証明数を用いたλ探索の効率的な実装

副田俊介 † 美添一樹 †† 田中哲朗 †††

df-pn 駆動 λ 探索および df-pn 駆動 dualλ 探索の性能をシンペイの局面を対象として評価した．df-pn 駆動 λ 探索は脅威度を用いた探索アルゴリズムである λ 探索と証明数を用いた探索アルゴリズムである df-pn を組み合わたアルゴリズムであり，将棋の必至問題で及び囲碁の捕獲問題での有効性を示されている．ただし，λ 探索と df-pn を組み合わせる方法である拡幅戦略の比較と，双方のプレイヤの脅威度を利用する探索アルゴリズムである df-pn 駆動 dualλ 探索の有効性は将棋の必至問題のみを対象に評価した．しかし，将棋の必至問題には強い脅威度が存在する局面のみが存在するため，弱い脅威度の存在する局面での拡幅戦略による違いや df-pn 駆動 λ 探索と df-pn 駆動 dualλ 探索の比較はおこなわれていない．そこで，弱い脅威度の存在する局面で拡幅戦略の比較および df-pn 駆動 λ 探索と df-pn 駆動 dualλ 探索をおこなうため，シンペイを対象として実験をおこなった．シンペイは全ての局面が解かれているゲームで，弱い脅威度の存在する局面があることや λ 探索が正解できない局面があることも知られており，実験の対象として興味深い．シンペイの全局面のうち勝ちとなる局面の 1000 分の 1 をランダムで取り出し，各アルゴリズム・拡幅戦略の性能を比較した．その結果，拡幅戦略の違いにより，大きな性能の変化は見られなかった．一方で，df-pn 駆動 λ 探索と比べて df-pn 駆動 dualλ 探索は，多くの問題で性能が改善されることがわかった．

Efficient Implementation of the Lambda Search based on Proof Numbers

Shunsuke Soeda,† Kazuki Yoshizoe†† and Tetsuro Tanaka†††

AND/OR tree search algorithms called the df-pn driven λ-search and the df-pn driven dualλ-search were tested on the game of Simpei. Df-pn driven λ-search is the combination of a threat based search algorithm called the λ-search algorithm and a proof number based search algorithm called the df-pn search algorithm. Df-pn driven λ-search was tested on brinkmate problems of Shogi and capture problems of Go. Several other methods to combine the λ-search algorithm and the df-pn search algorithm were also proposed as an expansion to df-pn driven λ-search called widening schemes. The df-pn driven dual λ-search algorithm was also proposed, which takes into account of threats by both players. Widening schemes and df-pn driven dual λ-search were tested for Shogi brinkmate problems. However, as Shogi brinkmate problems contain positions with strong threats only, there were no experiments of comparing these methods in positions involving weak threats. In this paper, the comparison of widening schemes and the comparison of df-pn driven λ-search and df-pn driven dual λ-search are made for positions with weak threats. The game of Simpei is a strongly solved game. It is known to have positions with weak threats, and it has some positions that λ-search can never solve, which make it an interesting target. 1 out of 1000 positions were randomly sampled from all the positions in Simpei, and positions where the winner is to play were extracted as the target, and the resource needed to solve the target was measured for each algorithms. As the result, no significant difference were seen between the widening schemes, while df-pn driven dual λ-search performed better than df-pn driven λ-search in most positions.

† 公立はこだて未来大学
Future University-Hakodate
shnsk@fun.ac.jp
†† 中央大学理工学部電気電子情報通信工学科
Dept. of Electrical, Electronic, and Communication Engineering, Chuo University
yoshizoe@is.s.u-tokyo.ac.jp
††† 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
ktanaka@ecc.u-tokyo.ac.jp

1. はじめに

近年の計算機能力の向上により，ゲームプログラミングにおいて終盤，すなわち勝敗を読み切れる局面の範囲が広がってきている．一般にゲームプログラミングでは末端での評価値を元にミニマックス法などのゲーム木探索を用いて局面の評価値を決定するが，終盤での探索で最後まで読み切る場合は末端の評価値が 2 値になるので，一般のゲーム木探索ではなく AND/OR

木の探索となる．

AND/OR 木の探索に関して，最近成功を収めているアプローチが二つある．一つは証明数に基づく探索で、一つは脅威度に基づく探索である．証明数に基づく探索は，展開途中の木の形から導き出される予想コストを使った最良優先探索であり，中でも df-pn 探索[3] というアルゴリズムが，詰将棋問題を解くプログラムなどで使われて成功を収めている．脅威度に基づく探索は，ゴールとの距離を用いた最良優先探索である．中でも脅威度をゲーム固有の性質を用いずに片方のプレイヤーが何回パス可能かを示す λ 次数で表現し，λ 次数を探索によって求める λ 探索[2] が，多くのゲームに適用可能だという点で有望だと考えられている．

筆者らは以前の研究で df-pn 探索と λ 探索を組み合わせた df-pn 駆動 λ 探索を提案し，将棋の必至問題[4),8)] と囲碁の捕獲問題[9] での有効性を示した．また，λ 探索と df-pn 探索を組み合わせる方法はいろいろ考えられるため，λ 探索と df-pn 探索を組み合わせる方法 (拡幅戦略) を複数提案し，それぞれについて将棋を対象として性能を評価する実験をおこなった[8]．さらに，λ 探索を，敵からの反撃も自然に表現できる dual λ 探索に拡張し，これも将棋の必至問題を用いた実験で評価もおこなった[4),8)]．

しかし，将棋の必至問題は強い脅威度を含む局面のみからなるため，弱い脅威度を含む局面での df-pn 駆動 dualλ や拡幅戦略の違いについて評価はされていない．そこで，本論文ではシンペイを対象に，高い次数の脅威度を含む探索で，拡幅戦略の比較及び df-pn 駆動 λ 探索と df-pn 駆動 dualλ 探索の比較をおこなった．シンペイは 2005 年に提案された，比較的簡単なボードゲームであり，既に筆者らによって解かれている[5]．次数の高い局面で，現実的な時間内に解くことができるものが存在することが知られており，今回の実験の対象にふさわしいと考えられる．

シンペイの全局面から 1000 分の 1 を選び，更にその中から手番のプレイヤが勝ちとなる局面を選んだ．各局面についてそれぞれの戦略・アルゴリズムで探索をさせ，使用したリソースを計測した．その結果，各拡幅戦略の間に大きな違いは見られなかった．一方，大くの局面で df-pn 駆動 dualλ 探索は df-pn 駆動 λ 探索より良い成績をおさめた．

2. Df-pn 駆動 λ 探索

df-pn 探索ではノードごとに証明数 (proof number)，反証数 (disproof number) を定義するが，df-pn 駆動 λ 探索ではこれを次数ごとのベクトル値とする．また，df-pn 駆動 dualλ 探索ではこのベクトル値をプレイヤごとに持つ．ここでは，局面 P をプレイヤ p の立場から次数 i で証明しようとする場合の証明数，反証数をそれぞれ $pn_p^i(P)$, $dn_p^i(P)$ と書くことにする．

λ 探索の性質から，$pn_p^j(P) = 0 (j \leq i)$ が成り立っているとき，P がプレイヤ p の立場から次数 i で証明できる (したがって，$pn_p^i(P) = 0$ として良い) ことが言える．一方，$dn_p^j(P) = 0 (j < i)$ が成り立っていても，次数 j で証明できないことがわかるだけで，次数 i での証明に関するヒントは何も与えてくれない．

また，ある j について $pn_{\bar{p}}^j(P) = 0$(ただし，$\bar{p}$ はプレイヤ p の敵のプレイヤを意味するものとする) が成り立っている時，すべての次数 i について，P がプレイヤ p の立場から次数 i で証明できない (したがって，$dn_p^i(P) = 0$ として良い) ことが言える．

探索によらずに決めることができるのは，次数 0 の証明数，反証数のみである．局面 P がプレイヤ p にとって直ちに勝つ手がある局面の場合は，$pn_p^0(P) = 0, dn_p^0(P) = \infty$ となり，それ以外の場合は $pn_p^0(P) = \infty, dn_p^0(P) = 0$ となる．

探索によって決まる証明数，反証数は OR ノード，AND ノードごとに定義される．

まずは，AND ノード (プレイヤ $\bar{p}$ の手番の局面)P の次数 i での証明数，反証数を求める．プレイヤ $\bar{p}$ はまずパス後の局面 ($pass(P)$ で表す) が次数 $i-1$ で反証されることを試み，それが失敗した後に通常の手生成をおこなうので，以下のようになる．

$$pn_p^i(P) = pn_p^{i-1}(pass(P)) + \sum_{Q \in children(P)} Pn_p^i(Q)$$

$$dn_p^i(P) = \begin{cases} dn_p^{i-1}(pass(P)) \\ \quad (\text{if } pn_p^{i-1}(\text{pass}(P)) \neq 0) \\ \min_{Q \in children(P)} dn_p^i(Q) \\ \quad (\text{otherwise}) \end{cases}$$

一方，OR ノード (プレイヤ p の手番の局面)P の次数 i での証明数，反証数は難しい．次数 i で証明をしたい時でも，次数 $j (j < i)$ で証明できれば良いし，そちらの方が効率的である可能性もある．そこで，OR ノードは図 1 のように仮想的に複数のノードに分けて考え，トップに次数に応じたノードに枝分かれする OR ノードがあるものとする．

図 1 で，P^0, P^1, P^2 は疑似ノード (pseudo node) と呼ばれ，次数をそれぞれ 0, 1, 2 に固定して P を探索した結果を表わす．$\bar{pn}_p^i(P)$ は P^i の証明数を，$\bar{dn}_p^i(P)$ は P^i の反証数を表する．次数を決めうちにした後のノードの証明数，反証数を $\bar{pn}_p^i(P), \bar{dn}_p^i(P)$

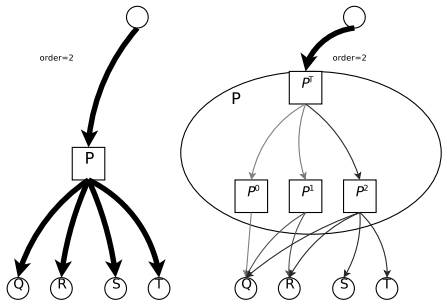

図 1 仮想的な OR ノード

とすると，以下のように定義される．

$$\bar{pn}_p^i(P) = \min_{Q \in children(P)} pn_p^i(Q)$$

$$\bar{dn}_p^i(P) = \sum_{Q \in children(P)} dn_p^i(Q)$$

この $\bar{pn}, \bar{dn}$ を求め，どの次数での証明を試みるかを決定する戦略を拡幅戦略 (widening scheme) と呼ぶ[8]．本研究では 4 つの拡幅戦略を用いた．以下その 4 つの拡幅戦略について説明する．ここではある i について， $pn_p^i(P)$ と $dn_p^i(P)$ を $\bar{pn}_p^j(P)$ と $\bar{dn}_p^j(P)$ (ただし $0 \geq j \geq i$) から求めるものとする．

証明数拡幅戦略は，疑似ノードの仮想的な親ノード (図 1 の P^T) を OR ノードとして扱うモデルに基づく戦略である．証明数拡幅戦略では各仮想ノードの証明数を比較し，証明数の小さい仮想ノードの証明数，反証数をノード全体の証明数，反証数として用いる．

繰り返し拡張戦略は，高い次数の探索のコストと比べて次数の低い探索のコストが無視できると仮定するモデルに基づく戦略である．繰り返し拡張戦略では反証されていない最小の次数の仮想ノードの証明数，反証数をノードの証明数，反証数として用いる．

- 証明数拡幅戦略 (pn)
  証明数拡幅戦略では証明数が最小となる疑似ノードの証明数，反証数をそのノードの証明数，反証数として用いる．疑似コードを以下に示す：

```
pn_p^i(P) = ∞
dn_p^i(P) = ∞
for j = 0 to i
    if p̄n_p^j(P) < pn_p^i(P) then
        pn_p^i(P) = p̄n_p^j(P)
        dn_p^i(P) = d̄n_p^j(P)
    fi
rof
```

- 漸次証明数拡幅戦略 (pn_2)
  漸次証明数拡幅戦略では i と $i-1$ に対応する疑似ノードの証明数を比べ，証明数が小さい方の疑似ノードの証明数，反証数をそのノードの証明数，反証数として用いる．疑似コードを以下に示す：

```
pn_p^i(P) = p̄n_p^i(P)
dn_p^i(P) = d̄n_p^i(P)
if i ≠ 0
    if p̄n_p^{i-1}(P) < pn_p^i(P) then
        pn_p^i(P) = p̄n_p^{i-1}(P)
        dn_p^i(P) = d̄n_p^{i-1}(P)
    fi
fi
```

- 繰り返し拡張戦略 (iw_all)
  証明数拡幅戦略では証明数が最小となる j の証明数，反証数をそのノードの証明数，反証数として用いる．疑似コードを以下に示す：

```
pn_p^i(P) = p̄n_p^i(P)
dn_p^i(P) = d̄n_p^i(P)
for j = 0 to i
    if p̄n_p^j(P) ≠ ∞ then
        pn_p^i(P) = p̄n_p^j(P)
        dn_p^i(P) = d̄n_p^j(P)
        break
    fi
rof
```

- 漸次繰り返し拡張戦略 (iw_2)
  反証されていない疑似ノードのうち，最小の次数を持つものの証明数，反証数をそのノー ドの証明数，反証数として用いる．疑似コードを以下に示す：

```
pn_p^i(P) = p̄n_p^i(P)
dn_p^i(P) = d̄n_p^i(P)
if i ≠ 0
    if p̄n_p^{i-1}(P) ≠ ∞ then
        pn_p^i(P) = p̄n_p^{i-1}(P)
        dn_p^i(P) = d̄n_p^{i-1}(P)
    fi
fi
```

3. Df-pn 駆動 Dual λ 探索

λ 探索は片方のプレイヤの攻めの速さだけに注目す

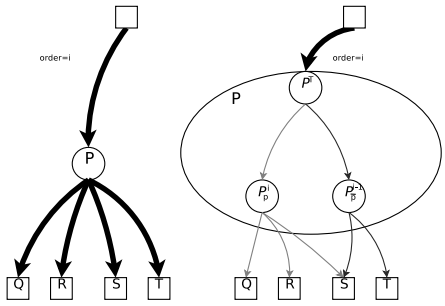

図 2 仮想的な AND ノード

るアルゴリズムであるのに対し，dualλ 探索は双方のプレイヤの攻めの速さに注目するアルゴリズムである[4),8)]．この節では dualλ 探索と df-pn 探索を組み合わせた df-pn 駆動 dualλ 探索について説明する．

df-pn 駆動 dualλ 探索では，OR ノードは df-pn 駆動 λ 探索と同様に扱う．

一方 AND ノードはプレイヤ $\bar{p}$ の攻撃も考えて証明数，反証数を計算する．プレイヤ p の攻撃の反証したい時でも，プレイヤ $\bar{p}$ の攻撃の証明をすれば良いし，そちらの方が効率的なこともある．そこで，AND ノードも図 2 のように仮想的に複数のノードに分けて考え，プレイヤ p の攻撃を考えた疑似ノードとプレイヤ $\bar{p}$ の攻撃を考えた疑似ノードとに分ける．

図 2 で, P_p^i, $P_{\bar{p}}^{i-1}$, はそれぞれプレイヤ p の攻撃を考えた疑似ノードとプレイヤ $\bar{p}$ の攻撃を考えた疑似ノードを表わす．P_p $\hat{pn}_p^i(P)$ は P^i の証明数を，$\hat{dn}_p^i(P)$ は P^i の反証数を表していると言える．

一方, 攻撃のプレイヤを決めうちにした後のノードの証明数, 反証数を $\hat{pn}_p^i(P), \hat{dn}_p^i(P)$ とすると，以下のように定義される．

$$\hat{pn}_p^i(P) = \sum_{Q \in children(P)} pn_p^i(Q)$$

$$\hat{dn}_p^i(P) = \min_{Q \in children(P)} dn_p^i(Q)$$

$\hat{pn}_p^i(P)$,$\hat{dn}_p^i(P)$ は図 2 での P_p の証明数，反証数に，$\hat{pn}_{\bar{p}}^i(P)$,$\hat{dn}_{\bar{p}}^i(P)$ は図 2 での $P_{\bar{p}}$ の証明数，反証数に対応する．

df-pn 駆動 dualλ 探索では疑似ノードの仮想的な親ノード (図 2 の P^T) を AND ノードとして扱うモデルに基づいている．現在の攻撃側のプレイヤ p が引き続き攻撃側として探索した場合のの反証数 $\hat{dn}_p^i(P)$ と，攻撃側を入れ替えて探索 した場合の証明数 $\hat{pn}_{\bar{p}}^i(P)$ とを比較し，より小さい方を用いる．ここで $\bar{p}$ が攻撃側のプレイヤの場合には証明数を用いることに注意されたい．

$pn_p^i(P) = \hat{pn}_p^i(P)$
$dn_p^i(P) = \hat{dn}_p^i(P)$
if $i \neq 0$
　if $\hat{pn}_{\bar{p}}^{i-1}(P) < dn_p^i(P)$ then
　　$pn_p^i(P) = \hat{dn}_{\bar{p}}^{i-1}(P)$
　　$dn_p^i(P) = \hat{pn}_{\bar{p}}^{i-1}(P)$
　fi
fi

また，ある j について $pn_{\bar{p}}^j(P) = 0$(ただし，$\bar{p}$ はプレイヤ p の敵のプレイヤを意味するものとする) が成り立っている時，すべての次数 i について，P がプレイヤ p の立場から次数 i で証明できない (したがって，$dn_p^i(P) = 0$ として良い) ことが言える．

4. 実　　験

以上で，df-pn 駆動 λ 探索と df-pn 駆動 dual λ 探索のアルゴリズムの概要を述べた．これらが，深さベースの反復深化を用いた λ 探索よりも効率がよいことを評価する実験をおこなった．

実験の対象としてはこの手の研究で良く使われている将棋や囲碁ではなくシンペイを用いる．シンペイ自体は既にすべての局面の勝敗が求められているので[5)]，ソルバ自体に需要があるわけではないが，評価の対象としては以下の点で興味深いので採用した．

- 全状態が 1200 万程度なのでまっとうなアルゴリズムならば実用的な時間で解ける (解ける解けないではなく，効率という基準で評価しやすい)．
- 勝ちまでの手数の分布が広い．勝つまでに 49 手必要な局面も存在することが知られている．
- ループが存在する．また，合流も多く木の探索としてだけとらえるのではなく，グラフの探索問題として考える必要がある．
- ZugZwang があるゲームであり，λ 探索では本質的に解けない局面が存在する．

λ 探索を実装する場合には，パスが定義されている必要がある．手元に駒がある時 (8 手目まで) にパスをした時の扱いを次のどちらにするかが，λ 探索では大きく影響する．

定義 1 パスをすると駒は手元に置き，次回以降の手番では手元に駒がある間は駒を置く (動かさない)．

定義 2 パスをすると駒は捨ててしまい二度と利用できない．盤面の自分の駒が 4 個未満でも，手元の駒がない場合は盤面の駒を動かす．

表 4 次数と最小証明木のサイズの平均値

次数	平均ノード数
1	16.76
2	223.586
3	1748.04
4	6634.11
5	9455.05
6	15505
7	8834.99

シンペイのルールではパスが合法手ではない☆ので，パスの定義としてはどちらを使っても良い．

一般のゲームではどちらが良いかを決定する際にゲームへの理解や予備実験が必要になるが，シンペイは全局面が求まっているので，それぞれの定義の場合の勝ち局面の次数 (以降は λ 次数のことを指す) の分布を求め，決定に役立てることができる．

結果を表 1 に示す．次数が ∞ になっているのは，λ 探索では勝ちを証明できない局面を表す．このように，定義 1 に従うと λ 探索では解けない次数 ∞ の局面数が多く，また最大次数も大きいため探索ノード数が大きくなることが予想される．そこで，以下では定義 2 を採用することにした．定義 2 は定義 1 よりも最大次数は小さいがそれでも 7 と大きめであり，λ 探索の実験対象としても興味深い対象であることが分かった．

ZugZwang が存在することから，次数が ∞ となる局面があることは予想されていが，20771(約 0.2%) という数字をどう評価するかは微妙である．本研究では λ を「限られたリソース内で勝ちがある場合に勝ちを効率よく証明する」という観点で考えると,「限られたリソース内で証明できない」問題の数が同程度になる領域に来るまでは問題にしないことにする．

また，親の次数とそのすべての子の次数の頻度を求めたものを表 2 と表 3 に示す．

λ 探索は次数が大きい局面ほど，勝ちの証明が難しい (コストがかかる) という仮定が成り立つゲームを対象としている．次数が n である局面の証明木は典型的には☆☆，証明木の深さは $2n$ 以上になるので，どのゲームでも一般に成り立つ性質ではあるが，シンペイでは全局面の最小証明木が求まっているので，直接確かめることができる．

図 3 に次数と最小証明木のサイズの度数分布を示す．このように，次数が大きくなるほど，最小証明木が大きくなる傾向を直接確かめることができた．一方で，次数が 1 上がるごとに，最初のうちはほぼ 10 倍になってきたのに，増え方が少なくなってきていて，たとえば λ 次数 6 と 7 では逆転しているように見える．

☆ 例外として，他に合法手がない場合のみ認められている．

☆☆ どの局面でもパスより悪い手しか存在しないということはないという仮定が成り立つ場合

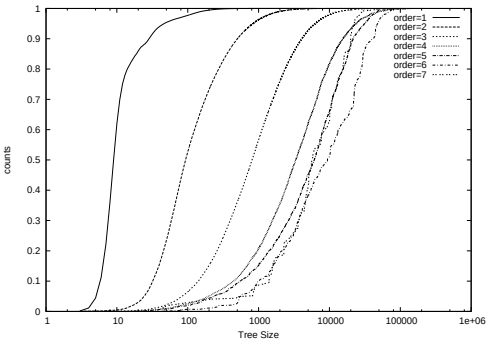

図 3 次数と最小証明木のサイズ (累積度数)

λ 探索は，次数 1 の探索が詰将棋ソルバ，次数 2 の探索が必至探索に対応するように，ドメインを限った探索アルゴリズムとして使うこともできるが，ここでは「勝ちが証明できる場合に勝ちを証明する」ことを目的にしたアルゴリズムとして評価することにする．この観点から勝ちが証明できる場合だけを評価に用いることにする．評価の際は，探索ノード数とテーブルに登録するノード数を用いる．それぞれ，時間的な制約と空間的制約のもとで利用することを想定している．ゲーム特有の知識により手生成を制限することが可能になる場合もあるが，シンペイに関しては有効な戦略が知られていないので，今回の実験では次数にかかわらず，すべての手を生成することにする．

全局面の 1000 文の 1 の局面をランダムに取り出したうち，勝ち局面を実験対象にして，以下の 5 種類のアルゴリズムのリソース消費量を計測した．

id 深さ打ち切りの全幅探索を深さを 2 ずつ増やしながら繰り返し反復する．

id-lambda トップレベルで次数を 1 からどんどん増やしていく．同じ次数内では深さを 2 ずつ増やしながら繰り返し反復する．

dfpn-lambda df-pn 駆動 λ 探索．岸本らによって提案されたループによる証明数無限増殖の防止法[6),7)] を実装している．証明数の二重カウントを防ぐための長井らの対策法[3)] は入れていないが，証明数のオーバーフローは起きずにすべての問題が解けた．

dfpn-plus-lambda 未展開の次数 i のノードの証明数，反証数を 1 ではなく 4^{i-1} としたもの．

dfpn-plus-dual-lambda dfpn-plus-lambda を dual にしたもの．

この結果をグラフにしたものが，図 4, 図 5 である．

表 1 パスの定義と次数ごとの局面数

パスの定義	次数											
	0	1	2	3	4	5	6	7	8	9	10	∞
定義 1	4380141	3446454	1132285	282643	47230	6911	2128	378	4	1	2	20998
定義 2	4380141	3446454	1134821	281263	46452	6801	2104	368	0	0	0	20771

表 2 親子の次数の関係 (親が OR ノード)

親の次数	子の次数								
	0	1	2	3	4	5	6	7	∞
1	0	17329252	8834710	2461392	347282	53838	19242	4018	52777298
2	0	0	4736644	1666900	251853	37093	10893	3133	27983178
3	0	0	0	635993	92901	12497	2863	854	7541536
4	0	0	0	0	57356	2017	573	70	1181530
5	0	0	0	0	0	7435	68	25	151991
6	0	0	0	0	0	0	2323	7	48147
7	0	0	0	0	0	0	0	410	9154

表 3 親子の次数の関係 (親が AND ノード)

親の次数	子の次数								
	0	1	2	3	4	5	6	7	∞
1	13912063	1926597	0	0	0	0	0	0	0
2	5236922	7960719	1341465	0	0	0	0	0	0
3	1638509	2144569	1239678	306708	0	0	0	0	0
4	270961	346563	183522	57850	42081	0	0	0	0
5	42893	58440	27897	6996	1337	6326	0	0	0
6	15851	20655	8686	1749	300	60	2147	0	0
7	3098	3824	1933	569	61	7	9	361	0

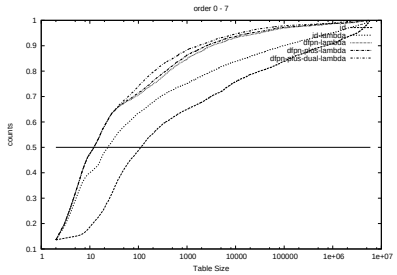

図 4 アルゴリズムごとの展開ノード数の比較

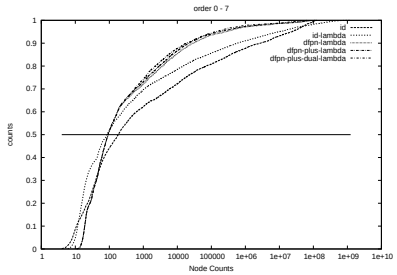

図 5 アルゴリズムごとの探索ノード数の比較

このうち，差がよく出ている次数 3 以上の部分だけをグラフにしたものが図 6, 図 7 となっている．後にいくほどリソース消費量が小さくなっている様子が分かる．

ただし，図 8, 図 9 のように，dfpn-plus-dual-lambda は dfpn-plus-lambda と比べて，多くの問題で改善されているが，問題によっては 10 倍程度のノード数を必要とする場合があることも分かっている．

以下の拡幅戦略の評価もおこなった．

pn 次数 1 から i まで証明数の一番小さい次数を選択する．

pn_2 次数 $i-1$ から i まで証明数の一番小さい次数を選択する．

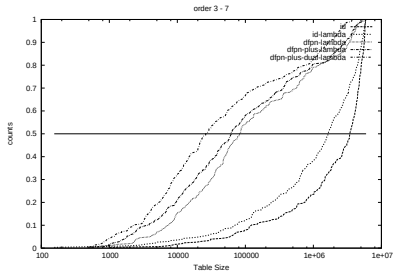

図 6 アルゴリズムごとの展開ノード数の比較 (次数 3 以上)

iw_all 次数 1 から i まで小さい次数を優先して選択する．

iw_2 次数 $i-1$ から i まで小さい次数を優先して

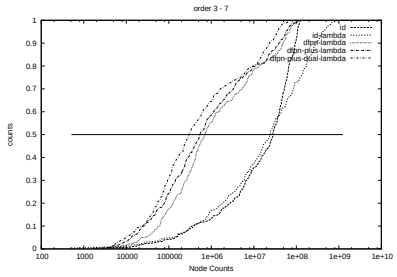

図 7 アルゴリズムごとの探索ノード数の比較 (次数 3 以上)

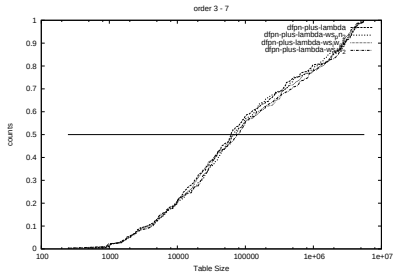

図 10 拡幅戦略ごとの展開ノード数の比較 (次数 3 以上)

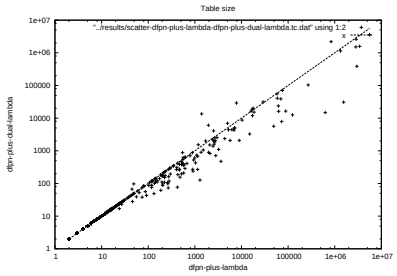

図 8 dfpn+ lambda dfpn+ dual lambda の展開ノード数の比較

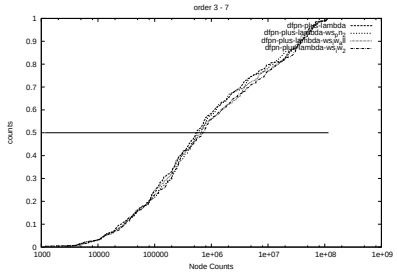

図 11 拡幅戦略ごとの探索ノード数の比較 (次数 3 以上)

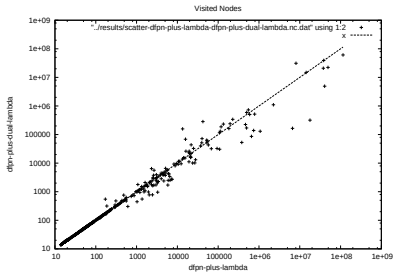

図 9 dfpn+ lambda dfpn+ dual lambda の探索ノード数の比較

選択する．

次数 2 以下ではほとんど差が出なかったので，次数が 3 以上のものだけを対象にグラフ化したものを図 10, 図 11 に示す．all が 2 よりも良い，`pn_based` が `iw` よりという傾向がみられる．ただし，この実験結果だけでは有意な差があるとは言えず，更に実験を重ねる必要があるものと思われる．

5. 結　　論

シンペイを題材とし，高い次数の脅威度が存在する局面での df-pn 駆動 λ 探索及び df-pn 駆動 dualλ 探索の評価をおこなった．全局面のうち 1000 分の 1 をランダムに取り出し，そのうち手番のプレイヤが勝ちとなる局面について探索させた，その結果以下がわかった：

- 拡幅戦略の間では大きな違いは見られなかった．
- 多くの局面で df-pn 駆動 dualλ は df-pn 駆動 λ より良い性能を示した．

今後の課題としては次のようなことが考えられる：

- 将棋や囲碁で良い性能を示した Proof Tree Tracing(Simulation)[1] を実装して評価をおこなう．
- ZugZwang の影響を調べる．λ 探索の性質から ZugZwang な局面を含む証明木は得られない．その結果，得られる証明木のが変化することが考えられるが，その影響について調べる．

参 考 文 献

1) Yasuhiro Kawano: Using similar positions to search game trees, Games of No Chance, pp. 193-202(1996).
2) Thomas Thomsen: Lambda-Search in Game Trees — with Application to Go, Lecture Notes in Computer Science 2063, pp. 19-38(2001).
3) 長井歩, 今井浩：df-pn アルゴリズムの詰将棋を解くプログラムへの応用, 情報処理学会論文誌, vol. 43, No. 6, pp-1769-1777(2002).
4) Shunsuke Soeda, Tomoyuki Kaneko and Tetsuro Tanaka: Dual Lambda Search and its Application to Shogi Endgames, Proceedings of the 11th Advances in Computer Games Conference(ACG 11), Sep 2005 Tapei(To appear).
5) 田中哲朗: ボードゲーム「シンペイ」の完全解析,

情報処理学会ゲーム情報学研究会資料 2006-GI-15-9, pp. 65 - 72(2006).
6) Akihiro Kishimoto and Martin Müller: Df-pn in Go: An Application to the One-Eye Problem, Advances in Computer Games 10, pp. 125-141(2003).
7) Akihiro Kishimoto: Correct and Efficient Search Algorithms in the Presence of Repetitions, PhD Thesis, University of Alberta,(2005).
8) Shunsuke Soeda: Game Tree Search Algorithms based on Threats, PhD Thesis, University of Tokyo, (2006).
9) Lambda Depth-first Proof Number Search and its Application to Go, Kazuki Yoshizoe, Akihiro Kishimoto and Martin Müller: In Proc. of IJCAI-07, Jan 2007 Hyderabad(To appear).