Pattern Recognition for Candidate Generation in the Game of Shogi

Reijer Grimbergen and Hitoshi Matsubara
Electrotechnical Laboratory,
1-1-4 Umezono, Tsukuba, Ibaraki, Japan 305.
E-mail: grimberg, matsubar@etl.go.jp

Abstract

Chess has been an important research area in Artificial Intelligence for decades. It seems that the strongest programs will soon play better than the best experts, so it is time to look at other game domains as possible test domains for AI research. We feel that shogi is well suited for this, because it is similar to chess, yet significantly different. In this paper we give the main differences between chess and shogi and possible solutions for problems arising from these differences. It is our assumption that all problems can be solved using pattern recognition. We describe a method for pattern recognition to do this and give some preliminary results for candidate generation using pattern recognition. For candidate generation, our pattern recognition program currently reduces the number of candidates generated by at least 77% and generates the optimal move in more than 75% of the test positions, using about 5,000 patterns. To play shogi at a high level these numbers need to be improved upon, but as a preliminary result this is encouraging.

Content areas: Pattern recognition, search, candidate generation, shogi

1 Introduction

In chess it is becoming clear that the combination of full width search techniques combined with pruning heuristics and fast (sometimes even special purpose) hardware might be enough to play the game at the level of the best human players. At the moment of writing it is still unclear whether the program Deeper Blue will be able to achieve this result by beating world champion Kasparov. However, its predecessor Deep Blue already scored a win under tournament conditions last year, so it will be only a question of time before no human player will be a match for the strongest programs. For a review of the development of chess research and last year's match between Kasparov and Deep Blue, refer to [Newborn 1997].

With this milestone in sight, it is unclear in what direction chess research will develop. Even with the best human players defeated, the ultimate goal of playing perfect chess may still be very distant. However, we feel that one of the main reasons for the progress of computer chess is the competitive element of building stronger and stronger programs and matching them up with human players. Therefore, it is time to think about the advantages of using other domains for developing and testing AI search techniques. Chess has shown the advantages of using a game for this purpose, so we have looked at the potential of other games for AI research. We have compared the complete information games chess, xiang qi (Chinese chess), shogi (Japanese chess) and Go [Matsubara et. al. 1996]. Our conclusion was that even though Go is probably the best domain in the long term, shogi would be well suited as the next target for AI.

In [Matsubara & Grimbergen 1997], we have given a detailed comparison between chess and shogi from a computational point of view, so here we will only review the most important points. The most important similarity is that in shogi, as in chess, mating the opponent's king is the ultimate goal of the game. However, the shogi board is slightly bigger (9x9) than a chess board and different pieces with different move possibilities are used. Shogi has no queen, but the king, rook and bishop move in the same way as their corresponding pieces in chess. The biggest difference between chess and shogi is that in shogi the pieces captured from the opponent can be put back on the board as a move (this type of move is called a *drop*).

The fact that shogi is very similar to chess makes it a reasonable assumption that results obtained from chess research will carry over to shogi. This is less likely for a completely different game like Go.

If shogi is so similar to chess, should it not be possible to change the game rules in the Deep Blue program today and have a world class shogi program tomorrow? In fact, this seems unlikely, due to the possibility of the re-use of pieces. Since there are only few exceptions to the dropping rule, the branching factor and the game tree complexity of shogi are considerably higher than those of chess. The game tree complexity of shogi is estimated at 10^{226}, based on an average branching factor of 80 [Matsubara & Handa 1994] and an average game length of 115 ply [Japanese Shogi Association 1994]. In contrast, the game tree complexity of chess is estimated at 10^{123}, based on an average branching factor of 35 [Hartmann 1989] and an average game length of 80 ply. Since the average branching factor of shogi is more than twice that of chess, we do not believe that the AI techniques used in chess will be as successful in shogi.

Strong chess programs combine a full width search and special hardware to search as deep as possible. Management of the search tree is based on pruning lines of play that are not promising, but this pruning is not done at the point were the next candidate moves are being generated. Thus, for building a program to play shogi at a high level, we need different pruning techniques to deal with the high branching factor.

In Section 2 we will describe how the differences between chess and shogi influence the different stages of the game. We will also give possible solutions to solve the problems arising from these differences. We will argue that in all cases pattern recognition can be used to solve these problems. In Section 3 we will describe our pattern recognition method for limiting the number of candidate moves generated in a shogi position. In Section 4 we then give the preliminary results of our pattern recognition program for candidate generation when tested on professional shogi games. Finally, Section 5 discusses our results and describes our intentions for future research.

2 Differences with Chess and Possible Solutions

As stated in the introduction, shogi is a game very similar to chess. However, the differences in piece movement and the possibility of dropping pieces have important consequences for building a shogi playing program. Below, we discuss the implications of these differences and possible solutions for problems resulting from these differences for some key concepts in game playing and search algorithms.

2.1 Opening

In shogi, most pieces have only limited movement. There is no piece as powerful as the queen in chess and the pawns can only move one square at a time. Add to this that the shogi board is bigger than the chess board and it should not be too surprising that in shogi the early stages of the game do not quickly lead to clashes between the opposing pieces. In a real shogi game each player usually tries to build a position he prefers, and does so much more independently of the moves of the opponent than would be possible in chess. This makes building an opening library like the ones used in chess problematic: there are no forced lines of play, and slightly different move orders may or may not lead to completely different positions. We built an opening database of about 12,000 positions, but it turned out that it was very easy to get around this opening library. That this is not a programming problem but part of the game of shogi is shown by the number of new opening strategies that are still introduced by professional players today (some of them as early as on the first move!).

One possible solution is to use transposition tables, but it is unclear if they would work well in shogi. For example, one of the killer heuristics in chess to limit space used for transposition tables is to stop when a piece is captured. Because of the possibility of dropping pieces in shogi, this heuristic can not be used and transposition tables might take up too much space.

Another solution is to use pattern recognition for generating candidate moves in the opening. Certain moves and certain castle formations are particularly suited for certain opening plans. Furthermore, evaluation of an opening position in shogi is usually based on positional features like the protection of the king and the potential of the attacking formation that is built. This makes pattern recognition suitable for the shogi opening. Only for the limited number of opening traps and in the few cases quick attacks can be played, an opening library is needed.

2.2 Middlegame and Endgame

The middle game of shogi usually is a collection of small, standard move sequences to quicken one's attack, slow down the opponent's attack, displace the opponent's pieces, promote pieces and winning material. The problem usually is to decide which of these *tesuji* is the most effective and in what order different tesuji should be played. Thus, the shogi middle game is well suited for pattern recognition. When tesuji are defined by patterns, candidate moves can be generated by looking at the features of the position. Standard search techniques can then be used to evaluate which of the tesuji works best.

However, evaluation of a shogi position is a little different from evaluation in chess. The limited movement of shogi pieces plays an important role here as well. Pieces that have been misplaced are hard to move back to better squares and are therefore prone to getting 'stuck' on useless squares where they can not participate in attack or defense. This has an important influence on the evaluation function. In chess, it is possible to make a reasonable evaluation function that is based almost completely on material. Also, it is possible to stop searching when a known book endgame position is reached (the material balance then defines the outcome of the game). However, in shogi the outcome of the game rarely depends on the material balance in this way. Because of the dropping of pieces, mate is the only way a shogi game can finish. Therefore, an evaluation function in shogi can only correctly judge a position if it is able to assess each pieces' potential for participating in attack or defense.

We feel that an evaluation function for shogi must put more emphasis on the positional features of a position than an evaluation function for chess. There is a connection with pattern recognition here as well. After all, an evaluation function that looks at the configuration of pieces to decide which pieces are well positioned and which pieces are not, is also doing pattern recognition.

Shogi programs usually rely on a special mating searcher for the final stages of the game. These sequences of continuously giving check and finally mating the king are the basis for the popular *tsume shogi* problems. Examples of such mating problems are frequently published in books and magazines. These problems can have solution sequences from three to several hundred ply. Until about five years ago, it was hard for computers to solve tsume problems with solution sequences longer than 10 ply, mainly because of the possibility of defensive drops. Recently, however, tsume shogi programs have outperformed the

experts [Ito et. al. 1995]. Most shogi programs use a tsume searcher and seem to use it well. Mating sequences of more than 20 ply are no exception. From a research point of view, the only remaining question seems to be at what point such a tsume searcher should be invoked.

The most important problem in the endgame is the stage just before mating. That is, bringing the king closer and closer to mate by using forcing attacking moves. Recent research [Iida & Abe 1996] has focused on this problem, which is vital to play the shogi endgame at a high level. We think that in this stage of the game pattern recognition can be used to suggest attacking and defending moves.

2.3 Tree Search

From the discussion above it follows that pattern recognition in shogi should focus on candidate generation and position evaluation. Standard tree search techniques can then be used to find the best move in a certain position. Therefore, no changes are needed in the well established scheme of a mini-max quiescence tree search with alpha-beta pruning that is used in almost all strong chess programs [Birmingham & Kent 1977][Harris 1975]. The only alteration is that in quiescence search the search should not only be deepened in case of checks and captures, but also to search for mate. This to accommodate the importance of mating in a shogi endgame.

3 Pattern Recognition for Candidate Generation

From the previous section we can conclude that there are two main areas where shogi is different from chess. One is the candidate generation in the different stages of the game and the other is the evaluation of shogi positions. We first focused on the problem of generating candidates by using pattern recognition. We think that in order to build a program that plays shogi at a high level, it is necessary to improve the quality of the candidate generation, thus decreasing the branching factor. This is consistent with the well-known cognitive result that the difference between strong and weak chess players is not based on the amount of search they perform, but on the number of candidate moves they consider [De Groot 1965].

To test the feasibility of pattern recognition in shogi, we first built a program that could solve tsume shogi [Grimbergen 1996]. We showed that it was possible to solve simple problems (5 and 7 ply) quickly. Here we we will describe how we adapted this approach to make it part of a full shogi playing program.

In research using chess patterns to generate candidate moves, usually only the actual configuration of a subset of the pieces on the board is used as a pattern [Berliner & Ebeling 1989][Levinson & Snyder 1991][Walczak 1992][Wilkins 1980]. Sometimes a tutorial component is added where an expert can enter piece configurations and rules for how to deal with them [Donninger 1996]. However, one of the theoretical problems of pattern recognition is that too many patterns need to be defined and searched, leading to slow and inaccurate results. Also, there is cognitive evidence [De Groot 1965][Chase & Simon 1973] that expert players not only use piece configurations, but also knowledge about the possible movement of pieces.

We have defined our patterns in such a way that we can deal with functional knowledge like this. The basic idea is not to define patterns as actual combinations of pieces, but to describe patterns as a list of features. This is a common way to make abstract descriptions of patterns [Watanabe 1985].

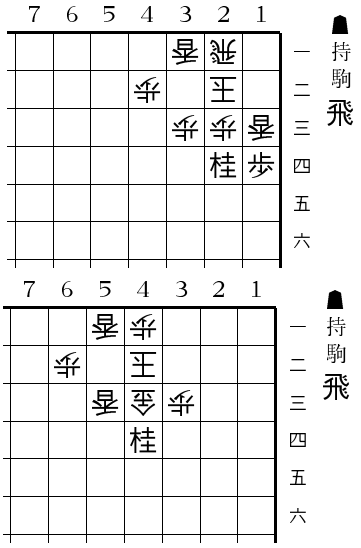

Diagram 1: Two similar mating problems

Let's look at an example to illustrate this idea. In diagram 1 we see two simple tsume shogi problems. It is not important to find the solution, nor to fully grasp the meaning of the kanji characters used in shogi diagrams. The only thing that matters is that the position in the top half of the diagram is different from the piece configuration in the bottom half. The bottom position is different both in the position of the pieces on the board and in the actual pieces on the board. However, the feature description of both positions as defined in our program is the same for both positions. Edited a little to make it more readable, the pattern features looks like this (here K_x and K_y are the horizontal and vertical position of the king):

- In_hand(R)
  "There is a rook in hand"
- Protected(K_x-1, K_y)
  "The drop square is protected"
- Free(K_x-1, K_y)
  "A piece can be dropped to the right of the king"
- ¬Defended(K_x-1, K_y)
  "The drop square is not defended by any piece other than the king"
- Can_not_move(K_x, K_y+1)
  "The king can not move straight up"
- Can_not_move(K_x, K_y-1)
  "The king can not move straight down"

- Can_not_move(K_x+1, K_y+1)
  "The king can not move up diagonally to the left"
- Can_not_move(K_x+1, K_y-1)
  "The king can not move down diagonally to the left"

By encoding this functional information it is possible to have one feature description that will fit many actual piece configurations, thus reducing the number of patterns that have to be defined.

Another way to improve the performance of pattern recognition is to describe a pattern relative to a piece on the board (in our program this is called an *anchor*). In the example all features except the first one have a reference to the position of the king. This means that instead of searching all patterns, only the patterns with an anchor piece on the board need to be searched. In shogi this usually means better performance in the endgame, when more pieces are likely to be in hand, not on the board. Because in chess captured pieces can not return in the game, it is likely that this approach will work even better in chess than in shogi.

To account for the differences between the opening, middlegame and endgame described in section 2, we have divided the patterns into three types. In the opening, building a position is done with actual piece configurations. To reflect this, our opening patterns do not have the functional component of the example above. As a consequence, opening patterns do not have a piece as an anchor. Instead, they use the classification of opening strategy and castle formation as anchor to guide the search for patterns. Using this type of anchor does not make candidate generation of opening patterns slower than the generation of middlegame and endgame patterns. On the contrary, because there are many more opening strategies than there are different pieces, the candidate generation is actually faster.

Middlegame patterns and endgame patterns are of the type described in the example. Apart from piece anchors, middlegame and endgame patterns also use the castling formation as anchor for patterns. However, in the endgame there are more patterns to deal with the attack of the opponent's king, with many patterns suggesting the sacrifice of pieces to weaken the defense.

The transition between finding candidate moves from opening, middlegame and endgame patterns is not clearly defined. Middlegame patterns are currently generated as soon as the number of moves suggested from opening patterns gets below a certain threshold. For the transition between middlegame and endgame patterns we are comparing several possibilities, but at the moment it is too early to report which of them works best.

In our program we have thus far coded about 5,000 patterns. These patterns are divided into opening patterns (± 1,900), middlegame patterns (± 1,500) and endgame patterns (± 1,600).

4 Results

To test the feasibility of our approach, we collected data for the candidate generation of five professional games. There are no quick games among the games and all were played by top level professionals. Thus it can be expected that the move played in the game is either the best move or a very reasonable one. Also, the types of positions played in each of these games were different. The collected data is represented in table 1

No	1	2	3	4	5
NoMv	154	105	98	101	93
Totmv	10162	8293	5759	7752	6988
Mv avg	66.0	79.0	58.8	76.8	75.1
Totcnd	2350	939	877	1159	1017
Cnd avg	15.3	8.9	9.0	11.5	10.9
Ratio	23.1	11.3	15.2	14.9	14.6
Hit	137	79	77	82	81
Hit %	89.0	75.2	78.6	81.2	87.1
MaxBr	171	218	129	167	182
MaxCnd	55	33	36	46	61

Table 1: Candidate generation data

This table contains the following information:

No Game number. For shogi players: Game 1 is a *Yagura* game, game 2 is an *Aigakari* game, game 3 is an *Ibisha against Shikenbisha* game, game 4 is a *Yokofudori* game and game 5 is a *Kakugawari bogin* game.

NoMv Number of moves of the game in ply. The average of 110.2 is a little lower than the average of 115 mentioned in the Shogi Yearbook, but this difference is not significant.

Totmv Total number of legal moves in the game.

Mv avg Average branching factor. In all cases this is lower than the average branching factor of 80 found by Matsubara et. al. The exceptionally low numbers of game 1 and game 3 can be explained as follows. Game 1 has an unusual high number of checks (20), which accounts for the relatively low number of legal moves. In game 3 there is an exceptionally low build up of the game. The first non-pawn is taken as late as on ply 48. Before that, the branching factor never got over 50.

Totcnd Total number of candidates generated. These are the candidate moves generated by the program.

Cand avg Average number of candidates generated per move. This number indicates that a search tree using this type of candidate generation will on average have branching factor between 9 and 16. This is well below the average branching factor of chess.

Ratio Percentage of total moves generated as candidates. This describes the number of candidates cut from the search, i.e. the merit of this approach. We see that we can expect cuts from about 75% to 90%.

Hit Number of positions were the game move was part of the candidates. This is an indication of the costs of this approach. If an optimal move is not among the candidate moves, this means

that however well the evaluation function and the tree search work, the program will not play the best move.

Hit % Percentage of hits. This shows that in the worst case, we can expect that the optimal move is not generated in 25% of the cases.

MaxBr The maximum number of legal moves in the game. An indication of how high the branching factor could get if we do not use some way of cutting the number of candidates. Actually, it has been shown [Nozaki 1990] that the maximum branching factor in shogi is 593.

MaxCnd The maximum number of candidates selected. This is an indication of the worst case branching factor in a search tree. With the current candidate generation, it is possible that more than 60 candidates are generated, which is far more than the average branching factor of chess.

5 Conclusions and Further Research

In the ideal case, we would like to find rules to cut the number of candidates while at the same time keeping a hit rate of 100%. That it is possible to find such an ideal cutting scheme is not very likely, but also not necessary. Human experts play shogi at a very high level, but still overlook moves by the opponent. And surely, when they are beginners, they overlook much more than when they are experts. We like our program to have the same kind of development (albeit a little quicker if possible). This is why we feel that a hit percentage between 75% and 90% is encouraging considering that we used only about 5,000 patterns to get this result. It has been estimated that expert chess players have knowledge of about 50,000 patterns [Chase & Simon 1973].

However, a shogi playing program that is at a loss in about one of every four positions it encounters is not very likely to play well. We estimate that a hit percentage of around 95% is necessary to play the game of shogi at a high level, but it is unclear how many patterns need to be added to get close to this number. In any case it would be nearly impossible to add all patterns by hand, so we intend to add learning capabilities to the program in the future.

However, first we need to further balance the candidate generation. The current reduction in candidates generated is in our view not enough to cut the search tree in such a way that it is worth the extra cost of searching for candidates from patterns. We need to speed up the candidate search (it now almost takes a second of computation time on a 200MHz Pentium to generate candidates from patterns) and at the same time improve the quality of the candidates generated. That is, we need more hits and fewer candidates.

It might also be a good idea to order the moves prior to the search in order to increase the chance of finding a reasonable move when the search is cut off because of time constraints. One idea would be to give moves generated by several patterns a higher search priority. Until now, this is not part of our program.

Also in the near future we need to design an evaluation function according to the ideas outlined in section 2. It will be interesting to see if the knowledge used by a positional evaluation function could also have a relation with the pattern recognition part of the program.

What is not being discussed in this paper is a comparison with other research in shogi. The reason for this is that until recently, research into game programming in general and research into shogi in particular has not been taken seriously in Japan. In 1987 the Computer Shogi Association (CSA) was founded, which organizes the annual Computer Shogi Championships since 1990. These tournaments have been dominated by professional game programmers from the start. In 1997 there were 33 entries, but only three of them were research programs (one of which was our program). The other two research programs did not use pattern recognition in their search. Information about the commercial (and strongest) programs is scarce, but from personal communication one can conclude that even though there is a difference in the amount of shogi knowledge they use, none of them uses pattern recognition. In short, it is impossible to compare our work with the work of others in shogi and we need to rely here on the similarity between our work and work previously done in chess.

The annual CSA tournament is a good possibility to evaluate the advances of our work. So, by adding the ideas outlined in this paper we hope to make a program that will do better than its predecessor HEISHINKU, which took part in the 1997 Computer Shogi championships and scored three wins and four losses.

References

[Berliner & Ebeling 1989] Berliner, H. & Ebeling, C. (1989). Pattern Knowledge and Search: The SUPREM Architecture. *Artificial Intelligence, 38*, 161-198.

[Birmingham & Kent 1977] Birmingham, J. & Kent, P. (1977). Tree-searching and Tree-pruning Techniques. In M.R.B. Clarke (Ed.), *Advances in Computer Chess 1* (pp. 89-97). Edinburgh: Edinburgh University Press.

[Chase & Simon 1973] Chase, W.G. & Simon, H.A. (1973). Skill in Chess. *American Scientist, 61*, 394-403.

[De Groot 1965] De Groot, A.D. (1965). *Thought and Choice in Chess*. The Hague, The Netherlands: Mouton & Co.

[Donninger 1996] Donninger, C. (1996). CHE: a graphical language for expressing chess knowledge. *International Computer Chess Association Journal, 19(4)*, 234-241.

[Grimbergen 1996] Grimbergen, R. (1996). Using Pattern Recognition and Selective Deepening to Solve Tsume Shogi. *Proceedings of the Game Programming Workshop in Japan '96*, 150-159.

[Harris 1975] Harris, L. (1975). The Heuristic Search and The Game of Chess. A Study of Quiescence, Sacrifices, and Plan Oriented Play. *Proceedings of*

the 4th International Joint Conference on Artificial Intelligence, 334-339.

[Hartmann 1989] Hartmann, D. (1989). Notions of Evaluation Functions Tested against Grandmaster Games. In Beal, D.F. (Ed.), *Advances in Computer Chess 5*, 91-141. Amsterdam, The Netherlands: Elsevier Science Publishers.

[Iida & Abe 1996] Iida, H. & Abe, F. (1996). Brinkmate Search. *Proceedings of the Game Programming Workshop in Japan '96*, 160-169.

[Ito et. al. 1995] Ito, K., Kawano, Y., Seo, M. & Noshita, K. (1995). Recent progress in solving Tsume Shogi by computers. *Journal of the Japanese Society for Artificial Intelligence, 10(6)*, 853-859. (in Japanese).

[Levinson & Snyder 1991] Levinson, R. & Snyder, R. (1991). Adaptive pattern-oriented chess. In Birnbaum, L.A. & Collins, G.C. (Eds.), *Proceedings of the 8th international workshop on machine learning (ML-91)*, 85-89.

[Japanese Shogi Association 1994] Japanese Shogi Association (ed.) (1994). *Shogi Nenkan [Year Book of Shogi]*. Tokyo, Japan: Japanese Shogi Association. (in Japanese)

[Matsubara & Handa 1994]
Matsubara, H. & Handa, K. (1994). Some properties of Shogi as a Game. *Proceedings of Artificial Intelligence, 96(3)*, 21-30. (in Japanese).

[Matsubara et. al. 1996] Matsubara, H., Iida, H. & Grimbergen, R. (1996). Natural Developments in Game Research. *International Computer Chess Association Journal, 19(2)*, 103-112.

[Matsubara & Grimbergen 1997] Matsubara, H. & Grimbergen, R. (1997). Differences between shogi and western chess from a computational point of view. *International colloquium on board games in academia 2.*

[Newborn 1997] Newborn, M. (1997). *Kasparov versus Deep Blue, Computer chess comes of age.* New York: Springer.

[Nozaki 1990] Nozaki, A. (1990). *Logical Shogi Introduction.* Chikuma-shobo, Tokyo. (in Japanese)

[Walczak 1992] Walczak, S. (1992). Pattern-based tactical planning. *International Journal of Pattern Recognition and Artificial Intelligence, 6(5)*, 955-988.

[Watanabe 1985] Watanabe, S. (1985). *Pattern recognition: human and mechanical.* New York: John Wiley & Sons.

[Wilkins 1980] Wilkins, D. (1980). Using Patterns and Plans in Chess. *Artificial Intelligence, 14*, 165-203.