3a)**	4.	**S-7g(8h)**	**K-4b(5a)**
5.	**S-3h(3i)**	**S-6b(7a)**	6.	**K-6h(5i)**	**G-5b(6a)**	7.	**P-4f(4g)**	**P-4d(4c)**	8.	**K-7h(6h)**	**G-4c(5b)**
9.	**S-4g(3h)**	**P-6d(6c)**	10.	**K-8h(7h)**	**S-6c(6b)**	11.	**P-2e(2f)**	**K-3a(4b)**	12.	**P-2d(2e)**	**Px2d(2c)**
13.	**Rx2d(2h)**	**P-2c**	14.	**R-2h(2d)**	**S-5d(6c)**	15.	**G-5h(4i)**	**P-6e(6d)**	16.	**G-7h(6i)**	**P-9d(9c)**
17.	**P-3f(3g)**	**R-6b(8b)**	18.	**P-9f(9g)**	**P-7d(7c)**	19.	**N-3g(2i)**	**K-2b(3a)**	20.	**S-5f(4g)**	**B-7c**
21.	**G-4g(5h)**	**P-1d(1c)**	22.	**R-2e(2h)**	**N-3c(2a)**	23.	**R-2f(2e)**	**S-5e(5d)**	24.	**Sx5e(5f)**	**Bx5e(7c)**
25.	**N-2e(3g)**	**P-7e(7d)**	26.	**Nx3c+(2e)**	**Sx3c(3b)**	27.	**P-5f(5g)**	**Bx7g+(5e)**	28.	**Gx7g(7h)**	**Px7f(7e)**
29.	**Gx7f(7g)**	**P-9e(9d)**	30.	**Px9e(9f)**	**N-8d**	31.	**G-8f(7f)**	**P-6f(6e)**	32.	**B-8e**	**G-3b(4a)**
33.	**N-5e**	**G-4b(4c)**	34.	**N-6c+(5e)**	**Rx6c(6b)**	35.	**Bx6c+(8e)**	**Px6g+(6f)**	36.	**R-7a**	**Lx9e(9a)**
37.	**Lx9e(9i)**	**P-9f**	38.	**R-2h(2f)**	**P-7g**	39.	**P-7i**	**N-9g**	40.	**Nx9g(8i)**	**S-9i**
41.	**Kx9i(8h)**	**Px9g+(9f)**	42.	**S-1c**	**Lx1c(1a)**	43.	**S-1a**	**K-1b(2b)**	44.	**B-2a**	**Resign**

Bibliography

[1] C. Shannon: "Programming a Computer for Playing Chess", Philosophical Magazine, Ser.7, Vol.41, No.314, 1950.

[2] R.D. Greenblatt, D.E. Eastlake and S.D. Crocker: "The Greenblatt Chess Program", Proceedings of Fall Joint Computer Conference, pp.801-810, 1967.

[3] A.L. Zobrist: "A new hashing method for game playing", Reprinted in ICCA Journal, pp.69-73, Vol.13, No.2, 1990.

[4] R.S. Sutton: "Learning to predict by the methods of temporal differences", Machine Learning, pp.9-44, Vol.3. No.1, 1988.

[5] M. Buro: "ProbCut: An effective selective extension of the alpha-beta algorithm", ICCA Journal, pp.71-76, Vol.18. No.2, 1995.

[6] R.E. Korf: "Macro-operators: a weak method for learning", Artificial Intelligence, pp.35-77, Vol.26, Issue 1, 1985.

[7] J. Schaeffer and A. Plaat: "Kasparov Versus Deep Blue: The Rematch", ICCA Journal, pp.95-101, Vol.20, No.2, 1997.

[8] H. Matsubara, H. lida, and R. Grimbergen: "Natural Developments in game research", ICCA Journal, pp.103-112, Vol.19, No.2, 1996.

[9] M.H.M. Winands: "Analysis and Implementation of Lines of Action", M.Sc. Thesis, Universiteit Maastricht, The Netherlands, 2000.

[10] H. Matsubara, and T. Takizawa: "How Shogi Programs Become Such Strong As Amateur 4-dan", Journal of the Japanese Society for Artificial Intelligence, pp.379-384, Vol.16, No.3, 2001. (in Japanese).

[11] H. Iida, M. Sakuta, and J. Rollason: "Computer Shogi", Artificial Intelligence, pp.121-144, Vol.134, 2002.

[12] L.V. Allis, M. van der Meulen and H.J. van den Herik: "Proof-number search", Artificial Intelligence, pp.91-124, Vol.66, 1994.

[13] R. Grimbergen: "A survey of Tsume-Shogi programs using variable-depth search", in: H.J. van den Herik, H.Iida (Eds.), Proceedings of First International Conference on Computers and Games, CG'98, Lecture Notes in Computer Science, pp.300-317, Vol.1558, Springer, Heidelberg, 1999.

[14] M. Seo, H. Iida and J.W.H.M. Uiterwijk: "The PN-search algorithm: Application to Tsume-Shogi", Artificial Intelligence, pp.253-277, Vol.129 (1-2), 2001.

[15] A. Nagai: "A new AND/OR tree search algorithm using proof number and disproof number", in: Proceedings of Complex Games Lab Workshop, pp.40-45, ETL, Tsukuba, Japan, 1998.

[16] A. Nagai: "A new depth-first-search algorithm for AND/OR trees", M.Sc. Thesis, Department of Information Science, University of Tokyo, Japan, 1999.

[17] A. Nagai and H. Imai: "Application of df-pn+ to Othello endgames", The 5th Game Programming Workshop (GPW 1999), pp.16-23, 1999.

[18] A. Nagai: "The recent achievements of computer Tsume-Shogi: Challenges in solving problems with extremely long steps", Report of Computer Shogi Association, pp.34-42, vol.13, 2000. (In Japanese)

[19] G. Tesauro: "Temporal Difference Learning and TD-Gammon", Communications of the ACM, pp.58-68, Vol.38, No.3, 1995.

[20] J. Schaeffer, R. Lake, P. Lu and M. Bryant: "Chinook: The Man-Machine World Checkers Champion", AI Magazine, pp.21-29, Vol.17, No.1, 1996.

[21] M. Campbell: "Knowledge Discovery in Deep Blue", Communications of the ACM, pp.65-67, Vol.42, No.11, 1999.

[22] M. Campbell, A.J. Hoane Jr. and F. Hsu: "Deep Blue", Artificial Intelligence, pp.57-83, Vol.134, 2002.

[23] M. Buro: "Improving heuristic mini-max search by supervised learning", Artificial Intelligence, pp.85-99, Vol.134, 2002.

[24] G. Tesauro: "Programming backgammon using self-teaching neural nets", Artificial Intelligence, pp.181-199, Vol.134, 2002.

[25] J. Feinstein: "Perfect play in 6x6 Othello from two alternative starting positions", http://www.feinst.demon.co.uk/Othello/6x6sol.html

[26] J. Schaeffer, Y. Björnsson, N. Burch, R. Lake, P. Lu and S. Sutphen: "Building the Checkers 10-piece Endgame Database", The 10th Advances in Computer Games Conference (ACG 10), pp.193-210, 2003.

[27] E. Trice and G. Dodgen: "The 7-piece Perfect Play Lookup Database for the Game of Checkers", The 10th Advances in Computer Games Conference (ACG 10), pp.211-230, 2003.

[28] J. Schaeffer, Y. Björnsson, N. Burch, A. Kishimoto, M. Müller, R. Lake, P. Lu and S. Sutphen: "Solving Checkers", Proceedings of IJCAI-05, pp.292-297, 2005.

[29] R. Hayward, Y. Björnsson, M. Johanson, M. Kan, N. Po and J. van Rijswijck: "Solving 7×7 Hex: Virtual Connections and Game-State Reduction", The 10th Advances in Computer Games Conference (ACG 10), pp.261-278, 2003.

[30] K. Noshita: "Union-Connections and A Simple Readable Winning Way in 7×7 Hex", The 9th Game Programming Workshop (GPW 2004), pp.72-79, 2004. (In Japanese)

[31] K. Mishima, H. Sakurai and K. Noshita: "New Proof Techniques and Their Applications to Winning Strategies in Hex", The 11th Game Programming Workshop (GPW 2006), pp.136-142, 2006. (In Japanese).

[32] M. Müller: "Solving 5×5 Amazons", The 6th Game Programming Workshop (GPW 2001), pp.64-71, 2001.

[33] T. Goto, K. Shibahara, N. Inui and Y. Kotani: "Solving Small Boards of Shogi", The 8th Game Programming Workshop (GPW 2003), pp.25-32, 2003. (In Japanese)

[34] M. Buro: "Solving the Oshi-Zumo Game", The 10th Advances in Computer Games Conference (ACG 10), pp.361-366, 2003.

[35] K. Shibahara, Y. Tajima and Y. Kotani: "Approach to Solve Small Boards of Shogi", The 10th Game Programming Workshop (GPW 2005), pp.134-137, 2005. (In Japanese)

[36] T. Tanaka: "Complete analysis of a board game *SIMPEI*", IPSJ SIG Technical Report, pp.65-72, Vol.15, 2006.

[37] K. Shibahara, Y. Tajima and Y. Kotani: "Solving Utsurigi", The 11th Game Programming Workshop (GPW 2006), pp.143-150, 2006. (In Japanese)

[38] T. Kaneko, K. Yamaguchi and S. Kawai: "Automatic Feature Construction and Optimization for General Game Player", The 6th Game Programming Workshop (GPW 2001), pp.25-32, 2001.

[39] T. Kaneko, K. Yamaguchi and S. Kawai: "Pattern Selection Problem for Automatically Generating Evaluation Functions for General Game Player", The 7th Game Programming Workshop (GPW 2002), pp.28-35, 2002.

[40] Y. Kakinoki : "Algorithms of Shogi program K3.0", In H.Matubara, editor, The progress of Computer Shogi, pp.1-23 Kyoritsu Shuppan Co, 1996. ISBN 4-320-02799-X. (In Japanese).

[41] H. Yamashita: "YSS - About its Data structures and Algorithm", In H.Matubara, editor, The progress of Computer Shogi 2, pp.112-142, Kyoritsu Shuppan Co, 1998. ISBN 4-320-02892-9. (In Japanese).

[42] Y. Tanase: "Algorithm of IS Shogi", In H.Matubara, editor, The progress of Computer Shogi 3, pp.1-14, Kyoritsu Shuppan Co, 2000. ISBN 4-320-02956-9. (In Japanese).

[43] S. Kanazawa: "Algorithm of Kanazawa Shogi", In H.Matubara, editor, The progress of Computer Shogi 3, pp.15-26, Kyoritsu Shuppan Co, 2000. ISBN 4-320-02956-9. (In Japanese).

[44] Y. Tsuruoka: “Shogi program Gekisashi”, In H.Matubara, editor, The progress of Computer Shogi 4, pp.1-17, Kyoritsu Shuppan Co, 2003. ISBN 4-320-12074-4. (In Japanese).

[45] M. Arioka: “Search algorithms of Shogi program KFEnd”, In H.Matubara, editor, The progress of Computer Shogi 4, pp.18-40, Kyoritsu Shuppan Co, 2003. ISBN 4-320-12074-4. (In Japanese).

[46] H. Yamashita: “YSS - about improvements after the progress of Computer Shogi 2”, In H.Matubara, editor, The progress of Computer Shogi 5, pp.1-32, Kyoritsu Shuppan Co, 2005. ISBN 4-320-12154-6. (In Japanese).

[47] K. Hoki: “Applications of Brute-Force Search and Futility pruning to Computer Shogi”, IPSJ Magazine, pp.884-889, Vol.47, No.8, 2006. (in Japanese).

[48] K. Hoki: “Optimal control of minmax search results to learn positional evaluation”, The 11th Game Programming Workshop (GPW 2006), pp.78-83, 2006. (In Japanese)

[49] Y. Tsuruoka, D. Yokoyama, T. Maruyama and T. Chikayama: “Game-Tree Search Algorithm Based on Realization Probability”, The 6th Game Programming Workshop (GPW 2001), pp.17-24, 2001. (In Japanese).

[50] Y. Tsuruoka, D. Yokoyama and T. Chikayama: “Game-Tree search algorithm based on realization probability”, ICGA Journal, pp.132-144, Vol.25, No.3, 2002.

[51] S. Soeda and T. Tanaka: “Categories for Amazons Moves”, The 8th Game Programming Workshop (GPW 2003), pp.118-121, 2003. (In Japanese)

[52] M. Seo: “The Algorithm for Checkmate Problems with conspiracy numbers”, In H.Matubara, editor, The progress of Computer Shogi 2, pp.1-21, Kyoritsu Shuppan Co, 1998. ISBN 4-320-02892-9. (In Japanese).

[53] A. Nagai: “Algorithm of df-pn and its application for solver of shogi checkmate problems”, In H.Matubara, editor, The progress of Computer Shogi 4, pp.96-114, Kyoritsu Shuppan Co, 2003. ISBN 4-320-12074-4. (In Japanese).

[54] A. Nagai and H. Imai: “A New Algorithm to Remove henbetu and to Find Yodume of Tsume-Shogi”, The 6th Game Programming Workshop (GPW 2001), pp.9-16, 2001. (In Japanese).

[55] T. Kaneko, T. Tanaka, K. Yamaguchi and S. Kawai: “Evaluation Functions for df-pn$^+$ in Shogi based on Prediction of Proof and Disproof Numbers after Expansion”, The 9th Game Programming Workshop (GPW 2004), pp.14-21, 2004. (In Japanese)

[56] T. Kaneko, T. Tanaka, K. Yamaguchi and S. Kawai: “Df-pn with Fixed-Depth Search at Frontier Nodes”, The 10th Game Programming Workshop (GPW 2005), pp.1-8, 2005. (In Japanese)

[57] F. Okabe: “Application for solving tsume shogi problem by route branch number”, The 10th Game Programming Workshop (GPW 2005), pp.9-16, 2005. (In Japanese)

[58] A. Kishimoto: "A Correct Algorithm to Prove No-Mate Positions in Shogi", The 9th Game Programming Workshop (GPW 2004), pp.1-8, 2004. (In Japanese)

[59] T. Hashimoto, M. Sakuta and H. Iida: "Brinkmate search with guarantee of solution", The 6th Game Programming Workshop (GPW 2001), pp.1-8, 2001. (In Japanese).

[60] M. Taketoshi, T. Hashimoto, M. Sakuta and H. Iida: "Search efficiency in Computer Shogi: A case study using TACOS", The 6th Game Programming Workshop (GPW 2001), pp.164-167, 2001.

[61] Y. Kajihara, J. Nagashima, M. Sakuta and H. Iida: "A Turning Point in Speculative Play", The 6th Game Programming Workshop (GPW 2001), pp.191-194, 2001. (In Japanese).

[62] M. Taketoshi, T. Hashimoto, Y. Kajihara, J. Nagashima and H. Iida: "Realization-Probability Search in Computer Shogi", The 7th Game Programming Workshop (GPW 2002), pp.87-92, 2002. (In Japanese).

[63] Y. Kajihara, T. Hashimoto and H. Iida: "A Speculative Play in Shogi Endgame", The 7th Game Programming Workshop (GPW 2002), pp.57-64, 2002. (In Japanese).

[64] T. Hashimoto and H. Iida: "Note on an evaluation function design for Shogi -absolute coordinate system and relative coordinate system-", The 9th Game Programming Workshop (GPW 2004), pp.88-91, 2004. (In Japanese)

[65] K. Matsubara, T. Hashimoto and H. Iida: "Using master games statistics on the squares played", The 9th Game Programming Workshop (GPW 2004), pp.100-103, 2004. (In Japanese)

[66] T. Hashimoto and H. Iida: "A technique of completely preventing strong horizontal effect", The 9th Game Programming Workshop (GPW 2004), pp.115-118, 2004. (In Japanese)

[67] T. Murata, T. Hashimoto, J. Nagashima and H. Iida: "Improvement of Shogi program in the opening and middle game", IPSJ SIG Technical Report, pp.1-7, Vol.14, 2005.

[68] A. Sano, T. Hashimoto, J. Nagashima and H. Iida: "An enhancement for endgame search using one ply mating search", The 9th Game Programming Workshop (GPW 2004), pp.9-13, 2004. (In Japanese)

[69] A. Sano, T. Hashimoto and H. Iida: "Implementing defense moves against 1-ply threatmate and enhancement for endgame search", The 10th Game Programming Workshop (GPW 2005), pp.17-22, 2005. (In Japanese)

[70] K. Matsubara, T. Hashimoto and H. Iida: "A technique of pruning wrong direction moves", The 10th Game Programming Workshop (GPW 2005), pp.110-113, 2005. (In Japanese)

[71] T. Hamada, T. Hashimoto and H. Iida: "Effective usage of heavy evaluation function and a design of Directional Asymmetric Orientation evaluation function for Shogi", The 10th Game Programming Workshop (GPW 2005), pp.130-133, 2005. (In Japanese)

[72] T. Hashimoto: "Algorithm of shogi program TACOS", In H.Matubara, editor, The progress of Computer Shogi 5, pp.33-67, Kyoritsu Shuppan Co, 2005. ISBN 4-320-12154-6. (In Japanese).

[73] J. Hashimoto and T. Hashimoto: "The use of Killer heuristics in Computer Shogi", IPSJ SIG Technical Report, pp.57-63, Vol.15, 2006.

[74] T. Murata, T. Hashimoto and J. Nagashima: "Automatic Extraction and Using of Technical Moves from Game Records", The 11th Game Programming Workshop (GPW 2006), pp.17-24, 2006. (In Japanese)

[75] M. Buro: "Toward Opening Book Learning", ICCA Journal, pp.98-102, Vol.22, No.2, 1999.

[76] R.M. Hyatt: "Book Learning - a Methodology to Tune an Opening Book Automatically", ICCA Journal, pp.3-12, Vol.22, No.1, 1999.

[77] R.M. Hyatt: Dr. Robert Hyatt's home page,
http://www.cis.uab.edu/hyatt/hyatt.html
(Here you can download source code of CRAFTY.)

[78] H. Kume: "Shogi Club 24 Collection of 240,000 game records vol.1" Naitai Shuppan Co, Japan, 2002, ISBN: 4931538037, (in Japanese).

[79] H. Kume: "Shogi Club 24 Collection of 240,000 game records vol.2" Seikou Syobou Co, Japan, 2004, ISBN: 4880861693, (in Japanese).

[80] N. Ohuchi: "Shogi Hashizeme zensyu", Japan Shogi Association, 1998, ISBN: 4819703404. (In Japanese).

[81] In Syukan-Shogi, editor: "Next move tests in endgame for 4th dan players", Mainichi Communications inc., 2001, ISBN: 839905711. (In Japanese).

[82] Web page of Shogi Club 24,
http://www.shogidojo.com/

[83] R. Grimbergen: "Using Castle and Assault Maps for Guiding Opening and Middle Game Play in Shogi", The 6th Game Programming Workshop (GPW 2001), pp.102-109, 2001.

[84] D.M. Breuker, J.W.H.M. Uiterwijk and H.J. van den Herik: "Replacement Schemes for Transposition Tables", ICCA Journal, pp.183-193, Vol.17, No.4, 1993.

[85] S. Lazar: "Analysis of Transposition Tables and Replacement Schemes", Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, 1995.

[86] R. Grimbergen: "A Neural Network for Evaluating King Danger in Shogi", The 7th Game Programming Workshop (GPW 2002), pp.36-43, 2002.

[87] M. Miwa, D. Yokoyama and T. Chikayama: "Prediction of mates in shogi using SVM and its application", The 9th Game Programming Workshop (GPW 2004), pp.143-150, 2004. (In Japanese)

[88] Y. Tsuruoka: "Learning moves from game records with maximum entropy method", Report of Computer Shogi Association, pp.38-41, vol.17, 2004.

[89] M. Miwa, D. Yokoyama and T. Chikayama: "Automatic construction of estimation of mates based on positions and effects of pieces", The 10th Game Programming Workshop (GPW 2005), pp.48-55, 2005. (In Japanese)

[90] T. Otsuki: "Extraction of "Forced Move" from *N*-gram Statistics", The 10th Game Programming Workshop (GPW 2005), pp.89-96, 2005. (In Japanese)

[91] M. Miwa, D. Yokoyama and T. Chikayama: "Category Extension Based on Extraction of Move History", The 11th Game Programming Workshop (GPW 2006), pp.64-69, 2006. (In Japanese)

[92] K. Shibahara, N. Inui and Y. Kotani: "Effect of ProbCut in Shogi -by Changing Parameters According to Position Category-", The 7th Game Programming Workshop (GPW 2002), pp.73-80, 2002. (In Japanese).

[93] S. Takeuchi, T. Kaneko and S. Kawai: "Application of ProbCut into Quiescence search in Shogi", IPSJ SIG Technical Report, pp.9-15, Vol.14, 2005.

[94] K. Usui, T. Suzuki, and Y. Kotani: "Parameter Learning Using Temporal Differences in Shogi", The 5th Game Programming Workshop (GPW 1999), pp.31-38, 1999. (In Japanese).

[95] T. Suzuki and Y. Kotani: "Nonlinearity of an Evaluation Function in Shogi", The 6th Game Programming Workshop (GPW 2001), pp.110-116, 2001. (In Japanese).

[96] S. Watanabe and Y. Kotani: "Position Evaluation Using Decision Tree in Shogi", The 6th Game Programming Workshop (GPW 2001), pp.117-123, 2001. (In Japanese).

[97] A. Hondoh and Y. Kotani: "An Adjustment Method of Position Evaluation Function by Repetition Regression Analysis", The 6th Game Programming Workshop (GPW 2001), pp.168-171, 2001. (In Japanese).

[98] H. Matsumoto, T. Suzuki and Y. Kotani: "Learning Evaluation Function of Shogi based on TD(λ) using Neural Network", The 6th Game Programming Workshop (GPW 2001), pp.176-178, 2001. (In Japanese).

[99] T. Kaneko, T. Tanaka, K. Yamaguchi, and S. Kawai: "Evaluation Functions Based on Pairs of Pieces", The 8th Game Programming Workshop (GPW 2003), pp.14-21, 2003. (In Japanese)

[100] S. Takeuchi, Y. Hayashi, T. Kaneko and S. Kawai: “Adjustment of Evaluation Functions Based on Relation Between Static Values and Win Ratios -Evaluation of Safety Difference Between Both kings in Shogi-”, The 11th Game Programming Workshop (GPW 2006), pp.56-63, 2006. (In Japanese)

[101] Y. Hori, H. Saito and T. Maruyama: “An FPGA-Based Hardware Platform for Tsume-Shogi”, The 8th Game Programming Workshop (GPW 2003), pp.44-51, 2003. (In Japanese)

[102] Computer Shogi Association Web Site, http://www.computer-shogi.org/index_e.html

[103] R.S. Sutton, and A.G. Barto: “Reinforcement Learning: An Introduction”, translated by S. Mikami, and M. Minagawa, Morikita Shuppan Co, 2000. ISBN 4-627-82661-3. (In Japanese).

[104] N. Baba, H. Kojima, and S. Ozawa: “The basis and application of neural network”, Kyoritsu Shuppan Co, 1994. ISBN 4-320-02714-0. (In Japanese).

Publications

Journal Publication

[1] M. Sakuta, J. Nagashima, T. Hashimoto and H. Iida: “Application of the Methods Developed in the Endgame Search of Shogi to Lines of Action”, Journal of the Information Processing Society of Japan, pp.2964-2972, Vol.43, No.10, October, 2002.

[2] M. sakuta, T. Hashimoto, J. Nagashima, J.W.H.M. Uiterwijk and H. Iida: “Application of the killer-tree heuristic and the lambda-search method to lines of action”, Journal of Information Sciences, pp.141-155, Vol.154, 2003.

[3] J. Nagashima, T. Hashimoto and H. Iida: “Self-Playing-based Opening Book Tuning”, New Mathematics and Natural Computation, pp.183-194, Vol.2, No.2. July, 2006.

Refereed Conference Proceedings

[4] T. Hashimoto, J. Nagashima, M. Sakuta and H. Iida: “Realization-Probability Search: from Shogi to Any Games”, 7th Joint International Conference on Advanced Science and Technology (JICAST 2002), pp.123-126. Hamamatsu, Japan. October, 2002.

[5] J. Nagashima, M. Taketoshi, Y. Kajihara, T. Hashimoto and H. Iida: “Several Enhancements for Better Development in Computer Shogi”, 7th Joint International Conference on Advanced Science and Technology (JICAST 2002), pp.306-309. Hamamatsu, Japan. October, 2002.

[6] T. Hashimoto, J. Nagashima, M. Sakuta, J.W.H.M. Uiterwijk and H. Iida: “Application of Realization Probability Search for Any Games -a case study using Lines of Action -”, The 7th Game Programming Workshop (GPW 2002), pp.81-86. Hakone, Japan. November, 2002.

[7] J. Nagashima, T. Hashimoto, M. Sakuta, J.W.H.M. Uiterwijk and H. Iida: “An Automatic Tuning of Game Playing System based on the Realization-Probability Search”, The 8th Game Programming Workshop (GPW 2003), pp.1-7. Hakone, Japan. November, 2003.

[8] J. Nagashima, T. Hashimoto and H. Iida: “An enhancement for realization-probability search using machine learning”, INTER-ACADEMIA 2004, pp.215-222. Budapest, Hungary. September, 2004.

[9] J. Nagashima, T. Hashimoto and H. Iida: “Opening Book Tuning”, The 9th Game Programming Workshop (GPW 2004), pp.129-134. Hakone, Japan. November, 2004.

[10] J. Nagashima, T. Hashimoto and H. Iida: “Finding Fruitful Positions”, 8th Joint International Conference on Advanced Science and Technology (JICAST 2004), pp.161-164. Chechiang, China. December, 2004.

[11] J. Nagashima, T. Hashimoto and H. Iida: “Master-like Opening Strategy in Computer Shogi”, 8th Joint Conference on Information Sciences (JCIS 2005). Salt Lake City, USA. July, 2005.

[12] J. Nagashima, J. Hashimoto and T. Hashimoto: “Opportune Time Recognition of Edge Attacks in Computer Shogi”, The 11th Game Programming Workshop (GPW 2006), pp.1-8. Hakone, Japan. November, 2006.

Non-Refereed Conference Proceedings

[13] J. Nagashima, M. Taketoshi, Y. Kajihara, T. Hashimoto and H. Iida: “An Efficient Use of Piece-Square Tables in Computer Shogi”, IPSJ SIG Technical Reports 2002-GI-8, pp.29-36. Edmonton, Canada. July, 2002.

[14] T. Hashimoto, J. Nagashima, M. Sakuta and H. Iida: “Realization-Probability Search : Its application to Shogi and LOA”, The 7th Computer Olympiad Computer-Games Workshop, Maastricht, Netherlands. July, 2002.

[15] J. Nagashima, T. Hashimoto and H. Iida: “Alternative use of two transposition tables for game tree search”, IPSJ SIG Technical Reports 2006-GI-15, pp.49-55. Tokyo, Japan. March, 2006.