A Plausible Move Generator for Shogi Using Static Evaluation

Reijer Grimbergen
Electrotechnical Laboratory,
1-1-4 Umezono, Tsukuba-shi, Ibaraki-ken, Japan 305-8568
E-mail: grimberg@etl.go.jp

Abstract

To make a program that can perform well under tournament conditions, shogi poses some extra challenges when compared with chess. The main problem is that in shogi there are on average many more legal moves than in chess. To deal with this large branching factor, it seems inevitable to revive research into plausible move generators, which has not been successful in chess. In this paper a first step in developing a plausible move generator will be presented for moves involving winning and losing material. It will be explained how such a plausible move generator can use the static evaluation of shogi positions to generate moves. Test results show that this plausible move generator generates on average only 30% of the total number of moves. The time spent on this analysis is worth the effort, as is indicated by a major increase in performance on a test set of shogi problems. As preliminary results, these are very encouraging.

1 Introduction

Strong chess programs are capable of reaching search depths of up to 13 plies in games under tournament conditions [4]. In computer chess tournaments the average thinking time is about 3 minutes per move. To reach the same level of playing strength, shogi programs need to have a similar performance. There are two important problems facing the shogi programmer who is building a high performance shogi program.

First, because of the possibility of re-using captured pieces (the *drop* move), the number of possible legal moves in shogi is on average much larger than in chess. The average branching factor of the search tree for chess is only about 35 while in shogi the average branching factor is about 80 [8].

These numbers do not tell the whole story. In shogi the branching factor increases as the game progresses. This is illustrated with data from 10 expert games that will be used in this paper for different tests. The details of the games chosen are given in Table 1. The games were selected to have

No	Players	Date	Opening	Length
1	Y.Habu(9-dan) - A.Shima(8-dan)	December 13th 1994	Yagura	123
2	N.Yashiki(7-dan) - T.Fujii(6-dan)	April 3rd 1996	Shikenbisha	165
3	H.Kamiya(7-dan) - K.Kodama(7-dan)	March 6th 1998	Aigakari	115
4	M.Tachi(6-dan) - H.Nozuki(4-dan)	December 16th 1997	Yokofudori	82
5	K.Tanigawa(9-dan) - M.Nakahara(10-dan)	June 12th 1990	Kakugawari	146
6	M.Chuza(4-dan) - N.Ito(4-dan)	March 11th 1997	Hineribisha	128
7	S.Anzai(5-dan) - S.Sato(5-dan)	November 18th 1997	Mukaibisha	112
8	N.Hatakeyama(6-dan) - D.Nakagawa(6-dan)	August 22nd 1997	Yodofuribisha	139
9	K.Horiguchi(6-dan) - D.Suzuki(5-dan)	October 21st 1997	Nakabisha	154
10	K.Ishida(9-dan) - K.Manabe(8-dan)	March 1998	Sankenbisha	104

Table 1: 10 Test games. *Opening* gives the opening strategy of each game in Japanese and *Length* is the length in ply of each game.

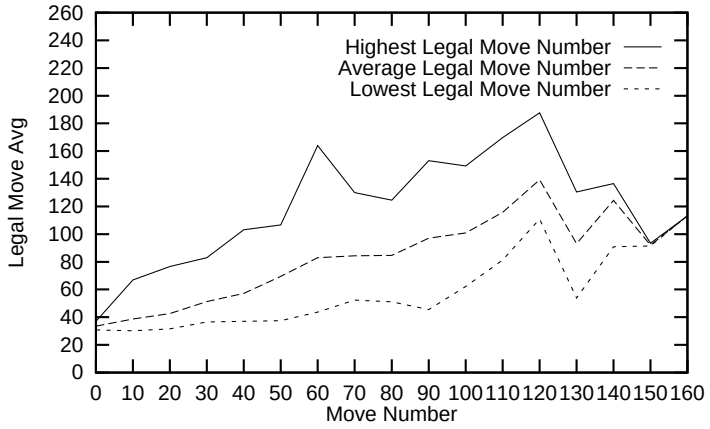

Figure 1: Highest number of legal moves, average number of legal moves and lowest number of legal moves by move number in the 10 test games.

a wide variety of different types of shogi positions. Therefore, each game is played by different players and has a different opening strategy.

In Figure 1 the results of analysing the number of legal moves in these test games have been given. The figure shows the increasing branching factor of shogi, which is caused by a larger number of possible drop moves. Pieces can be dropped on any vacant square, so there are many more move possibilities to drop pieces than for pieces already on the board. In the endgame, which is the vital stage in shogi, the branching factor can easily exceed a 100 and stretches of branching factors of 140 or more are not uncommon. In contrast, in chess the game often becomes easier towards the end as less and less pieces remain in play. All strong chess programs use endgame databases to stop the search when perfect knowledge about the game result is available.

Not only the high branching factor is a problem, but in shogi also the average thinking time per move in a game under tournament conditions is much less than in chess. There are two reasons for this. One is that the average game length of shogi is about 115 ply [1], while the average game length of chess is about 80 ply. A shogi program can therefore use less time per move than a chess program. The second reason is that the tournament conditions for shogi programs are much stricter than for chess programs. In the annual CSA tournament, the unofficial world championship tournament, a program has only 25 minutes for the full game (in the preliminary stages this is even less: 20 minutes). Based on the average length of a game, a program has to play about 60 moves in that time period, leaving only 25 seconds per move on average. This average is misleading, since most games between computer programs take longer than games between human players. Most programs therefore have some method to dynamically change the time alloted to a move. This time can for example be based on quiescence, position evaluation or a guess about the expected number of moves remaining.

To deal with a large search tree and severe time constraints, shogi programmers have realised that it is necessary to make good decisions about the moves on which search effort should be spent. All strong shogi programs use plausible move generators to cut as many unlikely moves from the search as possible. In chess, this approach has not been successful. In the 1960s most chess programs used plausible move generators (see for example [2]), but since the development of the CHESS 4.5 program in the early seventies [10], the basic approach of chess programs has been a full width search.

The features of shogi and the tournament conditions of computer shogi tournaments make a revival of plausible move generators necessary. This might also be of interest to cognitive scientists, who have been complaining that chess programs are not "real AI", since the way computer programs play chess has no connection with the way human players play the game (for a small discussion on this, see [6]).

Unfortunately, there are not many publications on the methods used in strong shogi programs, since no top programmers has a research interest (most are commercial programmers). Exceptions

are Kakinoki (author of KAKINOKI SHOGI) and Yamashita, author of the YSS, who gave some details about their programs in Japanese ([5],[12]). My work is in part inspired by these publications, but also aims at giving a more solid basis for the use of plausible move generators in shogi.

In earlier work, I presented a plausible move generator based on Candidate Relevance Analysis or CRA [3]. CRA analysed the meaning of moves regarding the goal of the game, discarding all moves for which no meaning could be found. CRA was very conservative about the moves to be discarded to make the chance of losing vital moves as small as possible. As a result, too many moves remained to be searched and the cuts resulting from CRA turned out be insufficient to allow deep search.

In this paper I present a first step in the construction of a plausible move generator for shogi which can discard enough moves to reach search depths comparable to those in strong chess programs. The 13 ply depth of chess programs seems a good target, since the strongest shogi programs are not yet able to search this deep, reaching only 9 to 11 ply. The plausible move generator described here is based on the information collected by a static evaluation of shogi positions.

2 A Plausible Move Generator for Shogi

To build a plausible move generator for shogi, it is necessary to understand when a move is a plausible shogi move. Evaluation of the current position, for example to find mating threats, forks, attacked pieces, favourable material exchanges and so on are vital to understand which moves are likely in a position. To better understand what types of moves are most often played in shogi, the meaning of the moves of the 10 test games were analysed by hand. If a move had more than one meaning, the most important meaning was picked. If it was hard to decide between the importance of different meanings, one was picked at random. The results of this analysis are presented in Table 2.

It is surprising that all moves of the test games can be described by only 11 basic move categories. It is also interesting to note that the number of sacrifices that do not fit into any other category is quite low. Only 1.0% of the moves were sacrifices of material that can not be explained by any of the other move categories.

From the table it can also be concluded that shogi is a very tactical game. Only 30.9% of the moves in the test games belongs to the category *Strategic Move*. All other moves belong to a category with a tactical meaning.

The only previous study into the classification of moves in shogi by using professional game records is by Kotani and Iida [7]. They give a more general classification of shogi moves, distinguishing only between drops, captures, promotions and "other" moves. This classification does not seem sophisticated enough to give good suggestions for pruning moves. Kakinoki [5] uses only 8 basic move categories, but all of the 11 categories in Table 2 are implicit in his categorisation. Yamashita [12]

Move Type	PMG	1	2	3	4	5	6	7	8	9	10	Tot	%
Strategic Move	(√)	36	30	46	29	48	49	48	49	32	25	392	30.9%
Attack Piece	√	15	22	19	15	24	20	19	16	23	15	188	14.8%
Win Material	√	16	30	15	14	21	18	12	15	28	13	182	14.4%
Attack King		23	29	7	0	17	11	9	17	27	16	156	12.3%
Defend King		15	31	8	1	17	8	4	12	22	14	132	10.4%
Defend Piece	√	10	10	10	10	13	10	12	12	12	10	109	8.6%
Exchange Material	√	3	7	7	6	2	5	6	6	5	5	52	4.1%
Threaten Promotion	√	1	1	1	6	1	5	1	6	0	2	24	1.9%
Promote Piece	√	3	2	1	1	2	0	1	4	1	1	16	1.3%
Sacrifice Piece		1	3	1	0	0	2	0	0	3	3	13	1.0%
Defend Promotion	√	0	0	0	0	1	0	0	2	1	0	4	0.3%

Table 2: Move meaning classification in the 10 test games. *PMG* are the categories that have been implemented in the plausible move generator described in this paper.

describes a much more sophisticated scheme, where the generation of moves is based on the number of plies left to search. For example, a move that attacks a piece is not generated if the remaining search depth is only a single ply. The idea is that the advantages of such a move can not be evaluated in a single ply of search. The 29 move categories that Yamashita describes are all subcategories of the categories of Table 2. Such a detailed analysis of move meaning improves the quality of the plausible move generator, but also takes more time. To find out which categorisation works best is one of the subjects of our research.

As a first step, I have looked at the move categories relating to loss and gain of material. That is, I have implemented a move analyser that recognises moves in all of the categories in Table 2 except *Strategic Move*, *Attack King*, *Defend King* and *Sacrifice Piece*. Actually, the category *Strategic Move* is also partially implemented. The CRA-based plausible move generator used previously had a number of positional patterns to recognise strategic moves. These patterns are still present in the current plausible move generator, and give an indication of the performance of the program in non-tactical positions.

To analyse the meaning of moves relating to win and loss of material, a static capture evaluator of possible capture sequences is used (briefly described in [11]). With this evaluator, for every square on the board it can be analysed statically if a piece on the square is attacked, if there is a favourable capture or if there is a promotion threat. To this I have added rules to decide if the intended defence of the piece or defence against a promotion threat improved the capture value, thereby introducing a static 1-ply lookahead. Only moves improving at least one of the square values given by the capture evaluator, are generated by the plausible move generator.

The static 1-ply lookahead has a number of implementation problems, as the examples in Figure 2 show. In the left diagram of Figure 2, the knight on 6i is attacked by the lance on 6g, but can be defended indirectly by moving the rook to 6f. This type of indirect defence requires a detailed analysis of the position. In the middle diagram, the silver on 7g is attacked by the two bishops. The problem here is to recognise that moving the silver to 8h is not a sufficient defence against the white threat. In the right diagram of Figure 2, the problem is to decide whether the bishop on 7g is attacked or not. The knight on 8i is pinned, but the pinning piece (the rook on 7i) is also the only piece attacking the bishop, so the bishop is actually not threatened.

These are just a few of the problems that adding a static 1-ply lookahead to a static capture evaluator faces. However, static capture analysis is just the start of what has recently become the core of a strong shogi program. For example, Rollason's SHOTEST also has a sophisticated static capture evaluator called SUPER-SOMA [9]. Rollason's program has done very well in recent computer shogi tournaments.

My current static capture evaluator can deal with all single-square captures with and without promotions, pinned pieces that lead to capture of the king (in contrast to pinned pieces that lead to major loss of material) and promotions. It can deal with the first two of the three cases of Figure 2, but the rules for dealing with the case in the rightmost diagram have not been implemented yet.

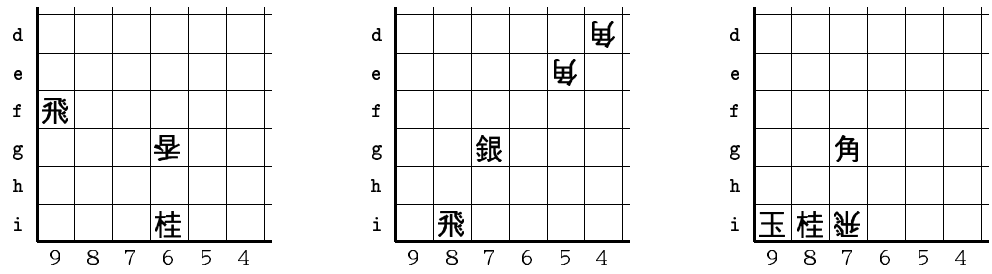

Figure 2: Examples of problems with static 1-ply lookahead

No	#Legal	#Gen	Sv(%)	NG	%
1	10003	3323	66.7%	11	8.9%
2	15793	4520	71.3%	26	15.7%
3	8146	2203	72.9%	22	19.1%
4	4310	1172	72.8%	7	8.5%
5	11248	3694	67.1%	21	14.3%
6	8399	2856	65.9%	20	15.6%
7	6396	1804	71.7%	11	9.8%
8	7802	2443	68.6%	18	12.9%
9	14261	4485	68.5%	18	11.7%
10	8221	1874	77.2%	23	22.1%
Tot	94579	28374	70.0%	177	14.0%

Table 3: Results of the plausible move generator in the 10 test games. *Legal* is the total number of legal moves, *Gen* is the number of moves generated by the plausible move generator, *Sv* is the savings of the method, and *NG* is the number of moves not generated by the plausible move generator

3 Results

To test the performance of the plausible move generator, we have compared the expert moves played in the test games with the moves generated by the plausible move generator. In Table 3 the results of this analysis have been given. It can be seen that the current plausible move generator saves between 65.9% and 77.2% of the total number of legal moves for an average of 70.0%. The cost for this is between 8.5% and 22.1% of the moves played by the expert player, for an average of 14.0%.

Losing a considerable number of the moves played by the expert player was unavoidable, since the plausible move generator is only a partial implementation. Therefore, the moves not generated must be analysed in more detail. In Table 4, the moves are classified by the move categories introduced previously. As expected, the categories *Strategic Move*, *Attack King*, *Defend King* and *Sacrifice Piece* are responsible for most of the moves not generated, namely 93.6%. The remaining moves point to errors in the move generator. One of these problems is the correct analysis of pinned pieces, as described in Figure 2.

Note that the plausible move generator is often able to generate multi-purpose moves. For example, there are 156 moves in the 10 test games categorised as *Attack King*. Of these moves, the current plausible move generator is able to generate 97 moves even though there are no rules for recognising attack moves. This is because most attacking moves are also at the same time a promotion, threaten a piece, threaten promotion or win/exchange a piece.

Move Type	1	2	3	4	5	6	7	8	9	10	Tot	%
Strategic Move	4	3	14	6	8	11	8	7	8	3	72	40.7%
Attack Piece	0	0	1	0	2	0	0	0	0	4	7	4.0%
Win Material	0	0	0	0	2	0	0	0	0	1	3	1.7%
Attack King	3	16	4	0	3	6	2	9	5	11	59	33.3%
Defend King	3	4	2	1	5	2	1	1	2	1	22	12.4%
Defend Piece	0	0	0	0	0	0	0	0	0	0	0	0.0%
Exchange Material	0	0	0	0	0	0	0	0	0	0	0	0.0%
Threaten Promotion	0	0	0	0	0	0	0	0	0	0	0	0.0%
Promote Piece	0	0	0	0	0	0	0	1	0	0	1	0.1%
Sacrifice Piece	1	3	1	0	0	2	0	0	3	3	13	7.3%
Defend Promotion	0	0	0	0	0	0	0	0	0	0	0	0.0%

Table 4: Move analysis of the moves not generated by the plausible move generator.

Pr	Move Generator		SU	Same
No	Without	With	fact	Result?
1	5:32	0:33	10	◯
2	0:39	0:01	39	◯
3	2:17	0:08	17	◯
4	17:53	1:18	14	◯
5	6:52	0:11	37	×
6	4:32	0:30	9	◯
7	3:22	0:24	8	◯
8	1:05	0:12	5	◯
9	2:57	0:16	11	◯
10	8:46	0:58	9	◯

Table 5: Time taken for 5 ply searches in 10 tactical shogi problems with and without a plausible move generator. *SU fact* is the speed up factor.

As a second test, 10 shogi problems were selected to see how much the search performance would improve if a plausible move generator was used. For this test, all shogi problems had a solution which leads to a material advantage for the side to move. Our shogi program is an iterative alpha-beta searcher using PVS/Negascout with transposition tables and the history heuristic. The full-width version of this algorithm can only search 5 ply deep on a 450 MHz Pentium when given a time limit of 20 minutes. Results of a 5 ply search with and without the plausible move generator are given in Table 5.

The results for these 10 test problems are a strong indication that the addition of a plausible move generator will have an important impact on the performance of the search. Even in the worst case, search with the plausible move generator is more than 5 times faster than full-width search, and in most cases more than 10 times faster.

We can also see a risk of the approach. In one of the test positions (number 5) the solution move was a sacrifice that the plausible move generator did not generate. Therefore, the solution to this problem could not be found.

We also see that the search is still not fast enough for good tournament play. Under some circumstances, even a 5 ply search will not be able to finish within 25 seconds.

4 Conclusions

Plausible move generators are a vital part of strong shogi programs. It is often unclear how these move generators are developed. This paper is a first attempt to use the analysis of professional games as the basis for a plausible move generator. The plausible move generator presented here deals with the generation of moves leading to a possible win or loss of material. The analysis of these types of moves is based on a static capture analysis with a static 1-ply evaluation of the moves defending against a material threat.

The plausible move generator presented here gives important savings in the moves that need to be searched. Preliminary results indicate that this will lead to an important improvement of the search performance in shogi programs.

However, the plausible move generator needs to be further expanded to deal with moves attacking and defending the king. Also, the current plausible move generator needs to be improved, since preliminary results indicate that the number of move generated is still too large to search deep enough for strong play under tournament conditions.

References

[1] Japanese Shogi Federation. *Shogi Yearbook Heisei 10.* Nihon Shogi Renmei, 1999.

[2] R. Greenblatt, D. Eastlake III, and S. Crocker. The Greenblatt Chess Program. In *Proceedings of the Fall Joint Computer Conference*, pages 801–810, 1967.

[3] R. Grimbergen. Candidate Relevance Analysis for Selective Search in Shogi. In *9th Advances in Computer Chess Conference*, Paderborn, Germany, 1999.

[4] E.A. Heinz. DARKTHOUGHT Goes Deep. *ICCA Journal*, 21(4):228–229, December 1998.

[5] G. Kakinoki. The Search Algorithm of the Shogi Program K3.0. In H. Matsubara, editor, *Computer Shogi Progress*, pages 1–23. Tokyo: Kyoritsu Shuppan Co, 1996. ISBN 4-320-02799-X. (In Japanese).

[6] R.E. Korf. Does Deep Blue Use Artificial Intelligence? *ICCA Journal*, 20(4):243–245, December 1997.

[7] Y. Kotani and H. Iida. Which Moves should be Pruned? –Classification of Shogi Moves and the Rate of Played Moves. In *Game Programming Workshop in Japan '95*, pages 148–156, Kanagawa, Japan, 1995. (In Japanese).

[8] H. Matsubara and K. Handa. Some properties of Shogi as a Game. *Proceedings of Artificial Intelligence*, 96(3):21–30, 1994. (In Japanese).

[9] J. Rollason. Personal communication, 1999.

[10] D. Slate and L. Atkin. Chess 4.5: The Northwestern University Chess Program. In P. Rey, editor, *Chess Skill in Man and Machine*, pages 82–118. Springer Verlag, New York, 1977.

[11] P. Van Diepen and J. Van den Herik. *Schaken voor Computers (Chess for Computers).* Schoonhoven, The Netherlands: Academic Services, 1987.

[12] 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).