Using Castle 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 are quite strong in the endgame, but continue to have problems in the opening and early middle game. This is caused by the strategic nature of the early stage of a shogi game. Long-term strategies are known to be hard to program in games. Particularly in chess this is supposed to be the Achilles' heel of strong chess programs. This paper presents a new method for guiding opening and middle game play in shogi. Board maps for castle formations are used to evaluate the strategic value of a position. These maps have importantly improved the playing strength of the shogi program SPEAR.

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 [11]. 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. The main problem lies in the early stage of the game. The opening phase in shogi is very strategic in nature. In chess, this 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 [10]. 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 especially in chess the use of an opening book is very important [6]. In general the opening book is implemented and updated manually, but there are also a number of programs in different games where the opening

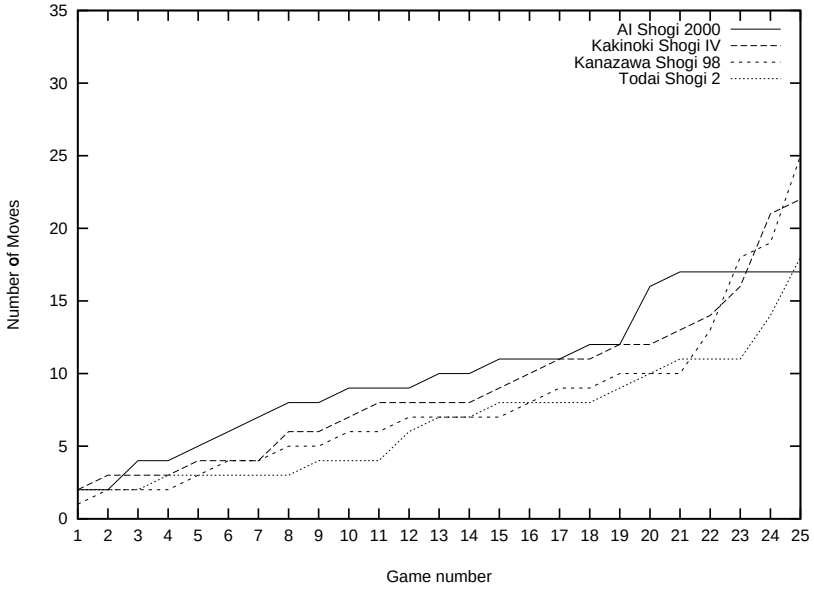

Figure 1: Full board database matches in 25 games against AI Shogi 2000, Kakinoki Shogi IV, Kanazawa Shogi 98 and Todai Shogi 2.

book is extended automatically by playing the program against itself and other programs. Examples are the chess programs Deep Blue [4], Crafty [5], the checkers program Chinook [9] 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.

The problem of using an opening book in shogi became clear when we tested the use of the opening book in games against strong shogi programs. We played 25 games each against four of the strongest shogi programs on the market: AI Shogi 2000 (winner of the CSA tournament, the computer shogi world championships, in 1997), Kakinoki Shogi IV (winner of the Computer Shogi Grand Prix in 1999), Todai Shogi 2 (winner of the CSA tournament in 1998, 2000 and 2001) and Kanazawa Shogi 98 (winner of the CSA tournament in 1995, 1996 and 1999). In these games, we looked at the point where our program went out of book. The results of this test are given in Figure 1.

Figure 1 shows that despite the large number of positions in the opening book, the program still gets out of book rather quickly. In almost one third of the games (32) our program is out of book within five moves. In 71 games our program is out of book in ten moves or less. On average, the program is out of book after 8.5 moves.

The conclusion from these results is that an opening book in shogi is not very effective. Other ways to deal with the strategic build-up in the opening are needed. In this paper we will present a new method for guiding the opening and middle game play in shogi. This method will use board maps of castle formations to guide the opening play in the early stages of the game.

In Section 2 we will describe how castle maps can be used to strategically guide a shogi

program in the opening and middle game. In Section 3 we will give results from self-play experiments showing that this method improves the playing strength of a shogi program. In Section 4 we will end with conclusions and thoughts on further research.

2 Castle Maps

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, in shogi castle formations 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 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 2. In this data structure we define the 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*).

Next, for each individual piece we need to define where it is best positioned in the castle. An example for the mino castle is given in Figure 3. 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 4.

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 4, there are two natural paths to this 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.

```
char **castles[] = {

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

   boat,         id_ranging,

} ;
```

Figure 2: Castle definitions.

[1] *Takamino* in Japanese.
[2] *Ginkanmuri* 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 3: Individual piece definitions for the mino castle.

```
   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 4: The castle map for the king in the mino castle.

In our programs, the castle maps are used to select the castle formation that can best be built from the current position (i.e. which castle formation resembles the current formation most). After this selection, the search is guided towards moves that improve the chosen castle 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 castle 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.

There are several ways in which the castle maps can be used in a shogi program. First of all, in the evaluation function. Based on the values in the castle maps, the values of the positional component of the evaluation function change. The second way the castle maps can be used is to make plans for building castle formations. If the 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 in which a castle formation is built is often important. From the castle maps an optimal move order can be derived that can be used to guide the search.

A third use of the castle 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 a program blindly follows the opening book and ends up with 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 castle 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.

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 should not be discarded.

3 Results

The castle 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 castle maps were implemented in the program SPEAR by the first author. For SPEAR the number of castle maps was increased considerably and extensively hand-tuned by playing hundreds of games against other commercial programs.

The castle 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 castle maps is very important. The castle 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 some self-play experiments with SPEAR to show the importance of using castle maps for playing strength. We played two versions of SPEAR against each other. One

Book	SpM-SpNM	%
10	12-8	60%
20	13-7	65%
30	10-10	50%
40	9-11	45%
50	9-11	45%
Total	53-47	53%

Table 1: Results of self play experiments between a version of SPEAR using castle maps (SpM) and a version not using castle maps (SpNM). *Book* is the maximum number of moves the book was used in each game.

version was using castle maps (*SpM*) and one version was not using castle maps (*SpNM*). The basic SPEAR program has the following features:

- Iterative alpha-beta search.
- Principal variation search [7].
- Quiescence search [2].
- History heuristic and killer moves [8].
- Null-move pruning [1].
- Hashtables for transposition and domination [12].
- Specialised mating search [11].

In this experiment, we specifically wanted to know how the interaction between the opening book and the castle 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 castle maps will play a less important role as the program can use the opening book to build a good castle 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 twenty times, i.e. 10 times with black and 10 times with white. The time limit was 25 minutes per side per game on a 1.2 GHz 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.

From this table we can see that the results are as expected. When the program has almost no support from the opening book, it benefits significantly from having the castle maps. If the database was used for 10 or 20 moves, the version with castle maps beat the version without castle maps with 12-8 and 13-7 respectively. The 12-8 result gives a 80.8% probability that the program with castle maps is stronger than the program without castle maps. For the 13-7 result, this probability is 90.5%.

For 30 moves and higher, there is almost no difference in playing strength with the version without castle maps actually winning the two matches with the maximum opening book use.

4 Conclusions

In this paper we have given a new method for building castle 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 is improved considerably. In the introduction we saw that in games against other strong shogi programs, the opening book can almost never be used for more than 20 moves, so we expect that the castle maps will improve the results against these programs. Testing this will be a future work.

There is still a lot of work to do before this method can be used to its full potential. One improvement is adding maps for attacking formations. Not only castle formations, but also the piece formation of attacks against castles are built in a standard way. We have already implemented attack formations, but we have no space here to fully report on their use, so this will be done in a future paper.

One problem with this approach has been that despite the tuning there are still castles that are very similar in build-up, especially in the early stages. The checks that need to be performed when selecting a castle grow considerably with each castle map that is being added. At the moment we are working on a more transparent method to deal with this.

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

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

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

[8] 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.

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

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

[11] 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.

[12] 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).