

## Heptamond 問題の全解数推定

是川 空, 五十嵐 力, 但馬 康宏, 小谷 善行  
東京農工大学 工学府 情報工学専攻

はめ込みパズルの一種である Heptamond 問題は膨大な分岐数と解局面を持つことで知られている。この問題について未知数である全解数の推定を行った。探索を高速化する手法として、複数の分岐の可能性から最少の分岐数を選択して探索を行うアルゴリズムを用いた。各探索深さにおける選択された最少の分岐数の平均が、その探索深さの局面数の変化率に値することに着目し、各深さにおける平均最少分岐数を得るための実験を行った。一つ目の実験は探索を一定確率で打ち切るシミュレーション実験、もう一つは探索深さに閾値を設けた全探索を行った。この実験によって、平均最少分岐数の推定を行い、Heptamond 問題の各深さにおける総局面数を求め、全解数がおおよそ  $10^{11}$  であると推定した。

### Estimating the Number of Heptamond Problem Solutions

Takashi Korekawa, Chikara Igarashi, Yasuhiro Tajima, Yoshiyuki Kotani

Department of Computer, Information Sciences  
Tokyo University of Agriculture and Technology

Heptamond is one of packing puzzles, which has great number of branching factors and solutions. We estimate the number of heptamond problem solutions. In order to solve this problem efficiently, our algorithm selects a cell of minimum branching factor, then it reduces the amount of nodes in a search tree. Average of selected branching factor can be used for the estimated rate of change for the number of situation, between now and next depth at each depth. We estimate this rate by random cutting with some constant probability, and count branching factor at all node with less than 8 depths. By this experimentation, we estimated number of heptamond problem solutions is about  $10^{11}$ .

#### 1. Heptamond 問題

##### 1. 1 Polyamond

複数の正三角形を辺と辺で接合した形を Polyamond と言う。構成される正三角形の数によって呼び方が異なり、正三角形 7 つで構成されるものの総称を Heptamond と言う。Heptamond は回転や反転によってできる同じ形を取り除くと 24 種類存在する。(図 1)

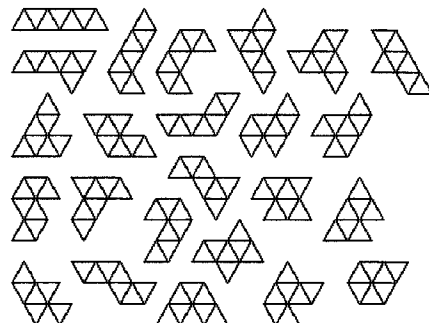


図 1 Heptamond

## 1. 2 Heptamond 問題のルール

Heptamond の 24 種類の形をブロックピースとして、ある形の枠内に全てのピースを隙間無くはめ込むパズルを Heptamond 問題と言う。ピースはそれぞれを回転反転してはめ込むことができるが、いくつかのピースが重なり合ったり、枠からはみ出たりするようにはめ込んではいけない。枠の形は無数に考えられるが、市販されている Heptamond 問題で採用され、もっとも解の形が美しいとされる図 2 の形の枠について考える。

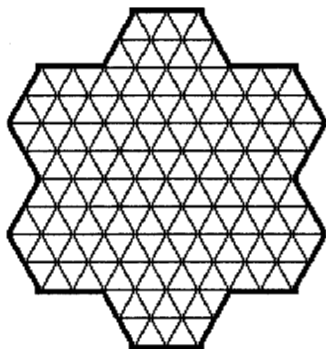


図 2 Heptamond 問題の枠の形

## 1. 3 Heptamond 問題の探索空間

Heptamond 問題のピースの状態を局面とすると、ピースを一つはめるという操作による状態の変化によって、局面が分岐していく。局面が一つ進むごとにピースが一つはめられることになる。

Heptamond 問題の局面数は膨大で、全解探索は非常に困難であり、また全解数は未知数である。

## 1. 4 本稿の概要

Heptamond 問題の膨大な全解数に対して複数の分岐候補の中から現局面での最少の分岐数を持つものを選択して探索を行うアルゴリズム提案する。この手法が通常の探索よりも分岐数の分散を抑え

ることができることを利用し、各深さでの総分岐数を局面の変化量として考え、深さ毎の局面数を求めることで全解数の推定を行う。

## 2. 探索アルゴリズム

### 2. 1 バックトラック探索

Heptamond 問題はピース 24 種類に対して各ピースの反転回転によって作られる形が 12 種類あることから、局面の分岐が多く探索量が膨大になる。この局面を高速に探索するために深さ優先の探索を行う。

ある局面に対してピースを一つはめるという操作を行うにあたって、「①どのピースをどの方向で」「②枠内のどこに」はめ込むかを決める必要がある。どちらも選択することで局面が分岐するが、両方について全ての通りを探索することは冗長である。このアルゴリズムでは枠内の三角形マスの一つを探索マスとして先に②を決定する。その後に①について探索マスを埋まった状態にし、かつ Heptamond のルールを満たしたはめ込み方を全て列挙し次の局面への分岐とする。

探索マスを決定し局面の分岐を行った後に、実際のピースの操作を行う。その局面に与えられた分岐のうち一つを選択し、それにあたるピースの操作を行う。もし局面に分岐が存在しないか、全ての分岐が探索済みであった場合、最後に入れたピースを外す操作を行って局面をバックトラックする。これらを局面に対して再帰的に繰り返すことで解局面の探索を行う。全てのピースがはまった状態が解である。

## 2. 2 探索マスの決定方法

探索において局面の分岐数を抑えるような探索順に変更することは全体の局面における大きな探索量の減少をもたらす。Heptamond 問題では、2. 1 節で示したように一つの分岐に際して探索マスを決定することになる。探索中のある局面について、枠内に残った空きマスの周辺の状態は各マスによって異なり、そのマスを埋めるためにピースを置く方法の数はマスの周囲の状態に大きく依存する。ここで探索マスの決定の際に残りの空きマスからもっとも分岐数の少ないものを選択する。この手法は序盤から分岐数を少なくすることができ、探索量の減少に大きな効果をもたらす。

新たな局面を探索する際に、探索マスが未定である局面の各空きマスが探索マスになった場合の分岐数をカウントし、分岐数が最少のものを選択することで探索マスを選択する。この決定方法では、副次的に分岐数がゼロであるマスを見つけることができるため、そのような局面の探索を打ち切ることができる枝刈り効果がある。

未探索の局面で各空きマスに対して次の分岐数を数え上げるため、探索1ノードあたりの計算量は大きくなるが、分岐数の大きな減少が見込めるため、全体の計算量は大きく減少する。(図3)

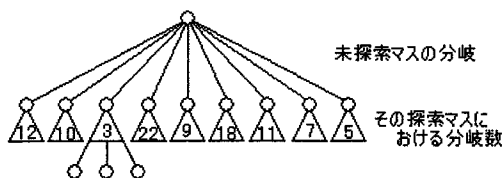


図3 探索マスの決定による探索木の変化

## 2. 3 その他の枝刈り

残りの空きマスが2つ以上の領域に分断された場合、残りの領域を埋めるには全ての分散された領域に複数個のピースが隙間無く入る必要がある。局面の分岐を考慮する際に残った領域の一つの大きさをカウントし、それがピースを構成する三角形の数の倍数であるかをチェックする。この条件を満たさない場合はその後の探索を続けても解局面に至ることはありえないので探索を打ち切る。

以上の手法を盛り込んだ探索アルゴリズムのフローチャートを図4に示す。

## 3. Heptamond 問題の全解数推定法

### 3. 1 Heptamond 問題の局面数推定

Heptamond 問題において、一部の枝を元に全解局面数を推定するのは非常に難しい。枝ごとの局面の形は多様で、その局面数は平均値に対して非常に分散が大きくなる。そこで、探索途中で局面数の推移を推定に用いるのではなく、局面における分岐数を元に局面数を推定する。

この探索手法は、探索する順番が探索マスをベースに考えられている。探索マスにおける分岐数は主に残りのピース数に依存していて、また奥まったマスが探索マスになるときの多く、このとき分岐数が非常に小さい状態になる。

残りのピースが多い序盤の局面では、ピースの入れ方も多種にわたるため、この手法でも分散が大きくなるが、終盤になるにつれて、分岐数の可能性が大幅に少なくなるので、ピースが多くはまるにつれて分岐数の分散が小さくなる。

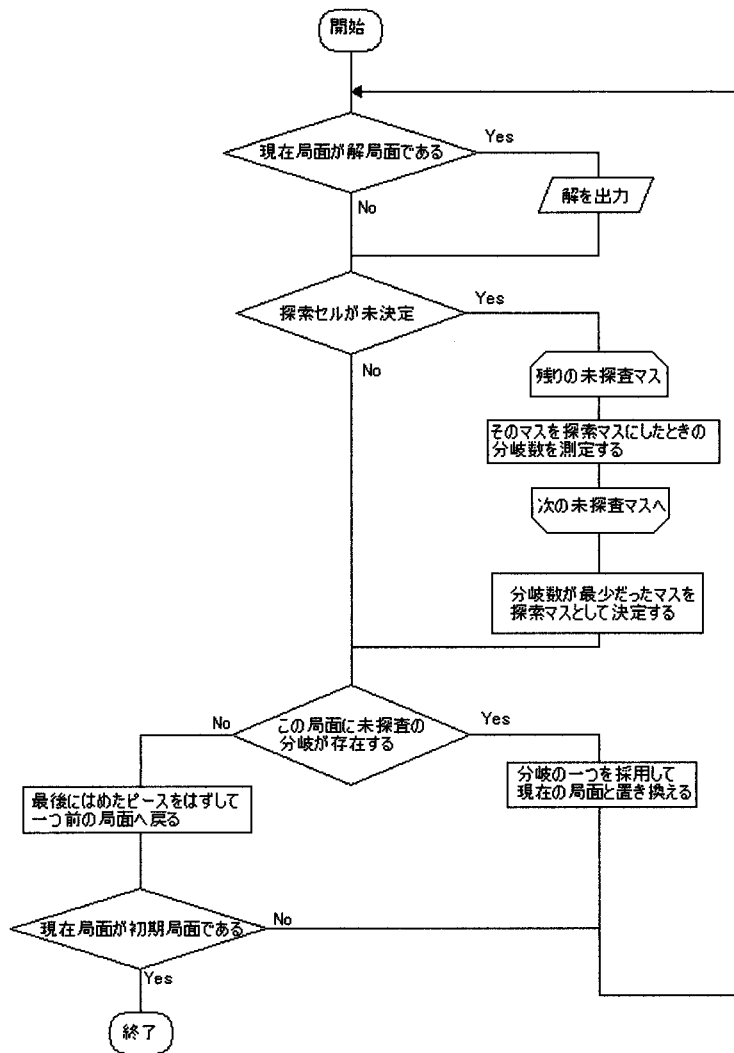


図4 アルゴリズムのフローチャート

### 3. 2 平均最少分岐数と局面数

Heptamond 問題のある探索深さにおける全ての局面の総分岐数は、次の深さにおける局面数に等しい。そのため各探索深さにおける局面の平均的分岐数は局面数の変化量に値する。これを利用し、各深さにおける分岐数の平均値を観察し、局面数の変化を推測する。

各深さにおける分岐数には残りのピース数が関係している。深さが一つ増える

ごとにピースが一つ使用済みになるために全てのマスにおいて分岐数は同じか小さくなる。また探索マスの決定に際して分岐数を常に小さく抑えることで分岐数の分散を抑えている。

#### 4. Heptamond 問題推定のための実験

##### 4. 1 確率による打ち切りを利用したシミュレーション

###### 4. 1. 1 実験方法

Heptamond 問題の全解探索を行う際に、各探索ノードにおいて一定の確率で探索を打ち切る。これによって、全ての探索空間の中から一部分の探索木をサンプルとして得ることができる。この探索で得られる木の各ノードにおける次の局面への分岐数を測定して、局面の深さ毎に分析する。

探索を打ち切る確率は、ある程度の時間内に 1 回の実験が終了し、多くのサンプルを得ることが望ましい。今回の実験では、局面の深さが 2 以下、または 20 以上の局面では探索の打ち切られる確率を 0% に設定し、それ以外の深さでは 70% に設定した。

#### 4. 1. 2 実験結果

4. 1. 1 の条件で 20 回の実験を行い、それぞれの深さにおける分岐数のサンプルを得た。1 度の実験ごとに平均最少分岐数を求め、その平均値を得た (表 a)

表 a 探索深さごとの最少分岐数の平均値

探索深さ	サンプル数	平均最少分岐数	標準偏差
1	1,785	15.90588	16.65330
2	7,558	17.26912	15.82202
3	35,003	14.68797	14.39378
4	136,812	11.56250	11.99969
5	419,965	9.75265	10.69441
6	1,087,891	9.09476	10.14039
7	2,627,538	8.52066	9.32693
8	5,944,777	7.58927	8.36218
9	11,976,866	6.64457	7.50520
10	21,125,941	5.68746	6.53901
11	31,908,840	4.73925	5.61250
12	40,153,097	4.07848	5.00182
13	43,474,112	3.67676	4.49891
14	42,426,575	3.11779	3.79206
15	35,115,853	2.42429	3.12522
16	22,601,406	1.99499	2.75211
17	11,972,325	1.76591	2.40203
18	5,613,338	1.31149	1.79945
19	1,955,107	0.82094	1.31211
20	1,605,034	0.58956	1.05913
21	946,269	0.40004	0.74043
22	378,550	0.11686	0.36033
23	44,237	0.05556	0.22908
24	2,458	0.00000	0.00000

#### 4. 2 閾値を設定した全探索

##### 4. 2. 1 実験方法

Heptamond 問題の全深さにわたる全解探索を行うことはその膨大な計算量から不可能だが、ある程度の深さまでなら全探索が可能である。序盤の局面は分岐数においても非常に分散が大きく、シミュレーションによって正しい値を得ることが難しい。また、今回の推定方法において、序盤の結果が正確な値を示していないと、全体の局面数を求める際に非常に大きな誤差を生むことになる。

そこで、上記の表 a がどの程度予想値

として使えるかを推定するために、ある探索深さに到達した場合探索を打ち切ることで、その深さまでの全ての探索木を得る。これによってその深さまでの局面の正確な変化量を得る。実時間内に終了するできるだけ大きな探索深さとして、探索深さ 8 に到達した局面の全ての探索をそこで打ち切り、その深さまでの全てのノードにおける分岐数を測定する。

#### 4. 2. 2 実験結果

閾値を探索深さ 8 に設定し、その深さまでの全局面数を調べた。表 b に局面数とその比、すなわち平均最少分岐数を示す。

表 b 閾値 8 までの全探索の結果

探索深さ	その深さの局面数	平均最少分岐数
0	1	85.00000
1	85	15.75294
2	1,339	17.66094
3	23,648	14.59032
4	345,032	11.44923
5	3,950,350	9.78742
6	38,663,744	9.02098
7	348,784,683	8.48040
8	2,957,832,611	

#### 5. 考察

表 b の結果より、表 a の数値は平均最少分岐の値とほぼ一致する。表 b の数値の方が全探索による実測値であるので、探索深さ 8 までの局面数はこれを使用する。残りの深さの部分に対して、シミュレーションの結果得られた平均最少分岐数を用いると、平均最少分岐数の推移は図 5 のような曲線を描くことが予想される。

図 5 より、探索深さ 2 の数値に序盤特有の外れ値が見られるほかは、平均最少分岐数は常に減少している。特に探索深さ 18 から 19 の間に平均最少分岐数は 1 以下になる。平均最少分岐数が 1 以下になった後の探索では局面数は減少することになるため、この深さでの探索量が全体の探索量に大きくかかわっていることがわかる。

この曲線を元に各探索深さにおける総局面数を推定した。この結果によると Heptamond 問題における探索深さ 24 の局面数はおよそ  $2.65 \times 10^{12}$  であることが推定される。

Heptamond 問題は局面を枠ごと回転反転した場合に全てのピースが一致する類解が存在する。普通全解数とはこれらの類解を取り除いた局面数を指す。一つの解について、回転反転した解の形は 12 通り存在するから、実際の全解数は  $2.21 \times 10^{11}$  になる。

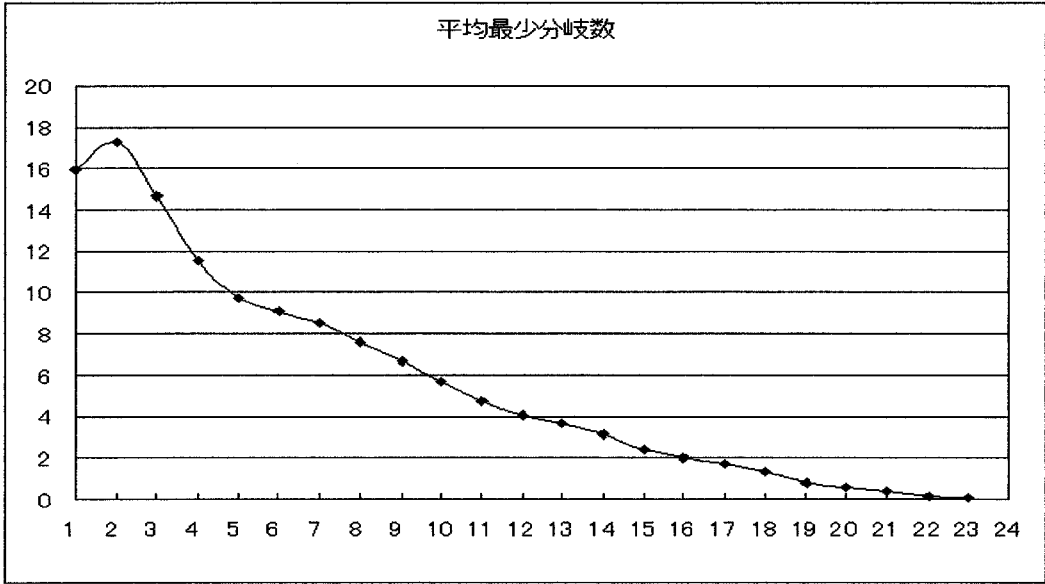


図5 平均最少分岐数の予想曲線

表c Heptamond 問題の推定局面数

探索深さ	平均最少分岐数	その深さの局面数	log(局面数)
0	85.00000	1	0.00
1	15.90588	85	1.93
2	17.26912	1,339	3.13
3	14.68797	23,648	4.37
4	11.56250	345,032	5.54
5	9.75265	3,950,350	6.60
6	9.09476	38,663,744	7.59
7	8.52066	348,784,683	8.54
8	7.58927	2,957,832,611	9.47
9	6.64457	22,447,776,496	10.35
10	5.68746	149,155,716,345	11.17
11	4.73925	848,316,496,929	11.93
12	4.07848	4,020,386,697,188	12.60
13	3.67676	16,397,082,651,589	13.21
14	3.11779	60,288,057,636,251	13.78
15	2.42429	187,965,322,368,439	14.27
16	1.99499	455,682,824,613,109	14.66
17	1.76591	909,082,880,780,256	14.96
18	1.31149	1,605,359,883,308,070	15.21
19	0.82094	2,105,406,668,655,240	15.32
20	0.58956	1,728,421,660,307,280	15.24
21	0.40004	1,019,013,825,300,470	15.01
22	0.11686	407,651,189,637,927	14.61
23	0.05556	47,637,737,884,066	13.68
24		2,646,959,778,444	12.42

## 6. まとめ

Heptamond 問題の全解数を推定するために、まず問題の解探索全体の探索量を減少させるために、複数の分岐候補から最少的分岐数を持つものを局面の分岐として選択するアルゴリズムを設計した。

この手法によって探索深さごとの平均最少分岐数を得たとき、その値が次の探索深さとの局面数の変化率に相当することに着目し、各探索深さにおける平均最少分岐数を得るための2つの実験を行った。

確率による探索打ち切りを利用したシミュレーションは終盤の分岐数の分散が少ない部分に、閾値を用いた全探索では局面数の少ない序盤について正確な値を示すことができるため、この2つを組み合わせることで Heptamond 問題の平均最少分岐数を推定した。

これらの結果より、Heptamond 問題の全解数は  $2.21 \times 10^{11}$  である。

## 参考文献

[1] 是川 空,五十嵐 力,柴原 一友,但馬 康宏,小谷 善行:heptamond 問題における全解探索のための手法,第10回ゲームプログラミングワークショップ2005 ,pages 122-125,2005.

[2] Chikara Igarashi, Yasuhiro Tajima, Nobuo Inui and Yoshiyuki Kotani: "A logic Puzzle Solver by Selecting Smallest Branching Factor", IPSJ-SIG Technical Report, 2005-GI-14, pages 41-45,2005.