Comparison of Df-pn based Search Algorithms on Shogi Brinkmate Problems

Shunsuke SOEDA

2006.4.25 GPS

1 Introduction

The currently best method is the one by Nagai, using df-pn+ to solve brinkmate problems. We have proposed a new threat based and proof number based algorithm called df-pn driven λ-search. We have conducted some preliminary experiments, showing that df-pn driven λ-search seems to be better than Nagai's method in solving Shogi brinkmate problems. Although more invesigation is requiered, df-pn λ-search seem to be able to:

- Solve brinkmate problems better than Nagai's method.
- Work much better with mixed problems of brinkmate positions and checkmate positions than Nagai's method.

2 Related work

Some works on solving Shogi brinkmate problems:

	Iida [4]	Arioka [1]	Hashimoto [3]	Nagai [7]	Soeda
Heuristic Pruning	Attack	Both	Attack	Attack	None
λ^2	ID	ID or PN^*	ID	df-pn+	df-pn+
λ^1	ID	?	PDS	df-pn+[1]	df-pn+
Notes	Hard/soft brinkmates		TDSS		

Df-pn driven λ-search is based on the following two algorithms:

- λ-search was introduced in [9].
- Df-pn search was introduced in [8].

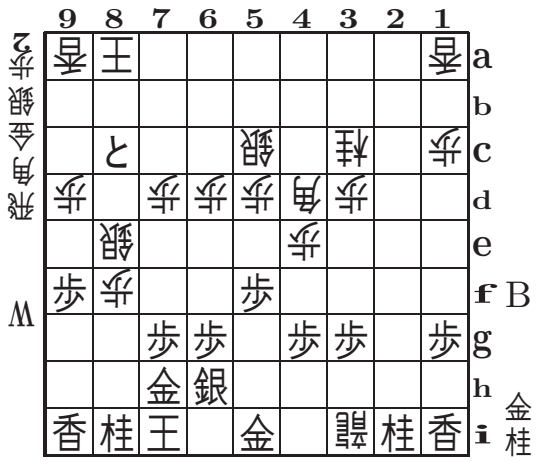

Figure 1: A position where it is difficult do disproof a λ^1 win by WHITE.

3 Df-pn driven Lambda Search

3.1 Extended lambda search

When iterative deepening is combined with λ-search, there is a design choice on how to conduct the iteration. Usually, a scheme where the depth threshold is iterated more than the threat order threshold, is chosen, as increaseing the threat order threshold increases the search space much more than increasing the threshold on depth in most games.

In the original λ-search, λ-move at an AND node is a move that is not followed by a tree which is proved for i where $0 \leq i \leq n-1$. This requires a multiple iteration on threat orders at each OR nodes. To know if a given move by the defender is a λ^n-move, first λ^i-search (where $0 \leq i \leq n-1$)must be conducted. If all search finish with the result value of $false$, then the move is a λ^n-move, and if any of the search results in the value of $true$, the move is not a λ^n-move. This is equivalent to generalized widening [2].

For most games and positions, conducting a multiple iteration on threat orders at each OR node is not a big problem. It is usually worth searching for a winning way with a stornger threat orders than finding a winning way with a weaker threat orders, as they are cheaper. The size of the tree to be searched for a stornger threat order is usually smaller than the size of the tree to be searched for a weaker threat order.

However, it is well known that in most games, it usually harder to disprove positions than prove positions (cite something here).Some positions are extreamely hard to disprove, and exapmle shown in figure 1.

Thus, we define an extended verision λ-tree as follows:

Definition 1 : extended λ^n-tree *An extended λ^n-tree is a search tree which*

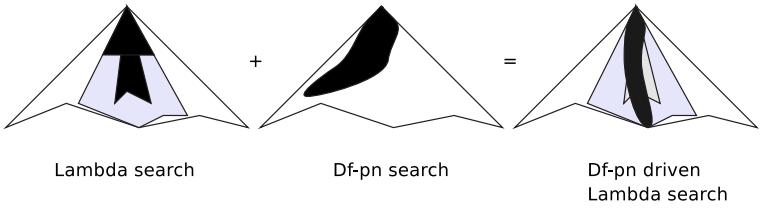

Figure 2: Df-pn driven λ-search.

consists solely of λ^i-moves where $0 \leq i \leq n$; a λ^n_{attack}-tree is a λ^n-tree where the attacker moves first.

3.2 Df-pn driven Lambda Search

In this section, the combnation of λ-search [9] and df-pn search [8] is introduced. Df-pn search algorithm is the currentyl best known search algorithm for searching AND/OR trees. Df-pn search alogritm and its variations have been successful in Shogi checkmate problems and brinkmate problems [8], and in one eye problem of Go [6, 5] and in Checkers [5]. The combination of df-pn search with λ-searh the part of the game tree which is likely to have the winning way, that requires a deep search to find the solution (figure 2).

Df-pn driven λ-search introduced in this chapter could model various search algorithms, including search algorithms proposed for searching Shogi checkmate problems and brinkmate problems. As this algorithm uses null moves to do define threats, it could be directly applied to problems where weaker threats get involved.

3.3 Schemes of iteration on threat orders

3.3.1 Search from Stronger threats

In the original λ-search by Thomsen, λ moves for the AND node are defined as follows:

> If the defender is to move, it is a move that implies that there does not exist any subsequent λ^i_a-tree with value 1, $0 < i \leq n-1$.

Under this scheme every, at every OR nodes, is searched to check if there is a solution by λ^{n-1} when searching for λ^n solutions. In other words, at OR nodes, iteration on threat order is performed, which is similar to generalized widening by Cazenave [2].

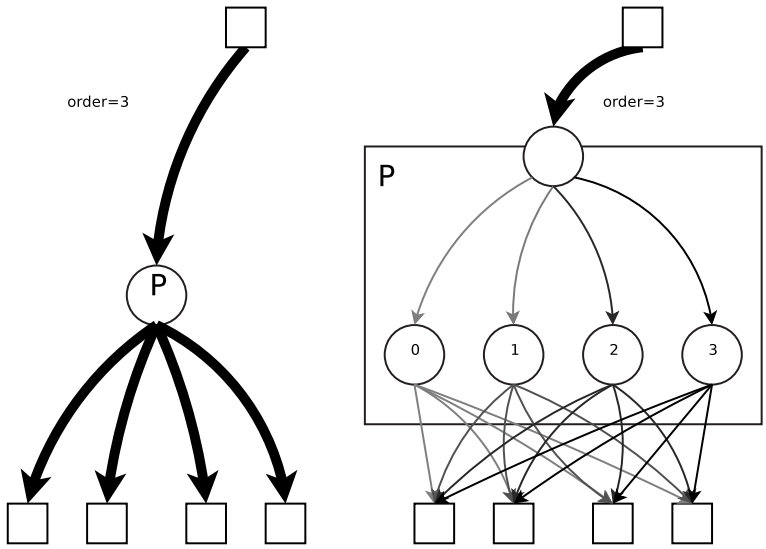

Figure 3: Multiple iteration on threat orders at OR nodes. P is an OR node searched with a threat order threshold of 3, which has 4 children nodes.

3.3.2 Fixed Threat Orders

The most simple combination of threat based search algorithm and df-pn search algorithm is to use df-pn to search threats of fixed order. This works best if the threat order of the given problem is known. For example, Checkmate problems in Shogi are problems that could be solved with λ^1 moves, while Brinkmate problems require search of λ^2 moves.

When the order of the threat of the solution is know prior to the search, and move candidates for different threat orders could be easily distinguished, this scheme could be efficient.

3.3.3 Proof Number based Widening

Treat an OR nodes as a complex of OR nodes called *pseudo-OR-nodes*, each representing the node searched at each threat level lower and equal to the on in consideration (figure 3.3.3).

The proof number and the disproof number of the node is calculated as follows:

- Proof numebr of P is the smallest proof number of the pseudo-OR-nodes.
- The disproof numebr of P is disproof number of the pseudo-OR-node representing the search with the threat order in consideration.

$$P.p = min(P_{\lambda^i.p})$$
$$P.d = P_{\lambda^3}$$

!!! Disscussion on Lemma & Proof !!!

4 Experiments

Conducted experiments on Brinkmate problems and Checkmate problems. The following problems were used:

- Mixture of brinkmate and checkmate problems from *Jitusryoku Youskei 100 Dai Tsugi No Itte – Tsume To Hisshi*by Keiji Mori (55 checkmate and 45 brinkmate problems).
- Brinkmate problems from *Tsumi yori Hisshi* by Takashi Kaneko (33 brinkmate problems).

The comparison was made on the following three types of algorithm:

	No move ordering	Pass stritcly first
Fixed threat order (λ^2)	Nagai	Nagai+
Iteration on threat order	(N/A)	Proposed

!!! Results of raw data handed in separetly !!!

5 Conclusion and Discussion

- Nagai's original method seems to consume much memory. Why?[2]
- Proposed method seems to be better than Nagai+ method.

References

[1] M. Arioka. Inside kfend - brinkmate search, 2000. (in Japanese).

[2] Tristan Cazenave. Generalized widening. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI*, pages 156–160. IOS Press, 2004.

[3] Tsuyoshi Hashimoto. *Searching for Solutions in Complex Games: Shogi endgames and Amazons*. PhD thesis, Shizuoka University, 2002.

[4] H. Iida and F. Abe. Brinkmate search. In *Game Programming Workshop in Japan '96*, pages 160–169, Kanagawa, Japan, 1996.

[2]It might be that there are more λ^2 moves and position that gets involved in this search. Must investigate.

[5] Akihiro Kishimoto. *Correct and Efficient Search Algorithms in the Presence of Repetitions*. PhD thesis, University of Alberta, 2005.

[6] Akihiro Kishimoto and Martin Müller. Df-pn in go: An application to the one-eye problem. In *Advances in Computer Games 10*, pages 125–141, 2003.

[7] Ayumu Nagai. *Df-pn Algorithm for Searching AND/OR Trees and Its Applications*. PhD thesis, Department of Information Science, University of Tokyo, Japan, 2002.

[8] 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.

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