THREAT STACKS TO GUIDE PRUNING AND SEARCH EXTENSIONS IN SHOGI

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

Abstract In most cases, game programs have to decide their next move under strict time constraints. Iterative deepening is the most common method to make sure a move is played that is the result of a completed search to a certain depth. To produce a reasonable move at each iteration a combination of pruning techniques and search extensions is used. Often, the decisions for pruning and extending the search are based on game-specific knowledge. In this paper we propose to store the game-specific knowledge into *threat stacks* and use general rules regarding the content of these threat stacks for pruning and extension decisions. Preliminary results in shogi show that if threat stacks are used instead of a standard quiescence search, an improvement of the tactical performance can be expected.

Keywords: Threat stacks, pruning, search extensions, horizon problem, shogi

1. Introduction

In most game-playing environments, time is the most important factor. A game program is not allowed to search until a win is found as for most interesting games this would take approximately to the end of the universe. It is very likely that a human opponent or spectators would lose interest at some point during this deliberation process.

Therefore, search needs to be terminated at some point and the move that is considered best at that point should be played. One problem is that it is hard to predict the search depth of the search that can be completed in this fixed time interval. To deal with this problem, most game programs use *iterative deepening* [7]: a search to depth N is completed before a search at depth $N+1$ is started. The next iteration is started until time runs out and the best move from the last completed search is played. Iterative deepening ensures that the next move is the

result of a completed search to a certain depth (this depth is called the *nominal search depth*).

For iterative deepening to work efficiently, the information of finished iterations is stored in a hash table and this information is used to guide the search at the next iteration. This only works well if the search at each iteration gives a reasonable estimate of the actual evaluation of the moves, i.e. the evaluation value of moves doesn't change dramatically between iterations. This is an important problem, as simple iterative deepening to depth N would not be able to detect a sudden change in the evaluation at depth $N + 1$ (for example the recapture of a queen in chess). A common method to improve this is to extend the search beyond the nominal search depth if a situation is detected where such a sudden change in evaluation value is likely. *Quiescence search* [2] aims at extending the search until a *quiescent* position is found that is unlikely to exhibit wild swings in the evaluation function value. Another example of search extensions are *singular extensions* [1], where the search is extended when there are forcing moves like attacking an important piece or threatening mate.

To speed up the termination of each iteration and to free time for search extensions, *pruning* is used. Pruning methods terminate the search before the nominal search depth is reached. One example of a pruning technique is *futility pruning* [4], where the search is terminated in a bad position from which it is highly unlikely to recover in the remaining search (for example, being a rook down in chess and having only one more move until the nominal search depth is reached and no possible captures). Another idea is ProbCut [3], which uses shallow searches and a likelihood of the result of the shallow search actually changing the overall evaluation. If the likelihood is below a certain threshold, the search is terminated based on this shallow search.

To produce a reasonable move at each iteration it is also important to properly deal with the *horizon problem*. This problem arises when one player has an unavoidable threat, but the other player can continue to play stalling moves to push the problem beyond the search horizon. Detecting a horizon problem usually requires game-specific knowledge.

In this paper we will present a new general framework called *threat stacks* for making pruning and extension decisions. This method can also be used to detect horizon moves. We have applied this method to *shogi* (Japanese chess), as we encountered problems using the common methods described above in our shogi program SPEAR. In Section 2 we will give the features of shogi and explain why these features cause problems when applying common pruning and extension techniques. In Section 3 we will explain the idea and implementation of threat stacks.

We will explain how threat stacks can be used to make pruning and extension decisions and we will show how threat stacks can be used to reduce the horizon problem. In Section 4 we will give some preliminary results of the method, indicating that the use of threat stacks can improve the tactical ability of a shogi program. Finally, in Section 5 we will give some conclusions and ideas for future work.

2. Pruning and Extensions in Shogi

Shogi is a two-player perfect information game that is similar to chess. The goal of the game is the same as in chess, namely to capture the king of the opponent. The most important difference between shogi and chess is that in shogi captured pieces can be re-used. When it is a player's turn to move, a choice can be made between moving one of the pieces on the board or putting one of the pieces previously captured back on an empty square. This second type of move, where a captured piece is returned on the board, is called a *drop*.

The drop rule of shogi is generally considered to be the distinguishing feature of this game. Most research efforts have been spent on dealing with the search explosion that is caused by this rule. Because of this rule, the average branching factor of the search tree in shogi is 80, while in chess the average branching factor is 35 [6].

Another important difference between chess and shogi is the promotion rule. In chess only pawns can promote and the pawn can promote to any piece (but in actual games almost always to a queen). In shogi most pieces can promote, but the only choice is between promotion and non-promotion. Another difference between chess and shogi is that in shogi promotion is possible on the top three ranks instead of only on the top rank like in chess.

For extension methods, these features have important consequences. As a result of the drop rule, the evaluation function is no longer dominated by material, but the major features of the evaluation function in shogi are both material and king danger. As a result, the set of non-quiescent positions is much larger. A position is no longer quiet when there are no more capture moves. In shogi there can also be a strong attack against one of the kings that might or might not be defendable with a single move. The problems for finding the right search extensions are further enhanced by the extended promotion rules in shogi, which can lead to a sudden change in the material balance much more often than the promotion in chess.

The features of shogi also make it harder to make the right pruning decisions in shogi. Strong attack moves, defence moves and promotions

can change the evaluation function value significantly, making it more difficult to judge the inability to recover from a bad position (as in futility pruning) or to determine the likelihood whether or not a deep search can be replaced by a shallow search (as in ProbCut).

Finally, the horizon problem is much worse in shogi than in chess. In most positions a tactical problem can be pushed over the search horizon by a sequence of drop moves attacking pieces and/or the king.

Strong shogi programs use static analysis of threats for pruning and extension decisions and to detect horizon moves [5]. Therefore, threats play a vital role in shogi, but thus far there has been no attempt to build a general framework for using threats in the search. We will now propose such a framework by defining *threat stacks* and explain how the information in these threat stacks can be used for pruning and extension decisions and to detect horizon moves.

3. Threat Stacks

3.1. Description of the Method

To store the threats of the black and white player we use two stacks. After each move, the threats by black and white are analysed and pushed on the respective threat stacks. Each threat entry consists of a threat identification number, up to three squares that define the pieces involved in the threat and a value indicating how strong the threat is.

When a move is undone, the threat stacks are popped. Using stacks ensures that at search depth N not only the current threats can be analysed, but also the threats at previous depths. In Section 3.4 we will explain the importance of this for detecting horizon moves.

The value of each threat is used to make a difference between strong threats and weak threats. Weak threats can be ignored when a strong threat needs to be dealt with. In shogi, weak threats are relative to the phase of the game. What is considered a weak threat in the endgame (i.e. winning a pawn) can be a strong threat in the opening.

In shogi there are two basic categories of threats: material threats and king threats. Here are the threats that have currently been implemented:

- **Check**
  A threat to capture the king.
- **Threat to gain material**
  To determine threats to capture material, a *static exchange evaluator* is used to determine favourable captures.
- **Promotion**
  The square of the piece threatening to promote and the promotion

square are being stored. Only promotion threats that significantly change the material value of a piece are stored (in shogi this is the promotion of the rook, bishop or pawn).

- **Discovered attack**
  In case of discovered attacks, three squares are stored: the square of the piece potentially attacked, the square of the piece potentially attacking and the square of the piece that can move to make the attack possible.

- **Pins**
  The square of the pinned piece and the piece that can be captured if the pinned piece moves is being stored.

- **King threats**
  King threats are analysed by comparing the piece covering of the attacking and defending side on the eight squares adjacent to the king. A strong threat is a threat where one side is controlling a square adjacent to the opponent king. A weak threat is a position where one side is only covering a square adjacent to the opponent king.

3.2. Search Extensions

Instead of a normal quiescence search, the search is extended by a threat quiescence search that aims at emptying the threat stacks as quickly as possible. At each node in the threat quiescence search, the strongest threats by each side are determined. The next action is then determined by the following rules:

- If the side to move has a stronger threat than the opponent, the threat is executed.

- If the side to move has a weaker threat than the opponent, moves to defend against the strongest opponent threat are generated.

If the strongest threats are weak threats, the threat quiescence only generates moves to execute or defend against these threats in the opening and middle game. In the endgame, in case of weak threats the threat quiescence is terminated and the current evaluation returned.

3.3. Pruning

The information stored in the threat stacks can be used for a number of pruning decisions:

- Delete sacrifices played one move before the nominal search depth. This ensures that the threat quiescence doesn't spend too much time searching the consequences of meaningless sacrifices.
- Prune if the number of threats is larger than the remaining search depth. It is important to have a safety margin here as sometimes multiple threats can be introduced or resolved with a single move.
- If there is a threat in the current position, prune moves that neither execute nor resolve a threat and also don't introduce a threat.

These pruning decisions are only made if there is a strong threat in the current position. When there are only weak threats, no moves are pruned.

3.4. The Horizon Problem

Horizon moves are dealt with in two steps. In the first step a potential horizon move is detected. A material sacrifice or a move introducing a threat with a lower value than the threat of the opponent is a potential horizon move. The second step is to analyse the threat stacks at the nominal search depth to determine if the move was actually a horizon move. To achieve this, a flag is set when a potential horizon move is detected and this flag is transported down the search tree until the nominal search depth is reached. At the nominal search depth, the threat stacks are analysed to see if the opponent threat that was the reason for flagging the potential horizon move is still unresolved. If so, the move is judged to be a horizon move and given a low evaluation.

4. Results

As a preliminary test of our method, we compared the performance of a program with threat stacks to a program using standard quiescence search on tactical shogi problems. The first test is a set of problems taken from the weekly magazine *Shukan Shogi*. The test set consists of 300 problems published in issues 762 (November 4th 1998) to 811 (October 20th 1999). The goal in each of these problems is to win material, avoid the loss of material, get a decisive attack or defend against the attack of the opponent.

The problem level ranges from starting level to expert level. It should be noted that the starting level is already quite advanced and is too hard for beginners. Two of the problems in the test set are incorrect and have been removed from the test set, leaving 298 problems.

The two program versions were given a maximum of 30 seconds on a 1.2 GHz Pentium III to solve each problem. This time limit is the same

Table 1. Comparison of threat stacks and quiescence search on 298 tactical shogi problems.

Version	*Total*	*Solved*	*Percentage*	*Time Used*
Threat Stacks	298	125	42%	02:14:12
Quiescence search	298	97	33%	02:34:45

Table 2. Comparison of threat stacks and quiescence search on a material test set.

Version	*Total*	*Solved*	*Percentage*	*Time Used*
Threat Stacks	115	93	81%	00:59:31
Quiescence search	115	99	86%	01:00:45

Table 3. Comparison of threat stacks and quiescence search on an attack/defence test set.

Version	*Total*	*Solved*	*Percentage*	*Time Used*
Threat Stacks	175	89	51%	01:18:27
Quiescence search	175	57	33%	01:38:12

as the expected average available time per move in the annual Computer Shogi World Championships.

The results of this test are given in Table 1. In this table we see that the overall tactical performance of a program using threat stacks improves significantly when compared to a program that uses quiescence search. It is also important to note that the threat analysis is not slowing the program down. On the contrary, the program using threat stacks finished the test set 20 minutes faster than the program using quiescence search.

Quiescence search is supposed to deal adequately with threats involving material, so the overall performance improvement was expected to be caused by the improvement in tactical endgame positions. To test this assumption, we used two different test sets. One test set consisted of 115 tactical problems in which winning material or avoiding the loss of material was the objective. The second test set was a set of 175 tactical problems about attack and defence. The results of the material test are given in Table 2 and the results of the attack/defence test in Table 3.

From these results it can indeed be concluded that threat stacks are improving the tactical performance in the endgame. When only material is involved, quiescence search gives better results.

Unfortunately, thus far we have not been able to show improvement of general playing strength. A self-play experiment between the program using threat stacks and the program using quiescence search ended in a victory for the program using quiescence search. Further tuning of the method is needed to improve playing strength.

5. Conclusions and Future Work

In this paper we have described how threat stacks can be used to make pruning and extension decisions and detect horizon moves in shogi. Our method has been applied to shogi, but we expect that it can also be applied in other games. Game-specific knowledge is used in the threat analysis, but the pruning decisions, the threat quiescence search and the detection of horizon moves are game-independent.

Preliminary results indicate that using threat stacks improves the tactical ability of a shogi program. However, it was also shown that further refinement and tuning is necessary to show the importance of this improvement and whether this improvement can be translated into winning more games.

References

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

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

[3] M. Buro. ProbCut: An Effective Selective Extension of the alpha-beta Algorithm. *ICCA Journal*, 18(2):71–76, June 1995.

[4] E. Heinz. Extended Futility Pruning. *ICCA Journal*, 21(2):75–83, June 1998.

[5] H. Iida, M. Sakuta, and J. Rollason. Computer Shogi. *Artificial Intelligence*, 124:121–144, 2002.

[6] H. Matsubara, H. Iida, and R. Grimbergen. Natural developments in game research: From Chess to Shogi to Go. *ICCA Journal*, 19(2):103–112, June 1996.

[7] 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.