福山大学工学部紀要
第25巻 2001年12月

スーパーパズにおける順位優先探索

新谷 敏朗†

Priority-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 appear in the tree many times. I use a data structure called 'Patricia' so that there is no duplication of nodes in the game tree. I have made clear that 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. In this paper, I found that the number of nodes searched in a priority-first search is less than those in the simple depth-first search in average case. The priority adopted here is decided from the nature of the game that the nearer is a node to the state of success, the more Aces are on the right-most column. It is expected that the number of nodes which will be searched until the first solution is found will be about 10^8 in case of the full-size Superpuzz.

キーワード：スーパーパズ，探索，順位優先，GPCC
Keywords: Superpuzz, Search, Priority-first, GPCC

1．まえがき

「スーパーパズ(Superpuzz)」はCMU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ている。筆者はこれまで、文献 [2]～[4] でスーパーパズの性質について考察を加えてきた。それらによると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」を作っていく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、それを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。よって筆者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。それにより、任意の初期局面を初期状態とする状態空間の全探索を行えば、原理的には解が見つかるかあるいはその初期局面には解が存在しないことが判明するかの２通りの結果のいずれかになる。スーパーパズは本来５２枚のカードをすべて使って４行１３列に配置し

† 情報処理工学科

てプレイするゲームであるが、ルール上たとえばAから6までのカードのみを使用してもプレイが可能である。これを「縮小サイズ」と呼ぶことにすると、列数4から6までの縮小サイズの場合に上記の方法を適用して、本来の13列の場合（フルサイズと呼ぶことにする）を外挿することにより、フルサイズの場合は状態空間中の局面数は平均で 10^9 個程度になることが予想されている。一つの局面の情報を保持するのに数10バイトは必要なので、フルサイズの場合に全探索するには 10^{11} バイト≒100G バイト程度のメモリが必要となる。しかし、現在のパーソナルコンピュータでは、アドレスは32ビット幅なので扱えるメモリは 2^{32} バイト≒4G バイトが上限である。従っていわゆるパソコン上でスーパーパズを解くためには、盲目的に「深さ優先探索」などを用いて全探索するのではなく、ゲームの性質を利用して探索する局面数を減らす工夫をする必要がある。本論文では、このような見地から局面の探索に際して、ゲームの性質を考慮した優先順位を設定することによって、解を見つけるまでに探索すべき局面数がある程度減らせることを、縮小サイズの場合の計算結果から明らかにする。

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。

まえがきでも述べたように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。

図1に6列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、空白の中にH6などと表記している。

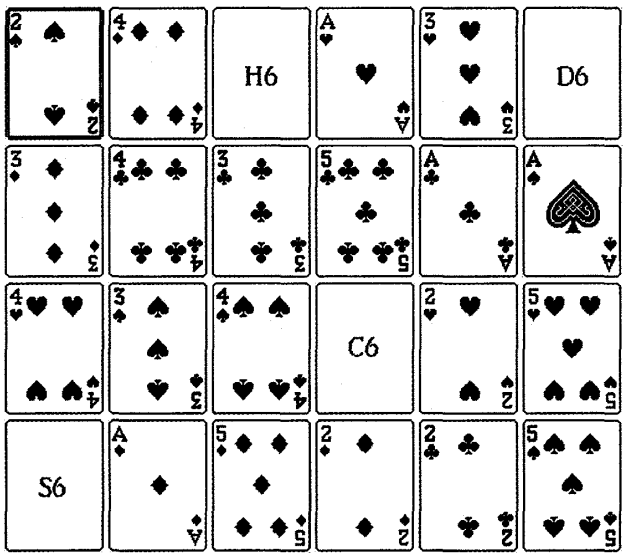


図 1 初期局面の例（縮小サイズ）
Fig.1 An example of initial states

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、筆者の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。

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

図 2 解答手順の例
Fig.2 An example of solution

2.2 ゲーム木と状態遷移図

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。状態遷移図において、節点（局面）を左端にあるエースと空白の数で分類することができる。たとえばエースの数が１で、空白の数が２の場合を、(1,2)要素と呼ぶことにすると、以下のように15個の部分集合（要素成分と呼ぶ）に状態空間が分割される。

(0,0), (0,1), (0,2), (0,3), (0,4)
(1,0), (1,1), (1,2), (1,3)
(2.0), (2,1), (2,2)
(3,0), (3,1)
(4,0)

ルールからすぐわかるように、(i,j)要素の節点はカードの移動によって、(i,j), (i, j+1), (i-1,j)のいずれかの要素になる。よって、状態遷移図は大まかには図３のように書くことができる。

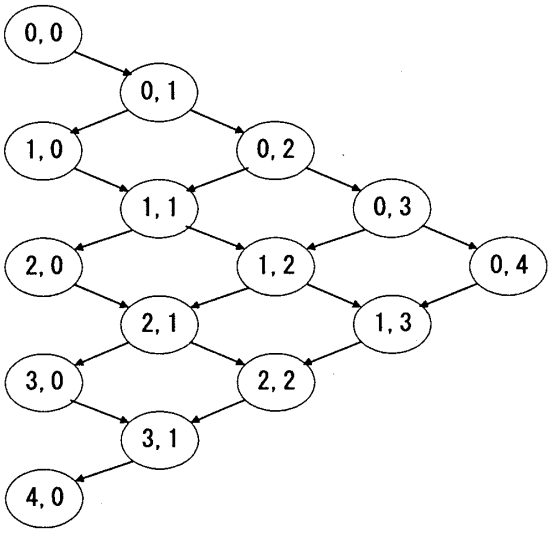


図３　要素成分に着目した状態遷移図
Fig.3 State transition in terms of component

３．ゲーム木の探索と探索の優先順位

スーパーパズを解くアルゴリズムは基本的に文献[2]のものとおなじである。

3.1 探索アルゴリズム

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（成功局面の４行について）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。順位キューをスタックで実現すれば、「深さ優先探索」になり、通常のキューで実現すれば「幅優先探索」になる。

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「ハッシュ」や「平衡木」なども考えられるが、筆者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」[5]を用いている。

3.3 探索における優先順位

ある局面の子孫の局面に成功局面が存在するとき、その局面を「成功可能」と呼び、存在しなければ、「成功不可能」と呼ぶことにする。もし、ある局面でカードの並び方をみて、その局面が「成功不可能」であることが判断できれば、その局面の子孫の局面を枝狩りできる。しかし、現段階ではそのような判断ができるための条件は見いだされていない。従って、ここでは次のように考えて、「成功可能であれば少ない探索局面で成功局面に至る」ことを目標とする。

図３で成功局面はすべて(4,0)要素成分の中にある。よって、できるだけ速く(4,0)要素成分に至るように、(i,j)要素の局面の優先順位を p(i,j)と表したとき、

$$p(i,j) < p(i,j+1) \qquad (1)$$
$$p(i,j) < p(i-1,j) \qquad (2)$$

が成り立つように各要素成分の優先順位を定めることにする。つまり、同じ要素成分の中の状態遷移は優先順位が低く、別の要素成分に速やかに移行するように優先順位を決める。このとき、(i,j)要素から(i,j+1)要素へ（図３で右下への遷移）と(i-1,j)要素への遷移（図３で左下への遷移）のどちらの優先順位を高くするかで２通り試みる。（図４と図５）また、比較の対象として単純な深さ優先の場合を考える。この場合も子局面をスタックにプッシュする際にエースが移動する場合とそうでない場合とどちらを先にプッシュするかで２通り試みる。

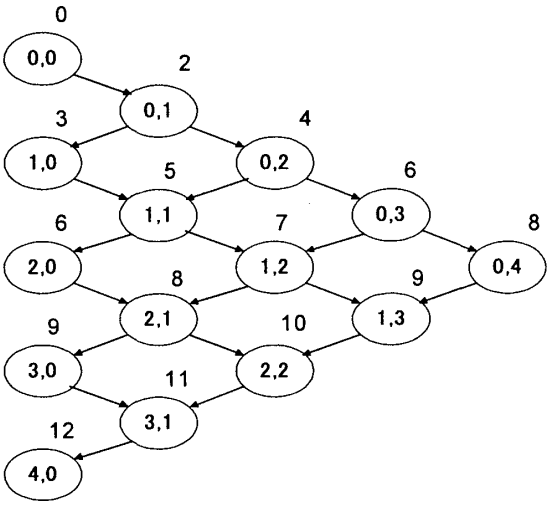


図４ 要素成分に対する優先順位１
Fig.4 Priority for component No.1

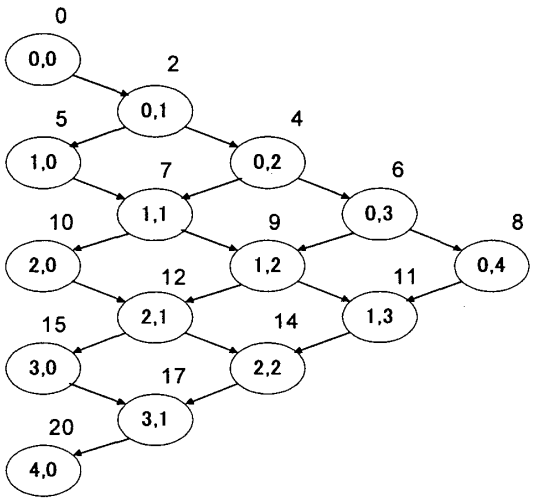


図５ 要素成分に対する優先順位１
Fig.5 Priority for component No.2

４．計算結果

疑似乱数によりシャッフルした初期局面を生成させて、上記の４通りの場合について探索を実行してみた。使用した計算機はPC/AT互換機で、

CPU：PentiumⅢ1GHz，メモリ：2GB，スワップ領域：4Gバイト

OS：FreeBSD 4.2 Release,

コンパイラ： gcc version 2.9.5.2

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、４列から９列までの縮小サイズについて各々100 個の初期局面について実行した。それらをまとめた結果を表１と図６に示した。表１と図６では成功した局面(100 個のうち、全探索して解が存在しなかったものをのぞいた 82～89 個)における平均をとっている。また、そのうち９列の場合に解が探索された例を３例だけ図７～９に示す。図７～９では、初期局面と４つの探索方式のそれぞれの場合における解の手順、探索節点数を示している。

この結果から、単純な深さ優先探索で解を探すよりは、成功局面に近い要素の優先順位を大きくして順位優先探索にした方が、平均的には探索節点数が少ないことがわかる。よって、順位優先探索の有効性が確認できたと考えられる。ただし、各々の初期局面についてみれば、単純な深さ優先探索のほうが速く解を見つ

表１列数と探索節点数の関係
Table1 Relation between number of columns and number of nodes searched

列数	順位優先1	順位優先2	深さ優先1	深さ優先2	成功数	失敗数
4	124	134	416	404	89	11
5	1830	1847	2650	3629	87	13
6	21927	21187	19956	32984	88	12
7	22724	22287	112229	55632	84	16
8	183771	182103	348681	371815	82	18
9	861365	917207	1062466	996120	82	18

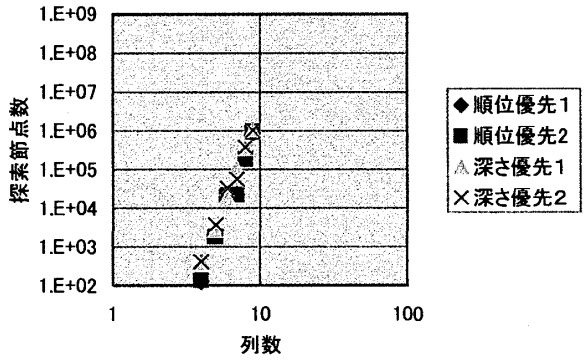
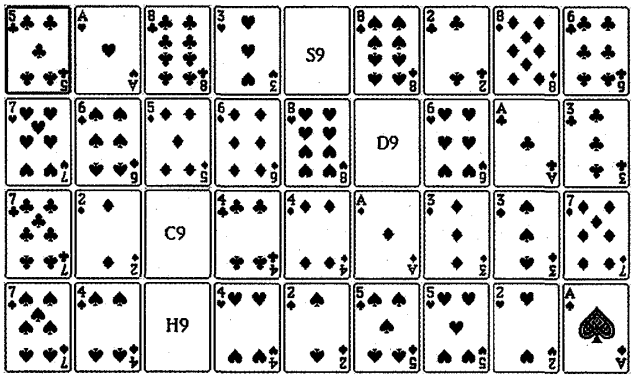


図６ 列数と探索節点数の関係
Fig. Relation between number of columns and number of nodes searched



(C9, H8) <- (C9, C8) <- (H7, D9) <- (H6, C9) <- (D9, C7) <-
(C9, C6) <- (S8, C9) <- (C9, H7) <- (H5, D9) <- (D9, C5) <-
(S9, C4) <- (S7, D9) <- (S6, S9) <- (D8, S9) <- (S9, C3) <-
(S5, S9) <- (D7, S9) <- (S9, C2) <- (S4, S9) <- (D6, S9) <-
(D5, H9) <- (H9, H4) <- (D4, H9) <- (H9, H3) <- (D3, H9) <-
(H9, C8) <- (D4, H9) <- (C4, H9) <- (S9, S8) <- (D9, H6) <-
(H2, D9) <- (D2, D9) <- (D9, HA) <- (D9, CA) <- (S3, S9) <-
(D9, S7) <- (S2, D9) <- (D9, H8) <- (SA, C9) <- (C9, H7) <-
(C9, CA) <- (C9, C7) <- (H9, C6) <- (C3, D9) <- (D9, D8) <-
(S9, D5) <- (D3, C9) <- (D7, S9) <- (H8, S9) <- (S9, S6) <-
(DA, H9) <- (H9, C5) <- (H9, D4) <- (H9, S3) <- (S5, H9) <-
(H4, S9)

No. 1, 56 moves, 9103 states searched.

（図４と図５の優先順位で探索した場合）

(C9, H8) <- (C9, C8) <- (H7, D9) <- (H6, C9) <- (D9, C7) <-
(C9, C6) <- (S8, C9) <- (C9, H7) <- (H5, D9) <- (D9, C5) <-
(S9, C4) <- (S7, D9) <- (S6, S9) <- (D8, S9) <- (S9, C3) <-
(S5, S9) <- (D7, S9) <- (S9, C2) <- (S4, S9) <- (D6, S9) <-
(D5, H9) <- (H9, H4) <- (D4, H9) <- (H9, H3) <- (D3, H9) <-
(H9, C8) <- (D4, H9) <- (C4, H9) <- (S9, S8) <- (D9, H6) <-
(H2, D9) <- (D2, D9) <- (D9, HA) <- (D9, CA) <- (S3, S9) <-
(D9, S7) <- (S2, D9) <- (D9, H8) <- (SA, C9) <- (C9, H7) <-
(C9, CA) <- (C9, C7) <- (H9, C6) <- (C3, D9) <- (D9, D8) <-
(S9, D5) <- (D3, C9) <- (D7, S9) <- (H8, S9) <- (S9, S6) <-
(DA, H9) <- (H9, C5) <- (H9, D4) <- (H9, S3) <- (S5, H9) <-
(H4, S9)

No. 1, 56 moves, 9103 states searched.

（図４と図５の優先順位で探索した場合）

(D9, H8) <- (D9, C8) <- (H7, H9) <- (H6, D9) <- (H9, C7) <-
(D9, C6) <- (S8, D9) <- (H5, H9) <- (H9, C5) <- (S7, H9) <-
(C9, C4) <- (S6, C9) <- (D8, C9) <- (C9, C3) <- (S5, C9) <-
(D7, C9) <- (C9, C2) <- (S4, C9) <- (D6, C9) <- (C9, S8) <-
(S3, C9) <- (D5, C9) <- (C9, H4) <- (D4, C9) <- (C9, H3) <-
(D3, C9) <- (C9, H2) <- (D2, C9) <- (C9, HA) <- (C9, CA) <-
(SA, C9) <- (DA, C9) <- (C9, CA) <- (C9, SA) <- (DA, C9) <-
(C9, SA) <- (C9, DA) <- (C9, S7) <- (S2, C9) <- (C9, D2) <-
(D9, H7) <- (SA, D9) <- (D9, DA) <- (D2, C9) <- (DA, D9) <-
(D4, S9) <- (H9, H6) <- (H2, H9) <- (H9, C8) <- (D3, H9) <-
(C9, D2) <- (D9, DA) <- (D9, C7) <- (D2, D9) <- (D9, H8) <-
(C4, S9) <- (C3, D9) <- (D9, D8) <- (S9, C6) <- (DA, S9) <-
(S9, H7) <- (CA, S9) <- (S9, C5) <- (S9, D4) <- (D7, S9) <-
(H8, S9) <- (S9, S6) <- (H4, S9) <- (H9, D3) <- (H9, S3) <-
(S5, H9) <- (C9, D2) <- (D3, C9)

No. 1, 73 moves, 3680 states searched.

（深さ優先探索その１；エースを後でプッシュした場合）

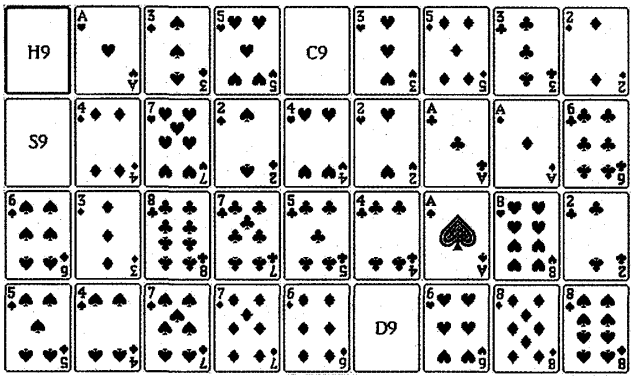
(D9, H8) <- (H7, C9) <- (D9, C8) <- (C9, C7) <- (H6, D9) <-
(H5, C9) <- (D9, C6) <- (C9, C5) <- (S8, D9) <- (S7, C9) <-
(H9, C4) <- (S6, H9) <- (D8, H9) <- (H9, C3) <- (S5, H9) <-
(D7, H9) <- (H9, C2) <- (H9, S8) <- (S4, S9) <- (S3, H9) <-
(D6, S9) <- (D5, H9) <- (H9, H4) <- (D4, H9) <- (H9, H3) <-
(D3, H9) <- (H9, H2) <- (S2, H9) <- (D2, H9) <- (H9, HA) <-
(SA, H9) <- (H9, CA) <- (SA, H9) <- (DA, H9) <- (H9, CA) <-
(H9, SA) <- (DA, H9) <- (SA, H9) <- (H9, DA) <- (H9, S7) <-
(S2, H9) <- (D2, H9) <- (D9, H7) <- (SA, D9) <- (DA, D9) <-
(H9, D2) <- (D9, DA) <- (H9, C8) <- (D4, H9) <- (C4, H9) <-
(D9, C7) <- (H9, C6) <- (D2, D9) <- (DA, H9) <- (C9, H6) <-
(D9, H8) <- (H9, H7) <- (CA, H9) <- (H2, C9) <- (C9, C8) <-
(C9, D2) <- (D3, C9) <- (H9, C5) <- (H9, D4) <- (H9, S3) <-
(S9, S8) <- (S4, S9) <- (C3, D9) <- (D9, D8) <- (D7, S9) <-
(H8, S9) <- (S9, S6) <- (S5, H9) <- (H4, S9)

No.1, 74 moves, 313083 states searched.

（深さ優先探索その２；エースを先にプッシュした場合）

図７　初期局面と解の例（１）

Fig.7 Example No.1 for initial state and solution



(H8, C9) <- (C9, S8) <- (C8, C9) <- (H7, C9) <- (H6, S9) <-
(C9, S7) <- (S9, S6) <- (C9, D8) <- (S9, D7) <- (D8, S9) <-
(C7, S9) <- (S9, S5) <- (S9, D6) <- (D7, S9) <- (C6, S9) <-
(S9, S4) <- (S9, D5) <- (D6, S9) <- (C5, S9) <- (S9, S3) <-

(S9, D4)<-(S8, S9)<-(H5, S9)<-(S9, S5)<-(H6, S9)<-(S9, C4)<-(S9, S2)<-(S9, D3)<-(S7, S9)<-(H4, S9)<-(S9, S4)<-(H5, S9)<-(S9, C3)<-(C2, H9)<-(S9, D5)<-(H9, D4)<-(H9, C8)<-(H8, H9)<-(S9, C7)<-(H9, C6)<-(C4, H9)<-(H9, D2)<-(SA, S9)<-(S6, S9)<-(H3, S9)<-(S9, S3)<-(H4, S9)<-(S9, D5)<-(S9, H7)<-(S9, DA)<-(S5, S9)<-(S4, H9)<-(S9, H6)<-(S8, D9)<-(H9, H5)<-(H2, H9)<-(D8, C9)<-(H6, C9)<-(D9, D7)<-(HA, H9)<-(CA, S9)

No. 1, 61 moves, 138222 states searched.

（図４の優先順位で探索した場合）

(H8, C9)<-(C9, S8)<-(C8, C9)<-(H7, C9)<-(H6, S9)<-(C9, S7)<-(S9, S6)<-(C9, D8)<-(S9, D7)<-(D8, S9)<-(C7, S9)<-(S9, S5)<-(S9, D6)<-(D7, S9)<-(C6, S9)<-(S9, S4)<-(S9, D5)<-(D6, S9)<-(C5, S9)<-(S9, S3)<-(S9, D4)<-(S8, S9)<-(H5, S9)<-(S9, S5)<-(H6, S9)<-(S9, C4)<-(S9, S2)<-(S9, D3)<-(S7, S9)<-(H4, S9)<-(S9, S4)<-(H5, S9)<-(S9, C3)<-(C2, H9)<-(S9, D5)<-(H9, D4)<-(H9, C8)<-(H8, H9)<-(S9, C7)<-(H9, C6)<-(C4, H9)<-(H9, D2)<-(SA, S9)<-(S6, S9)<-(H3, S9)<-(S9, S3)<-(H4, S9)<-(S9, D5)<-(S9, H7)<-(S9, DA)<-(S5, S9)<-(S4, H9)<-(S9, H6)<-(D8, C9)<-(S8, D9)<-(D9, D7)<-(H9, H5)<-(H6, C9)<-(H2, H9)<-(HA, H9)<-(CA, S9)

No. 1, 61 moves, 138621 states searched.

（図５の優先順位で探索した場合）

(H8, S9)<-(S9, S8)<-(C8, S9)<-(H7, S9)<-(H6, C9)<-(S9, S7)<-(S9, D8)<-(C9, S6)<-(C9, D7)<-(D8, S9)<-(C7, S9)<-(S9, S5)<-(S9, D6)<-(D7, S9)<-(C6, S9)<-(S9, S4)<-(S9, D5)<-(D6, S9)<-(C5, S9)<-(S9, S3)<-(S9, D4)<-(S8, S9)<-(H5, S9)<-(S9, S5)<-(H6, S9)<-(S9, C4)<-(S9, S2)<-(S9, D3)<-(S7, S9)<-(H4, S9)<-(S9, S4)<-(H5, S9)<-(S9, C3)<-(S9, D5)<-(C2, H9)<-(H9, D4)<-(H9, C8)<-(S9, C7)<-(H8, H9)<-(H9, C6)<-(C4, H9)<-(H9, D2)<-(S6, H9)<-(H3, H9)<-(H9, S3)<-(H4, H9)<-(H9, D5)<-(H9, H7)<-(H6, C9)<-(CA, C9)<-(C9, SA)<-(C9, DA)<-(HA, C9)<-(C9, SA)<-(C9, DA)<-(C9, HA)<-(DA, C9)<-(C9, HA)<-(S8, D9)<-(SA, C9)<-(C9, DA)<-(HA, C9)<-(DA, C9)<-(C9, HA)<-(D9, D7)<-(SA, C9)<-(DA, C9)<-(C9, SA)<-(DA, C9)<-(HA, C9)<-(C9, SA)<-(C9, DA)<-(HA, C9)<-(DA, C9)<-(C9, HA)<-(S5, C9)<-(SA, S9)<-(S9, DA)<-(HA, S9)<-(DA, H9)<-(H9, HA)<-(H9, S6)<-(S4, H9)<-(H9, H5)<-(H2, H9)<-(HA, H9)

No. 1, 86 moves, 551635 states searched.

（深さ優先探索その１；エースを後でプッシュした場合）

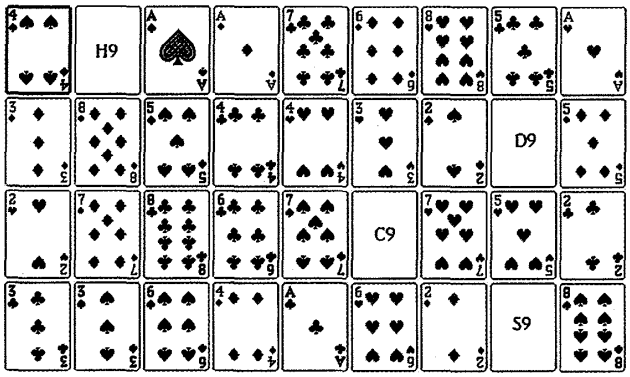
(C9, H8)<-(C8, C9)<-(C9, H7)<-(S9, H6)<-(C9, D8)<-(S9, D7)<-(D8, S9)<-(C7, S9)<-(S5, S9)<-(S9, H5)<-(S9, D6)<-(D7, S9)<-(C6, S9)<-(S4, S9)<-(S9, H4)<-(S9, D5)<-(D6, S9)<-(C5, S9)<-(S3, S9)<-(S9, H3)<-(S9, D4)<-(S8, S9)<-(H5, S9)<-(S9, S5)<-(H6, S9)<-(S9, C4)<-(S2, S9)<-(S9, H2)<-(S9, D3)<-(S7, S9)<-(H4, S9)<-(S9, S4)<-(H5, S9)<-(S9, C3)<-(C2, H9)<-(S9, D5)<-(H9, D4)<-(H9, C8)<-(S9, C7)<-(SA, S9)<-(S9, DA)<-(HA, S9)<-(CA, S9)<-(DA, S9)<-(S9, HA)<-(DA, S9)<-(HA, S9)<-(H8, H9)<-(H9, C6)<-(C4, H9)<-(H9, D2)<-(S9, DA)<-(CA, S9)<-(HA, S9)<-(DA, S9)<-(S9, HA)<-(H2, H9)<-(HA, S9)<-(S9, CA)<-(DA, S9)<-(S9, HA)<-(S9, DA)<-(HA, S9)<-(H9, H2)<-(S9, HA)<-(S6, H9)<-(H3, H9)<-(H9, S3)<-(H4, H9)<-(H9, D5)<-(H9, H7)<-(S5, S9)<-(S9, H6)<-(CA, S9)<-(S9, DA)<-(HA, S9)<-(DA, H9)<-(H9, HA)<-(H9, S6)<-(S4, H9)<-(H9, H5)<-(H2, H9)<-(HA, H9)<-(D8, C9)<-(H6, C9)<-(S8, D9)<-(D9, D7)

No. 1, 87 moves, 2917 states searched.

（深さ優先探索その２；エースを先にプッシュした場合）

図８　初期局面と解の例（２）

Fig.8 Example No.2 for initial state and solution



(S8, H9)<-(H9, H8)<-(D8, H9)<-(D9, S7)<-(H9, S6)<-(C8, H9)<-(D9, H7)<-(S9, H6)<-(H9, H5)<-(D7, D9)<-(D6, S9)<-(D5, H9)<-(D9, S5)<-(S9, S4)<-(H9, S3)<-(H9, D8)<-(C7, D9)<-(D9, H4)<-(D4, D9)<-(D3, H9)<-(D9, S2)<-(D9, D7)<-(D2, D9)<-(C6, S9)<-(S9, H3)<-(D9, H2)<-(C5, S9)<-(C4, D9)<-(S7, S9)<-(S6, D9)<-(H9, SA)<-(DA, H9)<-(H9, HA)<-(C3, H9)<-(S5, H9)<-(H9, H8)<-(H6, C9)<-(H5, H9)<-(S4, C9)<-(S3, H9)<-(C9, D6)<-(H6, C9)<-(C2, C9)<-(C9, D8)<-(C9, C8)<-(H9, C7)<-(S7, H9)<-(H9, H4)<-(S2, H9)<-(D9, C5)<-(S6, D9)<-(D9, C4)<-(H9, DA)<-(H2, H9)<-(HA, S9)<-(S9, S4)<-(S3, D9)<-(S9, D5)<-(CA, S9)<-(S9, D3)

No. 1, 60 moves, 3793 states searched.
（図４と図５の優先順位で探索した場合）

(H6, D9) <- (D9, C8) <- (D5, C9) <- (D4, H9) <- (D3, D9) <-
(C9, H5) <- (H9, H4) <- (D9, H3) <- (C9, C7) <- (S8, C9) <-
(H5, C9) <- (H9, C6) <- (C9, C5) <- (S7, H9) <- (S6, C9) <-
(H4, H9) <- (H9, C4) <- (S5, H9) <- (C5, H9) <- (D2, H9) <-
(H9, H2) <- (H3, C9) <- (C9, C3) <- (S4, C9) <- (S3, D9) <-
(C4, C9) <- (DA, C9) <- (C9, HA) <- (H2, H9) <- (H9, S2) <-
(SA, C9) <- (HA, C9) <- (C9, SA) <- (HA, C9) <- (H9, H2) <-
(C9, HA) <- (S2, H9) <- (H9, H2) <- (HA, C9) <- (H9, C2) <-
(C9, CA) <- (SA, C9) <- (C9, HA) <- (H2, H9) <- (C9, SA) <-
(HA, C9) <- (H9, H2) <- (S2, H9) <- (C9, SA) <- (C9, HA) <-
(SA, C9) <- (C9, HA) <- (H2, H9) <- (HA, C9) <- (H9, C2) <-
(H9, H2) <- (H9, S2) <- (C3, D9) <- (C9, SA) <- (HA, C9) <-
(SA, C9) <- (S2, H9) <- (C9, CA) <- (C9, SA) <- (C9, HA) <-
(SA, C9) <- (C9, HA) <- (H2, H9) <- (HA, C9) <- (C2, H9) <-
(H9, H2) <- (C9, HA) <- (H9, S2) <- (C9, SA) <- (HA, C9) <-
(C9, SA) <- (C9, HA) <- (CA, C9) <- (C9, HA) <- (H2, H9) <-
(HA, C9) <- (C9, CA) <- (SA, C9) <- (C9, HA) <- (C9, SA) <-
(HA, C9) <- (CA, C9) <- (H9, C2) <- (C9, SA) <- (C9, HA) <-
(D9, S3) <- (CA, C9) <- (C9, HA) <- (C9, CA) <- (HA, C9) <-
(H9, H2) <- (C9, HA) <- (SA, C9) <- (C9, HA) <- (C9, CA) <-
(C9, SA) <- (S2, H9) <- (HA, C9) <- (C9, SA) <- (C9, HA) <-
(CA, C9) <- (C9, HA) <- (C9, CA) <- (SA, C9) <- (C9, HA) <-
(C9, SA) <- (HA, C9) <- (C9, CA) <- (C9, HA) <- (SA, C9) <-
(H9, S2) <- (C9, SA) <- (C2, H9) <- (HA, C9) <- (SA, C9) <-
(H9, S2) <- (C9, HA) <- (CA, C9) <- (C9, SA) <- (H2, H9) <-
(S2, H9) <- (SA, C9) <- (H9, C2) <- (C9, HA) <- (C9, SA) <-
(S2, H9) <- (HA, C9) <- (C9, CA) <- (SA, C9) <- (C2, H9) <-
(H9, S2) <- (H2, H9) <- (C9, HA) <- (CA, C9) <- (C9, SA) <-
(CA, C9) <- (S2, H9) <- (SA, C9) <- (C9, HA) <- (H9, S2) <-
(C9, CA) <- (SA, C9) <- (C2, H9) <- (HA, C9) <- (C9, SA) <-
(S2, H9) <- (HA, C9) <- (CA, C9) <- (SA, C9) <- (H9, H2) <-
(S2, H9) <- (C9, SA) <- (H9, C2) <- (C9, HA) <- (SA, C9) <-
(H9, S2) <- (C9, SA) <- (CA, C9) <- (SA, C9) <- (C9, HA) <-
(C9, CA) <- (SA, C9) <- (H2, H9) <- (HA, C9) <- (C9, SA) <-
(HA, C9) <- (H9, H2) <- (C9, HA) <- (C2, H9) <- (CA, C9) <-
(SA, C9) <- (C9, HA) <- (H2, H9) <- (SA, C9) <- (HA, C9) <-
(H9, H2) <- (H9, S2) <- (C9, SA) <- (HA, C9) <- (SA, C9) <-
(C9, HA) <- (H2, H9) <- (HA, C9) <- (H9, S2) <- (H9, H2) <-
(C9, HA) <- (H9, C2) <- (C9, CA) <- (SA, C9) <- (HA, C9) <-
(H2, H9) <- (C9, SA) <- (HA, C9) <- (H9, H2) <- (S2, H9) <-
(SA, C9) <- (C9, HA) <- (SA, C9) <- (HA, C9) <- (H9, H2) <-
(C9, HA) <- (H9, S2) <- (H2, H9) <- (HA, C9) <- (H9, C2) <-
(C9, CA) <- (C9, SA) <- (HA, C9) <- (H9, H2) <- (C9, SA) <-
(C9, HA) <- (H2, H9) <- (S2, H9) <- (SA, C9) <- (C9, HA) <-
(C9, SA) <- (HA, C9) <- (H9, H2) <- (C9, HA) <- (S2, H9) <-
(H2, H9) <- (HA, C9) <- (C2, H9) <- (CA, C9) <- (C9, SA) <-
(C9, HA) <- (H9, H2) <- (SA, C9) <- (C9, HA) <- (H2, H9) <-
(H9, S2) <- (C9, SA) <- (HA, C9) <- (C9, SA) <- (C9, HA) <-
(H2, H9) <- (HA, C9) <- (C9, H2) <- (S9, HA) <- (D8, D9) <-
(D7, H9) <- (H9, H8) <- (H5, H9) <- (S3, H9) <- (S2, C9) <-
(SA, S9) <- (D9, C5) <- (D9, H4) <- (D9, S2) <- (C4, D9) <-
(H9, C3) <- (C9, C2) <- (S9, CA) <- (S9, SA) <- (C3, S9) <-
(C9, D8) <- (C9, C8) <- (C9, S8) <- (D2, C9) <- (C9, C7) <-
(C9, S7) <- (S9, S6) <- (H6, C9) <- (C2, C9) <- (CA, S9) <-
(D9, C4) <- (S3, D9) <- (C9, S5) <- (S9, S4) <- (S6, S9) <-
(S9, SA) <- (C9, D4) <- (S9, D3) <- (S8, C9) <- (S5, H9)

No. 1, 280 moves, 614385 states searched.
（深さ優先探索その１；エースを後でプッシュした場合）

(C9, H6) <- (C8, C9) <- (C9, H5) <- (C7, C9) <- (C9, H4) <-
(D4, S9) <- (C6, S9) <- (S8, S9) <- (S7, D9) <- (S6, S9) <-
(S5, D9) <- (S4, S9) <- (S3, D9) <- (S9, C4) <- (D9, C3) <-
(D9, D3) <- (H3, D9) <- (D9, D8) <- (C4, S9) <- (S2, S9) <-
(S9, C2) <- (S9, D2) <- (D3, D9) <- (C3, D9) <- (SA, D9) <-
(H2, S9) <- (D9, HA) <- (D9, DA) <- (CA, D9) <- (D9, DA) <-
(D9, CA) <- (HA, D9) <- (D2, S9) <- (D9, DA) <- (S9, H2) <-
(D9, HA) <- (DA, D9) <- (D9, HA) <- (D9, CA) <- (D9, DA) <-
(D2, S9) <- (HA, D9) <- (D9, DA) <- (D9, HA) <- (CA, D9) <-
(D9, HA) <- (D9, CA) <- (DA, D9) <- (D9, HA) <- (D9, DA) <-
(HA, D9) <- (D9, CA) <- (D9, HA) <- (DA, D9) <- (S9, D2) <-
(D9, DA) <- (C2, S9) <- (HA, D9) <- (DA, D9) <- (S9, D2) <-
(D9, HA) <- (CA, D9) <- (D9, DA) <- (H2, S9) <- (D2, S9) <-
(DA, D9) <- (S9, C2) <- (D9, HA) <- (D9, DA) <- (D2, S9) <-
(HA, D9) <- (D9, CA) <- (DA, D9) <- (C2, S9) <- (S9, D2) <-
(H2, S9) <- (D9, HA) <- (CA, D9) <- (D9, DA) <- (CA, D9) <-
(D2, S9) <- (DA, D9) <- (D9, HA) <- (S9, D2) <- (D9, CA) <-
(DA, D9) <- (C2, S9) <- (HA, D9) <- (D9, DA) <- (D2, S9) <-
(HA, D9) <- (CA, D9) <- (DA, D9) <- (S9, H2) <- (D2, S9) <-
(D9, DA) <- (S9, C2) <- (D9, HA) <- (DA, D9) <- (S9, D2) <-
(D9, DA) <- (CA, D9) <- (DA, D9) <- (D9, HA) <- (D9, CA) <-
(DA, D9) <- (H2, S9) <- (HA, D9) <- (D9, DA) <- (HA, D9) <-
(S9, H2) <- (D9, HA) <- (C2, S9) <- (CA, D9) <- (DA, D9) <-
(D9, HA) <- (H2, S9) <- (DA, D9) <- (HA, D9) <- (S9, H2) <-
(S9, D2) <- (D9, DA) <- (HA, D9) <- (DA, D9) <- (D9, HA) <-
(H2, S9) <- (HA, D9) <- (S9, D2) <- (S9, H2) <- (D9, HA) <-
(S9, C2) <- (D9, CA) <- (DA, D9) <- (HA, D9) <- (H2, S9) <-
(D9, DA) <- (HA, D9) <- (S9, H2) <- (D2, S9) <- (DA, D9) <-
(D9, HA) <- (DA, D9) <- (HA, D9) <- (S9, H2) <- (D9, HA) <-
(S9, D2) <- (H2, S9) <- (HA, D9) <- (S9, C2) <- (D9, CA) <-
(D9, DA) <- (HA, D9) <- (S9, H2) <- (D9, DA) <- (D9, HA) <-
(H2, S9) <- (D2, S9) <- (DA, D9) <- (D9, HA) <- (D9, DA) <-
(HA, D9) <- (S9, H2) <- (D9, HA) <- (D2, S9) <- (H2, S9) <-

(HA, D9)<-(C2, S9)<-(CA, D9)<-(D9, DA)<-(D9, HA)<-
(S9, H2)<-(DA, D9)<-(D9, HA)<-(H2, S9)<-(S9, D2)<-
(D9, DA)<-(HA, D9)<-(D9, DA)<-(D9, HA)<-(H2, S9)<-
(HA, D9)<-(S9, H3)<-(D9, H2)<-(C5, S9)<-(S7, S9)<-
(S9, HA)<-(H6, H9)<-(C4, D9)<-(S6, D9)<-(C3, S9)<-
(S5, S9)<-(S4, H9)<-(H9, C8)<-(S8, H9)<-(H9, H3)<-
(H9, D7)<-(D2, H9)<-(C9, C6)<-(S9, H8)<-(H5, S9)<-
(S3, S9)<-(S9, C7)<-(S7, S9)<-(S9, H4)<-(S2, S9)<-
(DA, S9)<-(H9, C2)<-(S9, CA)<-(H9, D8)<-(C8, H9)<-
(H9, D6)<-(H6, H9)<-(C2, H9)<-(CA, S9)<-(C9, D4)<-
(S9, S4)<-(S9, D5)<-(CA, S9)<-(S8, C9)<-(D9, C5)<-
(S6, D9)<-(D9, C4)<-(S3, D9)<-(S9, D3)

No. 1, 224 moves, 21834 states searched.

（深さ優先探索その２；エースを先にプッシュした場合）

図９　初期局面と解の例（３）

Fig.9 Example No.2 for initial state and solution

けている場合がある。図７，８の例も単純な深さ優先探索のどちらかが順位優先探索を用いた場合よりも最初の解を見いだすまでの探索節点数が少なくなっている。なお、図７と図９のように図４と図５の優先順位のどちらでも同じ探索節点数で同じ解を見つけている例も多かった。表１を見ても２通りの順位優先探索についてはあまり差がないことがわかる。

また、図６を外挿してみると、本論文で採用された順位優先探索によってもフルサイズの 13 列の場合には最初の成功局面までの探索局面数は平均 10^8 のオーダー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（32 ビットアドレス）のパソコンではフルサイズの場合をすべて解くことは依然として難しいと考えられる。

５．あとがき

本論文では、スーパーパズを解くために単純な深さ優先探索ではなく、局面の左端の列に存在するエースと穴の数に応じた優先度を用いた順位優先探索を行うことにより、解が見つかるまでの探索節点数を平均的にはかなり減らすことが可能であることを、縮小サイズの場合の計算結果から示した。ただし、特定の局面についてみた場合は、順位優先探索を採用することによって逆に探索節点数が増える場合もあることが明らかになった。また、２通り試みた優先順位の設定についてはあまり差がないこともわかった。

これらの結果から、今後の課題として個々の局面に対する優先度の設定に関しては改良の余地がかなりあると考えられる。理想的には、「成功不可能な」局面に対して一番低い優先順位を与える（すなわち、子局面をつくらない）ことが、探索節点数を減らすために最も効果がある。よって、任意の局面が「成功不可能」かどうかを判定する条件を今後考察していくことが必要である。しかし、単純な判定条件は存在せず、結局全探索するのと同じ程度の手間を必要とする可能性もあるので、本論文の考え方のように「解が存在する場合にできるだけ少ない探索節点数で解を見いだす」アプローチも意義があると考えられる。

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] 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論文集，IPSJ Symposium Series Vol99, No.14, pp84-91(1999)

[3] 新谷，スーパーパズにおける巾優先探索，福山大学工学部紀要，Vol.23，pp95-102(1999)

[4] 新谷，スーパーパズの状態遷移に関する考察，情報処理学会研究報告，2000-GI-3，pp41-48(2000)

[5] 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