福山大学工学部紀要
第23巻 1999年10月

スーパーパズにおける巾優先探索

新谷 敏朗

Breadth-first Search in Superpuzz

Toshio SHINTANI†

ABSTRACT

Superpuzz is a solitaire game with one deck of cards and was adopted as a problem on GPCC in 1991. Superpuzz 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 a game tree has nodes of 10^8 order and about 100 levels in the average case of the full size (13 columns) before the first solution is found. In the case of the full size, the program found solutions in 2 cases after searching entirely in the game tree and aborted with no solution because of memory limitation in 98 cases out of 100 times of try.

キーワード：スーパーパズ，探索，巾優先，GPCC
Keywords: Superpuzz, Search, Breadth-first, GPCC

1．まえがき

「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。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」を作って「深さ優先探索」を行うと同じ局面を表す節点が多く現われる。さらには状態遷移図は本来木ではなくグラフなので木として扱うと、元のグラフの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。よって本論文で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ゲーム木とは別に用意した。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ゲーム木の全探索を可能にした。探索方法としては、最短の手数で解が得られる「巾優先探索」を用いて、実際にパーソナルコンピュータ上で種々の初期局面について解を求めて、考察を加えた。

2．ゲームの性質

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

† 情報処理工学科

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。

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、6 列でプレイする場合には、各スート A から 6 までの 24 枚のカードを 1 組として、上のルールで K を 6 に Q を 5 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。

図 1 にハーフサイズの場合における初期局面の例を示す。なお 4 つの穴を区別するために、空白とせずに D6 などと表記している。

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

図 1 初期局面の例

Fig.1 An example of initial states

また、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を解く際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 K、ハーフサイズの場合は 6）との交換であると解釈して図 2 のように(H3, S6)のように書くことにする。なお、後で述べるように本論文のアルゴリズ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。

```
(H3, S6)<-(C5, D6)<-(C4, H6)<-(S6, D5)<-(C6, D4)<-
(D6, D3)<-(H6, D2)<-(D6, S5)<-(H5, H6)<-(H2, D6)<-
(C3, H6)<-(H6, C4)<-(C2, D6)<-(CA, H6)<-(D6, S4)<-
(H6, S3)<-(HA, D6)<-(D4, H6)<-(H6, S2)<-(C6, SA)<-
(H6, H3)<-(S4, H6)<-(H6, D3)<-(H6, H2)
```

図 2 解答手順の例

Fig.2 An example of solutions

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。例えば、図 1 の初期局面から 2 手分の状態遷移図の一部分を描くと図 3 のようになる。図 3 の節点は局面の左端のみを描いたもので、枝の向きに遷移が可能であることを表している。したがって例えば、初期状態から節点 6 に至るのに、

$$1 \rightarrow 2 \rightarrow 6$$

$$1 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 7 \rightarrow 6$$

などの複数の経路が存在する。また、

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 1$$

あるいは、

$$4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 4$$

という閉路が存在する。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。図 3 の例では長さ 6 の閉路であるが、長さが２８８の有向閉路があることが知られている。［3］よってこれを避けるためには、かなり前の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。本論文のアルゴリズムではその

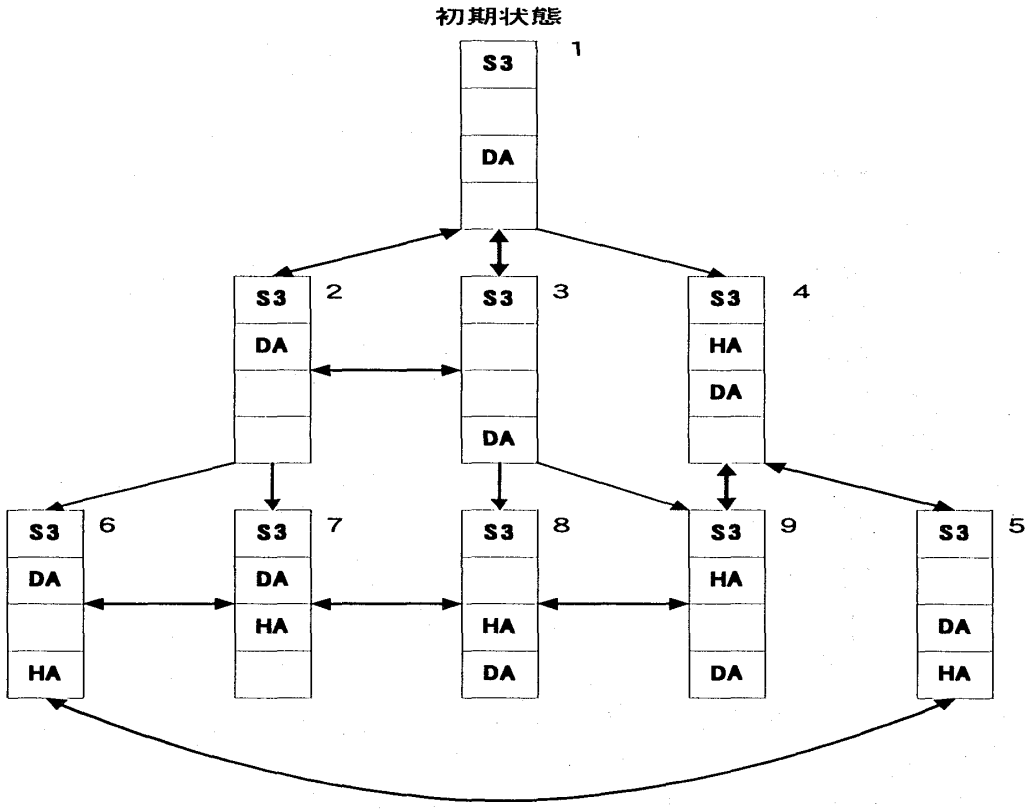


図 3 状態遷移図の例
Fig.3 An Example of state transition

ような重複節点のチェックを行なっている。

3．ゲーム木の探索アルゴリズム

前節で述べたことから、スーパーパズを解くアルゴリズムの流れ図は図４のようになる。

3.1 探索アルゴリズム

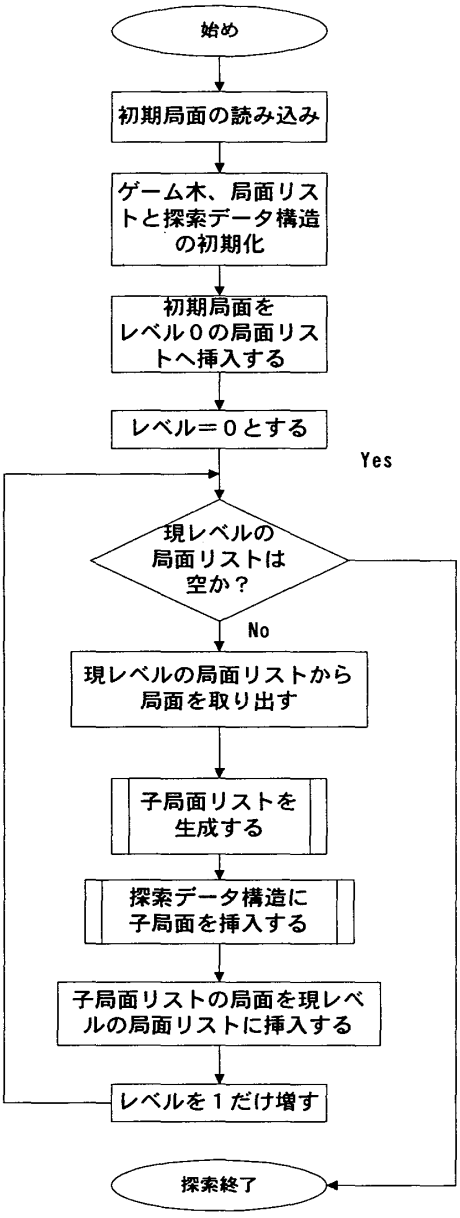
基本的な考え方としては、初期局面から生成される子局面（へのリンク）をレベルごとに初期化してある空のリストに挿入していく。一つの局面から生成される子局面の個数は 0 以上 16 (穴がすべて左端にある場合) 以下なのでそれらをリストにして一度に次のレベルの局面リストに挿入することにする。子局面がない場合は当然挿入は行われない。次のレベルに移ったときにそのレベルの局面リストが空であれば、ゲーム木をすべて探索したことになる。「スーパーパズ」ではスートの並び方は問題にしないので一つの初期局面に対して成功局面の数は最大で 24 個存在する可能性がある。成功局面がひとつ得られるだけでよい場合は最初の成功局面が発見された段階で探索を終了すればよい。なお、通常の幅優先探索ではキューを用いるが、本論文ではどのレベルまで探索が進んでいるかがわかりやすいことを重視して「レベル優先」の探索を行った。

3.2 重複局面のチェック

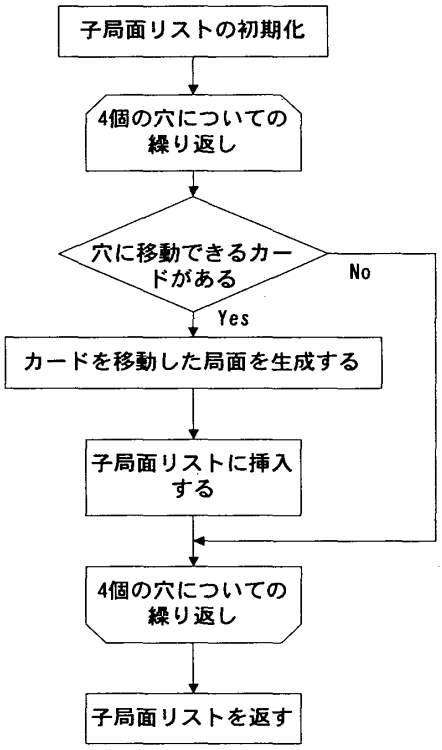
2 節で述べたように、探索の途中で生成された子局面は既にゲーム木の中に存在する可能性があるので、別に重複のチェック用のデータ構造を用意しておく。そのための探索用データ構造として「平衡木」なども考えられるが、本論文では探索のキーとなる局面を表す情報がカードのスートと数字という離散量からなることに着目して基数探索の一種である「パトリシア」[4] を用いた。

3.3 データ構造

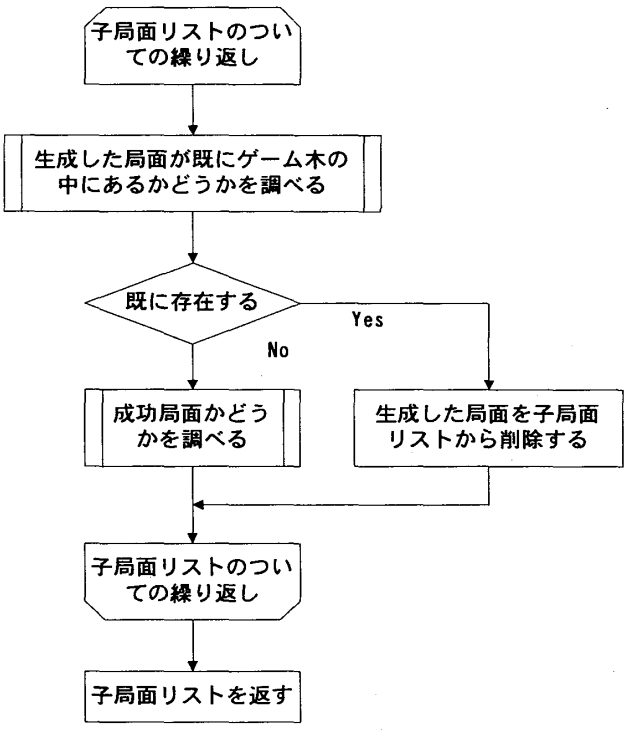
ゲーム木は一般に子節点の個数が不定の木であるが、この場合は局面の重複をチェックするためのデータ構造は別に用意するので、根から下に降りていく必要はない。逆に成功局面が得られた時に根までさかのぼることが必要なので、ゲーム木としての節点間のリンクは親節点へのリンクを保持することにする。そして、レベル別の局面リストのために弟局面に対するリンクも持つことにする。よって一つの局面について保持すべき情報は、52 枚のカードの情報



(a)全体の流れ図



(b)子局面リスト生成部分の流れ図



(c)探索データ構造への挿入部分の流れ図

図 4 スーパーパズの幅優先探索アルゴリズムの流れ図

Fig.4 Flow chart of the algorithm for Breath-first search in Superpuzz

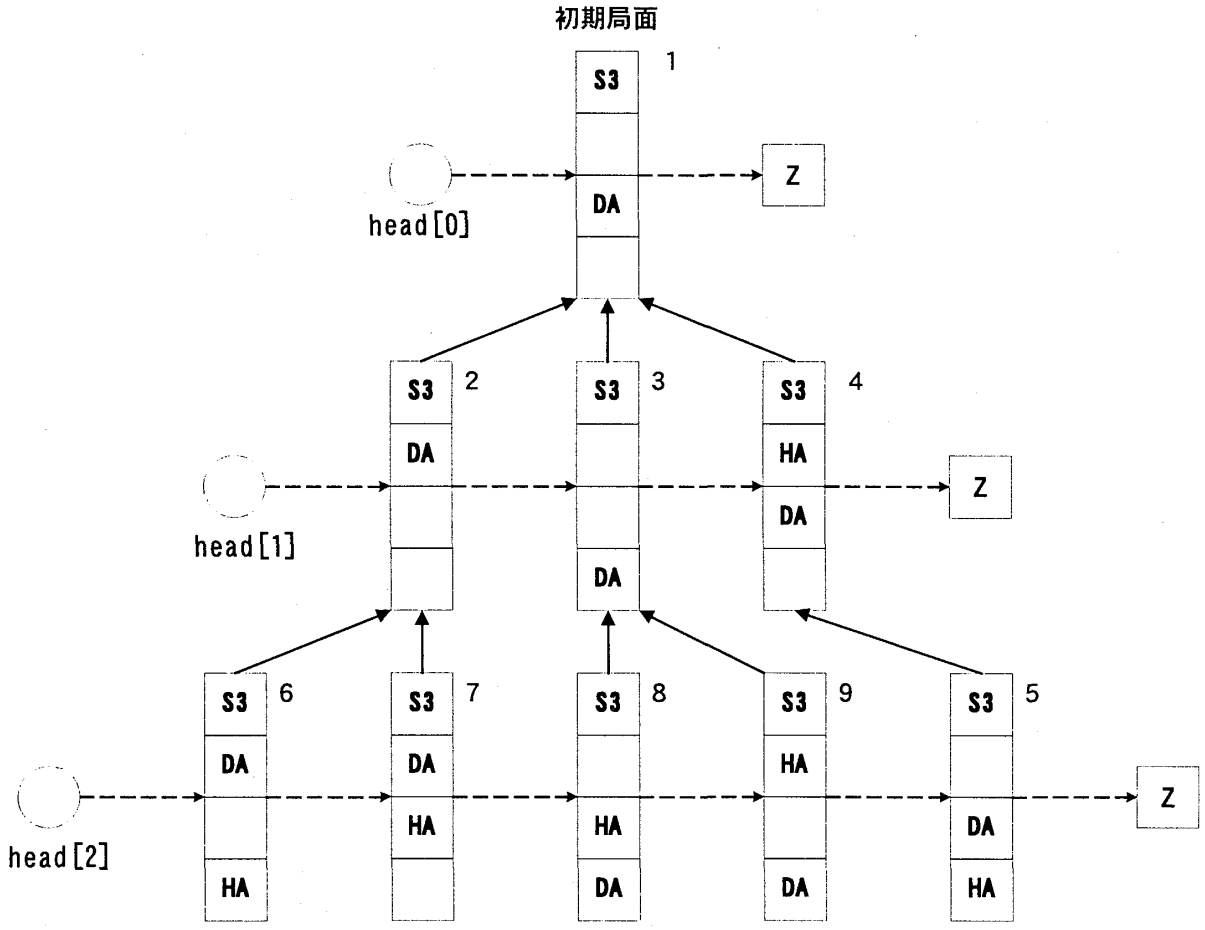


図 5 スーパーパズのゲーム木の例
Fig.5 An example of game trees in Superpuzz

と親節点、弟節点へのリンクである。よって例えば図 3 の状態遷移図に対応するゲーム木は図 5 のようになる。図 5 において、実線で描かれた枝は親節点へのリンクを表し、破線で描かれたリンクは弟節点へのリンクを示す。また、head[i]は第 i レベルの局面リストの先頭のダミー節点を、z は末尾のダミー節点を各々表している。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、60 バイトとなる。なお、実際のプログラムではさらに穴の位置も記憶しておくようにしているので、1 局面あたり 4 バイト使用している。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（各々 4 バイト）とその節点で調べるビットを保持する整数型変数（2 バイト）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。レベルごとの局面リストへの挿入・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。従って、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、全局面の情報を主記憶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．探索結果

疑似乱数によりシャッフルした初期局面を生成させて、本論文のアルゴリズムを適用してみた。図 1 と図 2 はその結果の一部である。使用した計算機は 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

という仕様のパソコンである。スーパーパズは列数を減らしても実行可能なので、まず 2 列から 6 列ま

表 1 ハーフサイズまでの場合の探索結果
Table 1 Results of search in case of 2 to 6 columns

列数 x	探索した全局面数	最大レベル	メジアンレベル	最初の解までの		局面生成時の重複の	
				手数 y	探索局面数 z	回数	割合
2	91	4.7	3.0	2.1	21	255	69.0%
3	3064	14.7	8.5	6.0	441	6295	60.2%
4	47026	28.7	15.9	10.0	3634	96413	61.6%
5	388476	42.3	23.6	15.5	28003	846803	60.3%
6	2142024	61.9	34.6	21.6	165006	4779494	65.6%

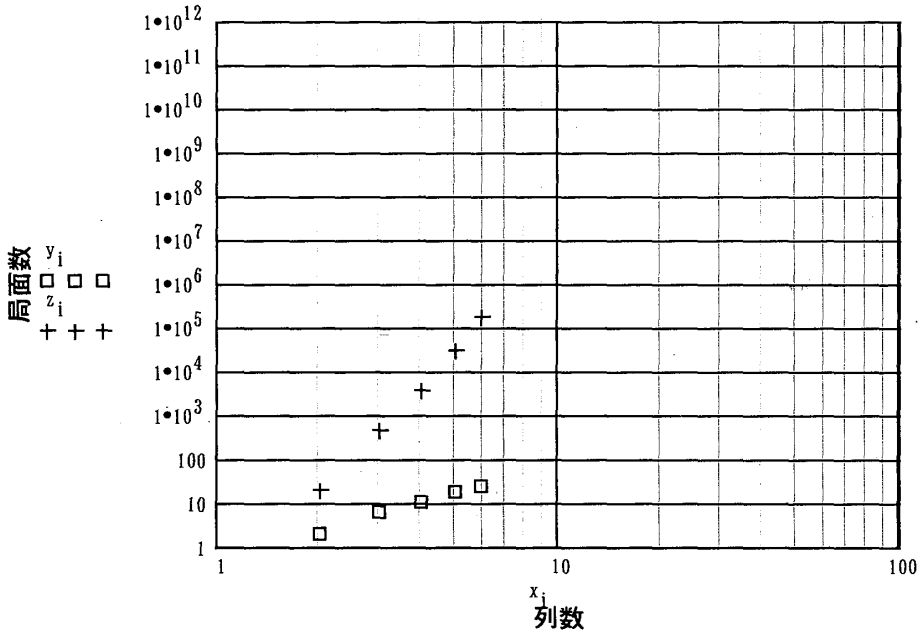


図 6 スーパーパズの列数と最初の解が見つかるまでの手数、探索局面数の関係
Fig.6 Relation between number of columns and number of moves and number of states before the first solution is found

での場合について各々 100 個の初期局面について実行した結果を表 1 に示す。この結果、子局面の約 6 割以上が既にゲーム木中に存在していることがわかった。このことは本論文で行った重複局面のチェックがスーパーパズの探索を成功させるために非常に重要であることを示している。表 1 の列数と最初の成功局面が得られるまでの手数およびそれまでに探索した局面数の平均値の関係を図示したものが図 6 である。この図から、フルサイズの 13 列の場合には最初の成功局面までの手数は 100 手を越え、探索局面数は平均 10^8 以上になることが予測される。従って、今回使用した程度の性能のパソコンではフルサイズの場合を解くことは難しい。実際、100 通りの初期局面に対して本論文の幅優先アルゴリズムを適用してみたが、ほとんどの場合はゲーム木のレベルがおおよそ 70 程度まで達した段階でメモリ不足となり探索を打ち切らざるを得なかった。図 7 に解が見つかった例をふたつ示す。図 7 の解答の手順で 10 は 0 と表記している。メモリ不足になるのは幅優先の探索によって最短手数の解を得ることを重視したことによってメモリ中に保持すべき局面数が多くなることが原因である。もし深さ優先の探索を行えば、手数はかなり多くなるが最初の成功局面までに保持すべき探索局面数は当然少なくなる。文献 [3] では、深さ優先探索により本論文と同じ計算機によってフルサイズの場合にもランダムに生成した初期局面の

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

初期局面の例(1)

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

初期局面の例(2)

```
(SK, HQ)<-(CQ, CK)<-(CK, HJ)<-(SQ, CK)<-(CJ, CK)<-(CK, H0)<-(SJ, CK)<-(C0, CK)<-(CK, H9)<-(S0, CK)<-
(C9, CK)<-(CK, H8)<-(S9, CK)<-(C8, CK)<-(CK, H7)<-(S8, CK)<-(C7, CK)<-(CK, H6)<-(S7, CK)<-(H5, DK)<-
(DQ, CK)<-(H4, SK)<-(D4, CK)<-(S6, DK)<-(DJ, DK)<-(S5, SK)<-(D3, DK)<-(CK, S4)<-(D0, SK)<-(D9, CK)<-
(D2, SK)<-(D8, HK)<-(DA, CK)<-(HK, HQ)<-(DK, S3)<-(DK, H3)<-(SK, S2)<-(D4, DK)<-(CK, SA)<-(D7, DK)<-
(SK, H2)<-(DK, HJ)<-(CK, HA)<-(S4, HK)<-(D3, SK)<-(S3, DK)<-(D6, SK)<-(H3, CK)<-(HK, C6)<-(D8, HK)<-
(DK, C5)<-(DK, SQ)<-(CQ, DK)<-(H0, DK)<-(S2, DK)<-(DK, C4)<-(SA, CK)<-(CK, H5)<-(DK, SJ)<-(CJ, DK)<-
(H9, DK)<-(HK, H8)<-(C6, DK)<-(C3, HK)<-(DK, D4)<-(HK, S0)<-(S4, DK)<-(C0, HK)<-(SK, C2)<-(SK, S9)<-
(DK, S8)<-(C9, SK)<-(SK, CQ)<-(DK, CJ)<-(DQ, DK)<-(DK, S7)<-(DK, C0)<-(DJ, DK)<-(DK, S6)<-(SQ, DK)<-
(D7, DK)<-(DK, H7)<-(DK, H0)<-(H8, HK)<-(DK, CJ)<-(HK, S7)<-(C8, DK)<-(H2, HK)<-(S3, DK)<-(HA, CK)<-
(HK, D0)<-(CA, CK)<-(D2, HK)<-(CK, S5)<-(HK, D8)<-(SJ, CK)<-(CK, SQ)<-(S0, DK)
1-st solution, 98 moves. 5436026 states searched.
```

初期局面(1)に対して得られた解答手順

```
(SK, H9)<-(CQ, SK)<-(SK, H8)<-(CJ, SK)<-(SK, H7)<-(C0, SK)<-(SK, H6)<-(C9, SK)<-(C8, HK)<-(SK, H5)<-
(SK, HQ)<-(HK, H4)<-(HK, D4)<-(H3, CK)<-(H2, SK)<-(CK, D3)<-(SK, D2)<-(D4, CK)<-(D3, SK)<-(CK, HJ)<-
(SK, SQ)<-(HA, CK)<-(HK, SJ)<-(CK, DA)<-(DQ, HK)<-(CQ, CK)<-(C7, HK)<-(H9, CK)<-(HK, H3)<-(H8, CK)<-
(CJ, HK)<-(CK, C8)<-(HK, DJ)<-(C6, HK)<-(HK, H2)<-(C0, HK)<-(HK, D0)<-(C5, HK)<-(D9, DK)<-(D8, CK)<-
(DK, S0)<-(CK, S9)<-(DJ, CK)<-(CK, D7)<-(CK, S8)<-(HK, S7)<-(C7, HK)<-(S0, HK)<-(C4, HK)<-(HK, S6)<-
(C6, HK)<-(S9, HK)<-(C3, HK)<-(HK, S5)<-(S4, HK)<-(S3, HK)<-(S2, HK)<-(D2, HK)<-(SA, DK)<-(C2, HK)<-
(S0, DK)<-(HK, S4)<-(DA, CK)<-(CA, CK)<-(C5, HK)<-(H7, HK)<-(C4, DK)<-(HK, C7)<-(H6, DK)<-(H0, HK)<-
(DK, C6)<-(HQ, HK)<-(CJ, DK)<-(C8, HK)<-(HK, S3)<-(C2, DK)<-(DK, H7)<-(CK, CA)<-(H5, DK)<-(CK, C9)<-
(S6, DK)<-(CK, D4)<-(DK, D6)<-(H4, CK)<-(S5, DK)<-(H3, HK)<-(HK, C7)
1-st solution, 87 moves. 1855066 states searched.
```

初期局面(2)に対して得られた解答手順

図7 フルサイズのスーパーパズの初期局面と解答手順の例
Fig.7 Examples of Initial states and solutions for Superpuzz in case of full size

うち 6 割程度は少なくとも一つの成功局面が得られることが示されている。ただし、ゲーム木を全探索する場合にはどのような探索方法を用いても同じ数の局面を探索することになる。そのためには、今回使用したようないわゆる 32 ビットアドレスのパソコンではメモリが不足するので、64 ビットアドレスの計算機（OS,コンパイラ）が必要となる。

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、新しく

作成された子局面が既にゲーム木の中に存在する割合が 6 割以上であることを確かめた。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。実際に列数が 6 のハーフサイズの場合には幅優先探索によっても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も示した。さらに 6 列までの結果を外挿することにより、フルサイズの 13 列の場合には最初の成功局面が得られるまでの平均手数は 100 以上で、必要な探索局面数の平均値は 10^8 のオーダーになることが予測されることを示した。従って、13 列の場合は現在の 32 ビットアドレスのパソコンでは幅優先探索によって解を得ることは難しいが、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。

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] 新谷, スーパーパズを解くプログラム, 第 5 回ゲームプログラミングワークショップ予稿集, 投稿中, 1999
[4] 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