難解な必至問題を解くアルゴリズムとその実装

長　井　　歩

将棋の終盤戦をパズル化した問題として，詰将棋以外に必至問題がある．その違いは，攻め側の手として王手以外に詰めろも許される点にある．詰将棋では攻め側の指せる手は王手のみであるのに対し，必至問題では王手と詰めろで相手の玉を受けなしに追い込む．攻め側の手が増える分，必至問題に強いアルゴリズムは，実際の将棋の終盤戦で役立つ機会は詰将棋に比べ格段に多くなる．しかし問題としての難易度は，一般に詰将棋よりも必至問題の方が難しい．

詰将棋を高速に解くアルゴリズムは近年著しく進歩したが，必至問題を高速に解くアルゴリズムは未開拓である．本研究では，難解な必至問題を高速に解くアルゴリズムとして df-pn$^+$ を応用し実装した．実験の結果，難解な必至問題集として有名な『来条克由必至名作集』全 81 問のうち 79 問を解くことに成功した．また，余必至探索にて３つの早必至を含む 27 の余必至を発見できた．

An Algorithm to Solve Hisshi Problems

Ayumu Nagai[*1]

Hisshi is an endgame of Shogi. It is different from Tsume Shogi because of permitting Tsumero besides Oute. Since a strong algorithm to solve Hisshi has more chance to solve actual endgames of Shogi, it may allow computer program to become still stronger. However, generally Hisshi is more difficult than Tsume Shogi.

While an algorithm to solve Tsume Shogi has greatly advanced, so far an algorithm to solve Hisshi has not advanced so much. We developed an algorithm based on df-pn$^+$ which can solve hard Hisshi problems. Experimental results on "Kitajyo Hisshi Problems" show that we solved 79 problems out of 81. Besides, we also found 27 Yohisshi including three Hayahisshi.

1. はじめに

計算機によって詰将棋を解くアルゴリズムは近年急激な進歩を遂げ，現在最長の 1525 手の詰将棋を含む 400 手以上の長編詰将棋がすべて (2002 年当時) 解かれる[18] など，殆どすべての詰将棋は計算機によって解くことができる[4]．計算機によって必至問題を解くのもその恩恵を受けていることは間違いないが，その能力や限界についてはこれまであまり本格的に論じられてこなかった．そもそも必至問題を解くことに焦点を当てた研究が非常に少ない．我々の知る限り，既存のすべての必至問題を解くための試み[11),13] は，２種類の探索エンジンからなる．詰将棋を解く詰み探索エンジンと，それを内部で呼び出す何らかの上流の探索エンジンである．上流の探索エンジンはその時点での探索木の末端近くで何度も詰み探索エンジンを呼び出す．必至探索専用プログラムの場合，上流探索エンジンはいわば必至探索エンジンである．詰み探索エンジンは詰めろの生成に用いる．指し将棋用プログラムは専用の必至探索エンジンこそ持たないが，上流探索エンジンである反復深化法から何度も詰み探索エンジンを呼び出す構造は同じである．

これに対し本研究で開発したアルゴリズムには，必至探索エンジンと詰み探索エンジンのような明確な区別はない．df-pn$^+$[6] の一つの応用として実装した．さらに詰将棋を解くための工夫を取り入れ，高速に必至問題を解くことに成功した．難解な必至問題集として有名な『来条克由必至名作集』を解かせたところ，81 問中 79 問を解くことに成功した．

2. df-pn$^+$ アルゴリズム

提案法は df-pn$^+$ アルゴリズムの応用である．そこで，まず最も単純なバリエーションである df-pn の基本的な考え方を述べたあとで df-pn$^+$ を説明する．

df-pn は証明 (できるか否か) を目的とした探索法である．詰将棋に適用する場合，詰みか不詰かを証明しようとする．必至問題に適用する場合，必至 (詰みを含む) か不必至かを証明しようとする．出力は，必至がかかるか否かの結論に加えて，その手順も返すことができる．

df-pn では，現在までに探索している各末端節点が (その時点では未知だが) 等確率で必至もしくは不必至と仮定する．それらの末端節点の中から次に展開する節点の選択基準として，根節点 (問題局面) が必至もし

*1 群馬大学大学院工学研究科
Department of Computer Science, Gunma University
E-mail: nagai@cs.gunma-u.ac.jp

くは不必至を示すために最も多くの情報を得られる末端節点 (これを most-proving node[3] という) を選択し展開する．そのために，証明数と反証数という 2 種類の数字を導入する．これをナイーブに実装すると最良優先探索法 (pn-search[1]) になる．それに対し df-pn は，深さ優先でありながら最良優先の pn-search と同じ振る舞いをする[7] ように開発された探索法である．

df-pn の枠組みでは，末端節点が等確率で必至もしくは不必至と仮定した．しかし実際には五分五分ではなく，もっと精度よく予測 (評価) できることも多い．そのような問題設定のために評価関数を盛り込んだのが df-pn$^+$ である．評価関数には 2 種類ある．一つは末端局面 n において，必至および不必至への遠さを与える関数 $\mathbf{h}_{\rm pn}(n)$ および $\mathbf{h}_{\rm dn}(n)$ である．もう一つは任意の親子局面 n と $n_{\rm child}$ の間で定義され，必至および不必至への遠ざかり方を与える関数 $\mathbf{cost}_{\rm pn}(n, n_{\rm child})$ および $\mathbf{cost}_{\rm pn}(n, n_{\rm child})$ である．

詰方にとっての証明数と受方にとっての反証数は同等なので，その違いを吸収するため $\mathbf{h}_\phi$，$\mathbf{h}_\delta$，$\mathbf{cost}_\phi$，$\mathbf{cost}_\delta$ を下記のように定義する．

定義 1　$\mathbf{h}_\phi$，$\mathbf{h}_\delta$，$\mathbf{cost}_\phi$，$\mathbf{cost}_\delta$ の定義

(1)　n が *OR* 節点，すなわち詰方 *(*攻め側*)* の手番

$$\begin{cases} \mathbf{h}_\phi(n) & \equiv & \mathbf{h}_{\rm pn}(n) \\ \mathbf{h}_\delta(n) & \equiv & \mathbf{h}_{\rm dn}(n) \\ \mathbf{cost}_\phi(n, n_{\rm child}) & \equiv & \mathbf{cost}_{\rm pn}(n, n_{\rm child}) \\ \mathbf{cost}_\delta(n, n_{\rm child}) & \equiv & \mathbf{cost}_{\rm dn}(n, n_{\rm child}) \end{cases}$$

(2)　n が *AND* 節点，すなわち受方 *(*玉側*)* の手番

$$\begin{cases} \mathbf{h}_\phi(n) & \equiv & \mathbf{h}_{\rm dn}(n) \\ \mathbf{h}_\delta(n) & \equiv & \mathbf{h}_{\rm pn}(n) \\ \mathbf{cost}_\phi(n, n_{\rm child}) & \equiv & \mathbf{cost}_{\rm dn}(n, n_{\rm child}) \\ \mathbf{cost}_\delta(n, n_{\rm child}) & \equiv & \mathbf{cost}_{\rm pn}(n, n_{\rm child}) \end{cases}$$

詰方にとっての証明数と受方にとっての反証数は同等なので，まとめて ϕ と定義する．同様に詰方にとっての反証数と受方にとっての証明数は同等なので，まとめて δ と定義する．

定義 2

$n.\phi$: *OR(AND)* 節点 n の証明数 *(*反証数*)*

$n.\delta$: *OR(AND)* 節点 n の反証数 *(*証明数*)*

(1)　n が葉節点

(a)　n が *OR(AND)* 節点で必至 *(*不必至*)*

$$n.\phi = 0$$
$$n.\delta = \infty$$

(b)　n が *OR(AND)* 節点で不必至 *(*必至*)*

$$n.\phi = \infty$$
$$n.\delta = 0$$

(c)　必至でも不必至でもない

$$n.\phi = \mathbf{h}_\phi(n)$$
$$n.\delta = \mathbf{h}_\delta(n)$$

(2)　n が内部節点

$$n.\phi = \min_{n_i \in n \text{ の子節点}} (n_{\rm i}.\delta + \mathbf{cost}_\phi(n, n_{\rm i}))$$
$$n.\delta = \sum_{n_i \in n \text{ の子節点}} (n_{\rm i}.\phi + \mathbf{cost}_\delta(n, n_{\rm i}))$$

df-pn$^+$ は ϕ, δ と $\mathbf{h}, \mathbf{cost}$ を用い，下記のように記述できる．

Procedure Df-pn$^+$:

Step 0)　根節点 r にて下記を代入.

$$r.th_\phi = \infty$$
$$r.th_\delta = \infty$$

Step 1)　各節点 n にて，$n.\phi \geq n.th_\phi$ もしくは $n.\delta \geq n.th_\delta$ となるまで **Step 2), 3)** を実行 *(*これが終了条件*)*

Step 2)　節点 n の子節点 n_i のうち *(*$n_i.\delta + \mathbf{cost}_\phi(n, n_i)$*)* が最小な子節点を n_c，*2* 番目に小さい子節点を n_2 とする.

Step 3)　n_c のしきい値を下記のように計算し，n_c を探索．**Step 1)** を再帰呼び出し.

$$n_c.th_\phi = n.th_\delta + n_c.\phi - \sum(n_{\rm i}.\phi + \mathbf{cost}_\delta(n, n_{\rm i})) \quad (1)$$
$$n_c.th_\delta = \mathbf{min}(n.th_\phi, n_2.\delta + \mathbf{cost}_\phi(n, n_2) + 1) - \mathbf{cost}_\phi(n, n_c). \quad (2)$$

Step 4)　終了条件が成立したら，n の親節点へ戻る．もし n が根節点なら探索終了.

$\mathbf{cost} = 0$ かつ $\mathbf{h} = 1$ のとき，df-pn$^+$ は df-pn と同じになる.

3. 提案アルゴリズム

提案法は df-pn$^+$ の一つの応用である．それを実現する鍵は,「必至探索と同時並行な詰めろ判定」と「指し手生成の切り替えのためのフラグ」の存在である.

提案法のアイデアは，受方の手としてパスも敢えて許容し，パスの手も他の受けの手と同様に探索し詰みを確認させることによって，直前の詰方の攻撃手が最終的には詰めろであることを保証する．パスの手は直前の攻撃手が王手でない場合に許容し，パスのあとの攻撃手は王手に限定する．従来法ではまず最初に詰み探索エンジンによって詰めろかどうかの判定を行い，詰めろであると判定できたなら必至探索を行っていた．同時並行に行うメリットは，王手でない攻撃手の直後にパスしたあとの詰みの発見が容易でない場合，(つまり詰めろの確認が容易でない場合，) そのような攻撃手による必至探索を控えめにすることができる点にあ

る．人間のエキスパートも簡単な詰めろの手から読んでいるものと考えられるためである．もちろんパス後に詰みが発見できれば全力で必至探索するが，これは df-pn$^+$ にて実装すれば自然にそのような性質を持つ．

またこのような探索の実現のため，現在探索中の手順にパスが含まれているかどうかのフラグを持ち，フラグが立っていれば攻め側の手として王手しか生成しない．フラグが立っていなければ王手を含むその他の攻撃手も生成する．提案法に詰み探索エンジンと必至探索エンジンの区別はないが，このフラグによってその違いを吸収できる (図 1 参照)．

このように指し手生成に特異性を持たせた df-pn$^+$ を実装したのが提案法である．df-pn$^+$ を採用した理由は，まず df-pn が難解な詰将棋を解くアルゴリズムとして標準的になっていることと，df-pn$^+$ はその拡張で様々なパラメータを導入することによって domain-specific なチューニングができることである．必至問題を解く場合，王手とそれ以外の手を同等に扱うべきでないことは容易に想像できるため，そのような調整に適した df-pn$^+$ を採用した．df-pn$^+$ は **cost** と **h** の 2 種類の関数を導入できる．本研究では，このうちの証明数用の **cost** 関数を王手以外の手の場合に限り負荷 (ペナルティー) を与えている．具体的には n が OR 節点で，王手以外の手で子節点 n_i に移動できるとき，$\mathbf{cost}_\phi(n, n_\mathrm{i}) = 2$ とした．それ以外は $\mathbf{cost} = 0$ とした．**h** については df-pn と同じく 1 に固定した．

基本的には詰将棋用のアルゴリズムと同様のヒューリスティックを導入したが，次節以降に詰将棋の探索と事情が違い工夫した点を説明する．

3.1 手の生成

実際に生成している手を下記にまとめる．受方の手については最終的にはすべての手を生成する．これに対し詰方の手については一部の手しか生成していない．その意味するところは，プログラムが必至との結論を出した場合には間違いなく必至がかかる反面，不必至と結論付けたとしても実際には必至がかかる可能性がある．それは詰方の手に読み抜けがあるためである．

- 詰方の手
  - 王手
  - 相手玉の 15 近傍に効きを付ける手 (図 2 参照)
  - 香以上の駒を取る手
  - 焦点の歩・香
  - 筋当り・1 筋違いの飛・角・香
- 受方の手 (直前の攻撃手が王手の場合)
  - 玉の移動手
  - 王手している駒を取る手
  - 合駒と中合い
  - 移動合い
- 受方の手 (直前の攻撃手が王手以外)

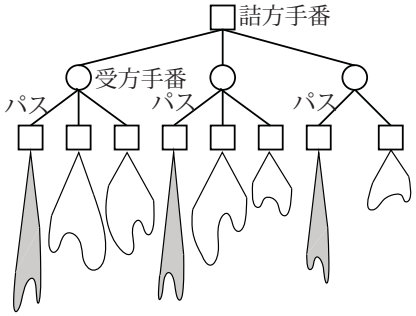

図 1 提案法は df-pn$^+$ の応用である．詰み探索エンジンと必至探索エンジンの明確な区別はない．受方がパスしたあとは王手のみを生成するので，詰み探索となる (網掛けの部分)．その切り替えのためのフラグを持つ．

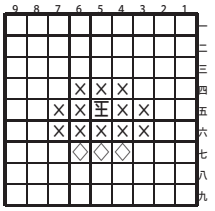

図 2 玉の 15 近傍に効きを付ける手．×のマスに効きを付ける手と◇のマスへ桂によって効きを付ける手．

  - パス
  - 玉の移動手
  - 駒を取る手
  - 玉の逃げ道を作る手
  - 大駒の移動手
  - 8+3 近傍への対抗効き (3 は図 2 の◇)
  - 直前の飛角香の飛び効きを止める手
  - 反駁手 (直前の攻撃手が詰めろと判明後生成)
  - その他すべての合法手

手が重複する場合がある．例えば受方の手として，詰方の駒を取りつつ玉の逃げ道を作る手も存在しうる．そのような場合には詰方の駒を取る手として扱っている．上に列挙した順に手の分類上の優先順位を付け，重複して生成することのないようにしている．

受方 (AND 局面) での証明数のカウント法についてであるが，df-pn では本来はすべての受け手の証明数の総和とするが，提案法では次のような工夫をしている．直前の攻撃手が王手の場合は詰将棋の場合と同じである．すなわち合駒や中合については無効合いと予想できる場合は全くカウントせず，有効合いと予想できる場合もそれらの手のうち最大の証明数にて代表させるなどの工夫をしている．直前の攻撃手が王手以外の場合は，駒を取る手については取られる駒ごとに最大の証明数にて代表させる．対抗効きの手，玉の逃げ道を作る手，飛び効きを止める手，反駁手についても 1 手だけカウントする．取る手については，取られる駒が銀以上の駒か，玉の 8 近傍内か，直前の駒を取る

手を除いてカウントしていない．大駒の移動手，その他すべての合法手についてもカウントしていない．

詰方 (OR 局面) での反証数のカウント法についても同様に，総和を計算する代わりに，同じ方向からの攻撃手については，それらの反証数の最大値で代表させたり，駒を取る手に関しては，取られる駒ごとにそれらの反証数の最大値で代表させるという工夫をした．

3.2 局面間の優越関係

現在探索中の手順にパスが含まれているかどうかのフラグについて前述した．この詰み・必至フラグは探索エンジンが持つフラグであるが，提案手法では，局面をハッシュに登録する際にこのフラグの情報も書き込んでいる．それによって詰み探索中の局面情報なのか，必至探索中の局面情報なのかを区別する[⋆1]．たとえ同じ局面 (盤面と持ち駒が同じ) であっても，フラグが立っているかどうかで区別するのである．しかし両者の持つ情報 (特に証明数や反証数の情報) が全く無関係なわけではなく，優越関係が成り立つ．提案手法ではその優越関係を利用している．

具体的には，詰み探索中は攻撃手が王手に限定されている意味で勝ちにくい．つまり必至探索中の局面は，詰み探索中の同一局面に比べて詰方にとって優越している．詰み探索中の局面の証明数は，必至探索中の同一局面の証明数と同じかそれ以上になる．同様に，必至探索中の局面の反証数は，詰み探索中の同一局面の反証数と同じかそれ以上になる．

持ち駒による優越関係と組み合わせ，さらに広く優越関係を利用できる．すなわち，たとえ同一局面でなく，詰み・必至フラグが一致していなかったとしても，盤面さえ同じ (持駒だけの違い) であれば優越関係を利用できる場合がある．それは持駒の優越した局面が必至探索中の場合，持駒の劣った詰み探索中の局面に優越する．

提案法では以上の優越関係を利用しながらハッシュを参照する．

3.3 証明駒・反証駒の拡張

証明駒とは，詰ますために最低限必要な詰方の持駒の集合のこと[17]である．詰みを探索するアルゴリズムの発展の中から生まれた工夫である．探索中に詰みを発見したものの，よく考えてみると実は駒余りということがしばしばある．そのような時に，詰ますために必要な最小限度の持駒を計算したものが証明駒である．反証駒はその裏返しで，不詰のために最低限必要な受方の持駒の集合のことである．

必至探索においても証明駒・反証駒を用いて探索を効率化できる．注意が必要なのは受方手番の局面で詰む (必至がかかる) 場合の証明駒の計算と，詰方手番の局面で不詰 (不必至) の場合の反証駒の計算である．基本的には，詰め上がりや逃れに至るまでに使用した持ち駒の和集合を考えればよい．しかし (自分は打つ必要がないながら) 相手がその駒を打つ手を消すために自分が持駒にする場合がある．このような場合にも対応するには，仮に自分の持駒を相手の持駒に渡したと仮定したときに，新たな手が生じるかどうかを以って判断する．新たな手が生じる可能性があるのは，相手が持っていない種類の持駒を渡すときであるが，詰み探索に比べ必至探索の場合は少し状況が異なる．

詰み探索の場合，証明駒の計算において新たな手が生じるのは，飛び効きによる王手 (両王手を除く) の場合に限り合駒の形で持駒を打てる (合法手の判断は必要)．反証駒の計算においては，渡した駒によって王手できる場合に限られる．いずれも即座に判断できる．

必至探索の場合，証明駒の計算にて新たな手が生じるのは，詰めろに対する受けの局面においてである．王手がかかっていないので受方に持駒を打つ手は一般に多数生じる．提案法ではこのような手が生じないように，受方が持っていない種類の持駒は受方には渡さない．反証駒の計算にて新たな手が生じるのは，詰方の局面において，渡した持駒を打つ詰めろが生じるときである．提案法では詰めろ判定のコストが高いこともあり，万全を期して詰方が持っていない種類の持駒は詰方には渡さない．このように必至探索においては証明駒や反証駒の効果は，詰み探索ほどは期待できないと考えられる．

3.4 無駄合い処理の拡張

無駄合い処理とは本来は詰将棋に由来する用語で，詰方の飛び駒 (飛角龍馬香) による王手に対する受方の合駒のうち，無駄に手数を伸ばすだけの効果しかない手を除外する処理のことである．除外後に残った手の中から最善手 (最長応手) を選択する．実際の処理方法として，柿木の無駄合い処理[12]が有名である．これは合駒によって詰み手数が長くなったかを以って無駄かどうか判断する．すなわち，無駄合い判定対象の合駒を飛び駒で取ったうえで受方に渡し，その局面が何手で詰むかを調べる．合駒した手とそれを取った 2 手をカウントしないときの詰み手数が，合駒以外の受けをした場合の手数以下になるかどうかを以って無駄合い判定を行う．

提案法の必至探索においても，同様の処理を拡張して実行している．作品としての必至問題においても，詰将棋と同様に作意手順に無駄な手が登場しない傾向があるためである．その例を図 3 に示す．図 3 は『来条必至』第 34 番である．作意は▲２二金までの 1 手必至である．しかし厳密には図 3(b) 以下▽１五歩で即詰みはなくなる．だが▲１五同香で今度こそ必至が

⋆1 提案法では必至探索中も王手を探索する．王手を探索した結果詰みと判明した場合，ハッシュには必至フラグではなく，詰みフラグを記録する．

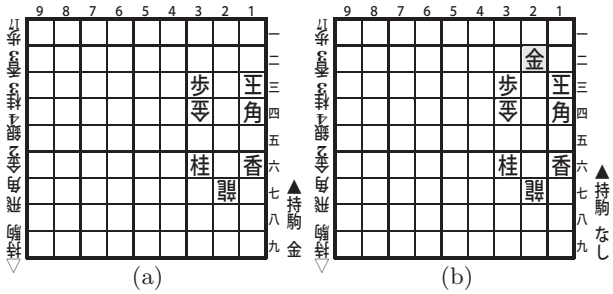

図 3 無駄合いの例．(a) は『来条必至』第 34 番の問題局面．作為は▲２二金までの 1 手必至．(b) がその局面．しかし厳密には▽１五歩で即詰みはなくなる．だが▲１五同香で今度こそ必至がかかる．つまり▽１五歩は一種の「無駄合い」である．

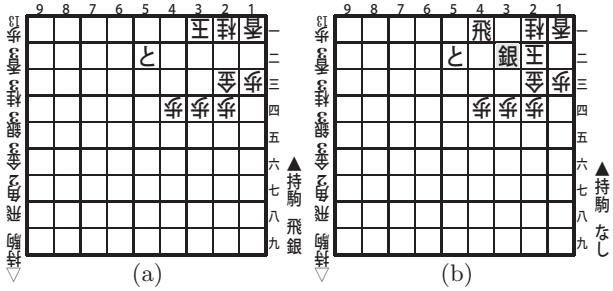

図 4 金子タカシ『詰みより必死』[14] の第 27 番．(a) は問題局面．作為は▲４一飛▽２二玉▲３二銀までの 3 手必至．(b) がその局面．しかし厳密には▽９一飛で即詰みはなくなる．だが▲９一同飛成で受けになっていない．広い意味で「無駄合い」と言える．詰将棋の無駄合いと違うのは，飛び駒と玉の間の合駒だけを考えればいいわけではない点に注意が必要である．

かかる．つまり▽１五歩は一種の「無駄合い」と言える．実際この事実を踏まえたうえで，作意手順を▲２二金までの 1 手必至としている．図 3(b) で▽１五歩を無駄合いと判定するため，柿木の無駄合い処理と同様の処理を行う．つまり，１五の歩を取ったうえで受方に渡し，その局面に何手で必至がかかるか調べる．合駒の手とそれを取った 2 手をカウントしないときの必至手数が，合駒以外の受けをした場合の手数以下になるかどうかを以って無駄合い判定を行う．

図 3 は最も単純な例である．それに対し，図 4 は詰将棋との違いがよく現れる．作為は▲４一飛▽２二玉▲３二銀までの 3 手必至である．しかし厳密には図 4(b) 以下▽９一飛で即詰みはなくなる．だが▲９一同飛成で受けになっていない．広い意味で「無駄合い」と言える．詰将棋の無駄合いと違うのは，飛び駒と玉の間の合駒だけを考えればいいわけではない点にある．

上述のように，必至探索では詰方の飛び駒の効きのあるマスに打つ駒はすべて無駄合いの可能性がある．現在我々の実装では，詰方のすべての飛び駒 (の効き) について調べてはいない．詰方が最後に触った飛び駒 (の効き) に限って調べている．正確には，(直前の手かどうかに限らず) 詰方が最後に触った飛び駒の効きがあるマスに打つ手だけを無駄合い判定の対象にしている．それは複数の飛び駒の効きがある場合を考えたくなかったためである．このような中途半端な実装のため，図 4(b) の▽９一飛は無駄合いと判定できるが，図 3(b) の▽１五歩は無駄合いと判定できない．

4. 実験結果

まず主題である必至探索の結果を示す．さらに，実装した必至探索プログラムを活用して発見した余必至，および変化長手数の結果を示す．ここに示す実験の対象は，難解な必至問題集として名高い『来条克由必至名作集』[20] (以下，単に『来条必至』) である．

4.1 必至探索

提案法を実装したプログラムにて『来条必至』を解かせたところ，81 問中 79 問を解くことに成功した．表 1 はその結果である．計算機環境は Core i7 860 (2.8GHz) メモリは 4GB であるが，ハッシュに用いているのは 100MB である．

解けなかったのは第 70 番と第 75 番である．我々はこれらの問題は不必至ではないかと考えている．その理由を述べる．まず第 70 番について，問題局面を図 5(a) に示す．作為手順は▲１一金▽同玉▲１三香不成▽２一玉▲２三桂不成▽３三歩▲４四桂▽同歩に▲４三金以下の 23 手必至であるが，7 手目▲４四桂 (図 5(b)) に対し，作為の▽４四同歩ではなく▽４四同飛と取る (図 5(c)) と，以下▲３三桂成▽２三馬▲同成桂▽３二銀 (図 5(d)) にて必至がかからなくなり，不完全作と考えられる[15),16)]．尚，8 手目に作為通り▽４四同歩と進めた局面には提案法にて必至がかかることを確認済みである．

また第 75 番について，問題局面を図 6(a) に示す．作為手順は▲５五馬▽同玉▲５三と▽３五とに▲５六桂以下の 11 手必至であるが，3 手目▲５三と (図 6(b)) に対し，作為の▽３五とではなく▽８一角と打って受ける手がある (図 6(c))．以下▲５六桂▽５四角▲４六金▽６五玉▲６六桂 (図 6(d)) までは殆ど必然であるが，ここで▽７六香と受けた局面 (図 6(e)) には必至がかからないと考えられる[16)]．少なくとも作意が最短応手とすると，図 6(c) には 7 手以内に必至がかかるはずであるが，それは無理である[16)]．▽８一角は作意から漏れてしまった受けと考えられる．尚，4 手目に作為通り▽３五とと進めた局面には提案法にて必至がかかることを確認済みである．

4.2 余必至探索

実装したプログラムを使って『来条必至』の余必至の探索を行った．

余必至とは詰将棋の余詰に相当し，要は別解のことである．作品として詰将棋を評価する際，余詰の存在は不完全作とされてしまう．必至のルールは詰将棋ほ

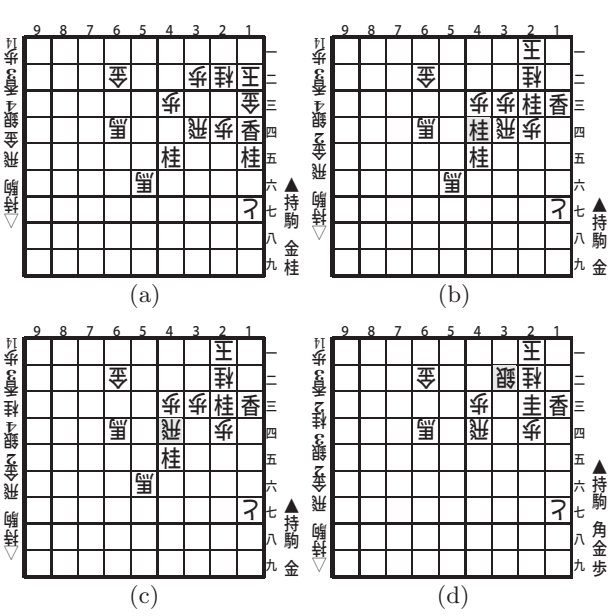

図 5　『来条必至』第 70 番．(a) は問題局面．作為は▲１一金▽同玉▲１三香不成▽２一玉▲２三桂不成▽３三歩▲４四桂▽**同歩**▲４三金以下の 23 手必至である．(b) は 7 手目▲４四桂までの局面．ここで作為は▽４四同歩だが，▽４四同飛と取った局面が (c)．(c) 以下▲３三桂成▽２三馬▲同成桂▽３二銀と進めた局面が (d)．この局面に必至がかかれば問題局面にも必至がかかる．

表 1　『来条必至』を解かせた結果

問題番号	作意手数	解答手数	時間 [秒]	局面変更回数
1	21	21	0.13	74748
2	5	7	0.05	26215
3	3	3	0.01	3264
4	5	5	0.02	13866
5	5	5	0.01	6977
6	9	9	0.04	24691
7	11	11	0.03	20706
8	9	9	0.06	35534
9	15	15	0.05	31450
10	15	17	0.20	99861
11	17	17	0.31	175792
12	13	13	0.08	45750
13	15	15	0.09	45640
14	19	29	16.33	7955824
15	15	15	0.28	154278
16	15	15	0.69	359878
17	17	17	0.66	310540
18	17	13	0.02	14654
19	21	21	0.11	57822
20	13	15	1.08	519249
21	19	27	0.50	248004
22	13	15	0.23	129867
23	19	19	0.16	91329
24	19	19	1.83	902761
25	21	21	0.35	197879
26	25	27	1.85	865733
27	19	19	0.17	98220
28	21	21	0.33	185416
29	19	21	1.31	695928
30	35	35	28.60	14013046
31	19	19	1.55	744034
32	25	25	1.09	572156
33	37	37	1.98	1046466
34	1	3	0.12	41342
35	1	1	0.03	17657
36	3	7	0.09	37118
37	3	3	0.03	11889
38	5	5	0.01	6267
39	5	5	0.14	84793
40	5	5	0.03	13543

問題番号	作意手数	解答手数	時間 [秒]	局面変更回数
41	5	5	0.24	127117
42	5	5	0.06	31299
43	9	9	0.10	51194
44	7	7	0.12	68869
45	29	29	7.34	3609126
46	9	13	0.92	435446
47	11	13	2.82	1465589
48	11	11	0.33	166175
49	9	9	0.36	179239
50	7	7	0.32	151095
51	7	7	0.06	35131
52	7	9	0.49	223605
53	9	9	0.05	26519
54	9	9	0.09	46817
55	9	9	0.09	42009
56	15	15	0.01	9964
57	9	13	0.50	264758
58	13	13	0.03	18810
59	9	11	1.38	651808
60	13	13	0.02	8311
61	9	19	0.20	101399
62	9	9	0.41	213234
63	9	17	0.74	341852
64	25	11	1.08	608019
65	37	43	2.29	1139130
66	27	31	3.79	1897154
67	27	27	0.98	449499
68	23	23	1.58	754697
69	33	15	0.16	82165
70	23	-	-	-
71	23	29	41.45	18411001
72	33	35	8.35	3688394
73	15	31	12.94	6616399
74	17	17	1.40	706142
75	11	-	-	-
76	9	11	0.35	191593
77	9	25	6.28	2863160
78	13	13	0.97	376682
79	13	29	9.43	4107534
80	11	11	0.15	76806
81	9	11	0.12	66794

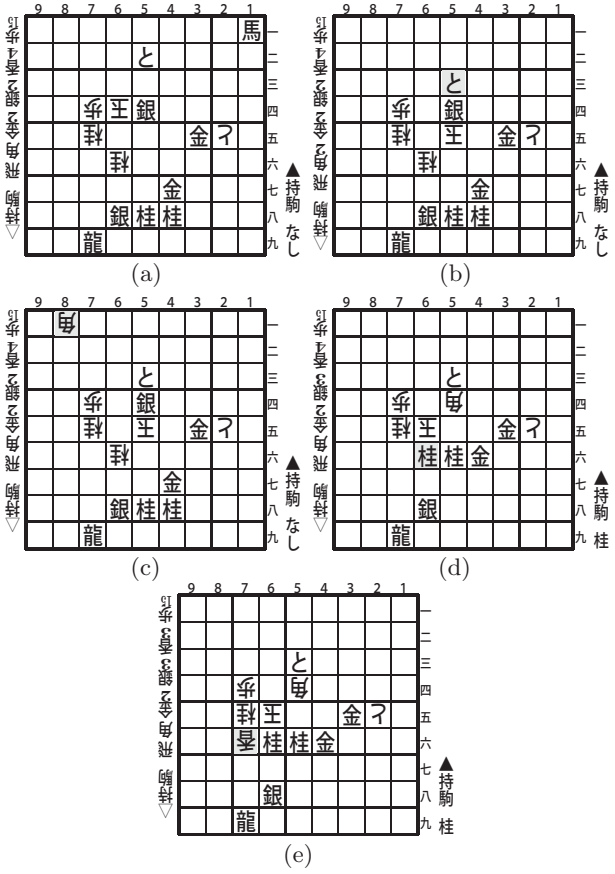

図 6　『来条必至』第 75 番．(a) は問題局面．作為は▲５五馬▽同玉▲５三と▽**３五と**▲５六桂以下の 11 手必至．(b) は 3 手目▲５三とまで．作為はここで▽３五とであるが，▽８一角と受けたのが (c)．(c) 以下▲５六桂▽５四角▲４六金▽６五玉▲６六桂までは殆ど必然．これが局面 (d)．(d) で▽７六香と打ったのが局面 (e)．この局面に必至がかからない限り，問題局面に必至をかけることはできないと思われる．

ど整備されていないが，余必至の存在はやはり「不完全作」とされる．ただしすべての余必至が不完全扱いされるのではなく，感覚的な部分がある．つまり軽微な余必至は「キズ」と表現されるが，非常に軽微だと「キズなし」扱いされる場合もある．

そこで計算機による余必至探索においても，キズなしとして扱われる次の 2 通りの余必至は対象外とした．それは作意が成る手のときの不成での余必至と，飛角香の打ち場所非限定 (つまり遠くから打っても近くから打っても必至) である．これらの余必至を除き『来条必至』の余必至探索の結果発見した 27 の余必至を表 2 に示す．「評価」の欄は篠田によるキズの程度の評価[16] である．「発見手数」は初期局面からの手数である．これは計算機にとって発見しやすかった手順の手数であり，これよりも短い必至手順が存在する可能性は十分にある．欄内の「早必至」とは余必至の一種で，作意手数よりも短手数の発見手数で必至がかかる余必至のことである．発見した余必至の例を図 7 に示す．

4.3　変化長手数

作意手順に余必至が存在してはならないルールがあ

表 2 発見した来条必至の余必至．(作意が成る手のときの不成と，飛角香の打ち場所非限定は対象外．)「評価」は篠田によるキズの程度の評価[16]．「発見手数」は初期局面からの手数．発見手数が作意手数より短い余必至を早必至と呼ぶ．

問題番号	作為手数	余必至の開始深さ	余必至	発見手数	評価
1	21	19	３二成銀	29	キズ
2	5	5	３四角成	7	キズ
5	5	3	３二金打	13	不完全
		5	３一金	13	小キズ
7	11	7	１四歩打	15	不完全
11	17	9	２一歩成	19	大キズ
12	13	7	６三金打	49	不完全
16	15	13	４四歩	27	キズなし
		15	３三歩打	21	小キズ
18	17	9	７二角成	13	不完全 (早必至)
20	13	13	６三香成	21	キズなし
22	13	13	４三金	21	キズなし
24	19	15	１四桂	21	キズなし
		17	３四桂	21	キズなし
28	21	5	２二角成	21	キズなし
		21	３三銀打	21	小キズ
30	35	5	１四歩	29	不完全
		29	１二歩成	43	キズなし
		31	２三金打	51	キズなし
		35	２三金打	59	キズなし
33	37	37	３二成桂	39	小キズ
45	29	29	３三金打	29	小キズ
64	25	5	１四桂打	11	不完全 (早必至)
69	33	7	２二香成	15	不完全 (早必至)
		29	３四馬	37	キズなし
73	15	15	２三角成	21	小キズ
74	17	5	７五金上	17	不完全

るのは前節で述べた通りである．必至問題のルールが詰将棋のルールに準じるとすると，他にも「受方は最長応手」のルールがある．これは受方は最長手数の応手で受けなければならないということである．つまり，作意手順中の受方の応手は最長手数の応手でなければならない．しかし必至問題では無駄合いの定義が曖昧なので，最長手数の応手を特定するのに困難が伴う．

そのような中,『来条必至』第 14 番は明確にこのルールに沿っていないので報告する．第 14 番の作為手順は下記の通り．

▲２三桂成　▽１二金　▲同成桂　▽同玉
▲３三香　▽同角　▲２三金　▽１一玉
▲３三金　▽同飛　▲２二角　▽１二玉
▲３三角成　▽同金　▲２二飛　▽１一玉
▲２三歩成　▽同金　▲同飛成

までの 19 手必至．

しかし我々に見落としがない限り，5 手目▲３三香に▽３三同角ではなく，▽３二金の方が必至手順が長くなる．最長応手の原則に則れば下記の手順が正解となる．(図 8 参照．)

▲２三桂成　▽１二金　▲同成桂　▽同玉
▲３三香　▽３二金　▲１一金　▽同玉
▲３二香成　▽２三金　▲同歩成　▽同飛
▲２二金　▽同角　▲同成香　▽同飛
▲３一角　▽３二金　▲２二桂成　▽同金
▲４二飛　▽３二桂　▲４三金　▽２五桂
▲３二飛成　▽１二金　▲１四桂

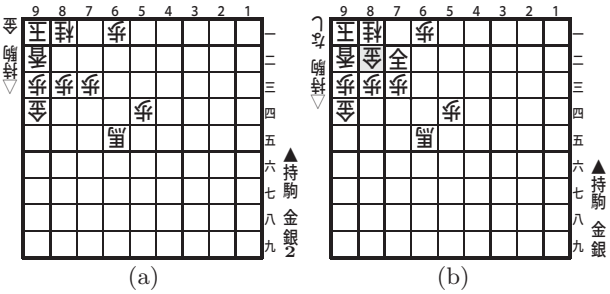

図 7 余必至の例．(a) は『来条必至』第 12 番の問題局面．作為は▲７二金▽８二金▲６三銀▽７二金▲同銀成▽８二金▲７一金▽７二金▲同金▽８二銀▲６三銀▽７一銀▲同金までの 13 手必至．(b) は 6 手目▽８二金まで．作意はここで▲７一金だが，▲６三金以下長手数の余必至を発見した．

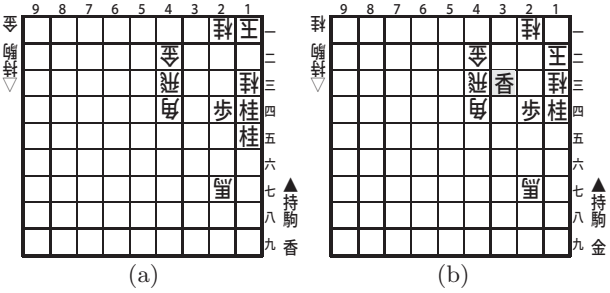

図 8 変化長手数の例．(a) は『来条必至』第 14 番の問題局面．作意通りに 5 手進めたのが (b)．すなわち▲２三桂成▽１二金▲同成桂▽同玉▲３三香の局面．作意は (b) のあと▽３三同角と受け，▲２三金以下の 19 手必至．しかし (b) では▽３二金の方が手順が長くなる．

までの 27 手必至．すなわち『来条必至』第 14 番も不完全作と考えられる．

5. 関 連 研 究

Threat-space search[2] はパスの手を考慮する意味で提案法に似る．仮に自分がパスしたら相手が指して脅威になる手を積極的に潰す探索法である．しかしこの論文は五目並べに特化した知識を多用する点に問題がある．

これを一般化したのが λ-search[10] で，やはりパスを考慮する意味で提案法と似る．パスの回数によって threat order(脅威度) が変化し，度合いの違いによる階層的な構造の中を探索する．提案法にはそもそも threat order の概念は明示的には存在しない．対応するのはパスの回数であるが，パスの回数が一定値に達したら王手のみを生成するためだけの存在である．また λ-search で λ^n-tree を生成するとき，λ^{n-1}-move でなく λ^n-move であることを保証するが，提案法はこの点が緩い．i を $i \leq n$ として，λ^i-move でありさえすればよいと考える．

df-pn と λ-search を組み合わせた探索法の研究[8),19)] もある．OR 節点で各 threat order に対応する多数の擬似節点を生成する点が提案法と大きく異なる．証明

数や反証数の計算も煩雑になる．提案法はこの点が単純である．AND 節点でも複数の擬似節点を生成するが，結局はパスとそれ以外の全合法手であるのでこの点は提案法と本質的に同じである．

提案法は詰方と受方という立場の違いを常に固定する．それに対し，Dual Lambda Search[9] は詰方と受方の立場が入れ替わる状況を想定する．将棋でいうと双玉問題を扱える．詰めろをかけた瞬間，詰めろ逃れの詰めろで逆転するというような問題も扱える．

df-pn$^+$ のしきい値計算の際，式 (2) のインクリメントの幅を 1 以外 (仮に α とする) にする研究[5] もある．これは df-pn$^+$ の **h** や **cost** の評価関数を α 倍したのと等価である．実装上は浮動小数を使わずに浮動小数 (有理数) の評価関数を導入するのと同じ効果がある．ただこの研究では **cost** は 0 にしている (と思われる) ので，α 倍しても **cost** は 0 のままである．

6. ま と め

必至問題を解くアルゴリズムを提案し実装した．提案法には，従来法にある詰み探索エンジンと必至探索エンジンの２つの探索エンジンという明確な区別はない．df-pn$^+$ の単純な応用として実装した．詰将棋を解くための工夫を取り入れ，高速に必至問題を解くことに成功した．難解な必至問題集として有名な『来条克由必至名作集』を解かせたところ，81 問中 79 問を解くことに成功した．また，余必至探索の結果３つの早必至を含む 27 の余必至を発見できた．

謝辞　本研究の一部は文部科学省科学研究費補助金若手研究 (B)「将棋の必至問題および最終盤を解くアルゴリズムの研究」の助成を得て行われた．また，本研究は篠田正人氏の協力なしにまとめることはできなかった．篠田氏は来条必至問題に精通しておられ，ご助言ご指摘はことごとく貴重なものばかりであった．ここに深謝の意を表する．本研究では他にも数多くの方々にご協力頂いた．黒田久泰氏，加藤徹氏，菊田裕司氏，山下宏氏には特に感謝の意を表する．

参 考 文 献

1) L.V. Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, Department of Computer Science, University of Limburg, Netherlands, 1994.
2) L.V. Allis, H.J. van den Herik, and M.P.H. Huntjens. Go-Moku Solved by New Search Techniques. *Computational Intelligence*, Vol. 12, pp. 7–23, 1996.
3) L.V. Allis, M.vander Meulen, and H.J. vanden Herik. Proof-Number Search. *Artif. Intel.*, Vol.66, pp. 91–124, 1994.
4) A. Kishimoto. Dealing with Infinite Loops, Underestimation, and Overestimation of Depth-First Proof-Number Search. In *Proc. of the 24th AAAI Conf. on Artif. Intel. (AAAI-10)*, pp. 108–113, 2010.
5) A. Kishimoto and M. Muller. Search versus Knowledge for Solving Life and Death Problems in Go. In *Proc. of the 20th AAAI Conf. on Artif. Intel. (AAAI-05)*, pp. 1374–1379, 2005.
6) A. Nagai. *Df-pn Algorithm for Searching AND/OR Trees and Its Applications*. PhD thesis, University of Tokyo, Japan, 2002.
7) A.Nagai and H.Imai. Proof for the Equivalence Between Some Best-First Algorithms and Depth-First Algorithms for AND/OR Trees. In *KOREA-JAPAN Joint Workshop on Algorithms and Computation*, pp. 163–170, 1999.
8) A.Reinefeld, J.Schaeffer, and T.A. Marsland. Lambda Depth-first Proof Number Search and its Application to Go. In *Proc. of the Int. Joint Conf. on Artif. Intel. (IJCAI-07)*, pp. 2404–2409, 2007.
9) S. Soeda, T. Kaneko, and T. Tanaka. Dual Lambda Search and shogi Endgames. In *Advances in Computer Games*, LNCS 4250, pp. 126–139, 2006.
10) T.Thomsen. Lambda-search in game trees - with application to Go. *ICGA J.*, Vol.23, No.4, pp. 203–217, 2000.
11) 有岡雅章. 必至探索. `http://www31.ocn.ne.jp/~kfend/inside_kfend/hissi.html`, 2000.
12) 柿木義一. 将棋プログラム K1.5 の思考アルゴリズム. 『コンピュータ将棋』4 章, 小谷善行ほか共著, サイエンス社, pp. 80–100, 1990.
13) 橋本剛, 作田誠, 飯田弘之. 必至問題を解くプログラムとその評価. 人工知能学会論文誌, Vol.16, pp. 539–547, 2001.
14) 金子タカシ. 詰みより必死. 毎日コミュニケーションズ, 1996.
15) 山下宏. 電子メールによる対話, 2011.
16) 篠田正人. 電子メールによる対話, 2011.
17) 脊尾昌宏. 詰将棋を解くアルゴリズムにおける優越関係の効率的な利用について. In *Game Programming Workshop in Japan '99*, pp. 129–136, 1999.
18) 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.
19) 副田俊介, 美添一樹, 岸本章宏, 金子知適, 田中哲朗, マーティン ミュラー. 証明数と反証数を用いたλ探索. 情報処理学会論文誌, Vol.48, No.11, pp. 3455–3462, 2007.
20) 来条克由. 来条克由必至名作集. 野口出版, 1984.