ゲーム情報学 5－8
（2001．3．16）

必死探索とその効率化

橋本 剛[1], 作田誠[1], 飯田弘之[2]

[1] 静岡大学理工学研究科 [2] 静岡大学情報学部

概要

我々が開発した必死探索プログラム TACO-H の手法を紹介する。TACO-H は手順をツリーとして記憶し再現させることで必死探索の合法手を効率良く見つける新しい手法 TDSS を使っている。TACO-H は解を保証しながら短時間で必死問題が解けることを示す。

また、玉の安全度という静的な評価値を閾値として、長手数の問題にもうまく対応できる安全度優先探索（SPH）を提案し、実験によってその効果を明らかにした。結果は、長手数の問題も含めて、必死問題をかなり高速に解くことに成功し、難問の内藤 17 手必死を、深さを閾値とした反復深化で 11.4 分、SPH では 3.75 分で解いたほか、SPH では 15 手必死をわずか 0.2 秒で解くなど、SPH が長手数問題を解くための非常に有効な手段であることを示した。

Brinkmate search and its efficiency

Tsuyoshi Hashimoto[1], Makoto Sakuta[2], Hiroyuki Iida[1]

[1]Shizuoka University Science and Engineering Faculty
[2]Shizuoka University Information Faculty

Abstract

We introduce some techniques of our shogi brinkmate solver program TACO-H, which guarantees the solution. Especially, we mainly introduce a new technique TDSS, which can efficiently distinguish legal moves of brinkmate search by representing tree of moves. We show our program can solve brinkmate problems in a short time by experimental results.

Furthermore, we propose a new method named safety priority hisshi search, SPH. SPH can solve problems with long steps by a static value of king safety as threshold, We succeed in solving brinkmate problems including long step problems at high speed. TACO-H solved the hard Naito's brinkmate problem with 17 steps in 11.4 minutes by depth-first iterative deepening and in 3.75 minutes by SPH, and solved a problem with 15 steps only in 0.2 seconds by SPH.

1 はじめに

必死は将棋の終盤において詰みと同様に重要な概念で、終盤でより正確な判断を求められるようになればなるほど、必死探索の必要性が増すものと思われる。だが、詰み探索でさまざまな研究が行われ目覚しい成果が挙げられているのとは対照的に、これまで必死探索に関する報告は少ない。

内藤 [1] は「あれだけ詰めの能力にすぐれているのに必死が全然ダメというのは、コン君の頭の中はいったいどうなっているのかと不思議な気分になる。」と必死を解くプログラムがないことを嘆いている。

必死は飯田 [2] の定義にあるように、詰めろあるいは王手とその応手の連続で、探索は詰め将棋探索と同様の AND/OR 木になる。だが、詰めろの判定には詰みルーチンを呼ばねばならず、合法手の生成に大きなコストがかかるのが大きな特徴である。必死問題を解くためには、この問題をうまく解決していかねばならない。

2 必死探索に関する研究

ごく初期の 1987 年に吉川 [3]、1996 年に飯田、安部 [2] がそれぞれ必死探索の結果を報告しており、飯田、安部は短手数（1 ～ 7 手）の問題を解くことに成功している。1998 年に山下 [4] は必死探索を実戦に応用することの難しさを述べている。

最近では、2000 年に橋本 [5]（筆者）が内藤 [1] の「今月はコンピュータで解けない上級必死問題です」との注釈付で出していた 17 手必死問題を 2.56 時間かけて解いたことを報告している。

また、有岡 [6] はホームページ上で必死探索に関する数々の手法や詳しい研究成果を報告している。有岡は必死探索の方法として、深さを閾値とする縦型の反復深化をはじめ、独自の方法で、証明数を閾値とした多重反復深化のアイデアも紹介し、金子 [7] の必死問題集を非常に高速に解き、優れた結果を残している。有岡の探索は、攻め方、受け方とも玉の周りに利きを付ける手や駒を取る手など、有効そうな手だけを生成し、それらの手を詰みルーチンにかけて合法手かどうか調べる、という非常に実戦的な方針を取っているのが大きな特徴である。

3 必死探索プログラム TACO-H

我々の開発した必死探索プログラム TACO-H は将棋プログラム TACOS を元に開発した。縦型探索で、基本的には深さを閾値として反復深化を行っている。以下 TACO-H の主な特徴を述べていく。

3.1 合法手の生成

必死探索において、合法手の生成は詰みルーチンを呼ぶ重い作業なので、反復深化にあたっては攻め方で合法手のリストをハッシュに登録するなど合法手の生成のコストを下げるよう気を配っている。

合法手の生成に関して、上述した有岡の方法は実戦的には非常に有効な手法だと思われるが、この方法だと受け方ですべての手を確認したことにならないため、必死であるという解が出ても解は保証されない。もっとも、詰めろ判定用詰みルーチンの探索最大深さを無限大にしないと厳密な意味で解は保証されないが、与えられた詰みルーチンの探索最大深さにおける必死解の保証ということにこだわり、TACO-H の基本方針は、攻め方は有岡と同様の手法で見込みのありそうな手だけを生成するようにし、受け方はあくまですべての指し手を合法手かどうかチェックするようにした。

だが、まともにすべての手で詰みかどうかを確認していると非常にコストがかかってしまう。そこで、TACO-H では下に述べる TDSS という方法を用いて、受け方で詰みルーチンを呼ばないで合法手かどうか判定してコストを下げている。

3.2 指し手の順序付け

探索の効率化には指し手の順序付けが重要であるが、TACO-H では詰みルーチンを呼ぶ前に玉の安全度と、指した手に関する情報（取り駒の価値、玉との距離など）を元にすべての手を順序付けしてから合法手

かどうかチェックしていく。玉の安全度とは、ここでは玉の移動できるマスの数を意味し、TACO-H ではこれに盤の端近くでの補正と、玉の周りのマスの利きの制圧度による補正を若干加味して計算される。

3.3 その他

有岡の示した「取られるだけの無駄な手数伸ばしへの対応」と「一手の先読み」の手法が TACO-H においても有効であったので、これを採用している。

4 TDSS (Threatmate defense search using similarity)

詰めろのかかっている局面で、玉方が何か手を指してもその手が詰めろに影響を与えていないと、本来の詰めろの手順と同じ手順で詰んでしまう。その場合、棚瀬 [8] が提案しているように本来の詰めろの手順で詰むかどうかだけを確認すれば、まともに詰みルーチンを呼ぶ場合に比べてかなり効率よく詰めろを防いだかどうかを確認できる。だが、棚瀬の方法では最善詰み手順を再現するだけなので、最善詰み手順には影響を与えていなくても、最善手順以外の個所で詰みを防いでいる手を見落としてしまうという欠点があるため、実戦ではともかく、問題を解く場合など解の保証が必要な場合には使えない。

そこで、TACO-H では一つの詰み手順を完全に再現できるかどうかを調べるという方法 TDSS (Threatmate-defense search using similarity) を使っている。ここで注意してほしいのは、詰み手順が複数あるときは全ての手順を同時に防ぐ手を指さないといけない、すなわち少なくとも一つの詰み手順を防いでいない手は、明らかに詰めろを防ぐことを失敗している手である、ということである。

TDSS の基本概念は以下のようになる

- 詰みがある局面を P、受け方が任意の可能手を指した局面を P ' とする
- P の詰み手順 T を記憶する
- ここで、T は攻め方の手は最善手一手だけ、受け方の手はすべて記憶する
- P ' 上で P の詰み手順を攻め方の手は一手だけ、受け方の手はすべて再現する
- 完全に再現できたら 、S は明らかに詰みを防いでいない手である

つまり、記憶する手順はツリー構造で、ツリーを完全に再現できるかどうかを調べることで、再現できた場合には少なくとも一つの詰み手順が存在することが保証される。

完全に再現出来ない場合、すなわち指し手が詰みに何らかの影響を与えている場合は、詰みルーチンを呼んで詰むかどうかを確認する。その場合 TDSS を呼んだことは無駄になってしまうが、実際には詰めろに影響を与えていない指し手の方が圧倒的に多いため、全体としてコストが削減される。

なお、TACO-H では TDSS をより有効に使うため、まず玉の安全度を計算し、詰みに影響を与えている可能性が高いと思われる手には TDSS を呼ばず直接詰みルーチンを呼ぶようにしている。また、TDSS を攻め方のノードにも応用して、受け方の手は最善手一手だけ、攻め方の手はすべて記憶し、詰めろにならないことを示す手順をツリーにし、明らかに詰めろにならない手を高速に判別するということも行っている。

今後さらに研究が進み、実戦形式の局面で 1 手勝ち、2 手勝ちなどを判定する場合には、TDSS の手法は無駄な王手などの判定で特に有効に働くと思われる。また、この TDSS の概念は囲碁など探索空間の大きいゲームなどにも有効ではないかと考えている。

5 実験:深さを閾値とした縦型反復深化

TACO-H に、有岡 [6] と同じく金子の必死問題集 [7] のうち 7 手必死以上の問題 NO.69～100 と上述した内藤 [1] の 17 手問題を解かせてみた。マシンは CPU が PentiumIII800Mhz である。実験用のパラメータとして重要なのは合法手の判別で呼ぶ詰みルーチンの最大探索深さだが、ここではすべての問題を解いた中で最小の深さ 5 を採用している。なお、有岡のように攻め方と玉方でこのパラメータを違う値にするということはしていない。必死探索の場合、探索ノード数の定義が難しいので、局面を更新する関数を呼び出した回数を調べた。

5.1 結果

表 1 は左から問題番号、問題に明記してある必死手数、TACO-H が実際に解いた手数、探索時間（秒）、局面更新数（単位は 10 万）である。

中には一時間以上かかってしまった問題もあるが、すべて解くことが出来た。特筆すべきは、筆者の 2000 年の報告 [5] で 2.56 時間、局面更新数 2.76 億回かかっていた内藤の 17 手問題が、12 分弱、局面更新数 95 万回で解けるようになったことである。時間短縮については CPU のスピードが上がったことも一つの理由であるが、攻め方の合法手候補をすべて読んでいたのを見込みのありそうな手だけに絞ったことや、指し手のソートに使う評価関数の改良などが大きく影響している。

なお、実際に解いた手数が有岡の結果と違う問題がいくつかあるが、これは合法手判別用の詰みルーチンの最大深さを 5 にしているため、実際よりも手数を長く数えてしまうためである。

6 安全度優先探索（safety priority hisshi search, SPH）

手数の短い必死問題は反復深化でかなり高速に解く事に成功した。次に内藤の１７手必死のように長手数の問題を解くために、PN* (あるいは C*)[9] のような手法が必死にも使えないだろうかという興味が起こってくる。詰め将棋では、玉方の応手の数を証明数に使いすばらしい成果があがっている。必死探索も同じ AND/OR 木の探索なので、証明数に玉方の応手の数を使えば詰め将棋と同じ手法が使える。だが、必死探索では着手の生成のために毎回詰みルーチンを呼ばねばならず、玉方の応手の数を証明数とする方法ではかなりのコストがかかってしまう。そのため、詰め将棋で開発された手法をそのまま使うことは出来ず、何か工夫が必要になる。

有岡 [6] は、受け方の応手数をきちんと数える代わりに、擬似的な証明数を使うという方法を取っている。受け方が必死にならないとき、その深さでの証明数への寄与を「まだ展開していない手の数」として評価値に反映させ、その証明数を閾値として脊尾の PN*を適応するという方法である。この手法は有岡本人が指摘しているように、「まだ展開していない手の数」に自玉が詰まされる手が含まれてしまうため、証明数が不正確になる恐れがある。

ここでは、必死探索に玉の安全度という局面の静的な評価を閾値として用いる新たな手法、安全度優先必死探索（safety priority hisshi search, SPH）を提案する。玉の安全度とは、ここでは玉の移動できるマスの数を意味し、TACO-H ではこれに盤の端近くでの補正と、玉の周りのマスの利きの制圧度による補正を若干加味して計算される。一般に玉の安全度が低い方が詰みやすいので、玉の安全度は受け方の応手の数と似た性質を持つが、静的に評価できるため、受け方の応手の数に比べて計算コストが圧倒的に少なくてすむという利点がある。

必死は玉を詰めろと王手で攻めて行くが、玉の安全度が高い局面はそもそも詰めろにならないことが多く、必死をかけるためには受け方が手を指した後も玉の安全度が低い局面が続くのが一般的であると思われ

表 1: 深さを閾値とした縦型反復深化実行結果

No.	問題手数	実際手数	時間 (秒)	局面更新数
69	7	7	5.20	8.0
70	7	7	3.53	5.5
71	7	7	20.87	36.3
72	7	7	2.69	4.1
73	7	7	1.56	2.2
74	7	7	635.91	1047.9
75	7	7	6.39	10.0
76	7	7	107.58	151.9
77	7	9	274.72	397.6
78	7	9	10.64	18.4
79	7	9	1620.52	2240.5
80	7	13	352.71	475.9
81	7	7	0.60	0.9
82	7	7	17.14	23.6
83	9	9	1.26	1.6
84	9	9	10.60	15.4
85	9	9	30.48	56.4
86	9	11	33.22	50.4
87	9	9	19.05	33.0
88	9	9	73.49	116.3
89	9	9	139.85	176.6
90	9	11	29.27	48.2
91	9	11	42.75	59.9
92	9	7	2.52	3.0
93	9	13	4205.22	6488.5
94	11	11	200.92	273.5
95	11	13	213.66	295.4
96	11	13	681.17	920.2
97	11	11	192.26	248.1
98	11	11	115.66	152.6
99	15	15	34.95	49.6
100	15	17	862.41	1197.4
内藤問題	17	17	685.90	954.6

*局面更新数の単位は 10 万

る。そこで、受け方が手を指した後の局面で玉の安全度がある閾値以上であった場合は探索を後回しにし、玉の安全度が低くなる手だけを探索していけば長手数必死でも効率よく探索が行える。だがここで注意しなければいけないのは、玉の安全度が高くなる手が詰めろを防いでいるとは限らない、すなわち合法手であるとは限らないということである。この点に注意をして、SPH では次の手順で探索を行う。

- まず適当に安全度の閾値を決める
- 攻め方のノードは反復深化の場合と同じ

  受け方のノードで以下のように探索を行う
- 受け方の応手をすべて生成し、玉の安全度が高くなる順にソートする
- ソートした上位の手から詰み探索をし、詰み探索をして合法手かどうか調べる
- 合法手が見つかったら、

  1．玉の安全度が閾値を超えている場合

  その手をさらに深く探索していき、値が帰ってきたらそのノードの他の手も調べていく。

  2．玉の安全度が閾値以下の場合

  そのノードの探索を打ち切る。
- 必死が見つからなかった場合、閾値をあげて再探索する

つまり SPH は玉の安全度を閾値として反復深化を行うわけだが、受け方ノードでは閾値にかかわらず合法手がみつかるまで応手を調べるので、安全度が閾値を超えていながら合法手でないという手が存在しても、必死かどうかきちんと見極めることができるのである。例えば閾値が 2 の場合だと、受け方が合法手を見つけ、その手を指した後の局面が玉の安全度 2 以下の場合、すなわち玉の動けるマスが 2 以下の場合はどんどん深く探索していき、安全度が 2 以上の場合は探索を後回しにするのである。

6.1 実験

SPH を実装した TACO-H で、深さを閾値とした反復深化と同じ方法で実験を行った。なお、第 3 章で述べた「取られるだけの無駄な手数伸ばしへの対応」と「一手の先読み」の手法はここでは外している。玉の安全度の閾値は、初期値 2 から反復深化させている。また、探索深さに上限を設け、問題番号 69 から 98 までは 11、それ以外は 17 としたが、11 で解けなかった問題は 17 にして解き直してみた。

6.2 結果

表の右から 2 番目は探索深さの上限で、一番右は表 1 の結果と比べて速度が何倍速くなったかを示す。問題 93、96 はハッシュ表がいっぱいになってしまい解けなかった。かえって遅くなったものもいくつかあるが、探索深さの上限を実際に解ける深さよりも深くしているためであると思われる。早く解けた問題も多く、問題によっては極端に早く解けたものも多く見られる。99 番は 15 手必死であるが、比較的わかりやすい一直線の手順であるため、0.2 秒という驚くべき速さで解いている。また、内藤問題では 225 秒 (3.75 分) という好結果が得られた。特に長手数の問題では早く解けており、長手数の必死問題を解くための有効な手段と成り得ることが示せたと思う。探索深さの上限をうまく設定すれば、短手数の問題でもかなり有効ではないかと思われる。

表 2: SPH 実行結果

No.	問題手数	安全度閾値	時間 (秒)	局面更新数	最大探索深さ	速度比較
69	7	2	0.27	47.6	11	19.2
70	7	4	24.28	3896.0	11	0.1
71	7	2	13.88	2100.6	11	1.5
72	7	4	27.04	3889.9	11	0.1
73	7	4	6.78	1017.4	11	0.2
74	7	3	518.58	81920.1	11	1.2
75	7	3	5.14	820.7	11	1.2
76	7	3	49.69	7121.0	11	2.2
77	7	2	26.66	3927.1	11	10.3
78	7	3	6.61	1113.6	11	1.6
79	7	3	264.60	35472.2	11	6.1
80	7	3	394.94	53945.4	17	0.9
81	7	3	2.74	384.5	11	0.2
82	7	2	22.74	3133.8	11	0.8
83	9	2	1.61	225.8	11	0.8
84	9	2	0.13	14.0	11	81.5
85	9	3	46.58	7749.4	11	0.7
86	9	2	12.38	1711.2	11	2.7
87	9	4	194.14	30056.2	11	8.4
88	9	3	57.54	8685.5	11	1.3
89	9	4	336.30	43397.0	11	0.4
90	9	2	1.90	333.1	11	15.4
91	9	2	34.60	4880.1	11	1.2
92	9	3	14.90	1697.6	11	0.2
93	9	x	x	x	x	x
94	11	3	0.83	115.1	11	241.5
95	11	2	433.93	56622.1	17	0.5
96	11	x	x	x	x	x
97	11	3	54.82	7073.9	11	3.5
98	11	2	7.11	880.5	11	16.3
99	15	2	0.18	21.1	17	194.2
100	15	3	542.41	73303.5	17	1.6
内藤問題	17	2	225.26	29083.4	17	3.0

7 まとめ

必死探索プログラム TACO-H の手法を TDSS を中心に紹介したが、実験によって短時間で必死問題が解けるようになったことを示した。解を保証を求めながら行う我々の方法でも、これまで難しいとされてきた必死探索を十分に実用的な時間で解くことに成功したと言えるだろう.

また、玉の安全度という静的な評価値を閾値として、長手数の問題にもうまく対応できる安全度優先探索（SPH）を提案し、実験によってその効果を明らかにした。長手数の問題を含む、必死問題を効率良く解くための有効な手段になると思われる。

主な結果としては、難問の内藤 17 手必死を、深さを閾値とした反復深化で 11.4 分、S PH では 3.75 分で解いたほか、SPH では 15 手必死をわずか 0.2 秒で解くなど、SPH が長手数問題を解くための非常に有効な手段であることを示した。

8 今後の方針

TACO-H を更に改良して、短手数問題も長手数問題もより高速に解けるようにしたい。目標として、最大 37 手の必死問題を含む来条克由必死問題集に挑戦するほか、実戦形式の寄せ合い問題（終盤の次の一手）を解くルーチンへと応用していく。

また、実戦の将棋プログラム TACOS へうまく実装し、強い終盤ルーチンを作っていきたい。

参考文献

[1] 内藤, 新妙手探し（26）, 近代将棋 8 月号, p26–27,1999.

[2] H.Iida and F.Abe, Brinkmate Search, Proceedings of Game Programming Workshop in Japan '96, p160–169, Hakone, Japan, Oct. 1996.

[3] 吉川, 将棋プログラムの研究. Programmers ' Σ, 第 3 巻, p102–115, 技術評論社,1987.

[4] 山下.6 章 YSS–そのデータ構造, およびアルゴリズムについて, コンピュータ将棋の進歩 2, 共立出版,1998.

[5] 橋本, 必死探索について, コンピュータ将棋協会誌 Vol.13,p44,2000.

[6] 有岡, 必死探索, 将棋プログラム KFEnd,2000. http://plaza9.mbn.or.jp/~kfend/inside_kfend/hissi.html.

[7] 金子タカシ, 詰みより必死, 毎日コミュニケーションズ,1996

[8] 棚瀬, IS 将棋のアルゴリズム. コンピュータ将棋の進歩 3, p1–13, 共立出版,2000.

[9] M. Seo, H. Iida, and J. W. H. M. Uiterwijk. The PN*-search algorithm: Application to Tsume-Shogi. *Artificial Intelligence.* to appear.