スーパーパズを解くプログラム

新谷敏朗
福山大学工学部情報処理工学科
shintani@fuip.fukuyama-u.ac.jp

1991 年度に GPCC の課題として採用されたスーパーパズを解くプログラムを提案する。このゲームはトランプのカード 1 組を使用する一人遊びであり、エースの動きの自由度が他のカードに比べて大きいことが特徴である。そのために、単純に新しい局面をゲーム木に追加していったのでは、同じ局面が繰り返し現われて成功局面に至ることができない。本プログラムでは、局面の重複をチェックするための探索データ構造として「パトリシア」をゲーム木とは別に用意して同じ局面がゲーム木中に存在しないように工夫した。それにより、ハーフサイズの 6 列までの場合ではいわゆるパソコンレベルのコンピュータでゲーム木の全探索が可能であることを確認した。フルサイズの 13 列の場合ではゲーム木の全節点数が平均で 10^9 のオーダーになることが予想され、100 回の試行のうち少なくとも 1 つの成功局面が得られた場合が 57 回、全探索して成功局面がなかった場合が 7 回であった。

A Program to Solve Superpuzz

Toshio Shintani
Dept. of Information Processing Engineering, Fukuyama University

I propose a program to solve Superpuzz which was adopted as a problem on GPCC in 1991. Superpuzz is a solitaire game with one deck of cards and has a characteristic that Aces can move more freely than other cards in the game. So you cannot find a solution by adding a new node simply to the game tree because the same node will be appeared in the tree many times. I use a data structure called 'Patricia' so that there is no duplication of nodes in the game tree. It is possible to search the game tree entirely up to the case of the half size (6 columns) on a typical computer for personal use. It is expected that there are nodes of 10^9 order in the game tree in the average case of the full size (13 columns). In the case of full size, the program found no solution in 7 cases after searching entirely in the game tree and at least one solution in 57 cases out of 100 times of try.

1．はじめに

「スーパーパズ(Superpuzz)」は MIT の Berliner 教授が提唱したトランプの一人あそびである。［1］ 1991 年度の GPCC の課題として採用されたが、文献［2］によるとまだ解かれていないようである。このゲームにおいて局面を節点に、カードの移動を枝に対応させるとゲームを表す状態遷移図は有向グラフになる。したがって、スーパーパズを解くために通常の完全情報ゲームを解く場合と同じように「ゲーム木」を作ると同じ局面を表す節点が多く現われる。さらには本来木ではなくグラフなので閉路を回り続けてしまい探索が終了しない可能性が高い。ここで提案するプログラムはそうならないように、重複局面をチェックするためのデータ構造を別に用意した。そして、同一局面が現われた場合は、その局面をゲーム木に追加しないようにして全探索を可能にしたものである。

2．ゲームの性質

まず、文献［1］によってルールとそれによって生じるゲームの性質を述べる。

2.1 ルール

ルールは以下の通りである。スートを H,D,S,C で表し、数字はエースを A，絵札を J,Q,K それ以外は数字で示す。

1. トランプ1組をよくシャッフルした後、表向きに13列4段に並べる。
2. 4枚の K を取り除く。それによって生じた空白を「穴」と呼ぶ。
3. 穴が左端にある場合は、任意のスートの A をそこに移動できる。穴が左端でない場所にある場合は、穴の左隣のカードに続くカードをそこに移動できる。例えば、H6 の次には H7 が続く。穴の左隣が Q または穴の場合は、いかなるカードもそこには移動できない。
4. カードの移動によって新たに生じた穴には、3.の規則に従ってカードを移動できる。
5. すべてのカードが左端の A を先頭に昇順に並べば成功である。スートの並び順は任意である。どのカードも移動できない状態（穴の左隣がすべて Q または穴）で成功状態でなければ失敗である。

図 1 に初期局面の例を示す。ただし、10 は 0 と表記した。さらに4つの穴を区別するために、空白とせずに K で表記している。

```
S3 D7 SQ D3 D2 DK D5 S2 C9 CJ H2 D4 H4
S4 SJ H3 D9 H5 S9 D8 C2 HJ SA DA HQ CA
DQ DJ D0 S7 C0 S0 S6 H9 C8 H8 H0 C3 C6
C5 C4 CQ H7 H6 HK SK S8 CK C7 D6 HA S5
```

図 1 初期局面の例(1)

このルールから、A 以外のカードは移動先が一意的に定まるので解答の表記の際に移動させるカードを示せばよいことがわかる。A については移動先が最大で 4 個所存在する。文献［1］では A も一意的な動きをするように便宜的に表記している。しかし後でわかるように A の左端での動きがこのゲームのキーポイントであると考えられるので、ここではカードの移動をそのカードと穴（を表すキング）との交換であると解釈して図 2 のように(HA, DK)のように書くことにする。なお、後で述べるように本プログラムでは解答を成功局面から初期局面までさかのぼるようにしているので、矢印が左向きになっている。

このゲームは列数を減らしてもプレイが可能である。例えば、10 列でプレイする場合には、各スート A から 10 までの 40 枚のカードを 1 組として、上のルールで K を 10 に Q を 9 と読み替えればよい。列数が 1 の場合はゲームとして成り立たないが、列数が 2 以上の場合はゲームとしてプレイできる。ただし、すぐわかるように、列数が 2 の場合は必ず成功する。文献［1］にも 6 列の場合の例が紹介されている。ここでは、6 列の場合を「ハーフサイズ」、13 列の場合を「フルサイズ」と呼ぶことにする。

```
(CQ, SK)<-(CJ, SK)<-(CO, CK)<-(SK, HQ)<-(CK, HJ)<-(C9, SK)<-(C8, CK)<-(SQ, SK)<-(SJ, CK)<-
(HK, C7)<-(SK, C6)<-(CK, C5)<-(DQ, CK)<-(DJ, DK)<-(DO, CK)<-(CK, HO)<-(SO, CK)<-(D9, DK)<-
(DK, H9)<-(HO, CK)<-(D8, CK)<-(CK, H8)<-(S9, DK)<-(S8, CK)<-(H9, DK)<-(D7, DK)<-(DK, H7)<-
(H8, CK)<-(D6, CK)<-(CK, H6)<-(S7, DK)<-(S6, CK)<-(HK, SO)<-(SK, S9)<-(HO, SK)<-(SK, C4)<-
(DJ, SK)<-(D5, SK)<-(SK, H5)<-(H7, DK)<-(H6, CK)<-(D4, DK)<-(D3, CK)<-(DK, H4)<-(CK, H3)<-
(S5, SK)<-(S4, DK)<-(S3, CK)<-(H5, SK)<-(D2, SK)<-(SK, H2)<-(H4, DK)<-(H3, CK)<-(CK, SQ)<-
(CK, S8)<-(H9, CK)<-(CK, C3)<-(DO, CK)<-(D4, CK)<-(CK, D5)<-(CK, H4)<-(D9, CK)<-(S2, SK)<-
(SK, SJ)<-(SK, S7)<-(SO, HK)<-(HK, S6)<-(CK, S5)<-(C7, CK)<-(H8, SK)<-(SK, C2)<-(CK, CA)<-
(SA, CK)<-(DA, CK)<-(CK, SA)<-(DA, CK)<-(CK, HA)<-(SA, CK)<-(CK, DA)<-(SK, S2)<-(HA, CK)<-
(DA, CK)<-(CK, SA)<-(DA, CK)<-(CK, HA)<-(CK, DA)<-(HA, CK)<-(SA, CK)<-(CK, HA)<-(H2, SK)<-
(HA, CK)<-(CK, SA)<-(CK, DA)<-(CK, HA)<-(DA, CK)<-(CK, HA)<-(SA, CK)<-(CK, DA)<-(HA, CK)<-
(SK, H2)<-(DA, CK)<-(CK, HA)<-(H2, SK)<-(SK, S2)<-(SA, CK)<-(DA, CK)<-(HA, CK)<-(CK, DA)<-
(HA, CK)<-(CK, SA)<-(HA, CK)<-(SK, H2)<-(CK, HA)<-(SA, CK)<-(DA, CK)<-(CK, HA)<-(DA, CK)<-
(HA, CK)<-(CK, SA)<-(DA, CK)<-(HA, CK)<-(H2, SK)<-(CK, DA)<-(HA, CK)<-(SK, H2)<-(S2, SK)<-
(SA, CK)<-(CK, DA)<-(CK, HA)<-(SA, CK)<-(DA, CK)<-(CK, HA)<-(CK, DA)<-(HA, CK)<-(SK, H2)<-
(CK, HA)<-(CK, DQ)<-(CK, S8)<-(CK, HO)<-(S9, CK)<-(H7, HK)<-(D8, HK)<-(HK, S4)<-(C6, HK)<-
(HK, S3)<-(C5, HK)<-(HK, S2)<-(C4, HK)<-(HK, HQ)<-(HK, CO)<-(S5, HK)<-(HK, CQ)<-(CK, CJ)<-
(C3, CK)<-(CK, HJ)<-(CK, C9)<-(D7, CK)<-(CK, S3)<-(HQ, CK)<-(CK, SA)<-(DA, CK)<-(HA, CK)<-
(CK, DA)<-(HA, CK)<-(S4, CK)<-(SK, DJ)<-(SK, S7)<-(SK, H9)<-(S8, SK)<-(SK, CO)<-(CK, CJ)<-
(CK, SO)<-(C7, CK)<-(H6, CK)<-(CK, S9)<-(HO, SK)<-(SK, C8)<-(D6, SK)<-(SK, S2)<-(DA, SK)<-
(SK, HA)<-(HK, C6)<-(SK, C5)<-(CQ, SK)<-(H2, SK)<-(SK, D7)<-(HA, SK)<-(SK, S3)<-(SK, C9)<-
(SK, H8)<-(HK, H7)<-(DK, D3)
No. 1, 192 moves, 5165923 states searched.
```

図 2 解答手順の例

2.2 ゲーム木

局面を節点に、カードの移動を有向の枝に各々対応させると、初期局面から生成可能な局面に至る状態遷移図を描くことができる。これはいわゆるゲーム木に対応するものであるが、スーパーパズでは4つの穴にそれぞれ異なるカードを移動させるので、ある節点から別の節点に至る道の数が非常に多くなる。従って、正確には状態遷移図はゲームを表すグラフである。このようなグラフで通常のゲーム木のように初期局面から始めて可能な手（カードの移動）で生成される局面を子節点として単に付け加えていくと、本来は同じ部分グラフが別々に扱われてしまう。従って、同じ局面を表す節点が一つのゲーム木の中に多数存在するという状態になり得る。

2.3 有向の閉路

さらに、A 以外のカードは一度移動すると左隣のカードが移動してその右隣が穴になるまでは移動できないが、A は左端の穴であれば自由に移動できる。よって、図 3 のようにゲームを表すグラフの中に有向の閉路が存在する。図 3 は局面の左端のみを描いたものであり、穴が一つと A（スートのみ示している）が 3 つある状態を示している。また、どちら向きにも遷移可能なので枝の向きは省略している。この場合は、左端だけを考えると 4×3！＝24 通りの局面が存在し、最大長さ 24 の有向閉路が存在する。（図の番号順に回る）そして第 2 列から第 4 列までを考慮に入れると、図 3 の 1 本の枝が 12 回のカードの移動に対応する場合があることが図 4 によりわかる。図 4 は局面のうち任意の 2 段を左端から 4 列だけ描いたもので、カードの数字だけを示している。（スートは 4 枚とも同じとする）図 4 については、図の番号順に進む長さ 12 の道が存在する。従って、図 3 と図 4 をあわせて考えると長さが 12×24＝288 の有向閉路が存在することがわかる。従って、スーパーパズのゲームグラフ探索を通常のゲーム木探索として実行したのではこのような有向閉路を回り続けるということが起こり得る。単純に考えるとこれを避けるために、かなり前の（少なくとも 288 手前）局面までさかのぼって局面の重複をチェックし既にゲーム木の中に存在する節点は破棄して追加しないようにする必要がある。

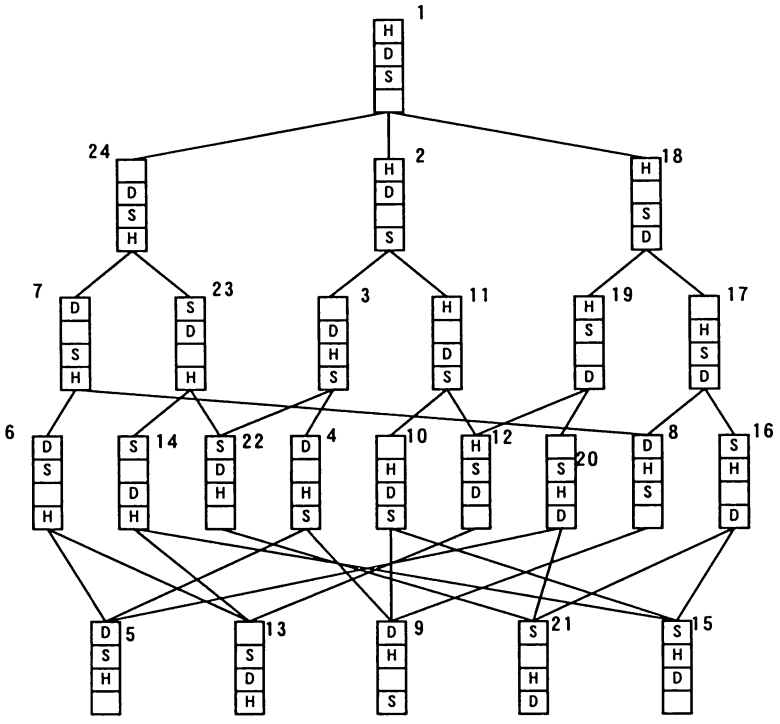


図 3 局面の左端のみに着目した状態遷移図の例

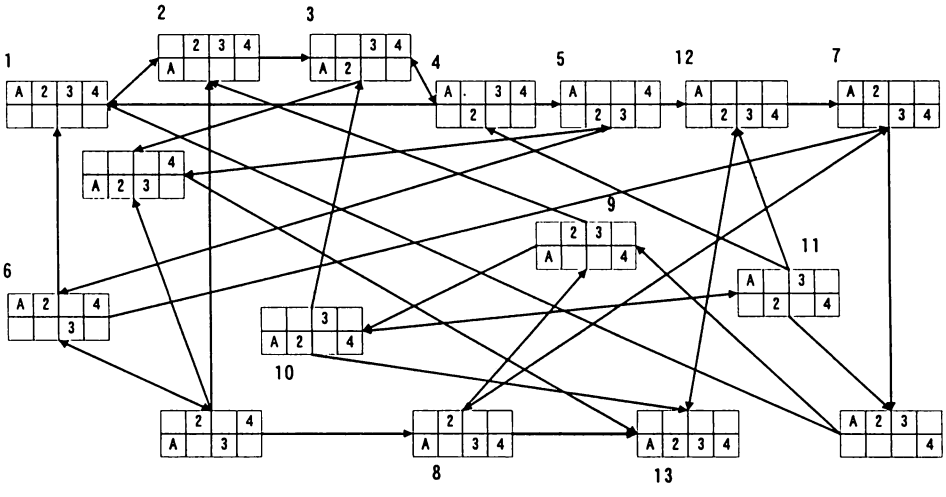


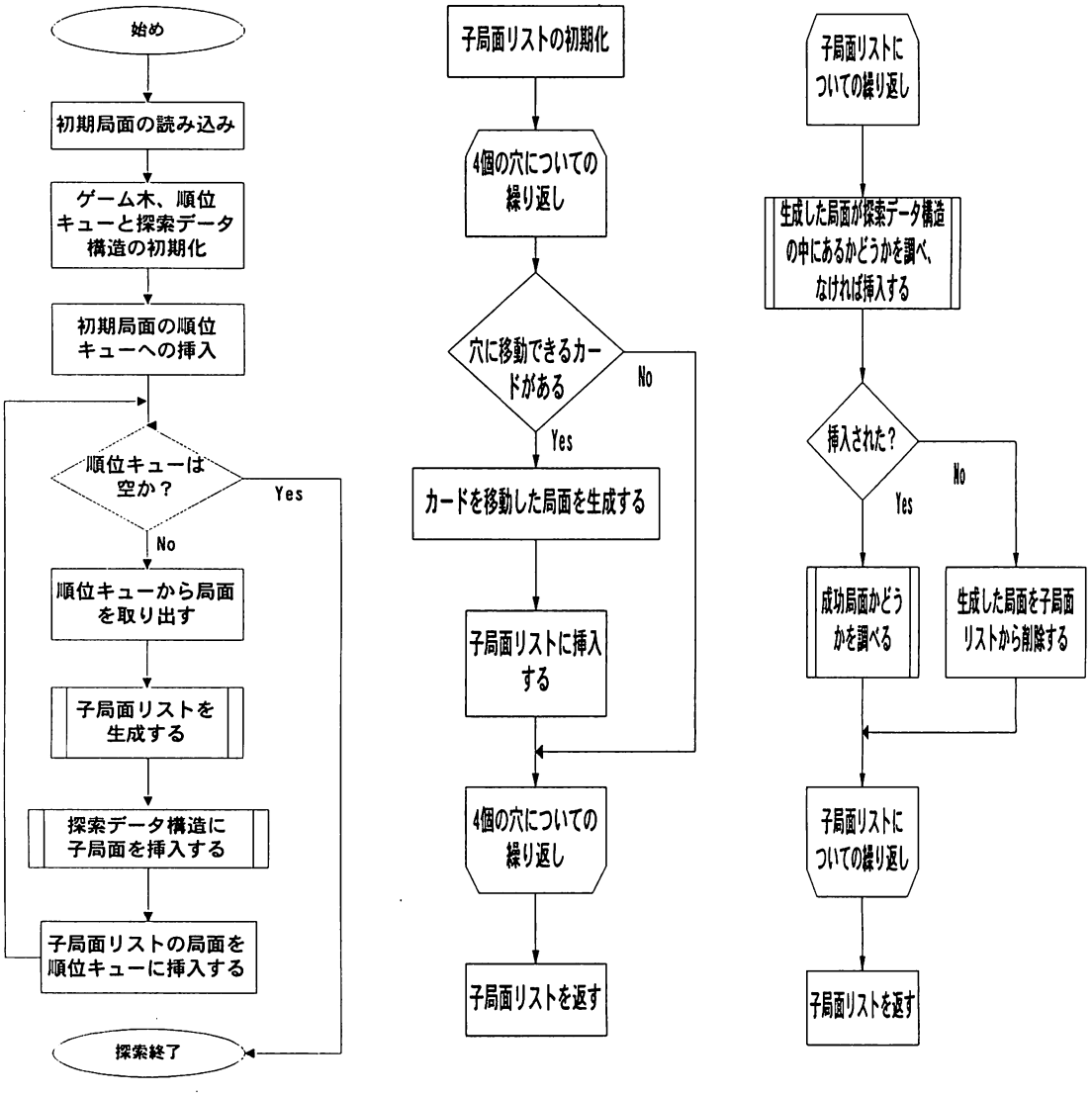
図 4 局面の二つの段と左端の 4 列のみに着目した状態遷移図の例

3．解法の概略

前節で述べたことから、スーパーパズを解くプログラムの流れ図は図 5 のようになる。

3.1 探索方法

基本的な考え方としては、初期局面から生成される子局面（へのリンク）を順位キューに挿入して取り出した順に更に子局面を生成していく。一つの局面から生成される子局面の個数は 0 以上



(a) 全体の流れ図　(b) 子局面リスト作成部分　(c) 探索データ構造への挿入部分

図 5 プログラムの流れ図

16 (穴がすべて左端にある場合) 以下なのでそれらをリストにして一度に順位キューに挿入することにする。子局面がない場合は当然挿入は行われない。順位キューが空になれば、ゲーム木をすべて探索したことになる。「スーパーパズ」ではスートの並び方は問題にしないので一つの初期局面に対して成功局面の数は最大で 24 である。成功局面がひとつ得られるだけでよい場合はひとつ発見された段階で探索を終了すればよい。なお、順位キューを通常のキューにすれば幅優先探索を、スタックにすれば深さ優先探索を行うことになる。

3.2 局面の重複

探索の途中で生成された子局面は既にゲーム木の中に存在する可能性があるので、別に重複のチェック用のデータ構造を用意しておく。このチェックの効率がプログラム全体の効率を左右する。

ここでは、探索のキーとなる局面がカードのスートと数字という離散量からなることに着目して基数探索の一種である「パトリシア」[3] を用いた。

3.3 データ構造

ゲーム木は一般に子節点の個数が不定の多分木であるが、この場合は局面の重複をチェックするためのデータ構造は別に用意するので、根から下に降りていく必要はない。逆に成功局面が得られた時に根までさかのぼることが必要なので、ゲーム木としての節点間のリンクは親節点へのリンクを保持することにする。よって一つの局面について保持すべき情報は、52 枚のカードと親節点へのリンクである。1 枚のカードにはスートに 2 ビット、数字に 4 ビットが必要である。通常 1 バイトは 8 ビットであることと、リンクのために 4 バイト必要であるとすると 1 局面あたり、56 バイトとなる。順位キューについては、最初の解が早く得られることを重視してスタックとした。パトリシアでは局面へのリンク、左右の子節点へのリンクとその節点で調べるビットを保持する整数型変数が必要である。スートと数字で表されるカードの情報に必要なビット数（6×52＝312）は一定なので、パトリシアによる局面の重複のチェックは局面数には関係しない定数時間で実行できる。順位キューへの挿入・削除の繰り返しは、重複局面をチェックしているので局面数に比例する時間で実行できる。したがって、このプログラムの時間計算量はゲーム木の全節点数（全局面数）に対して線形である。また、全局面をメモリ中に保持するので空間計算量も全局面数に対して線形である。

4．実行結果と考察

疑似乱数によりシャッフルした初期局面の例を図 6 に示す。先に示した図 1 もその一つである。

S6 D6 D0 CQ SK CK H2 D4 CA C2 SA SQ HK
S2 H3 DJ S0 H5 D8 H8 S7 D9 CJ C7 C4 D3
H4 S9 C5 D5 H0 C0 H9 H6 S3 HJ SJ HA D2
DK S5 C8 HQ S8 C6 DA C9 DQ D7 H7 C3 S4

(a)

S3 HJ HQ C6 S9 H3 S0 H6 D5 H2 D0 S4 D4
S5 SJ CA DA C9 D8 S7 CJ C0 C7 D3 DJ CK
C3 C5 C8 C2 S8 HA H0 H5 SQ D7 C4 S6 H8
H4 DQ S2 H9 D2 CQ HK D9 SA SK H7 DK D6

(b)

図 6 初期局面の例(2)

図 1 については約 3295 万個の局面を探索した結果、図 2 に示したものも含めて 3 つの異なる解を得た。図 2 の解では 57 手目から 64 手にわたって A とそれに付随した 2 の動きが繰り返されている。（斜体部分）これは本プログラムが基本的に深さ優先探索を行っているためである。この 64 手は局面を見ればすぐにわかるように 3 手に短縮できる。図 6 の(a)については、17574132 個の局面が生成されたが成功局面に至らずに探索を終了した。即ち、この初期局面については解を得ることが理論的に不可能である。(b)については、図 1 と同じく約 3295 万個の局面を探索したが、解が得られないままメモリ不足で探索を打ち切った。また 100 個の初期局面について本プログラムを実行した結果は、

成功（少なくとも 1 個の解が得られた）：57　　全探索して失敗（理論的に不可能）： 7

であった。残りの 36 個についてはメモリ不足で探索途中で実行を打ち切った。メモリ不足にならなかった例については、文献 [1] による予想成功率 82%より高い値（約 89%）が得られてい

る。また、成功した57の場合については、最初の成功局面に至るまでの

手数の平均値：　　190　　探索局面数の平均値：　6726362

であった。全探索して失敗した7つの場合については、

探索局面数の平均値：19767976

であった。次に、文献［1］に示されている 6 列の場合の初期局面と得られた 16 個の解のうち ひとつを図7に示す。

```
S3 H2 D2 D4 H6 D5
D6 C4 H3 C5 CA S6
D3 SA DA S4 S5 C6
C2 C3 H4 HA S2 H5
```

```
(S6, D5)<-(H5, D6)<-(D6, D4)<-(D6, C5)<-(D6, S5)<-(H2, D6)<-(D6, S4)<-(HA, D6)<-(D6, S3)<-
(C4, D6)<-(S4, D6)<-(D6, H2)<-(S2, D6)<-(C3, D6)<-(H6, C2)<-(D2, H6)<-(S6, CA)<-(S6, SA)<-
(DA, S6)<-(S6, SA)<-(S6, DA)<-(H6, D2)<-(DA, S6)<-(SA, S6)<-(S6, DA)<-(D3, D6)<-(D2, H6)<-
(DA, S6)<-(S6, SA)<-(S6, DA)<-(H6, C4)<-(H4, H6)<-(H3, D6)<-(H6, D3)<-(D6, D2)<-(DA, S6)<-
(D5, H6)<-(D4, D6)<-(D3, S6)<-(D6, D5)<-(S6, D4)<-(SA, S6)<-(S6, DA)<-(C2, S6)<-(DA, D6)<-
(D5, H6)
No. 1,  46 moves,  61705 states searched.
```

図 7 ハーフサイズ(6 列)の初期局面と解の例

列数が 6 以下の場合に、疑似乱数によりシャッフルした初期局面 100 個について本プログラムを実行した結果を表1に示す。

表 1 列数が 6 までの場合の実行結果

列数 x_i	100回の試行で初期局面から生成された局面数の			最初に発見した成功局面までの		成功局面に至らなかった（理論的に不可能な）	
	平均値 y_i	最大値 z_i	最小値	平均手数	探索局面数	初期局面数	平均局面数 w_i
2	90	209	1	3	19	0	
3	3064	10044	1	31	118	7	7
4	47452	200408	2	48	393	11	282
5	391324	3034017	1	77	2753	13	2178
6	2142024	18066011	109	79	17916	12	35760

6 列までは試みた 100 回すべての場合に全探索可能であった。表1によると 6 列の場合の最大局面数が 10^7 のオーダーである。上で述べたように1局面あたり数十バイトが必要なので、6 列までであればメモリが 1G バイト程度あれば十分全探索が可能であることがわかる。32 ビットアドレスの OS であれば現在のパソコンでもこの条件を満たしている。実際、今回の計算に用いたのは PC/AT 互換機で、

CPU：Pentium II 400MHz，メモリ：768MB，スワップ領域：4G バイト
OS：FreeBSD 2.2.8 Release，コンパイラ： gcc version 2.7.2.1

という特別に高性能ではないパソコンである。

表1の列数とゲーム木あたりの最大局面数、平均局面数と最初の成功局面までに探索した局面数の平均値の関係を図示すると図8のようになる。

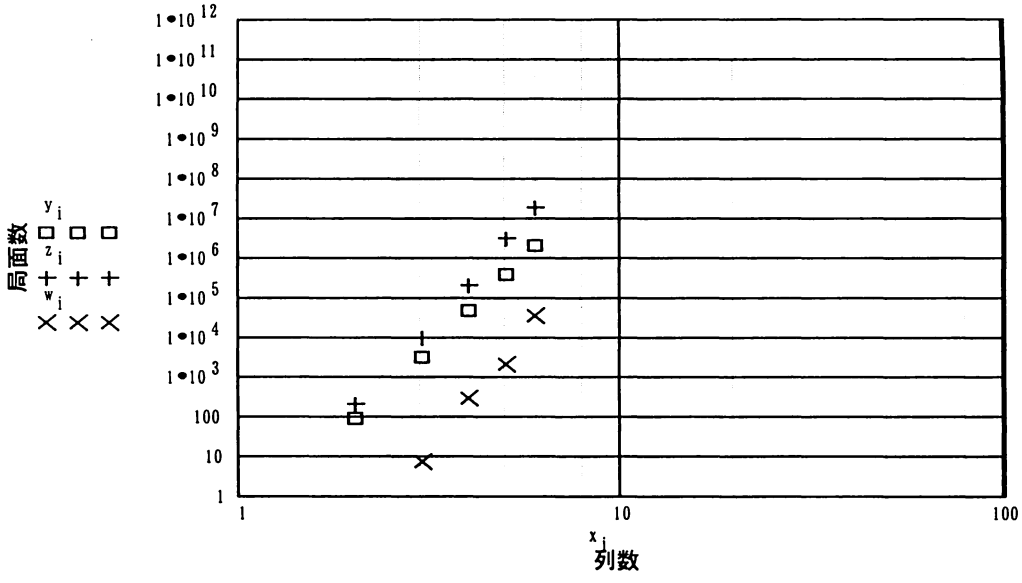


図 8 列数と局面数の関係

図 8 の結果を外挿すると、13 列では平均局面数は 10^9 個、最大局面数は 10^{10} 個のオーダーになると考えられる。よって 13 列の場合に全探索が可能なためには 100G バイト以上のメモリが必要であると予測される。ただしひとつの成功局面だけが必要なのであれば、10G バイト程度のメモリで充分ではないかと考えられる。

5．あとがき

スーパーパズを解くために、状態遷移図を単にゲーム木として探索したのでは、同じ局面が度々現われて堂々巡りに陥る可能性が高いことを示した。そして、局面の重複をチェックするための探索データ構造として「パトリシア」を用いることによってゲーム木の全探索が時間および空間計算量に関して全局面数に対して線形時間で実行できることを示した。実際に 6 列までの場合にはゲーム木の全探索が現在のパソコンレベルの計算機で可能であること示した。13 列の場合は現在の 32 ビットアドレスのパソコンでは 6 割程度しか解が得られないが、64 ビットアドレスが一般的になれば 13 列の場合を全探索することが可能になると考えられる。また、ルールで決められた移動方法と逆にカードを移動させてを成功局面からさかのぼることで、成功局面に至る可能性のある局面をすべて生成することも計算機資源が充分にあれば可能である。

参考文献

[1] 南雲，GPCC ウルトラナノピコ問題 フットステップとスーパーパズ，bit，Vol.23，No.5,共立出版，1991

[2] 小谷ほか，プログラミングシンポジウムと GPCC のゲームとパズル，情報処理学会研究報告，99-GI-1，pp55-61(1999)

[3] R.セジウィック，アルゴリズム C 第 2 巻，pp70-74，近代科学社，1996