デジタル・モデュロール２

○渡辺　俊 *1

キーワード：ル・コルビュジエ、羽目板遊び、Python、浮動小数、順列・組み合わせ

１．はじめに

前報 [1] では、ル・コルビュジエ著のモデュロール [2] に掲載された６つの羽目板遊びのうち最初の２つについて、その解法を提示した。一方、Python で実装したプログラムによる計算に時間がかかるなどの理由から、具体的な組み合わせ（解）の総数を明確に示すことはできなかった。

本報では、その後の研究成果を踏まえて、先のアルゴリズムにおける問題点を明らかにするとともに、残りの４つについても解法を示すことで、羽目板遊びの探求を完結したい。

２．羽目板遊び１

前報 [1] では、再帰二分法による各深度における解の概数（場合の数）を提示するに留まった。一方で、その値は多数の重複解を含むばかりでなく、特定の深度におけるパネルの枚数が統一されず、正確な解の総数を計算するアルゴリズムに展開するのに適したアプローチとは言い難い。

そこで、分割の回数を基準とした新たな探求方法を考える。n ＋１枚のパネルの構成は、n 枚のパネルの構成から任意の一枚を選択して分割ルールを適応することで生成できる。ここで､選択したパネルについて考えると､縦横が赤組の値の場合は 8 通り、縦横が青組の場合は 10 通り、縦が青組･横が赤組の場合は 9 通り、縦が赤組･横が青組の場合は 9 通りのパターンへの分割が可能である（図１）。初期枠パネルは縦横が青組（2260 ㎜）の正方形なので、各パネル枚数における解の概数は下記の通り計算できることになる。

パネル	解の概数	
1 枚	1	
2 枚	10	
3 枚	188	$= 10 \times 8 + 9 \times 12$
4 枚	5,172	$= 188 \times 8 + 158 \times 12 + 10^2 \times 8 + 9^2 \times 12$
5 枚	187,368	$= 5{,}172 \times 8 + 4{,}140 \times 12 + \ldots$
6 枚	8,421,456	$= 187{,}368 \times 8 + 144{,}336 \times 12 + \ldots$

一方で、これらの値も図２に示す通り異なる分割経路を通じた重複解を多数含んでいる。重複解を解析的に排除することは極めて困難であり、コンピュータの力技で全ての解を導出・保存しながら総数を求める方が現実的であろう。そこで、図１の分析に基づき、Python によるプログラムを準備することにする。まず、夫々の方形パネルを規定する４つのクラス（BB, BR, RB, RR）を定義し、各クラスには相当するモデュロール分割のルールをメソッドとして記述する。問題は､分割数が増えるに従って解の数も爆発的に多くなる中で、効率的に重複解を探し出す手続きである。

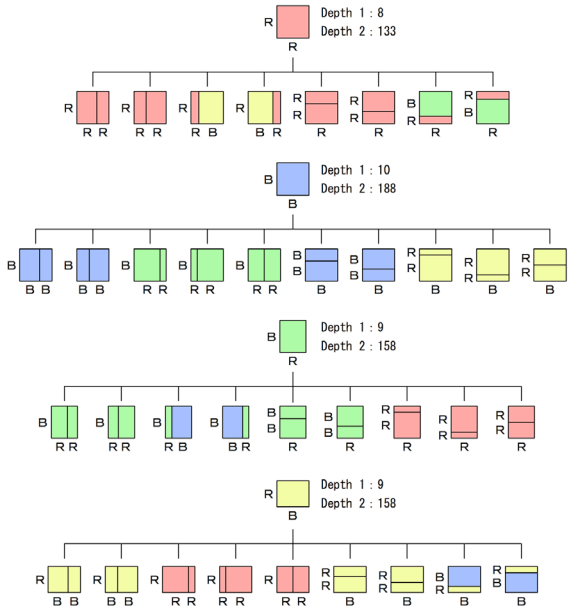

図１　羽目板遊び１の分割ルール

解を保存する最も簡便な方法は、Python であれば下記

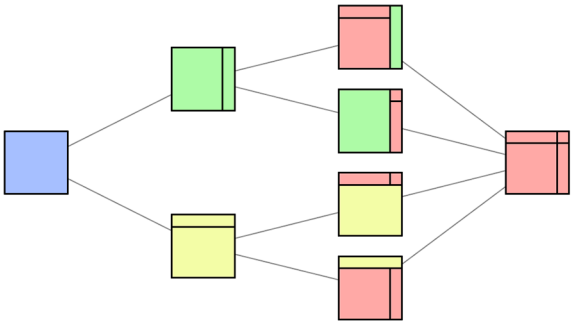

図２　異なる分割経路を通じた重複解

に示す通り、辞書オブジェクト（Dictionary）を用いて、それぞれの解番号（ID）をキーとして構成パネルのリスト（List）を保存する方法であろう。

```
{'1': [BB(0.0,0.0,863.243..,2260.0),
  BB(863.243..,0.0,1396.756..,2260.0),
  BB(1396.756..,0.0,2260.0,2260.0)],
'2': [BB(0.0,0.0,533.513..,2260.0),
  BB(533.513..,0.0,1396.756..,2260.0),
  BB(1396.756..,0.0,2260.0,2260.0)],
'3': .....}
```

しかし、この方法では重複解の判定に（一つずつ順に解を照合する）線形探索が必要であり、解の数が増えるに従って急速に効率が悪くなり実用に耐えない。別の方法として、下記に示す通り、解を構成するパネルに分解し、各パネルをキーとして解番号のリストを保存する方法が考えられる。

```
{'BB(1396.756..,0.0,1726.486..,2260.0)': {12},
'RR(1130.0,0.0,2260.0,1130.0)': {163, 86},
'BB(0.0,0.0,1396.756..,863.243..)': {107, 7},
'BR(0.0,0.0,266.7568..,2260.0)': {57, 74, 4},
'RB(863.243..,0.0,2260.0,1828.378..)': {37, 127},
'RR(1828.378..,431.621..,2260.0,2260.0)': {154, 55},
.....}
```

この方法であれば、検索対象のパネルがどこで使われているかが一目瞭然であり、高速な重複解の判定が可能である。以上による Python のプログラムで計算した各パネル数における正確な解の数は以下の通りとなる。先の概数と比較すると、相当数の重複解があることが確認できる。

パネル	解の総数[注1]	計算時間[注2]
1 枚	1	
2 枚	10	0. 36 秒
3 枚	172	5. 78 秒
4 枚	3, 453	115. 23 秒
5 枚	75, 186	902. 38 秒
6 枚	1, 724, 764	5, 219. 91 秒

３．羽目板遊び２

前報[1)]で示した２番目の羽目板遊びにおける探索木の枝刈りの様子を図３の示す。それぞれのパターンで全ての解を導き出すために、これらの手続きを実装する深さ優先探索のプログラムを記述することになるが、正しい計算を行うためにはいくつかの工夫が必要である。

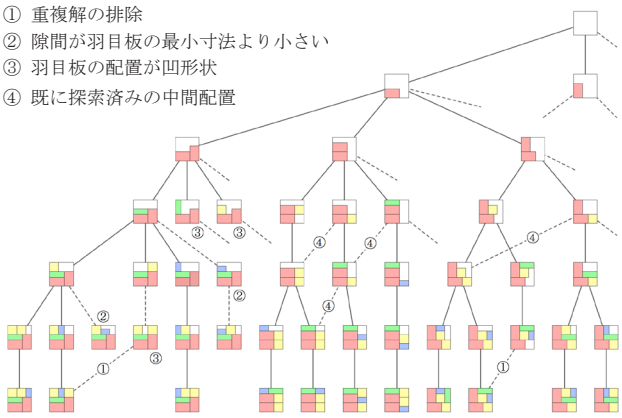

図３　羽目板遊び２での探索木の枝刈り

まずは、コンピュータは基本的に無理数をそのまま扱うことはできないため、通常は浮動小数などを用いざるをえない。一方、浮動小数は $1 / 2^n$ の有限級数であり、それ自体で微妙な誤差を含んでいることが知られており、例えば 0.1 + 0.1 + 0.1 = 0.3 にはならない。実際、黄金比 φ やモデュロールの値に Python の浮動小数型（float）を用いたプログラムでは、パネルの間に 10^{-13} の誤差で隙間が生じる場合があり、パネル配置の凹凸を正しく判定できないことが確認された（図４）。幸い、Python では Decimal モデュールが正確な小数の表現を提供しており、この種の誤差を解消することができた。

また、効率的な枝刈りのためには、最終的な解だけでなく途中段階のパネル配置まで含めた全ての状況を保存し、それぞれの段階で既に探索したパネル配置かどうかを高速で判定する必要がある。しかし前節で示した重複解を排除するデータ構造ではメモリー消費が多く、６種類12枚のパターンａではメモリー不足で計算不能となった。そこで、画像化した配置パターンを tobytes でバイト配列に変換し、md5 でハッシュして Python のセット（Set）に保存することで、メモリーへの負担を減らすとともに高速の探索を実現した。ただし、この方法は、パネルサイズが際限なく小さくなり、画像上の区別ができなくなる１番目の羽目板遊びでは採用できない。

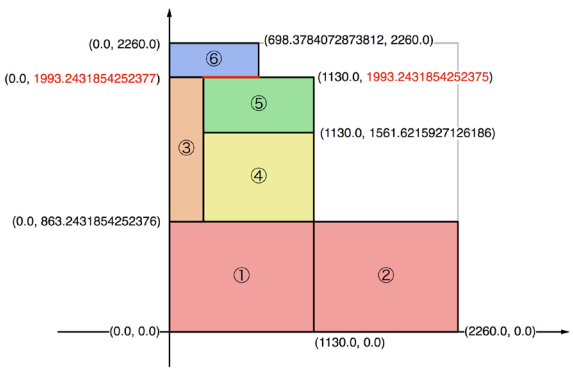

図４　浮動小数による誤差

以下、それぞてのパターンにおける解の総数と計算時間を示す。パネル数が増えた途端に、急激に“組み合わせ爆発”が起こることが確認できる。

パターン	解の総数[注1]	計算時間[注2]
a）6 種類 12 枚[注3]	1, 449, 912	10, 190, 990. 3 秒
b）4 種類 6 枚	232	32. 4 秒
c）3 種類 9 枚	315	425. 9 秒

4．羽目板遊び３

３番目の羽目板遊びは、１番目のルールはそのままに初期枠パネルの幅をモデュロールにより変化させるパズルである（図５）。それぞれの初期枠パネルによる解の探索は１番目と同じ Python のプログラムで計算可能であるが、解の総数だけについて言えば再計算をする必要はない。横の長さが青組の値であれば、解の縦横比が異なるだけで、解の総数は同じである。ちなみに、横の長さが赤組の値の場合も１番目の解法で計算でき、その総数は下記のとおりである。

パネル	解の総数[注1]	計算時間[注2]
1 枚	1	
2 枚	9	0. 33 秒
3 枚	138	4. 63 秒
4 枚	2, 564	85. 25 秒
5 枚	52, 955	1590. 32 秒
6 枚	1, 168, 932	4282. 57 秒

5．羽目板遊び４～６

４～６番目の羽目板遊びは、段階的な構成による一連のパズルである。

まず、４番目の羽目板遊びは、前節と同様に、２番目

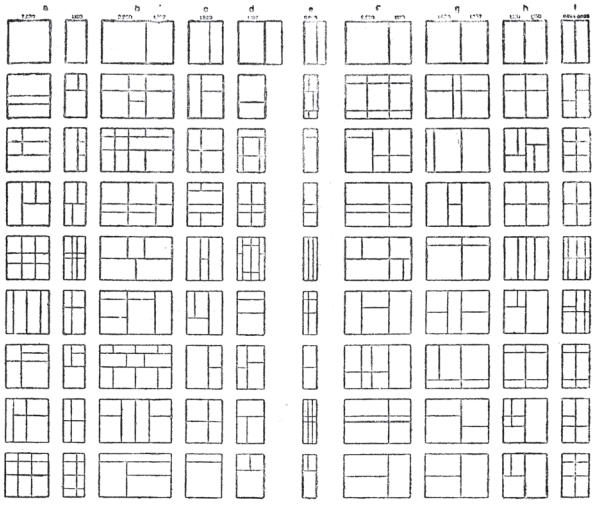

図5　「羽目板遊び３」（モデュロール 第41 図より）

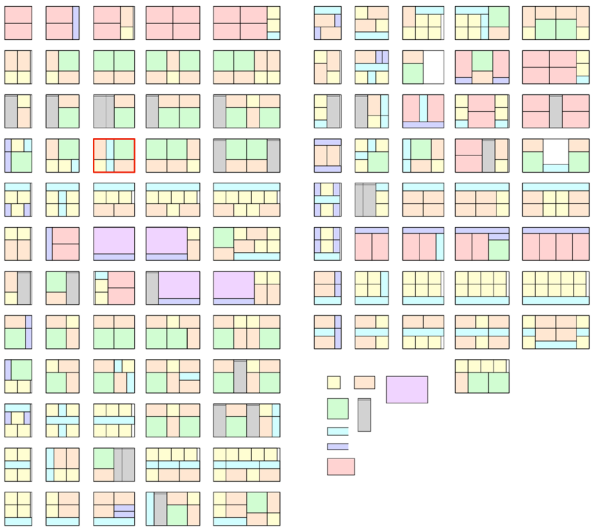

図６　羽目板遊び４（著者着彩）[注4]

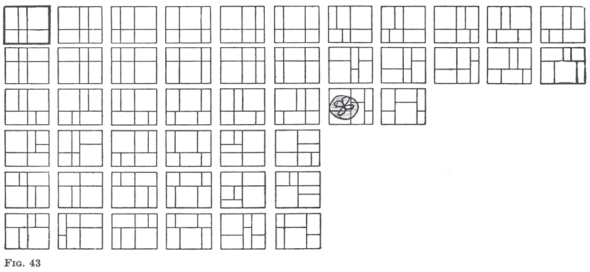

図７　羽目板遊び５（モデュロール 第 43 図より）

ルールはそのままに配置枠の幅をモデュロールにより変化させている（図６）。２番目との違いは、配置するパネルが固定されておらず、５種類のパネル、２種類の帯（可変長パネル）、入り口のための２種類の扉パネルから、任意に選択するところにある。加えて、所々に端数も許容されているのが確認できる。従って、解の探索には２番目と同じプログラムを利用できるものの、深さ優先探索に先立ち夫々の場合で配置するパネルの構成（パターン）を決める必要がある。

ル・コルビュジエは、４番目の羽目板遊びとして５つの配置枠について 101 種類のパターンを提示し、その中から１つを選択し（4 行 3 列目）、５番目の羽目板遊びとして選択したパターンについて 48 種類の組み合わせ（解）を提示した（図７）。この文脈において、コルビュジエは全体で 4848（101 × 48）の解が与えられたと結論づけているが[2)]、これは明らかに誤りである。101 種類のパターンには同じパネル構成のものが含まれているだけでなく、パターンによって解の総数は異なるからである。しかし、これをもって羽目板遊びの無限の可能性が否定されるものではない。

パネルの構成（パターン）を決めるに当たり、２種類の帯（可変長パネル）について任意の長さが許容される

となると、文字通り無限の解を持つことになるため、図６に従い５種類のパネルで用いられている長さに限定することにする。これにより、端数を許容しなければ、パネルの面積の合計が配置枠の面積と一致する暫定パターンを網羅的に洗い出すことができる。しかし、全ての暫定パターンが必ずしも解を持つとは限らない。また、帯パネルだけで埋め尽くすパータンもありうるが、羽目板遊びの趣旨には沿わないであろう。

図６に従い、５種類のパネルの必ずどれかは使い、２種類の帯の総数を５以下に、扉パネルの総数を２以下とすると、それぞれの配置枠における解の総数は下記の通りとなった（φは黄金比）。探索された解の事例を図８に示す。なお、幅が広がるほど配置するパネル数が増加し、特に 4）5）では 10 枚を超えるパターンが増えるとともに組み合わせ爆発が起きており、計算が間に合えばシンポジウムの際に明らかにしたい。

	幅	パターン数（暫定数）	解の総数[注1]
1）	1130 × φ	32 （207）	2,032
2）	2260	79 （560）	14,895
3）	2260 × 2 / φ	145 （1051）	39,084
4）	2260 × φ	--- （3072）	---
5）	2260 × 2	--- （6912）	---

ル・コルビュジエは端数に関して「それらは建築家の利用を俟つものであるか、あるいは建築家の意図してつくられたものである」と言及するに留まり[1)]、どのような端数を容認するのか明確には述べていない。理論的には、容認される端数が他のパネルと同様に定義されれば、提示したアルゴリズムで解を探索することは可能である。ル・コルビュジエは「これらもまたモデュロールのより小さい単位にあてはまるものであり、全体の中に含まれるのである」とも述べている[2)]。実際、図７に示された例を確認すれば、全ての端数の幅は、$2260 / \varphi^5$ か $1130 / \varphi^5$ の何れかであり、端数パネルを恣意的に規定することも可能ではある。しかし、暫定パターンの数はさらに増加し（2 万以上）、それに比例して探索空間も爆発的に拡大する。現状でも相当な計算時間が必要なため、最先端のスーパーコンピュータでも利用しない限り実時間で正確な総数を計算するとは不可能であろう。

なお、６番目の羽目板遊びは、５番目の解に対して５つの異なる素材を割り当てるだけなので、その解の総数は 5^n である。

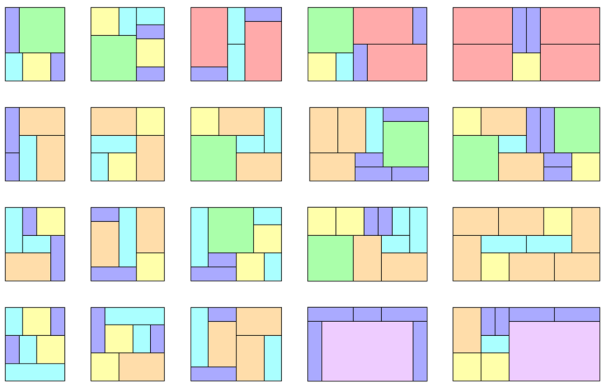

図８　探索された組合せの例

６．おわりに

羽目板遊びに類する構成は、ル・コルビュジエの建築作品の随所で確認することができる。ラ・トゥーレット修道院では、かつて事務所で働いていた現代音楽作曲家ヤニス・クセナキスにより、モデュロールで構成された 29 種類の市松状の PC パネルが羽目板遊びのごとく開口部に表情を与えている[5)]。同修道院では、モデュロールの応用して作り上げた音楽的パン・ド・ベール（波動式ガラス壁）も有名である。ル・コルビュジエはその構成を「水平には部材の密度の違いが連続し、ちょうど弾性体のなかの波のような具合に与えられる。垂直にはいろいろな密度の対位が調和して作られた」と評している[3)]。本報で紹介したプログラムを進化させれば、波動式ガラス壁の新たな構成を探求することも可能であろう。

本研究は、科学研究費補助金　基盤研究（B）「デザイン思考における記号操作の意味的構造と実践的役割に注目するデザイン知の探究」の支援を得て行われた。

注 1）モデュロール[2)] での事例に従い、回転、鏡像が一致するとしても、異なる組合せとして数える。

注 2）計算には MacPro（プロセッサ　3.7GHz Quad-Core Intel Xeon E5・メモリ　64GB）を用い、プロルラムは Python 3.5 で記述し、幾何演算のパッケージとして Shapely 1.6 を利用した。

注 3）原書では「5 つの異なった羽目板」となっているが誤植であり、図をみれば 6 つであることが確認できる。

注 4）原図の凡例などに誤りがあるため、判読しやすいように描き直して着彩した。

[参考文献]

1）渡辺俊：デジタル・モデュロール、第 39 回情報・システム・利用・技術シンポジウム論文集、日本建築学会、pp. 269-272、2016 年 12 月

2）ル・コルビュジエ、吉阪隆正訳：モデュロール、美術出版社、1953

3）ル・コルビュジエ、吉阪隆正訳：モデュロールⅡ、美術出版社、1959

4）Python 言語リファレンス、Python Software Foundation、(https://docs.python.jp/3/reference/index.html)

5）セルジオ・フェロ他、青山マミ訳：ル・コルビュジェ　ラ・トゥーレット修道院、TOTO 出版、1997

[*1] 筑波大学システム情報系・教授・博士（工学）