Towards Evaluation of Shogi Endgames with Speed of Attack

SHUNSUKE SOEDA,† TOMOYUKI KANEKO† and TETSURO TANAKA††

Finding a good move in Shogi endgames is considered to be a hard problem. Moves such as sacrifice moves often become important in Shogi endgames, but they could not be recognized to be a good move with simple evaluation functions. In this paper, we will introduce a concept called possible pass count , which is a value closely related to threatmates and brinkmates. Then we will propose an algorithm to calculate possible pass count . Finally, we will show the performance results of our algorithm and show that possible pass count could be calculated in reasonable time for many Shogi positions.

1. Introduction

The introduction of Df-pn search algorithm[6] has enabled Shogi programs to search complicated checkmate sequences. The ability of computers to solve checkmate sequence problems (or `Tsume-shogi` problems) has surpassed human grand champions.

However, finding a good move in Shogi endgames is still considered to be a hard problem. In the endgame, moves to peel off the pieces around the defender's King is required. Most top level Shogi programs use hand-tuned evaluation functions to find good moves in the endgame, which is both hard to understand and to maintenance. Hand-tuned evaluation function are also not so reliable in the sense that they cannot handle positions they were not designed for. Some moves in the endgame involve sacrifice of pieces. The effect of these kind of moves become evident only after when a few moves has been played. Thus a simple evaluation with deeper search is preferred.

Threatmates and brinkmates moves by the attacker, which if not defended properly by the defender, leads to a checkmate sequence by the attacker. The are both concepts that play an important role in Shogi endgames.

Brinkmates are moves by the attacker that often leads to a win by the attacker. Some algorithms to search brinkmates has been proposed an algorithm[3][1]. It is still slow and only few programs use brinkmate search.

Threatmates are moves by the attacker if overlooked by the defender leads to a checkmate sequence. Finding threatmates often leads to finding checkmates[4].

In this paper, we will introduce a new concept called *possible pass count* in shogi endgames, and will show that they are closely related to threatmates and brinkmates. Then we will propose an algorithm to calculate possible pass count . Finally, we will show the performance results of our algorithm.

2. Related Work

In this section, we will make a brief introduction to some related works.

2.1 Shogi

Shogi midgames is a successful research area, with the recent proposition of realization probability search[11]. One reason of their success is the use of simple evaluation function with deeper search.

However, they still have difficulties in finding a good move in the mid-to-endgames in Shogi.

2.2 Endgames in Other Games

Shogi endgames are difficult compared to endgames in other games. Endgames in Chess[9] and Checkers[7] are solved using a endgame database. However as pieces in Shogi does not decrease, it is impossible to create a endgame database for Shogi.

In Go, the endgame could be divided into some small independent subgame, thus making it easy to analyze[2]. Shogi divides to at most two subgames, and it is rare that each subgame is ever independent.

2.3 Pass-count Aware Methods

One basic concept of our method is counting the number of passes the defender can make before he is mated. This concept is well seen in Go programs, for example the concept of Possible Omission Number[8].

Another example is λ search[10], which is almost equivalent with brinkmate search in

† Department of General Systems Studies,Graduate School of Arts and Science, The University of Tokyo
†† Information Technology Center, The University of Tokyo

Shogi. Our algorithm is very close to λ search, but we foucus on a place between threatmate search and brinkmate search.

3. Possible Pass Count

In this section, we will define a value called *possible pass count*. Then we will define *threatmate*(n) and *brinkmate*(n) using this *possible pass count*. Finally, we will explain our algorithm to calculate these values.

We will use the word **attacker** to refer to the player who is trying to find a threat move, and **defender** to refer to the player who is trying to avoid the threat move.

3.1 Possible Pass Count

We will first define a value called **possible pass count**, which is a value defined for each position in the game.

There are no real passes in Shogi, but some moves by the defender will not help its defense. For example most attack moves by the defender. We will call these moves a **pass**.

Definition 1 : Pass
A move by the defender that does not give any influence to the defense of the King owned by the defender.

We will assume that a pass exists for every defender's turn. Also, we assume that there are no no zugzwang positions in Shogi ☆.

Possible pass count is a value representing the total number of passes that the defender can make before the attacker can find a checkmate sequence. When we calculate possible pass count , we will limit the moves that the defender can generate to **DEFENSE_MOVES**. This means that a possible pass count of a given position is defined for each set of DEFENSE_MOVES.

We will define possible pass count as follows:

Definition 2 : Possible pass count

- The possible pass count for a position in a checkmate sequence is 0.
- The possible pass count for a position in the attacker's turn is the minimum possible pass count of the position reachable by any legal move.
- The possible pass count for a position in the defender's after a check is the maximum possible pass count of the position reachable by any escape moves.
- The possible pass count for a position in the defender's turn after a non-check move is the maximum of the following:
  - (Possible pass count after a pass) +1.
  - Maximum possible pass count of the position reachable by DEFENSE_MOVES from the position.

3.2 Threatmate(n) and Brinkmate(n)

Given the definition of possible pass count we will define threatmate(n) and brinkmate(n) as follows:

Definition 3 : Threatmate(n)
A position with possible pass count of n, where DEFENSE_MOVES is empty.

Definition 4 : Brinkmate(n)
A position with possible pass count of n, where DEFENSE_MOVES is any legal move.

The definition of both threatmate(n) and brinkmate(n) is illustrated in figure 1.

3.3 Algorithm to calculate *possible pass count*

The basic idea of calculating the possible pass count of a given position is an enhanced version of Df-pn algorithm. For each position in search, we will try to proove its possible pass count from a smaller number. That is, we will first try to proove it is has a possible pass count of 0, then 1 ... until the proof succeeds.

Note that for each position kept in the table, we keep a proof number and disproof number separately for each possible pass count we are going to proove.

- In the start position, first try to prove it has a possible pass count of 0. If it fails, try again with the number increased, until the proof succeeds.
- For each position in search, first try to prove the position has a possible pass count of $N-1$ if $N \geq 1$. This is done recursively, so for any position, the search should start from prooving it is has a possible pass count of 0. If the proof succeeds, return success. If the search limit is reached while searching in $N-1$ or if the proof for $N-1$ fails, keep on with the proof of N.
- For attack positions trying to prove that the possible pass count is N, choose the child position with the smallest proof number for proving it has a possible pass count of N. If the proof succeeds, return success. If all the children are disproofed, or if there

☆ This assumption does not always stand

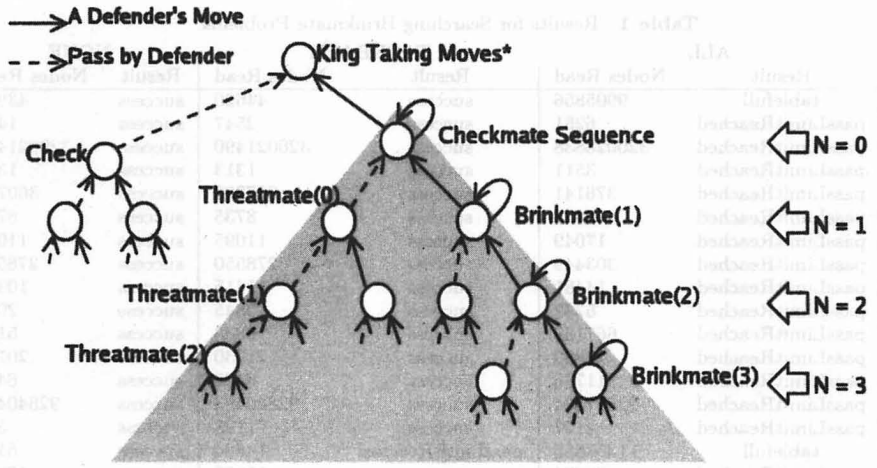

Fig. 1 Definition of *threatmate*(n) and *brinkmate*(n).

are no attack moves, return fail.

- For defense positions trying to prove that the possible pass count is N, first pass and see if the following position has the possible pass count of $N-1$. If it succeeds, try to prove that the other children of the positions has a possible pass count of less than N. If any proof for the pass move fails, return fail. If all the children generated succeeds, return success.

Generally, a proof with a larger set of DEFENSE_MOVES is harder than a proof with a smaller set of DEFENSE_MOVES. On the other hand, a proof with a larger set of DEFENSE_MOVES more accurate than a proof with a smaller set of DEFENSE_MOVES, in the sense that they are harder to defend.

4. Experiments

We conducted an experiment to measure the performance of our algorithm. We have implemented our algorithm with a simple Df-pn, without any enhancements. Our program still does not detect Pawn drop checkmates, and GHI problems[5].

We have chosen 37 brinkmate problems from `Shogi world 2001 April`. We compared the results with the following set of DEFENSE_MOVES:

- ALL (generate all defense moves, brinkmate search)
- TAKEBACK (generate moves that take back the last attack piece moved)
- NONE (generate no defense moves, threatmate search)

We have limited the size of the transposition table to 400,000 nodes, and try to find possible pass count with one. For most problems, the calculation ended in less than a second. The results for the experiment is shown in table1.

We could see from the results that searching for brinkmates (ALL) is still difficult even with our algorithm. However, searching for possible pass count with TACKBACK was successful for most problems, and took no more than three time longer to finish the search compared with NONE.

5. Conclusion

We have introduced a new concept called possible pass count , and showed that they are related to threatmates and brinkmates. We have proposed an efficient algorithm to calculate possible pass count , and showed that it could solve most problems in less than a second.

References

1) M. Arioka. *Search in Shogi Program KFEnd*, pages 18–40. Kyoritsu, 2003.
2) David Wolfe Elwyn Berlekamp. *Mathematical Go*. A K Peters, 1994.
3) H. Iida and F. Abe. Brinkmate search. In

Table 1 Results for Searching Brinkmate Problems.

Problem	ALL		TACKBACK		NONE	
	Result	Nodes Read	Result	Nodes Read	Result	Nodes Read
1	tablefull	9905856	success	44620	success	43931
2	passLimitReached	6251	success	2547	success	1481
3	passLimitReached	320028838	success	320021490	success	320021490
4	passLimitReached	3511	success	1313	success	1313
5	passLimitReached	376141	success	367207	success	360720
6	passLimitReached	16625	success	8735	success	8735
7	passLimitReached	17049	success	11095	success	11095
8	passLimitReached	303449	success	278550	success	278550
10	passLimitReached	14481	success	11415	success	10382
11	passLimitReached	5732	success	3845	success	2097
12	passLimitReached	667132	success	6838	success	5151
13	passLimitReached	22800	success	21730	success	20181
15	passLimitReached	11756	success	6481	success	6481
16	passLimitReached	92857334	success	92840493	success	92840493
17	passLimitReached	4197	success	1198	success	567
18	tablefull	11367855	passLimitReached	14894	success	5110
19	passLimitReached	28334	success	11695	success	10963
20	passLimitReached	96825	success	92357	success	92357
21	passLimitReached	7480	success	5844	success	4153
22	passLimitReached	30110	success	13068	success	13064
23	passLimitReached	100073	success	87494	success	83359
24	passLimitReached	47374104	passLimitReached	2593	success	1395
25	tablcfull	242066483	succcss	232691968	succcss	232691968
26	passLimitReached	63366	success	43731	success	43731
27	passLimitReached	9812	success	4912	success	1766
28	passLimitReached	8051	success	999	success	901
29	passLimitReached	29203	success	11969	success	11822
30	passLimitReached	105637	success	84180	success	81836
31	passLimitReached	2726	passLimitReached	2102	success	248
32	passLimitReached	35794	success	31214	success	31214
33	passLimitReached	6440	passLimitReached	7599	success	2936
34	passLimitReached	251037	success	206609	success	206600
35	passLimitReached	48825	success	23436	success	19815
36	passLimitReached	60561	success	56179	success	55636
37	passLimitReached	55854	success	43306	success	41818
38	passLimitReached	2672263	success	35354	success	32594
39	passLimitReached	9688381	success	9677096	success	9677091
total	0	738350366	35	656776156	39	656723044

Game Programming Workshop in Japan '96, pages 160–169, Kanagawa, Japan, 1996.

4) Kentaro Kayama. Toward a formulation of the shogi endgame. In *Computer Shogi Association Report*, volume 10, pages 67–69, 1997.

5) Akihiro Kishimoto and Martin Mueller. A solution to the ghi problem for depth-first proof-number search. In *7th Joint Conference on Information Sciences (JCIS2003)*, pages 489 – 492, 2003.

6) Ayumu Nagai and Hiroshi Imai. Application of df-pn algorithm to a program to solve tsume-shogi problems. In *IPSJ Journal*, volume 43, pages 1769–1777, 2002.

7) J. Schaeffer, Y. Björnsson, N. Burch, R. Lake, P. Lu, and S. Sutphen. *BUILDING THE CHECKERS 10-PIECE ENDGAME DATABASE*, pages 193–210. Kluwer, 2003.

8) Morihiko Tajima and Noriaki Sanechika. Estimating the possible omission number for groups in go by the number of n-th dame. In H. Jaap van den Herik and Hiroyuki Iida, editors, *Computers and Games*, volume 1558 of *Lecture Notes in Computer Science*, pages 265–279. Springer, 1999.

9) J.A. Tamplin and G.McC. Haworth. *Chess Endgames: Data and Strategy*, pages 81–87. Kluwer, 2003.

10) Thomas Thomsen. Lambda-search in game trees — with application to go. *Lecture Notes in Computer Science*, 2063:19–38, 2001.

11) Yoshimasa Tsuruoka, Daisaku Yokoyama, and Takashi Chikayama. Game-tree search algorithm based on realization probability. In *ICGA Journal*, volume25, pages 145–152, 2002.