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

脊尾 昌宏

松下電器産業株式会社

概要

探索問題において、優越関係を使って効率的に解を求める方法は一般的に使われている。本論文では詰将棋を解くプログラムにおいて、優越関係をより積極的に利用するために、探索中に「詰めに必要な持駒」という付加情報を随時計算して、より有益な情報をハッシュ表に登録する手法を提案し検証する。

On Effective Utilization of Dominance Relations in Tsume-Shogi Solving Algorithms.

Masahiro Seo

Matsushita Electric Industrial Co., Ltd.

Abstract

Dominance relations is generally used to solve search problems effectively. In this paper, we suggest and investigate the method to utilize dominance relations aggressively by calculating the additional information, that is "captured pieces to mate", and saving more valuable results in the hash table in Tsune-Shogi solver.

1. はじめに

詰将棋を解くプログラムにおいて、証明数を用いた解法が主流になってきている。証明数による探索といえば従来は最良優先探索であったが、少ないメモリ消費で最良優先探索と同じ探索を行なう多重反復深化法が脊尾により開発され、現存する最長手数（1525 手詰）の詰将棋「ミクロコスモス」を解くことに成功した。反復深化法などの縦型探索では、探索した結果をハッシュ表に登録しておき、後で別の手順を経て同じ局面（または類似した局面）に到達したときに、ハッシュ表の登録情報を検索して、可能ならば探索の省略を行なう。ハッシュ表に登録する情報が強いほど、探索が省略できる可能性が高くなる。

本論文では、詰将棋を解くプログラムにおいて、通常の探索に並行して「詰めに必要な持駒」という付加情報を計算しておき、将来の有効活用を見越してハッシュ表の１項目として登録する手法を提案し、検証する。

2. 優越関係とデータ構造

探索問題において、優越関係(dominance relation)を使って効率的に解を求める手法は一般的に使われている[2]。部分問題 A が部分問題 B を優越する（以後 DOM(A,B)と記述する）とは、B は A より良い解を持ち得ないことである。詰将棋の場合は詰みと不詰の２値なので、２つの局面 A,B の間に DOM(A,B)が成立するとき、B が詰みで A が不詰というケースはあり得ないこ

とを意味する。この場合、もしBが解ければ、Aは少なくともBを解くのと全く同じ手順で解けることが保証されるので、Aの探索を省略することができる。

詰将棋で局面Aと局面Bの間に優越関係が適用されるのは、盤面が同一で持駒だけが異なっており、なおかつBの持駒がAの持駒の部分集合になっている場合である。打歩詰禁じ手のルールから、それ以外の場合は優越関係が成立しない。例えば持駒が香なら詰むからといって、同じ盤面で持駒が飛でも詰むとは限らないし、ある詰む局面で角が馬に変わった局面でも詰むとは限らない。

先手の持駒情報を、一例としてFigure1のように1つの32ビットの変数で表現する。

31 ～ 25	24 ～ 20	19 18	17 16	15 ～ 12	11 ～ 8	7 ～ 4	3 ～ 0
0000000	PPPPP	RR	BB	GGGG	SSSS	NNNN	LLLL

P：歩、R：飛、B：角、G：金、S：銀、N：桂、L：香

Figure1: 持駒のビット表現

飛角金銀桂香は持っていればビットをONにして、持っていなければビットをOFFにする。ビットは下のビットから順番にONにする。歩だけは枚数が多くビット数が足りないので、5ビットをそのまま使って枚数を表現する。例えば、先手の持駒が「角1枚、銀3枚、桂2枚、歩5枚」の場合は、持駒ビットの値は"0x00510730"になる。

優越関係を効率的に活用するために、ハッシュ表に登録する局面情報を、盤面と持駒とに分離して登録しておくとよい。優越関係を満たすか否かは、Figure2のように持駒情報のビット演算により簡単に計算できる。

```
/* n1 が n2 を優越するか判定 */
int dominate (unsigned long n1, unsigned long n2)
{
    if (n1 < n2)
        return false;
    if (((n1 & n2) & 0xFFFFF) == (n2 & 0xFFFFF))
        return true;
    return false;
}
```

Figure2: 優越関係の判定法

3. 証明数探索における優越関係とハッシュ表

本論文では探索手法として、証明数(proof number)による多重反復深化法(multiple iterative deepening)を仮定する。証明数とは1980年代後半にMcAllesterにより提案されたミニマックス木における「共謀数」[6]という概念を、AllisがAND/OR木探索に応用したものである[1]。証明数はAND/OR木の探索において、ルート節点を証明する（解く）ために証明しなければならない先端節点数の最小値として定義され、証明数の値が0になれば、その節点が証明された（解けた）ことを意味する。証明数はより広い意味で共謀数であると解釈することができる。証明数による最良優先探索は本来メモリを多量に必要としたが、脊尾が開発した多重反復深化法[7][8]により、少ないメモリで最良優先探索と同等の探索が実行可能になった。

局面Aの証明数をPN(A)で表わす。証明数が大きいほど問題が解きにくいことを表すので、2つの局面A,Bの間にDOM(A,B)が成立するとき、PN(B)の値はPN(A)以上であるとしてよい。ハッシュ表の検索アルゴリズムとして、持駒を除いた盤面の情報をコード化して検索キーとして表引きし、優越関係を満たす局面をその近傍から検索する。局面Nの証明数をハッシュ表から検索するとき、Nと盤面が同一のX（N自身も含む）の中で、

DOM(N,X) かつ PN(X) = 0
を満たす局面 X がハッシュ表に登録されていれば、
PN(N) = 0
が成立する。また、そのような局面 X が存在しないとき、DOM(X,N)を満たす局面 X（N 自身も含む）の中での証明数の最大値を MAX_PN(X)としたとき、
PN(N) = MAX_PN(X)
と評価することができる。

盤面情報をキーとして、登録する番地へジャンプ。 その番地が空であるか、または盤面と持駒が同じであれば、そこに保存して終了。 そうでなければ、その隣の番地を調べて、空であるか、または盤面と持駒が同じであれば、そこに保存して終了。保存する番地が見つかるまで繰り返す。 もし一定個数の番地を調べても見つからなければ、候補の中から情報量の一番小さなものを削除して、代わりに保存する。

Figure3: ハッシュ表への登録

```
int findtable (unsigned long n)
{
    int   maxpn = 1;
    /* 局面 n と盤面情報が同じ局面 x すべてに対して */
    for (all nodes x having the same board with n)
    {
        if (dominate (n, x) == true && pn(x) == 0)
            return 0;
        if (dominate (x, n) == true && pn(x) > maxpn)
            maxpn = pn(x);
    }
    return maxpn;
}
```

盤面情報をキーとして、検索する番地へジャンプ。
その番地および近傍から、同一局面または優越関係が成立する局面の情報を検索する。

Figure4: 優越関係を使ったハッシュ表の検索

4. 優越関係による無駄合探索の効率化

優越関係が有効になる例として、無駄合探索の効率化がある。無駄合の判定法としては、柿木の無駄合判定アルゴリズム[5]が知られている。実際に無駄合探索を効率的に処理するための探索手法として、無駄合探索のための特別なルーチンを使うのが一般的であった。例えば筆者のプログラムでは、後手の中合の応手を先手がその直後に取った場合には、取った駒を後手に返して探索し（無駄合探索ルーチン）、もしそれで詰まなければ通常の探索を行なう。Figure5 に後手の応手に対する探索法をフローチャートで示す。もし無駄合探索ルーチン実行中に、ある種類の合駒が詰めば、ハッシュ表と優越関係および後手の応手の順序付け(priority)[4]により、他の合駒の探索を大幅に簡略化することが可能になる。例えば、歩の中合に対して無駄合探索で詰んだ場合、取った歩を後手が持っていても詰むという、より強い情報がハッシュ表に残るので、歩以外のどの合駒に対しても、すぐに取る手を探索すれば、優越関係からすぐに詰みと判定できる。つまり合駒の種類に依存しない結果がハッシュ表に残るので、7種類の合駒それぞれに対する探索を行なう必要がなくなる。その反面、もし中合が有効合であれば、無駄合探索ルーチンでは詰まないので、探索が2度手間になってしまい、最悪の場合約2倍の局面を探索しなければならない。

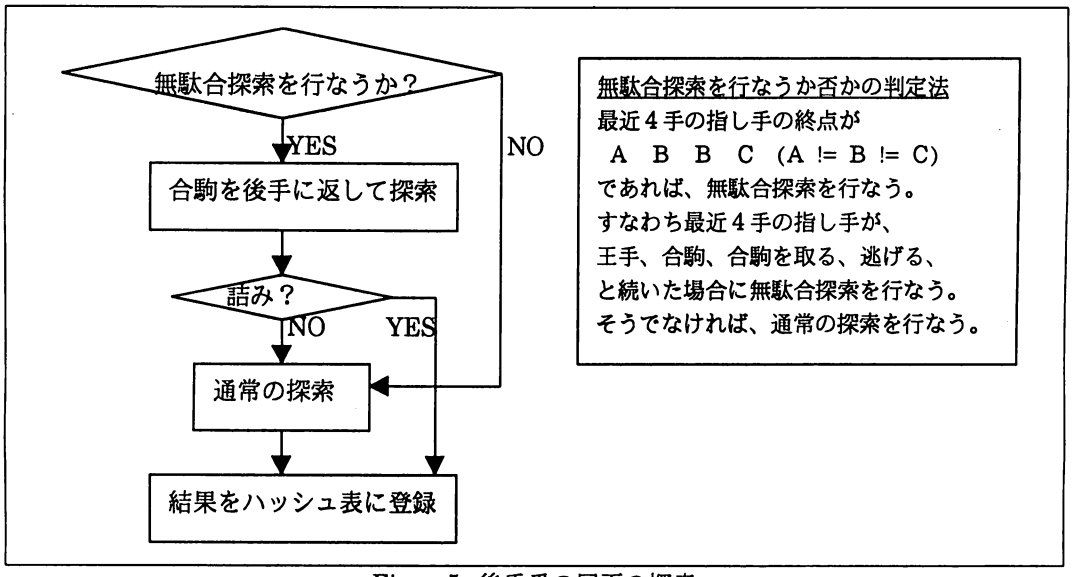

Figure5: 後手番の局面の探索

5. 付加情報による優越関係の拡張利用

本論文では、探索中の各局面において、もしそれが詰みの局面ならば、先手の持駒の中で実際に詰めに必要な持駒の集合（証明駒（proof pieces）と呼ぶ）という付加情報を計算して、ハッシュ表に登録することにより、優越関係をより有効に利用する手法について述べる。証明駒はFigure6に示すとおり、探索中に詰めに使用した持駒から再帰的に計算される。

(1) 局面Nが先端局面のとき、
詰み（詰め上がり）の局面ならば、
PP(N) = 0
それ以外のとき、
PP(N) = ∞

(2) 先手番の局面のとき、局面Nから指手MVにより詰みの子局面NCに至るとする。
(i)MVが先手の持駒BPを打つ手ならば、
PP(N) = PP(NC) + BP
(ii)MVが後手の駒WPを取る手ならば、
PP(N) = PP(NC) – WP
(iii)それ以外の手ならば、
PP(N) = PP(NC)

(3) 後手番の局面ならば、子局面NCのPPの和集合
PP(N) = Σ(PP(NC))

ただし(1)および(3)ではPP(N)に、
Σ(後手が一枚も持っていない種類の先手の持駒)
をさらに加算する。これは実際には王手に使わなくとも先手が持駒として所持しておくことで、後手に合駒として使わせない駒を考慮するためである。

Figure6: 局面Nの証明駒PP(N)の計算法

通常の探索後に結果をハッシュ表に登録するのに加えて、局面 N が詰みであるとき、もし証明駒 PP(N)が現在の先手の持駒と異なっていれば、つまりもっと少ない駒数でも詰むのであれば、盤面が現在の盤面で持駒が PP(N)で詰みという情報をハッシュ表に登録する。Figure7 に証明駒により拡張された優越関係による後手番での探索アルゴリズムを示す。

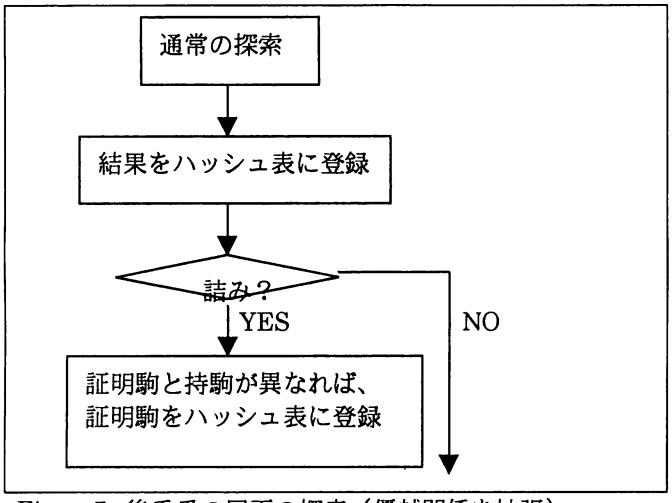

Figure7: 後手番の局面の探索（優越関係を拡張）

証明駒を利用すれば、優越関係を用いたハッシュ表の検索がより強力に行なえる。似た局面の探索の結果をそのまま利用できる機会が増えるからであり、より多くの局面の探索を省略することができる。またもう 1 つの利点として、前述の無駄合探索ルーチンなどの特別なルーチンを用意する必要がなくなる。なぜなら無駄合探索を行ったときにできる無駄合を取った駒をすぐ後手に返しても詰むというより強い情報を、通常の探索の中で自然に生成することができるからである。Figure5 のように探索が二度手間になる可能性がなく、より効率的でエレガントであるといえる。

6. 実験結果

「証明駒」という付加情報と優越関係による探索で、無駄合探索と優越関係による従来の探索より良い性能が得られることが期待できる。付加情報を利用した場合（5 節の手法）と利用しない場合（4 節の手法）について、実際に江戸時代の難解な作品集「将棋無双」[3]を解かせてそれぞれの結果を比較した。全 100 題中、不詰とされている 6 題を除いた 94 題に対して実験を行なった。結果を Figure8, 9 に掲載する。×印は制限時間（2 時間）以内に解けなかったことを表わしている。

実験に使用したマシン環境は CPU が Pentium200MHz, メモリ 256MB である。使用したメモリのほとんど（192MB）はハッシュ表が使っている。ハッシュ表では 1 局面あたり 16 バイトを使用しており、登録可能な局面数は最大 6291456 局面である。

96 題中 78 題の問題において、付加情報ありの方が生成局面数は少なくなっている。解答時間もほぼ同様の傾向を示しており、付加情報の計算に伴うオーバーヘッドはほとんど見られないと判断してよい。また付加情報ありの場合、すべての問題をそれぞれ 2 時間以内に解くことができた。

7. おわりに

本論文では優越関係をより積極的に活用するために、「証明駒」という付加情報を探索中に計算しておく手法を提案し、実際に問題集を解かせて有効性を確認した。

なお、現在は通常の登録データと証明駒を計算したデータの両方をハッシュに登録するアルゴリズムにしている。したがって、通常の登録データに加えて証明駒を計算したデータを登録するので、ハッシュ表を余分に消費する欠点がある。これを解消するため、ハッシュ表があふれてガベージコレクションを行なうときに、不要な登録データを優先的に削除するなどの手段をとるのがよいと考えられる。

付加情報を使って「将棋無双」を実用的な時間内で全題解くことができた。もっと長手数の問題でまだ解けていない問題がいくつかあるので、新手法のさらなる開発により、それらを含むすべての問題を実用的な時間内に解けるプログラムを開発することを目標と考えている。

私の詰将棋解図ソフト「脊尾詰」を使っていただき、貴重なご意見をいただいた励棋の会員の方々に感謝いたします。

参考文献

[1] Allis, L.V., Meulen, M., and Herik, H.J.: "Proof-number search," Artificial Intelligence, Vol.66, pp.91-124, 1994.

[2] Ibaraki, T., "The Power of Dominance Relations in Branch-and-Bound Algorithms," J. ACM, Vol.24, No.2, 1977, pp.264--279.

[3] 門脇芳雄 編: 「詰むや詰まざるや」, 平凡社東洋文庫, 1975.

[4] Kawano, Y., "Using similar positions to search game trees," Combinatorial Game Workshop, Berkeley, 1994.

[5] 小谷, 吉川, 柿木, 森田: 「コンピュータ将棋」, サイエンス社, 1990.

[6] McAllester, D.A.: "Conspiracy Numbers for Min-Max Search," Artificial Intelligence, Vol.35, pp.287-310, 1988.

[7] 脊尾昌宏: "共謀数を応用した詰め将棋プログラムについて," ゲームプログラミングワークショップ '95, pp.128-137, 1995.

[8] 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」第 1 章, 共立出版, 1998.

		付加情報なし		付加情報あり		あり／なし
番号	手数	解答時間	生成局面数	解答時間	生成局面数	生成局面数比
1	11	×	×	78分7秒	106892735	−
2	47	2分29秒	2790497	2分28秒	2782644	0.997
3	39	31秒	626037	30秒	578525	0.924
4	25	27秒	514310	36秒	665539	1.294
5	43	6分58秒	10390945	6分14秒	9086727	0.874
7	15	2分33秒	4356036	2分19秒	3817193	0.876
8	31	2分57秒	4409148	2分15秒	3269423	0.741
9	29	12秒	201957	10秒	144759	0.716
10	19	2分14秒	3993623	1分3秒	1772589	0.443
11	31	42秒	771347	57秒	1079967	1.400
12	61	5分27秒	5435341	5分3秒	4977114	0.915
13	41	10秒	160105	7秒	115081	0.718
14	83	46秒	811177	32秒	586341	0.722
15	33	20秒	303102	19秒	285511	0.941
16	53	1分41秒	1976638	53秒	915391	0.463
17	23	4分12秒	5233115	3分23秒	4142567	0.791
18	27	24秒	438842	29秒	561541	1.279
19	29	45秒	810348	42秒	749461	0.924
20	45	1分10秒	1116841	59秒	931805	0.834
21	21	34秒	736125	19秒	406137	0.551
22	33	26秒	586423	26秒	588743	1.003
23	25	13秒	255993	14秒	287029	1.121
24	31	15秒	228820	10秒	150298	0.656
25	17	29秒	553655	26秒	477214	0.861
26	25	36秒	934474	37秒	921183	0.985
27	25	2分24秒	3671946	1分31秒	2301273	0.626
28	47	×	×	3分46秒	3973379	−
29	21	23秒	414544	23秒	400637	0.966
30	119	1分58秒	2602010	1分25秒	1813827	0.697
31	107	1分24秒	1395070	1分19秒	1293843	0.927
32	35	37秒	674247	21秒	350139	0.519
33	25	10秒	168469	10秒	153020	0.908
34	31	48秒	1018308	42秒	886452	0.870
35	17	11秒	195149	11秒	210640	1.079
36	37	14秒	286199	13秒	261965	0.915
38	13	2秒	24990	2秒	24986	0.999
39	25	31秒	693119	20秒	420961	0.607
41	47	26秒	484536	25秒	459070	0.947
42	43	44秒	739408	36秒	577833	0.781
43	47	4分36秒	4348281	4分52秒	4562599	1.049
44	25	51秒	932366	47秒	833153	0.893
45	47	4分15秒	4453127	3分1秒	3111634	0.698
46	27	4分45秒	5548706	4分11秒	4802887	0.865
47	13	51秒	917221	35秒	631354	0.688
48	37	×	×	107分31秒	125237850	−
49	27	3秒	42666	3秒	31777	0.744
50	17	5秒	92861	5秒	80551	0.867

Figure8: 将棋無双の解図結果（第1番から第50番）

		付加情報なし		付加情報あり		あり／なし
番号	手数	解答時間	生成局面数	解答時間	生成局面数	生成局面数比
51	33	29分15秒	37000536	15分52秒	20355223	0.550
52	55	7分3秒	7353520	7分38秒	7842359	1.066
53	37	55秒	1011381	51秒	923579	0.913
54	19	41秒	990933	35秒	814046	0.821
55	11	4秒	55708	4秒	55529	0.996
56	45	9秒	160734	9秒	163003	1.014
57	29	55秒	1116218	24秒	460778	0.412
58	31	10秒	143877	10秒	142428	0.989
59	45	2分59秒	2975988	4分18秒	4470652	1.502
60	45	7秒	81847	6秒	75534	0.922
61	39	1分1秒	1024964	56秒	917440	0.895
62	23	11秒	174333	11秒	175019	1.003
63	25	1分22秒	1769854	1分9秒	1449066	0.818
64	19	1分15秒	1568606	2分17秒	2795696	1.782
65	9	1分55秒	2128925	1分54秒	2063713	0.969
66	29	23秒	307499	21秒	283254	0.921
67	33	1分44秒	2633905	1分23秒	2050011	0.778
68	39	6秒	66630	5秒	65251	0.979
69	31	9秒	149064	9秒	139733	0.937
70	79	4分22秒	4342465	4分3秒	3976944	0.915
71	25	18秒	257651	16秒	227472	0.882
72	41	45秒	754039	57秒	937600	1.243
74	15	38秒	652158	39秒	657575	1.008
75	225	3分45秒	3406290	2分14秒	1925398	0.565
76	31	44秒	908627	42秒	817475	0.899
77	11	0秒	2215	0秒	2215	1.000
78	13	14秒	299606	12秒	269214	0.898
79	11	5秒	93203	5秒	92316	0.990
80	35	49秒	982687	42秒	820631	0.835
81	23	20秒	359674	1分36秒	2106748	5.857
82	21	21秒	436917	9秒	160369	0.367
83	29	44秒	892141	41秒	850299	0.953
84	29	21秒	369697	13秒	239622	0.648
85	55	25秒	536109	18秒	360421	0.672
86	15	7秒	114822	7秒	108437	0.944
87	57	2分33秒	2578937	2分18秒	2323803	0.901
90	37	1秒	18096	2秒	18494	1.021
91	31	10秒	142528	10秒	139947	0.981
92	51	27秒	489899	26秒	460352	0.939
93	35	20秒	340856	18秒	311057	0.912
94	57	5分16秒	5886651	4分57秒	5572144	0.946
95	35	1分28秒	1805280	1分13秒	1468336	0.813
96	51	6分1秒	6680104	2分33秒	2743451	0.410
97	51	1分43秒	2116883	2分28秒	2995945	1.415
98	61	1分29秒	1585391	1分25秒	1489596	0.939
99	41	10秒	124218	6秒	67557	0.543
100	163	28分26秒	28509889	23分8秒	22852944	0.801

Figure9: 将棋無双の解図結果（第51番から第100番）