Using Pattern Recognition and Selective Deepening to Solve Tsume Shogi

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

Abstract

In chess brute-force search methods have been very successful. However, games like shogi have a game tree complexity that is much higher than chess. Therefore, it is unclear if the methods used for making strong chess programs will also be successful for other games. In our research we will attempt to use a more cognitively based method, pattern recognition, to build a strong shogi playing program. In chess, pattern recognition has not been very successful, but in this article we will take a different approach to patterns. We will describe a more general definition of patterns and how we used this definition and position evaluation to make a program to solve simple tsume shogi problems. The results for solving five-ply and seven-ply tsume shogi problems are encouraging. Furthermore, an evaluation function using only pattern recognition turned out to be working better than an evaluation function using both pattern recognition and position evaluation.

1 Introduction

Ever since Shannon's famous 1950 article "Programming a Computer for Playing Chess" [24], chess has been an important research subject in Artificial Intelligence. Early on, it was considered vital to the strength of a program to include as much chess knowledge as possible and the cognitive aspects of chess ability [7] were expected to be part of every strong chess playing program. However, chess research took a dramatic turn after Slate and Atkin, the programmers of the famous program CHESS, in 1973 more or less by accident discovered that speed made a chess program much stronger than knowledge. Since then, chess programs have focused on searching as much of the game tree as possible as fast as possible. Most programs use a simple mini-max algorithm with alfa-beta pruning [5], enhanced by other programming tools like quiescence search [3, 9], conspiracy numbers [21] and singular extensions [1]. The success of this approach is undeniable. In twenty years, chess programs have improved from weak amateur level to a level in which they can even give the world champion a scare [26]. The program to do so, Deep Blue, is capable of analysing 500,000,000 positions per second, looking ahead some 13 plies on average. From an engineering point of view, this is a remarkable achievement with concequences that far surpass the limited domain of chess research.

However, with such a large number of evaluated positions, there is no question that the link with human problem solving and thus with classic AI is lost. Chase and Simon [6] have shown more than 20 years ago that the difference between expert chess players and novices does not lie in the number of positions evaluated, but in the recognition of patterns (or chunks) in a chess position. Stronger players encode a chess position in larger chunks. In chess research there have been some attempts to use this psychological fact for building strong chess programs [4, 15, 27, 29], but these programs have been a minority and their success in actual game playing has been limited.

At the moment it is unclear what will happen with chess as a research domain after the strongest programs have surpassed the level of the strongest human experts. Chess might lose all its attraction for future research. Another possibility is that chess research will advance without human opposition, implying that the strongest programs will only play each other. The final possibility seems to be that other directions for chess research get more attention, leading chess research back to its AI roots.

Because future developments in chess are uncertain, we feel that it is time to look at other games as potential topics for research. In [17] we have compared chess, xiang qi, go and shogi and argued that especially shogi (Japanese chess) is a suitable topic for future research. Shogi has a game tree complexity that is considerably larger than that of chess. The branching factor of chess is estimated at 35 [10], while the branching factor of shogi is estimated at 80 [16]. As a result, chess has a game tree complexity of $\pm 10^{123}$ and shogi has a game tree complexity of $\pm 10^{226}$. On the other hand, it was argued that shogi and chess are very similar and that results from chess research can be expected to extend to the domain of shogi. Therefore, shogi is a similar, but a considerable more complex domain, so it is unclear if brute force methods will be as successful as in chess. We expect that more cognitive based methods have to be applied to make a shogi program that can play the game at a high level.

The aim of our research is to build a shogi program based on pattern recognition and position evaluation. In this article we will explain the first step of building such a program: using pattern recognition and position evaluation in a subdomain: tsume shogi (shogi mating problems).

2 Tsume Shogi with Pattern Recognition and Position Evaluation

In the past couple of years, tsume shogi has been an important topic of research in shogi [11, 12, 13, 19, 20]. In chess, endgame research is important because of the possibility of an exhaustive search, but mating problems have only been of brief interest to researchers [2]. In contrast, tsume shogi is a much more challenging subdomain. There are many tsume problems with a solution length of more than 20 ply and there are even quite a few problems with solutions over a 100 ply long. According to Seo [22], the average branching factor of tsume shogi problems is 5. As a result, building a computer program to solve long tsume shogi problems is a challenging task. Recently, a number of very fast programs have been developed [12]. The fastest program to date seems to be the latest version of the commercially available Morita Shogi [18]. This program solves most problems with a solution length of less than 30 ply within a couple of seconds. Therefore, the best tsume shogi programs at the moment outperform human experts, and solving tsume shogi seems to be no longer a challenging domain for research. However, as a test domain for methods to be applied to a complete shogi playing program, tsume shogi can still be useful. In this section we will describe how pattern recognition and position evaluation can be used to solve simple tsume shogi problems.

2.1 Pattern recognition

As mentioned in the introduction, cognitive research in chess [6, 7] has shown that pattern recognition plays an important role in chess expertise. Also, attempts to build chess programs based on pattern recognition have not been successful. Part of the reason for that has been the success of the brute force approach, but we feel that the method of pattern recognition used in these programs might also be a factor. A common feature of all the pattern based programs for chess that we know of [4, 15, 27, 29] is that a pattern is defined as a configuration of pieces on the board. However, research by Tichomirov and Poznyanskaya (cited in [23]) seems to indicate that chess positions are usually not literally perceived but that more abstract notions like the functionality of pieces is of vital importance.

To incorporate these findings into a pattern recognition program we turn to a more general definition of pattern recognition. In [28] pattern recognition is defined as the task of placing a particular object into a class corresponding to a concept. More formally, we need a set

$$C = C_1, \cdots, C_n$$

of classes into which patterns P are clustered. Patterns have associated with them a set of features F which are used to classify these patterns. The initial class C^I is defined by a small set of example patterns

$$E = E_1, \cdots, E_v$$

for which class membership has been established. For each class C_i in C there is at least one E_k in E. When given a pattern P_r with a set of features F_r the problem of pattern recognition is to find a class C_i of patterns $P_{i1}, \cdots, P_{im}$ so that P_r is similar to each of the patterns of this class C_i, i.e. if some relation between feature sets F_r and $F_{i1}, \cdots, F_{im}$ can be established. Determining similarity is now done by evaluating the relative importance of features in a class. This relative importance is then reflected by an evaluation function which assigns different weights to different features according to their relative importance for defining class membership [14, 8]. Some features can be so important for class membership that the absence of any one of these features makes it impossible for the pattern to be part of the class. We will call these features *essential features.*

As an example, let us look at a simple classification of animals. C consists of three classes: {Mammals}, {Fish} and {Other animals}. E is the set {Dog, Salmon, Swallow, Lion, Tortoise, Bee}. In C^I {Mammals} is defined by the set {Dog, Lion}, {Fish} by {Salmon} and {Other animals} by {Swallow, Tortoise, Bee}. Each of the example patterns in E has a set of features. For example, some of the features of {Dog} are: "Barks", "Gives birth to living young" and "Tail". For {Salmon} some of the features are: "Lays eggs", "Can live under water" and "Tail". In this simple example it is clear that the feature "Barks" is less important for defining the class membership of {Mammals} than "Gives birth to living young". The essential feature of members of the class {Mammals} seems "Gives birth to living young" and the essential feature of the class {Fish} seems "Can live under water". However, we need to be careful defining the essential features. If, for example, we want to classify the new pattern {Whale} with features "Tail", "Gives birth to living young" and "Can live under water" we see it has the essential features of two classes and the third feature "Tail" is also part of both classes {Mammal} and {Fish}. In this case we need to further refine our feature set to put the pattern {Whale} in the correct set.

The idea of our tsume shogi program is to have a set of patterns where the next move is the mating move, defined by essential and non-essential features relative to the essential piece, the king to be mated. For example, the one-move mating position on the left of diagram 1 can be described as follows:

Essential features:

- In_hand(飛) “There is a rook in hand”
- Protected(G_x-1, G_y) “The drop square is protected”

Non-essential features:

- Free(G_x-1, G_y) “A piece can be dropped to the right of the king”
- ¬Defended(G_x-1, G_y) “The drop square is not defended by any piece other than the king”
- Cannot_move(G_x, G_y+1) “The king can not move straight up”
- Cannot_move(G_x, G_y-1) “The king can not move straight down”
- Cannot_move(G_x+1, G_y+1) “The king can not move up diagonally to the left”
- Cannot_move(G_x+1, G_y-1) “The king can not move down diagonally to the left”

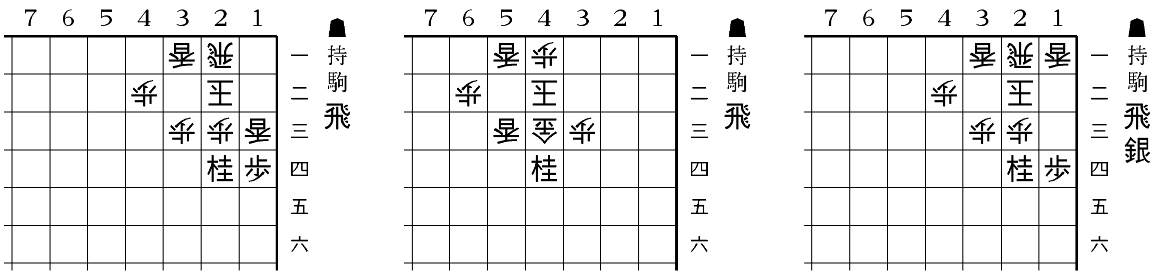

Diagram 1: Simple tsume positions

Intuitively, a mating pattern is defined by the shape of the attacking pieces, so these are the essential features. The defensive configuration can shift during the solution sequence of a tsume problem, so we consider these features as non-essential. Of course, there are many tsume problems where the attacking configuration also changes. We will return to this issue later.

By describing a mating pattern in this way, a mating pattern can be identified independent of the position on the board and independent of the exact pieces that block the escape of the king. For example, the position in the middle of diagram 1 matches the same pattern. Also, by using patterns with distance one from mate, the pattern suggests the next move. For tsume shogi this is not very important, but we expect that for actual shogi playing this suggestion of next moves from patterns can be used to cut the number of potential moves to be investigated by the program.

For tsume shogi, set C above ideally corresponds to all possible mating patterns. However, the large number of possible mating positions prohibits making such an ideal set C. Instead, for our program we concentrated on building a set E of example mating patterns, where each mating pattern corresponds to a class of mating patterns in C. Thus, in our program there is no difference between E and C. To solve an actual tsume shogi problem, the distance between the problem position after all the possible defenses to a check and the patterns in E are calculated. This distance is the difference between the non-essential features of the resulting problem position and pattern E_k in E. This distance can be used as an evaluation of the checks that are possible in a certain tsume problem position:

$$PATEVAL = \sum_{i}^{\#defmoves} min(patdis_i)$$

For each check, look at the difference to the set of mating patterns for all possible defense moves. The best check is the check where for all possible defenses the distance to mating patterns is the smallest. This is the check that should be investigated first. Let's return to diagram 1. If we look at the tsume problem on the right, the current feature difference between the position on the right and the mating pattern on the left is one: the square on the right of the king (12) is still defended by the lance on 11. If we assume that E only consists of the mating pattern on the left of diagram 1 and that an unknown pattern distance corresponds to a maximum distance of (arbitrarily) 10, the following evaluation of the checks in this simple tsume problem can be made:

1. Check: 13銀; Defense: 13香; PATEVAL = 0
2. Check: 12飛; Defense: 12香; PATEVAL = 10
3. Check: 32飛, Defense: 32香; PATEVAL = 10
4. Check: 12桂 + ("+" means promoting); Defense 12香 or 12玉; PATEVAL = 20
5. Check: 32桂 +; Defense: 32香 or 23玉; PATEVAL = 20

Therefore, 13銀 is evaluated as the next best check, followed by 12飛 and 32飛. 12桂 + and 32桂 + are considered to be the worst checks.

2.2 Position Evaluation

The second part of the evaluation function used for our tsume program is an evaluation of the position on the board. This idea has been used before [25], and has only be refined in our program. Our evaluation of a tsume position has the following aspects:

1. Escape routes: An evaluation of how easy it is for the king to get away from the attacking pieces.
2. King mobility: An evaluation of the freedom of movement of the king.
3. Thickness: The king position's thickness depends on the number of defenders close to the king.

4. Primary attack and defense: Evaluation of attack and defense on the 8 squares adjacent to the king.
5. Secondary attack and defense: Evaluation of attack and defense on the 16 squares at distance two from the king.
6. Indirect primary attack: Search for long range pieces like bishop and rook that are currently blocked by other pieces but indirectly attack the 8 squares adjacent to the king.
7. Indirect secondary attack: The same, but now for the 16 squares at distance two from the king.
8. Mobility of pieces: If the attacking pieces are blocking each other, they can not be used efficiently.
9. Material: Without material, a king can not be mated.

Using these aspects the following evaluation function for the position of the king has been used in our program:

$$\left(\sum_{j}^{\#attfeat} aw_j * af_j - \sum_{k}^{\#deffeat} dw_k * df_k\right)$$

With:

- #attfeat = number of attacking features
- aw = attacking feature weight
- af = attacking feature value
- #deffeat = number of defense features
- dw = defense feature weight
- df = defense feature value

This evaluation function is a measure for the amount of danger to the king to be mated. Checking moves in a certain tsume problem position can now be ordered by applying this evaluation function to the positions resulting from defending against these checks.

As already pointed out in [14], an evaluation function like this also fits in the abstract framework of pattern recognition of the previous section. The set C consists of a set of patterns "King in danger" and a set of patterns "King not in danger". A tsume problem pattern is considered to be in the set of "King in danger" if the evaluation function exceeds a certain value. We will not go into this any further for reasons that will become clear when discussing the results of our tsume shogi program.

2.3 Selective deepening

Pattern recognition is more important than position evaluation, so the total evaluation function of the positions resulting from defending against a check has a weight (pw) attached to the pattern recognition part of the evaluation function:

$$\sum_{i}^{\#defmoves} pw * min(patdis_i) + (\sum_{j}^{\#attfeat} aw_j * af_j - \sum_{k}^{\#deffeat} dw_k * df_k)$$

This evaluation function is now used to evaluate all checks in a certain tsume position. The most promising nodes according to the evaluation function are evalauted first. This process of selective deepening [21] looks at the tree as a whole, and postpones the expansion of a node if the evaluation function resulting from this expansion has a higher value than the evaluation of a node in any other part of the tree.

3 Results

At the moment, our program recognises about 300 standard mating patterns. The program also stops automatically after 10 minutes or if the search tree has grown bigger than 60,000 nodes. Some simple heuristics for filtering out useless intermediate drops have been added to avoid that the number of useless defensive moves spoils the results of the evaluation. We have used a test set of 100 five-ply tsume problems and 100 seven-ply tsume problems, all taken from the monthly shogi magazine “Shogi Sekai”. The results of our tsume program with the evaluation function above are:

	five-ply tsumes	seven-ply tsumes
$<$ 1 minute	98	82
$<$ 5 minutes	2	7
$>$ 5 minutes	0	0
Time limit exceeded	0	5
Too many nodes	0	6

To evaluate the importance of pattern recognition for our program, we also tested the program with the following evaluation function:

$$\sum_{i}^{\#defmoves} min(patdis_i)$$

This lead to the following results:

	five-ply tsumes	seven-ply tsumes
$<$ 1 minute	99	85
$<$ 5 minutes	1	6
$>$ 5 minutes	0	0
Time limit exceeded	0	1
Too many nodes	0	8

4 Conclusions and further research

The first conclusion from the tables is that our tsume shogi program, slow as it may be compared to the strongest ones, can solve simple tsume shogi problems quickly. Brute force methods may be much quicker, but it should be noted that only 300 patterns have been defined in our test program. Also, the previously mentioned emphasis on attacking configurations might influence the performance negatively. We expect that patterns in real shogi playing will be less general and that this will not be a major problem there.

Another observation is that position evaluation only seems to make the performance of our program worse. It is difficult to conclude that position evaluation is useless for a program that will be able to play a normal game of shogi. Tsume shogi problems are selected on artistic values like sacrifices and surprising ways of avoiding the king to escape. These artistic aspects do not play a major role in normal shogi play. Still, in our view strong players prefer positions that have known patterns and continuations and base their strategies on reaching these positions (a view also held by Walczak [27]).

Encouraged by the results of our tsume shogi test program, we will now continue our research by building a shogi playing program based on pattern recognition. However, cognitive research seems to indicate that an expert chess player recognises at least 50,000 patterns[23]. This is probably one of the most important reasons why chess programs based on pattern recognition have not been able to reach expert level. We feel that an algorithm to learn new patterns is vital to the success of our approach and building such an algorithm will be an important part of our future research.

Acknowledgements

I want to thank Dr.H.Matsubara and Dr.H.Iida for their support, their comments on my research and this paper and for helping me find my way in the Japanese literature on the subject of tsume shogi.

References

[1] Anantharaman, T., Campbell, M.S. & Hsu, F. (1990). Singular Extensions: Adding Selectivity to Brute-Force Searching. *Artificial Intelligence, 43*, 99-109.

[2] Baylor, G. & Simon, H. (1966). A chess mating combinations program. *Proceedings of the Spring Joint Computer Conference*, 431-447.

[3] Beal, D.F. (1990). A Generalised Quiescence Search Algorithm. *Artificial Intelligence, 43*, 85-90.

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

[5] 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.

[6] Chase, W.G. & Simon, H.A. (1973). Perception in Chess. *Cognitive Psychology, 4*, 55-81.

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

[8] Goos, K. (1994). Preselection strategies for case based classification. *Lecture notes in artificial intelligence, 861*, 28-38.

[9] 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.

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

[11] Ito, T. & Noshita, K. (1993). Two fast programs for solving Tsume Shogi. *34th Programming Symposium.* (in Japanese).

[12] 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).

[13] Koyama, K. & Kawano, Y. (1993). The database of Tsume Shogi problems and its evaluations. *SIGAI of Information Processing Society of Japan*, 88-95. (in Japanese).

[14] Lee, K. & Mahajan, S. (1988). A pattern classification approach to evaluation function learning. *Artificial Intelligence, 36*, 1-25.

[15] 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.

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

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

[18] Morita, K. (1996). Personal communication.

[19] Nakayama, Y., Akazawa, T. & Noshita, K. (1996). A parallel algorithm for solving hard Tsume Shogi problems. *International Computer Chess Association Journal, 19(2)*, 94-99.

[20] Noshita, K. (1991). A program for solving Tsume Shogi quickly and accurately. *Proceedings of Game Playing Workshop*, 56-59.

[21] Schaeffer, J. (1990). Conspiracy Numbers. *Artificial Intelligence, 43*, 67-84.

[22] Seo, M. (1995). Application of conspiracy numbers to a tsume shogi program. *Proceedings of the game programming workshop in Japan '95*, 128-137. (in Japanese).

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

[24] Shannon, C.E. (1950). Programming a Computer for Playing Chess. *Philosophical Magazine, Vol. 41 (7th series)*, 256-275.

[25] Tanaka, S., Iida, H. & Kotani, Y. (1995). Approximate estimation of solving tsume shogi problems using an evaluation function. *Computer Strategy Game Programming Workshop.*

[26] Uiterwijk, J. (1996). The Kasparov-Deep Blue Match. *International Computer Chess Association Journal, 19(1)*, 38-41.

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

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

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