# IS 将棋の詰将棋解答プログラムについて

岸本章宏

## 1　はじめに

月日の経つのは早いもので、IS 将棋の詰将棋プログラムを担当してからすでに 5 年の月日が流れ、アルバータ大学に留学してから 3 年の時が流れた。現在は、博士論文のための研究を行いながら、その合間に詰将棋のプログラムを書くといった状況である。具体的には、博士の研究としてはコンピュータ将棋の経験を活かして、詰碁を正確にかつ高速に解くプログラムを書き、逆に詰碁の研究で得た知見をコンピュータ将棋側に取り込んでいる。

IS 将棋の詰将棋解答プログラムは、昨年までは脊尾の手法 [3] に反証数を不完全な形で用いていたのであるが、今年に詰碁の研究において、長井の df-pn アルゴリズム [4] を実装したことを機会に、一念発起して、df-pn ベースに書き直すことにした。少しテクニカルな話になるのだが、その際に直面した問題と解決策について説明する。今回の記事は、証明数、反証数、df-pn などの概念は既知のものとするので、その点を御了承いただきたい。

なお、この手法自体は元々はコンピュータ囲碁用に開発したものであり、囲碁について興味がおありの方のために [1] を参考文献に挙げておく。

## 2　Df-pn の問題点

Df-pn では証明数と反証数の閾値を持っているが、$n$ を現在探索中の節点、$th$ を閾値、$pn$ を証明数、$dn$ を反証数とするとある節点を展開するかどうかの判定条件は、次のようになっている。

$$n.pn \geq th.pn \quad || \quad n.dn \geq th.dn$$

Df-pn では上の条件を満たしたときに探索をストップして、親節点に情報を返す。ところが、この df-pn の判定条件は、将棋などのサイクルを含む探索空間では深刻な問題を引き起こす。例えば図 1 を考える。OR 節点が攻め方であり、AND 節点が玉方とする。

この図において、$A$ を探索するとする。$A$ を展開するにあたっての閾値は、大抵の場合は、$A$ の証明数 (や反証数) よりも少しだけ大きな値に設定されている。例えば、$A$ での証明数の閾値を $th.pn = A.pn + 1$ として、$D$ の証明数

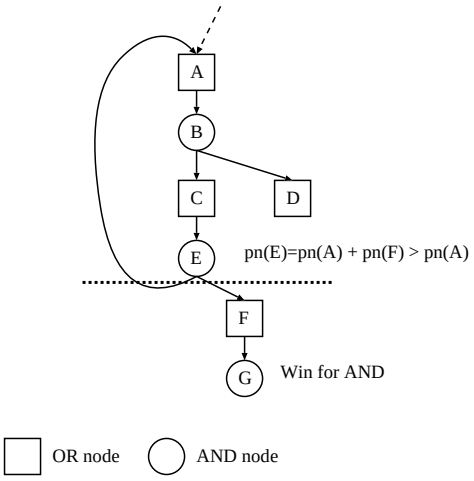


図 1: サイクルを含む空間における Df-pn の問題

を 1 とする。この状況で、$E$ にたどりついたとすると、$E$ での証明数の閾値は、$th.pn = A.pn$ である。$E$ の証明数は、$F$ の証明数を 1 と仮定する (まだ展開されたことがない) と

$$E.pn = A.pn + F.pn = A.pn + 1 > A.pn = th.pn$$

となる。

ここで、df-pn の判定条件を考えると、$E$ の証明数は、証明数の閾値より大きいので、$E$ は展開されずに親節点に戻ることになる。[1] このグラフ自体は、$F \rightarrow G$ を展開さえすれば、AND 節点の勝ち (つまり不詰) が証明されるはずなのだが、上の現象が永久に起こり続けるために $E$ 以下の節点が全く展開されない。

なお、頻度は少ないものの似た現象自体は脊尾の手法でも起こる。以前のバージョンでは、例えば、将棋無双の 100 番の「大迷路」が解けなかったのだが、この問題に起因していた。

# 3 最小距離法

## 3.1 証明 (反証) 数の計算の変更

上の問題を解決するためにハッシュのエントリーを増やして、節点を展開するたびに、ルート節点からのその節点までの最小距離 (minimal distance)

[1] 注意しないといけないことは、サイクルがあるからといって、無条件に不詰をハッシュに入れてはいけない。それを行うと詰みの問題を不詰と間違える (またはその逆) いわゆる GHI 問題 [2] を引き起こしてしまう。

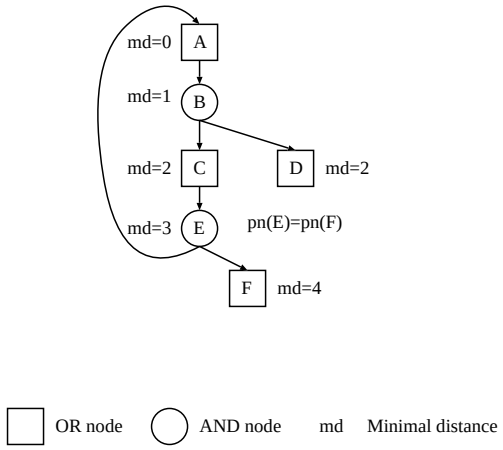


図 2: 最小距離を利用した証明数の計算の例

を計算して、その最小距離の情報を保持するようにする。基本的なアイデアとしては、探索の深さが深くなるにしたがって、最小距離の値が大きくなるはずであるので、その事実を利用して、サイクルを引き起こす可能性のある節点の証明数を無視するようにする。具体的には、$md$ を最小距離とすると、$n_1, \cdots, n_l$ というのは $n_i.md > n.md$ であるような $n$ の子節点 $n_i$(以下、普通の子節点と呼ぶ) であり、$n_{l+1}, \cdots, n_k$ は、$n_j.md \leq n.md$ となる子節点 $n_j$ (以下、異常な子節点と呼ぶ) として、AND 節点 $n$ での証明数の計算方法を以下のように変更する。

$$n.pn = \begin{cases} \displaystyle\sum_{i=1}^{l} n_i.pn & (\text{if } \displaystyle\sum_{i=1}^{l} n_i.pn \neq 0) \\ \displaystyle\max_{l+1 \leq j \leq k} n_j.pn & (\text{if } \displaystyle\sum_{i=1}^{l} n_i.pn = 0) \end{cases}$$

OR 節点の反証数の計算に対しても同様のことを行う。

図 2 が最小距離法を用いた証明数の計算例である。$F$ を未展開の節点とすると $F.pn = 1$ であるとして、$E$ の証明数を計算するとすると $A$ は $E$ にとっては異常な子節点であるので、$E$ の証明数には足しこまれない。その結果 $E.pn = F.pn = 1$ となる。

## 3.2 最小距離の更新

最小距離法を動かすためには上の方法だけでは不十分である。というのは、$n$ を AND 節点として、普通の子節点と異常な子節点を持っていたとして、さ

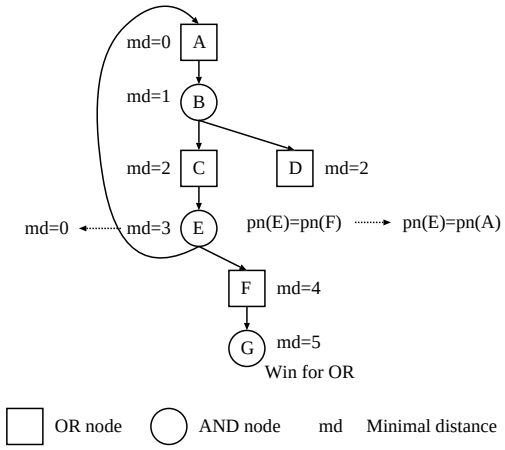


図 3: $E$ の最小距離の更新の様子

らに普通の子節点はすでに証明 (詰み) されてしまったとする。そうすると $n$ の結果を調べるためには、異常な子供の節点を展開しないといけなくなる。つまり、$n$ 自体が異常な節点になってしまうわけである ($n$ が OR 節点のときも同様)。この場合は、$n$ の最小距離を更新してやらないといけない。現在の実装では、この状況が起こったときには、まだ解かれていない異常な節点の中で、最小距離が一番小さなものに $n$ の最小距離を更新している。

図 3 と 4 に最小距離の更新の例を示す。$G$ を OR 節点勝ち (詰み) の局面とすると $E$ の勝ち負けを判定するには $A$ を展開しないといけない。つまり $E$ は異常な子節点しか持っていないので、$E$ の最小距離は、$A$ の最小距離で更新され、$E.pn = A.pn$ となる。同様に $C$ の最小距離は $E$ の最小距離で更新される (図 4)。その結果として、$C$ は $B$ にとっては異常な子節点に変化するので、$B$ の証明数の計算のときには、$C$ の証明数は無視され、$B.pn = D.pn$ となる。

# 4 最小距離法の限界

例えば、図 5 などはサイクルがないのにもかかわらず、反証数が足しこまれない。$C$ の反証数は $C.dn = D.dn + E.dn$ であるべきだが、$D$ が反証されない限りは、$C.dn = D.dn$ となってしまう。同様に、この方法はすべてのサイクルに起因する df-pn の問題を検出できるわけではない。現在のプログラムでは、このような例は起こっており、将棋依存の条件を書き下すことで解決している。

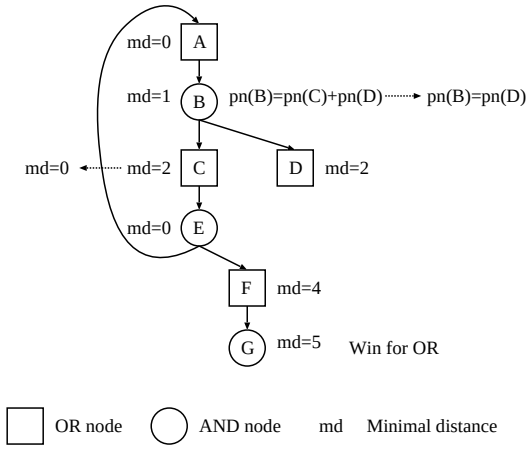


図 4: $C$ の最小距離の更新の様子

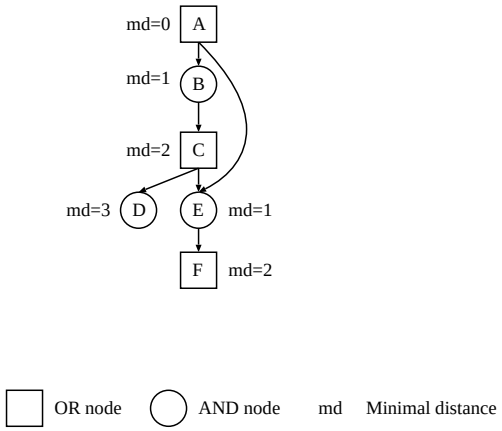


図 5: 最小距離法の問題点

## 5　最小距離法の有効性

[1] での実験では、最小距離法がなければ、囲碁では df-pn は効率よく動かないことが示されている。詰将棋の場合の具体的なデータは試していないが、df-pn の問題を解決する方法が何らかの形で実装されていなければ、[1] の場合と同じことが起こって実用にならなかった。

さきほども述べたように、前のバージョンの詰将棋プログラムにはサイクルに関わる問題に対して、いくつか解けないことがあるという弱点があったのだが、現在の IS 将棋の詰将棋は基本的にはその問題を解決しているようである。その結果として、「将棋図巧」、「将棋無双」、「続詰むや詰まざるや」のすべての詰む問題を解けるようになった。

## 6　おわりに

最後に今回のトピックとは直接は関係はないが、個人的な意見を書かせていただく。詰将棋の研究は終わったかのように述べられており、私自身もそのように思っていた時期があったが、現在は、次の理由から、そうとも言えないのではないかと考えている。

- 詰将棋プログラムの探索木が、証明木 (詰みの証明に必要な最小の場合) に比べてはるか大きい。
- 現状のプログラムは、不詰であることを示すのが遅い。

そういうわけで、現状の問題を解決するさらなる手法を開発して、コンピューター将棋の進歩に少しでも貢献できれば幸いである。

## 参考文献


[1] Akihiro Kishimoto and Martin Müller. Df-pn in Go: An application to the one-eye problem. In *Advances in Computer Games Many Games, Many Challenges*, pp. 125–141, 2003.

[2] A. J. Palay. *Searching with Probabilities*. PhD thesis, Boston University, 1985.

[3] 脊尾昌宏. 共謀数を用いた詰将棋の解法. コンピュータ将棋の進歩２, pp. 1–28. 共立出版, 1998.

[4] 長井歩. df-pn アルゴリズムと詰将棋を解くプログラムへの応用. アマ４段を超える コンピュータ将棋の進歩４, pp. 96–114. 共立出版, 2003.