Using Castle and Assault Maps for Guiding Opening and Middle Game Play in Shogi

Reijer Grimbergen
Department of Information Science, Saga University
Honjo-machi 1, Saga-shi, 840-8502 Japan
E-mail: grimbergen@fu.is.saga-u.ac.jp

Jeff Rollason
Oxford Softworks
198 The Hill, Burford, Oxfordshire, OX18 4HX, England
E-mail: 100031.3537@compuserve.com

Abstract

Shogi programs still have a lot of room for improvement in the opening and early middle game, which usually is a strategic stage in shogi. The planning of long-term strategies is required, which is known to be hard to program in games. This paper presents a new method for guiding opening and middle game play in shogi. Board maps for castle formations and assault formations are used to evaluate the strategic value of a position. These maps have importantly improved the playing strength of the shogi programs SPEAR and SHOTEST.

1 Introduction

The strength of shogi programs has improved considerably in the last couple of years. In the final stage of the game, the *tsume shogi* search, shogi programs have outperformed human experts [13]. Even in the endgame phase just before mating, shogi programs can outplay strong amateur players. Based on these observations, the strength of shogi programs is estimated to be at least 3-dan.

Despite these accomplishments, there is still a lot of work to be done before a shogi program can beat the best players in the world. One of the problems lies in the early stage of the game. The opening phase in shogi is very strategic in nature. In chess, long-term strategic planning is known to be the major weak point of the strong programs as well. So much so that world champion Kasparov in his famous 1997 match against DEEP BLUE did not play in his usual aggressive style, but chose quiet strategic openings [12]. This was not only to avoid tactical complications which are the strong point of deep searching chess programs, but also to stay out of the huge databases that are used for opening play in chess.

In two-player complete information games, the starting position is determined by the rules of the game. All strong game playing programs make use of this information to build an opening database from which moves can be played in the early stages of a game. This database is called an *opening book*. If the current position in the game no longer matches any of the positions in the database there are no more move suggestions: the program is *out of book*.

An opening book can have hundreds of thousands of positions, and the impressive results of strong game programs in chess [8], checkers [11] and Othello [3] would not have been possible without a large opening book and extensive tuning of this opening book to fit the program's play.

In general the opening book is implemented and updated manually, but there are also a number of programs in different games where the opening book is extended automatically by playing the program against itself and other programs. Examples are the chess programs DEEP BLUE [4], CRAFTY [6], the checkers program CHINOOK [11] and the Othello program LOGISTELLO [3].

To test how effective the use of an opening book is in shogi, we made an opening book containing more than a 1000 professional games and the opening variations given in more than 20 opening books. The complete opening book built in this way has more than 110,000 positions. This is smaller than the opening book of most strong chess programs, but much larger than most shogi programs. It is hard to build an opening book that is similar in size to the ones used in chess programs, as worldwide the number of expert shogi players is much smaller than the number of expert chess players. Therefore, in shogi the number of publicly available expert games and the number of books on opening play is much smaller than in chess.

In earlier work [5], we showed that even such a relatively large opening book is not very effective in shogi. In 100 games against four of the strongest shogi programs the program was out of book in 8.5 moves on average. In almost one-third of the games the program was out of book within 5 moves. Therefore, other ways to deal with the strategic build-up in the opening are needed. In this paper we will present a different method for guiding the opening and middle game play in shogi. This method will use board maps of castle formations and assault formations to guide the play in the early stages of the game.

In Section 2 the general data structure for the maps is given. In Section 3 details of the castle maps and assault maps will be given. In Section 4 we will explain how the maps can be used in the search. In Section 5 we will give results from self-play experiments showing that this method improves the playing strength of a shogi program. Section 6 explains about the *otoshiana* method, which is similar to the method presented here. In Section 7 we will end with conclusions and thoughts on further research.

2 General Data Structure for Castles

In chess there are only two different castle formations: castling on the king side or castling on the queen side of the board. Furthermore, castling in chess only takes a single move.

In shogi, there are numerous castle formations that take a number of moves to build. For example, the *anaguma* castle, which is the strongest castle formation is shogi, takes more than 10 moves to complete. Also, castle formations in shogi can have multiple stages. For example, the *mino* castle can be turned into the stronger *high mino*[1], which can be rebuild into a *silver*

```
char **castles[] = {

   mino,            a_mino,          id_static,
   high_mino,       a_mino,          id_static,
   silver_crown,    a_mino,          id_static,


   boat,            a_boat,          id_ranging,

} ;
```

Figure 1: Castle definitions.

[1] *Takamino* in Japanese.

```
char *mino[] = {
      // Non-promoted pieces
      mino_pawn,
      mino_lance,
      mino_knight,
      mino_silver,
      mino_gold,
      mino_bishop,
      mino_rook,
      mino_king,

      // Promoted pieces
      mino_gold,    // Promoted pawn
      mino_gold,    // Promoted lance
      mino_gold,    // Promoted knight
      mino_gold,    // Promoted silver
      void,         // Promoted gold
      mino_bishop,  // Promoted bishop
      mino_rook,    // Promoted rook
      void,         // Promoted king

      "Mino"
   } ;
```

Figure 2: Individual piece definitions for the mino castle.

crown[2].

To guide the building of castles, we have defined *castle maps* for a number of common castle formations in shogi. First, a data structure is needed in which each individual castle is defined. An example of such a data structure is given in Figure 1. In this data structure we define the castles, the best way to assault these castles and for which type of position this castle is best suited. The mino castles are best build when the opponent has adopted a static rook strategy (indicated by *id_static*), while the *boat* castle is best played when the opponent has a ranging rook formation (*id_ranging*). The best assault against each mino castle is defined by the maps in *a_mino*, while the best assault against a boat castle is defined by the maps in *a_boat*.

3 Castle Maps and Assault Maps

After building a general data structure for each castle, we need to define for each individual piece where it is best positioned in the castle. An example for the mino castle is given in Figure 2. Note that the two void entries indicate that a gold and a king in shogi can't promote, so there is no castle map for these two pieces. Pieces that promote to gold have the same castle maps as a gold. Promoted rook and promoted bishop have the same castle maps as the rook and bishop respectively.

Finally, for each individual piece we have an array defining where it is positioned best in that particular castle formation. An example of the castle map for the king in the mino castle is given in Figure 3.

[2] *Ginkanmuri* in Japanese.

```
char mino_king[] = {
      0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0,-9,-8,-1,-1,-1,-1,
      0, 0, 0,-9,-7, 0, 4, 8, 5,
      0, 0, 0,-8,-6, 2, 8,14, 6,
      0, 0, 0,-7,-5,-1, 8, 8, 6
} ;
```

Figure 3: The castle map for the king in the mino castle.

These values indicate that the king in the mino castle is best positioned on square 2h which has the value 14. To move the king to this square, optimal paths can be constructed with a hill climbing approach: move the piece to a neighboring square with a higher value. If more than one neighboring square has a higher value, choose the square with the highest value.

In the example of Figure 3, there are two natural paths to the optimal square from the starting position of the king on 5i (the square with value -5): K5i-K4h-K3h-K2h and K5i-K4h-K3i-K2h. Both paths have square values -5, 2, 8, 14.

The assault maps are defined in exactly the same way. For each assault there is a data structure defining the maps of each piece, like in Figure 2. For each individual piece there is a map like Figure 3, where the evaluation of the position of the piece for this particular assault is given.

4 How to Use the Maps

In our programs, the maps are used to select the castle formation that can best be built from the current position (i.e. the castle formation resembles the current formation most). When the castles for both sides have been identified, the assaults for both castles are defined by the data structure of Figure 1. After this selection, the hill climbing defined by the map values guides the search towards moves that improve the chosen castle formation and assault formation.

Especially in the very early stages of the opening, when it is still possible to build many different castle formations, this selection process can't select the correct castles based on the maps alone, but needs extra support in the form of common shogi opening knowledge. This has been implemented with a number of rules that check for special cases. These rules mainly trigger when neither player has committed to a certain strategy.

Note that one of the side effects of the method is that formations that for some reason have gone astray can be fixed. Even if the optimal move order is no longer possible, the hill-climbing approach might still generate non-optimal moves to get the pieces into the right formation. For example, if there is a piece on 4h in the mino castle but not on 4i, the map of Figure 3 can still generate a path for the king to move into the mino castle (K5i-4i-3h-2h).

There are several other ways in which the maps can be used in a shogi program. First, based on the values in the maps, the values of the positional component of the evaluation function change. The second way the maps can be used is to make plans for building castle formations. If the map values are only used in the evaluation function, there is no difference between different move orders that lead to the same final position in the search. In shogi the move order by which

a formation is built is often important. From the maps an optimal move order can be derived that can be used to guide the search.

A third use of the maps is that they can help with a completely different problem that often occurs in game programs with a large opening book. This is the problem that the program blindly follows the opening book and ends up in a position that human experts judge as better but the program doesn't understand because of the limitations of the evaluation function. Rather than trying to reach a position that is objectively better, the program should aim for a position it can understand. The maps provide a way to do this. If the opening book move decreases the castle value of the moving piece, it should not be trusted as it might put the pieces in a position the program doesn't like. Therefore, such a move should not be played without search. Of course, it shouldn't be discarded either, as it might have been the only way to avoid losing the game.

The fourth use of the maps is for establishing the game stage. In a shogi program, it is important to make a difference between the opening, middle game and endgame. Evaluation function features change dramatically based on the stage of the game. Especially the judgment of the transition from opening to middle game is a problem in shogi. The maps can provide extra information to make the decision whether the current position is still an opening position or the middle game has started. Our current program uses a combination of a minimal castle map value of all pieces combined and information about the optimal position of the king. If the king is in an optimal position and the castle value exceeds a certain threshold, the position is judged to be a middle game position.

Finally, the castle maps can be used for plausible move generation during search. Most shogi programs analyze the legal moves and discard certain moves without search. Moves that improve the castle value or assault value of the moving piece should not be discarded.

5 Results

The castle maps and assault maps were first implemented in version 3 of the shogi program SHOTEST by the second author, which participated in the CSA tournament in 1999. For the CSA tournament in 2001, the maps were implemented in the program SPEAR by the first author. For SPEAR the number of maps was increased considerably and extensively hand-tuned by playing hundreds of games against other commercial programs. The current version of SPEAR has maps for 35 different castle formations and 20 assault formations.

The castle maps and assault maps were the most important difference between the version of SPEAR that played in the CSA tournament in 2000 and the version that played in the CSA tournament in 2001. The results of SPEAR in 2001 were much better than the results in 2000, despite the average improvement in the playing level of the competing programs. Both versions cleared the first preliminary qualification stage, but in the second qualification tournament the 2000 version only scored 3 wins and 6 losses, while the 2001 version scored 5 wins and 4 losses, just missing qualification for the finals.

SHOTEST did only a little better (also 5-4 but against stronger opposition), despite its good results in previous years. This shows that tuning the maps is very important. The maps have consequences for different parts of the program and there was no time to do this tuning for SHOTEST in the build-up to the CSA tournament.

We also performed self-play experiments with SPEAR to show the importance of using maps for playing strength. We played three versions of SPEAR against each other. One version was using both castle maps and assault maps (*SpAllMaps*), one version was using only castle maps (*SpCasMaps*) and one version was not using castle maps (*SpNoMaps*). The basic SPEAR program has the following features:

- Iterative alpha-beta search.
- Principal variation search [9].
- Quiescence search [2].
- History heuristic and killer moves [10].
- Null-move pruning [1].
- Hashtables for transposition and domination [14].
- Specialized mating search [13].

In this experiment, we specifically wanted to know how the interaction between the opening book and the maps influenced playing strength. Therefore, we had the programs follow the opening book for 10, 20, 30, 40 and 50 moves. If the opening book can be used longer, the maps will play a less important role as the program can use the opening book to build a good castle formation and assault formation. After the opening book, the resulting position is played twice: once with either program version playing the black pieces. Each program version played the other versions fifty times, i.e. 25 times with black and 25 times with white. The time limit was 25 minutes per side per game on a 1GHz Pentium III. This is the same time limit as used in the annual CSA tournament. The results of this tournament are given in Table 1.

This table shows two interesting results. First, when the program has almost no support from the opening book, it benefits significantly from having the castle maps and assault maps. If the database was used for 10 or 20 moves, the version with both castle maps and assault maps beat the version without maps with 68-32 respectively. The version with only castle maps has almost the same result with a score of 67-33. The 68-32 result gives a 99.98% probability that the program with maps is stronger than the program without the maps. For the 68-32 result, this probability is 99.97%.

For 30 moves and higher, there is almost no difference in playing strength between the version without maps and the two other versions.

A second result is that the two versions with maps have almost the same result against the version without maps. Yet, the version with both castle maps and assault maps beat the version with only castle maps by a significant margin. Furthermore, the results are consistent throughout the use of the opening book, the version with both castle maps and assault maps winning three matches 29-21, one match 28-22 and one match 27-23. It seems that if the program has no clue

Match	*Book*					Result
	10	20	30	40	50	
SpAllMaps-SpNoMaps	38-12	30-20	27-23	25-25	25-25	145-105
	76%	70%	54%	50%	50%	58%
SpCasMaps-SpNoMaps	36-14	31-19	25-25	25-25	23-27	140-110
	72%	62%	50%	50%	46%	56%
SpAllMaps-SpCasMaps	27-23	29-21	29-21	29-21	28-22	142-108
	54%	58%	58%	58%	56%	57%

Table 1: Results of self play experiments between a version of SPEAR using castle maps and assault maps (SpAllMaps), a version using only castle maps (SpCasMaps) and a version not using any maps (SpNoMaps). *Book* is the maximum number of moves the book was used in each game.

about the castle of the opponent or makes a mistake in assessing the castle, the assault maps have no significant impact. However, if the opponent plays a known castle, the use of assault maps will lead to a significant improvement in playing strength which lasts for a longer period of the game.

6 Related Work

We believe that our work is very similar to the so-called *otoshiana* method, which was first used in the program YSS [15] (a non-Japanese explanation of this method will be available soon [7]). The *otoshiana* method also uses a hill climbing approach to build castle formations and it seems likely that the implementation is similar to ours. However, this is difficult to judge, as Yamashita's description of the *otoshiana* method is short and lacks the details for implementing the method as given in this paper. The original description mentions only 9 different castle formations and no assault formations. This could be an important difference, as our results indicate that the use of assault maps considerably improves the playing strength. Furthermore, it is unclear how the *otoshiana* method assesses the type of position that the opponent is building. Yamashita mentions that the position of the rook is taken into account, "among other things", but no details are given. Still, most strong shogi programs seem to use a method similar to the *otoshiana* method.

It is interesting that the *otoshiana* method also includes a number of positional elements. For example, having a lance on the back rank is better than having a lance higher up the board and there are penalties for having a knight high up the board or a rook on the 7th rank. Our current research has focused only on opening and early middle game formations, but the method might have a more general use.

7 Conclusions

In this paper we have given a new method for building castle formations and assault formations in the opening and middle game in shogi. We have shown that if the program gets out of book in the first 20 moves, the playing strength can be improved considerably by using castle maps and assault maps. Furthermore, using both castle maps and assault maps gives significantly better results than using only castle maps.

There is still a lot of work to do before this method can be used to its full potential. The current program only uses one castle during the search. It was expected that using the full map matching at each position evaluation would slow down the search too much to be of practical use. However, one idea would be to take a limited number of similar castles into the search and only evaluate the changes in this subset. This has not been implemented yet.

The second possible extension is the use of swapping rules. Castles and assaults may or may not be feasible based on certain piece formations. This can be either because one's own pieces are positioned in the wrong way or the opponent has made a formation that makes the position that was aimed for too dangerous. The checks that are now performed for swapping castles and assaults are primitive if-then rules. At the moment we are working on a more transparent method to deal with this.

Related to this is the idea of scaling castles. The importance of a castle formation and assault formation changes during the game and a global scaling factor attached to each map can account for this. Also, instead of discarding a castle or assault, it is better to scale the value of the castle down based on certain criteria. Even if a castle is not so good under the circumstances, it can still be the best one available.

Finally, it might be better to evaluate assaults independent from castle formations. For example, there are a number of ways in which a yagura castle can be assaulted (i.e. *bogin*, *yagura nakabisha*, *suzume sashi* etc.). This can be covered by a number of swapping rules, but it might be better to match assaults separately. All these ideas are currently being investigated and implemented and will hopefully lead to a flexible and robust way to play the opening and middle game in shogi at a high level.

References

[1] D. Beal. Experiments with the Null Move. In D.Beal, editor, *Advances in Computer Chess 5*, pages 65–79. Elsevier Science Publishers: The Netherlands, 1989.

[2] D. Beal. A Generalised Quiescence Search Algorithm. *Artificial Intelligence*, 43:85–98, 1990.

[3] M. Buro. Toward Opening Book Learning. *ICCA Journal*, 22(2):98–102, June 1999.

[4] M. Campbell. Knowledge Discovery in DEEP BLUE. *Communications of the ACM*, 42(11):65–67, 1999.

[5] R. Grimbergen and J. Rollason. Using Castle Maps for Guiding Opening and Middle Game Play in Shogi. In *Information Processing Society of Japan Meeting*, pages 51–57, Hakodate, Japan, 2001.

[6] R.M. Hyatt. Book Learning - A Methodology to Tune an Opening Book Automatically. *ICCA Journal*, 22(1):3–12, March 1999.

[7] H. Iida, M. Sakuta, and J. Rollason. Computer Shogi. *Artificial Intelligence*, 2001. (To appear).

[8] M. Newborn. *Kasparov versus Deep Blue: Computer Chess Comes of Age*. Springer Verlag, 1996. ISBN 0-387-94820-1.

[9] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison Wesley Publishing Company: Reading, Massachusetts, 1984. ISBN 0-201-05594-5.

[10] J. Schaeffer. The History Heuristic and Alpha-Beta Search Enhancements in Practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1203–1212, 1989.

[11] J. Schaeffer. *One Jump Ahead: Challenging Human Supremacy in Checkers*. Springer-Verlag New York, Inc., 1997. ISBN 0-387-94930-5.

[12] J. Schaeffer and A. Plaat. Kasparov Versus Deep Blue: The Rematch. *ICCA Journal*, 20(2):95–101, June 1997.

[13] M. Seo. The C* Algorithm for AND/OR Tree Search and its Application to a Tsume-Shogi Program. Master's thesis, Faculty of Science, University of Tokyo, 1995.

[14] M. Seo. On Effective Utilization of Dominance Relations in Tsume-Shogi Solving Algorithms. In *Game Programming Workshop in Japan '99*, pages 129–136, Kanagawa, Japan, 1999. (In Japanese).

[15] H. Yamashita. YSS: About its Datastructures and Algorithm. In H. Matsubara, editor, *Computer Shogi Progress 2*, pages 112–142. Tokyo: Kyoritsu Shuppan Co, 1998. ISBN 4-320-02799-X. (In Japanese).