論文名 Title	局所平坦折り可能な45 度格子展開図の効率的な数え上げ	Effiicient counting of locally flat-foldable 45-degree grid crease patterns
著者 Author(s)	櫻井亮佑、遠藤敏夫	Ryosuke Sakurai, Toshio Endo
受理年月日 Date of acceptance	2026/04/27	
掲載 First publish	『折り紙の科学』（”Science of Origami”）2026/06/14 Vol. 11 No. 1 page 1-17	
備考 Note		

日本折紙学会
Japan Origami Academic Society
www.origami.jp

局所平坦折り可能な 45 度格子展開図の効率的な数え上げ

櫻井亮佑　遠藤敏夫
東京科学大学
226-0026 神奈川県横浜市緑区長津田町 4259
sakusaku858@gmail.com

Efficient counting of locally flat-foldable 45-degree grid crease patterns

Ryosuke Sakurai, Toshio Endo
Institute of Science Tokyo
4259 Nagatsuta-cho, Midori-ku, Yokohama-shi, Kanagawa, 226-0026, Japan

要約　正方形の折り紙の各辺を等分するように格子状に折筋をつけ、さらに格子点を通る斜め 45 度、135 度の折筋をつける。このようにして得られたすべての線分に対し、折る、折らない、のいずれかを割り当ててできる展開図を 45 度系格子パターンにおける展開図と呼ぶ。また、展開図内部の任意の点について、その点の近傍を平坦に折りたたむための条件を満たす展開図を局所平坦折り可能な展開図と呼ぶ。本論文では、45 度格子系パターンにおける局所平坦折り可能な展開図を対称性を考慮して効率的に数え上げる手法を提案し、提案手法によって 12 × 12 マス以下の局所平坦折り可能な展開図の個数を求めた。
キーワード：局所平坦折り、45 度系格子パターン、数え上げ、動的計画法

Abstract: A 45-degree grid crease pattern is an origami pattern made up of some of the creases from a square grid with equally divided sides and some diagonal creases at 45 and 135 degrees that pass through the grid points. Locally flat-foldable means that any point within a crease pattern satisfies the conditions for its neighborhood to be folded flat. This paper proposes a method for counting locally flat-foldable 45-degree grid crease patterns. Using this proposed method, we determined the number of locally flat-foldable 45-degree grid patterns up to 12 × 12.
Keyword: locally flat-foldable, 45-degree grid, counting, dynamic programming

1　はじめに

近年、折り紙作品を創作する人々が増え、日々多くの折り紙作品が発表され続けている。折り紙創作の理論や技法も様々に発展しており、それらを駆使することによりある程度望みの形に近い形状を折り紙で表現することが可能である [11]。一方で、理論によらず紙を折ることで

偶然現れた形状や、シンプルな折りによって現れる形状も、折り紙の面白さであることには変わりない。さらに、折りという操作によって立体的な形状が得られるのも折り紙の魅力であるが、立体的な形状の折り紙作品は理論的に取り扱うのが難しい。また、壁面に設置するための折り紙など、むしろ平坦な形状が好まれる場合もある。

例えば、山本らは局所平坦折り可能な 45 度系格子パターン展開図を列挙するアルゴリズムを提案し、4×4 以下の大きさの局所平坦折り可能な 45 度系格子パターン展開図の列挙を行った。また、それらの展開図を折りたたむことで得られる形状の列挙も行い、アルファベットや数字を表現した平坦な折り紙作品の折り方を計算機を用いて発見した [14]。榎本らは局所平坦折り可能な 45 度系格子パターン展開図を漸化式を用いて数え上げる手法を提案し、$2 \times n$ の場合の一般項を示した。さらに、ZDD を用いて局所平坦折り可能な 45 度系格子パターン展開図を列挙する手法を提案した。数え上げにおいては $6 \times 1,000$ まで、列挙においては 8×8 まで行った [13, 4]。

Chiu らは局所平坦折り可能な山谷の割当を列挙する問題を、グラフの 3 彩色問題へと帰着する手法を考案した [3]。Hull らは各頂点において平坦折り可能であるという条件のもとで一様ランダムに山谷の割当を行い、局所平坦折り可能な展開図が大域的平坦折り可能である確率が指数関数的に小さいことを示した [5]。

本研究では、局所平坦折り可能な 45 度系格子パターン展開図に対し、数え上げに特化したアルゴリズムを提案し、計算機実験によって 12×12 までの大きさの局所平坦折り可能な 45 度系格子パターン展開図を数え上げることができた。

2　準備

本章では、本研究で取り扱う問題を定義し、既知の定理および既存研究について述べる。

2.1　45 度格子展開図

同じ大きさの正方形の集まりで構成されるグリッドとそれらの対角線からなるパターンを 45 度系格子パターンといい、折り紙の展開図上の折線および紙の外周部が 45 度系格子パターンの線と重なるものを本論文では 45 度格子展開図と呼ぶ（図 1）。また、折り紙用紙と対応するグリッド上の単位正方形の個数を 45 度格子展開図の大きさまたはサイズという。例えば、伝承折紙の風車、二双舟、だまし船は大きさ 4×4 の 45 度格子展開図を持つ作品であり、やっこさんは 8×8 の 45 度格子展開図を持つ作品である（図 2）。

2.2　局所平坦折り可能

展開図上の折線の交点を頂点と呼ぶ。また、頂点のうち、展開図の内部にあるものを内部頂点という。展開図が平坦に折りたたまれるには、少なくとも、すべての内部頂点において前川

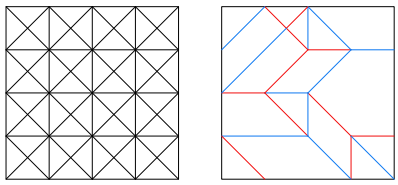


図 1: 4×4 の 45 度系格子パターン (左) と 45 度格子展開図（右）

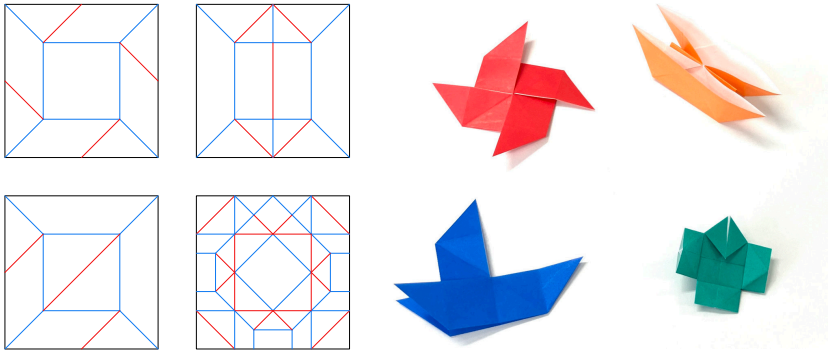


図 2: 45 度格子展開図を持つ伝承折り紙
左上：風車、右上：二双舟、左下：だまし船、右下：やっこさん

定理と川崎定理を満たしていなければならない [9, 15]。

- 前川定理　頂点周りの山折りと谷折りの本数の差は 2 である (図 3 左)
- 川崎定理　1 つおきの内角の和は 180 度である (図 3 右)

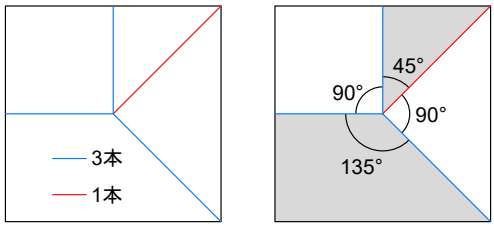


図 3: 前川定理 (左) と川崎定理 (右)

一方で、外周部に存在する頂点は折線の本数や角度によらず山谷をうまく割り当てさえすれば平坦に折りたたむことが可能であると考えられる。しかし、与えられた展開図に対し、大域的に平坦に折りたためるように山谷を割り当てる問題は NP 困難であることが Bern らによって示されている [1]。従って本研究では、山折りと谷折りを区別しない。すなわち、前川定理については考慮せず、任意の内部頂点において川崎定理を満たす展開図を局所平坦折り可能な展開図とする。以後、川崎定理を局所平坦条件と呼ぶ。また、格子パターン上の線分に対し、折る、折らないを決めることを辺の割り当てと呼ぶ。

2.3　対称性の考慮

展開図を数える際、回転させたり鏡に映したりして得られるような本質的に同一の展開図をひとつの展開図として数えたいというのは自然であろう (図 4)。そこで本研究では、折り紙用紙を正方形に限定し、二面体群 D_4 を考慮した。すなわち、以下の 8 つの写像によって互いに移り変わる展開図を同一のものとみなして数え上げを行った。

- 恒等写像 e
- 回転 r_k：重心周りの $\frac{\pi}{2}k$ ラジアン回転 $(k = 1, 2, 3)$
- 鏡映 s_h, s_v：水平および垂直な中心線を軸とした反転
- 鏡映 s_{d_1}, s_{d_2}：2 本の対角線を軸とした反転

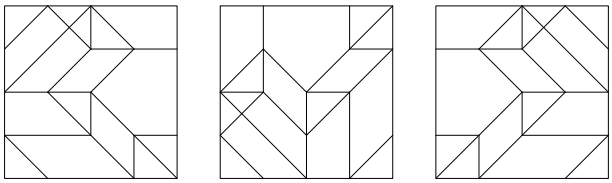


図 4: 本質的に同一の展開図

2.4　バーンサイドの補題

バーンサイドの補題とは、対称性を考慮して数学的な対象を数え上げるとき用いられる定理である [2]。

定理 1 (バーンサイドの補題)**.** 有限群 G が集合 X に作用しているとする。$\mathrm{fix}_X(g)$ は群 G の元 g によって固定される X の元すべてからなる集合を表す。このとき軌道の数 $|X/G|$ は

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |\mathrm{fix}_X(g)| \tag{1}$$

と表される。

本研究において、有限群 G を正方形の対称群（二面体群 D_4）とし、その位数は $|G| = 8$ である。集合 X_n を $n \times n$ グリッドの 45 度系格子パターンにおいて局所平坦折り可能な展開図（山谷の区別を除いた幾何学的構造）すべてからなる集合と定義する。

このとき対称性を考慮した展開図の総数 $|X_n/G|$ は、式 (1) により各対称操作 $g \in G$ に対して不変かつ局所平坦折り可能な展開図の個数 $|\mathrm{fix}_{X_n}(g)|$ を用いて

$$|X_n/G| = \frac{1}{|G|} \sum_{g \in G} |\mathrm{fix}_{X_n}(g)|$$

と表される。ただし以下の 3 つが成り立つから、実際に計算機を用いて数えるのは $e, r_1, r_2, s_v, s_{d_1}$ についてのみでよい。

$$|\mathrm{fix}_{X_n}(r_1)| = |\mathrm{fix}_{X_n}(r_3)|, \quad |\mathrm{fix}_{X_n}(s_h)| = |\mathrm{fix}_{X_n}(s_v)|, \quad |\mathrm{fix}_{X_n}(s_{d_1})| = \mathrm{fix}_{X_n}(s_{d_2})|$$

各 $|\mathrm{fix}_{X_n}(g)|$ の値の求め方は第 3 章で述べる。

2.5 山本らによる列挙手法

45 度系格子パターンには 2 種類の頂点が存在する。頂点から出る線の数が 4 本の 4 価頂点と頂点から出る線の数が 8 本の 8 価頂点である (図 5)。局所平坦条件を満たす 4 価頂点の辺の割り当ては 4 通りしかなく、局所平坦条件を満たす 8 価頂点の辺の割り当てパターンは回転や反転を含めちょうど 36 通りである (図 7)。各 4 価頂点 v に対し、v と接続する同一直線上の 2 辺を 1 つの辺とみなす。各 8 価頂点にこの 36 個のパターンを順次当てはめていくことにより、局所平坦折り可能な展開図を列挙する。このとき、互いに隣接する 8 価頂点が共有する辺 (図 6 の点線) の「折る・折らない」割り当てに矛盾がないことを制約とする。

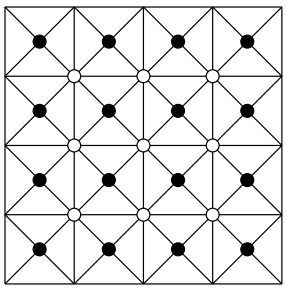


図 5: 4 価頂点（黒丸）と 8 価頂点（白丸）

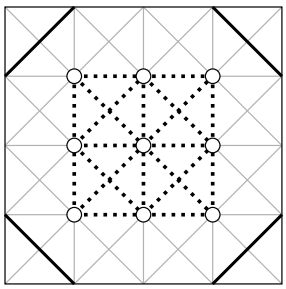


図 6: 8 価頂点間の辺と四隅の辺

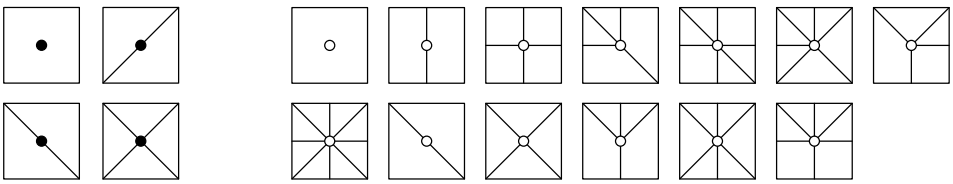


図 7: 局所平坦条件を満たす 4 価頂点の辺の割り当て (左) と 8 価頂点の辺の割り当て (右)

この方法では、四隅の斜めの線を折るか、折らないかが決定されない (図 6 の黒い斜め線)。逆に考えると、この四隅の斜めの線は局所平坦折り可能かどうかに関与しない。斜めの線 4 カ所について、折るか、折らないかを決定すればよく、対称性を考慮しなければ 2^4 通りの組み合わせになる。

2.6　岩下らによるグリッドグラフ上のパスの数え上げ

有名な数え上げ問題に「パスの数え上げ」がある。パスの数え上げは、グラフ G と G 上の 2 頂点 s,t が与えられ、同じ頂点を 2 度通らずに s から t へ行く経路が何通りあるかを数え上げる問題である。Knuth は ZDD というデータ構造と Simpath というアルゴリズムを用いて高速にパスを列挙する手法を提案した [10]。Kawahara らは制約条件によってさまざまなバリエーションを持つ Simpath を総称してフロンティア法と呼び、フロンティア法がなぜ高速であるかを明らかにした [8]。岩下らはグリッドグラフ上のパスの数え上げに対し、フロンティア法をベースとして最小完全ハッシュ関数を用いたさらに効率的な数え上げ手法を提案した [7, 6]。また、その論文では中国の剰余定理を使った効率化についても言及している。

2.7　最小完全ハッシュ関数

ハッシュ関数とは、入力データ（キー）に対し何らかの短い値（ハッシュ値）を出力する関数である。ハッシュ関数は大量のデータの中から特定のデータを高速に検索する際に用いられる。同じ入力データに対しては必ず同じ出力が得られるが、異なる入力データに対しては必ずしも異なる出力が得られるとは限らない。また、終域のハッシュ値に対応する入力が必ず存在するとも限らない。すなわち、全射とも単射とも限らない。

一方で最小完全ハッシュ関数は全単射（入力データとハッシュ値が一対一対応）であり、出力ハッシュ値が連続する整数値である。岩下らの研究 [7, 6] では、後述するフロンティアの状態を入力として最小完全ハッシュ関数を定義し、ハッシュ値を配列のインデックスとして用いている。

2.8　フロンティア法

無向グラフ $G = (V, E)$ と 2 つの頂点 $s, t \in V$ を入力としたパスの数え上げ問題を考える。各辺 $e \in E$ に対し、辺 e をパスの一部として「採用する・採用しない」のどちらかを割り当て、全体の割り当てが s-t パスを成すかどうかを判定することによって、s-t パスを数え上げることができる。この手法は、フロンティア法 [8] と呼ばれるアルゴリズムを用いて高速化できる。

パスの数え上げにおいて、ある辺集合まで探索を進めた際、それまでに選択された辺と未探索の辺の両方に接続している頂点の集合をフロンティアと呼ぶ（図 8）。

探索の過程では、以下の 2 つを区別して考える。

- 探索の進捗（部分グラフ）：どの辺を採用し、どの辺を採用しないかという、構築途中の具体的な情報
- フロンティアの状態：フロンティア上の各頂点 v に対する以下の 2 つの情報

– 頂点 v の次数 (次数が 0 なら「まだ接続されていない」、次数が 1 なら「端点になっている」、次数が 2 なら「パスの途中になっている」ことを表す)
– v が端点であるならば反対側の端点がどの頂点か

複数の異なる探索の進捗（部分グラフ）に対し、それらのフロンティアの状態が全く同一であれば、それ以降の未探索領域においてどの辺を採用すれば s-t パスが完成するかという条件は完全に一致する。

例えば、図 8 に示す 2 つの異なる探索の進捗（部分グラフ）はフロンティアの状態が全く同一であるため、これらを個別に探索する必要はない。したがって、フロンティアの状態をキーとして、同一のフロンティアの状態を持つ複数の探索の進捗をひとまとめに集約することで、探索空間を劇的に削減することが可能となる。

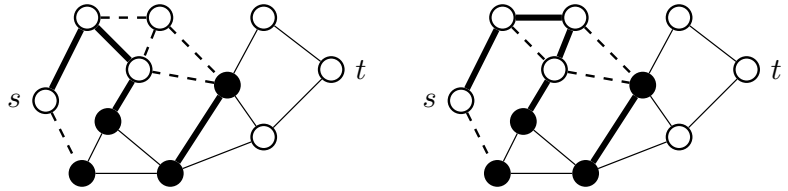


図 8: フロンティアの状態が同一の割り当て
太線：経路として採用する辺、破線：経路として採用しない辺
実線：未探索の辺、黒丸：フロンティアに含まれる頂点

2.9 中国の剰余定理

定理 2 (中国の剰余定理)**.** $m_1, m_2, \ldots, m_n > 0$ をどの 2 つも互いに素な整数とする。任意の整数 $a_1, a_2, \ldots, a_n$ に対し、

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

を満たす整数 x が $m_1 m_2 \ldots m_n$ を法として一意に存在する。

この定理を用いれば、巨大な整数を使った計算を複数の小さな整数の計算に置き換えることができる。巨大な整数を使った計算はメモリ使用量が大きくなるため、複数の小さな整数の計算に置き換えることでメモリ使用量を節約できる可能性がある。ただし、計算の最後に複数の剰余からもとの巨大な整数を復元する処理が必要となる。また、$n = 2$ の場合、すなわち合同

式が 2 本の場合には解 x は、$m_1p+m_2q=1$ を満たす整数 p,q を用いて $x=a_1m_2q+a_2m_1p$ と表せる。これを含めて中国の剰余定理と呼ぶこともある。

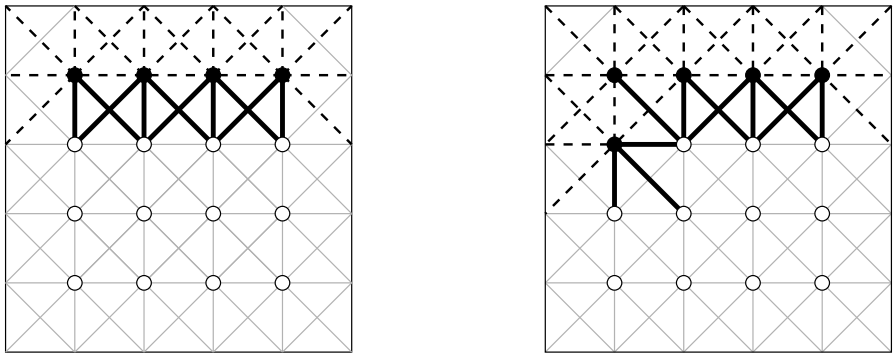
3　提案手法

本研究では、1 つの 8 価頂点の各辺に対する「折る・折らない」の割り当てであって局所平坦折り可能なものを「タイル」と呼び、8 価頂点にその割り当てを適用することを「設置」と呼ぶ。また、どの頂点にどのタイルが設置されたかを「タイル配置」と呼ぶ。

提案手法では、左上の 8 価頂点から行優先で図 7 右に示した 36 種類のタイル（回転、反転含む）のいずれかを設置していく。このとき、フロンティア法の考え方を用いて、探索の進捗では必要な情報だけを保持するようにし、異なるタイル配置であってもその情報が同一であれば集約する。

3.1　恒等写像 e によって固定される展開図の数え上げ

局所平坦折り可能な 45 度格子展開図の数え上げにおけるフロンティアは、タイルを設置した 8 価頂点と未設置の 8 価頂点の両方に接続する辺の集合である。図 9 において、黒丸はタイルを設置済みの 8 価頂点であり、白丸は未設置の 8 価頂点である。破線と太線は「折る・折らない」の割り当てが確定した辺であり、太線の辺の集合がフロンティアである。図 10 の 2 つの異なるタイル配置において、○印のついた辺は折ることが確定しているフロンティア辺、×印のついた辺は折らないことが確定しているフロンティア辺である。これらのタイル配置はフロンティアの状態が互いに同一であるため、残りの 8 価頂点のタイル配置が同一であれば局所平坦折り可能かどうかも同一となる。



(a) 4 枚のタイルを設置した盤面　　(b) 5 枚のタイルを設置した盤面

図 9: 局所平坦折り可能な展開図の数え上げにおけるフロンティア

次に局所平坦折り可能な 45 度格子展開図の数え上げにおけるフロンティアの状態の総数について考える。5×5 の問題について、ある行のすべての 8 価頂点にちょうどタイルを設置し

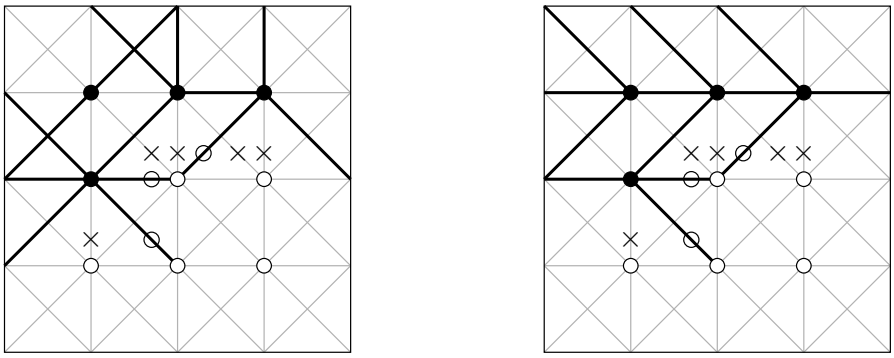


図 10: フロンティアの状態が同一のタイル配置

終わった盤面 (図 9a) と、そうでない盤面 (図 9b) について、前者の盤面のフロンティアは 10 本の辺からなり、後者の盤面のフロンティアは 11 本の辺からなる。$n \times n$ の問題において、フロンティアは最大 $3n-4$ 本の辺を持つ。これらの辺は各々が個別に「折る・折らない」という値を取りうる。従って、各フロンティアの状態の数は最大 2^{3n-4} となる。

これらの状態と、0 から $2^{3n-4}-1$ までの整数を一対一で対応づけることにより、最小完全ハッシュ関数が得られる。この最小完全ハッシュ関数を用いることにより、フロンティアの状態に番号をつけることができる。設置済みのタイルの枚数が異なる 2 つのタイル配置に同一の状態番号が対応付けられることがあるが、設置済みのタイルの枚数ごとに状態を区別するため、提案手法のアルゴリズムは問題なく動作する。

最後に、動的計画法を用いた数え上げの考え方について説明する。設置済のタイルの枚数を縦軸に取り、フロンティアの状態番号を横軸に取った表を用意する (図 11)。この表はアルゴリズムによって何度も更新されるが、最終的にこの表の (i,j) 成分は、タイルを i 枚設置した状態であって、フロンティアの状態番号が j であるようなタイルの設置の仕方の総数となる。

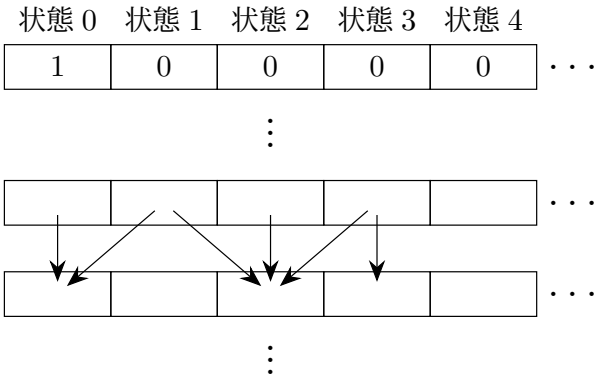


図 11: 動的計画法

初期状態として、表の $(0,0)$ 成分には 1 を入れ、それ以外の成分には 0 を入れる。0 行目に

おいて、$(0,0)$ 成分にのみ 1 を入れるのは、0 枚タイルを設置する方法がただ 1 つだけ存在することを意味する。

以後、i 行目の値をもとに、$i+1$ 行目を計算することを繰り返す。i 枚のタイルが設置されていてフロンティアの状態番号が j であることを状態 (i,j) と呼ぶと、表の定義より、表の (i,j) 成分は状態 (i,j) となるタイルの配置の仕方の総数である。状態 (i,j) に $i+1$ 枚目のタイルとしてタイル t を設置したあとのフロンティアの状態番号を $\mathrm{put}(i,j,t)$ とする。36 種類のタイル t に対し、状態 (i,j) にタイル t が設置できるかどうか判定し、設置できるなら、表の $(i+1,\mathrm{put}(i,j,t))$ 成分に (i,j) 成分を加算する。タイルの設置可能性は、タイル数 i とフロンティアの状態番号 j をもとに、フロンティア辺の「折る・折らない」の割り当てを復元して判定する。これを $(n-1)\times(n-1)$ 行目まで繰り返す。

このようにして表を埋めたのち、最終行のセルの値の和 sum を計算する。8 価頂点におけるタイル配置のみでは図 6 で示した 4 隅の辺の割り当て（2^4 通り）が考慮されないから、求める展開図の個数は $N_{\mathrm{corner}}=2^4$ として $|\mathrm{fix}_{X_n}(e)| = N_{\mathrm{corner}}\cdot\mathrm{sum}$ で得られる。

各行の計算に必要な情報はひとつ前の行のみに書かれているので、実際にはタイルの枚数分の行は不要で、偶数番目のタイル設置の結果は 1 行目に、奇数番目のタイル設置の結果は 2 行目に書き込めばよいから、2 行の表で足りる。

BDD や ZDD といったデータ構造では、根ノード (0 枚タイルを設置した状態) から葉ノード (すべての 8 価頂点にタイルを設置した状態) までの状態を保持しなければならないが、本手法では設置前の状態と設置後の状態のみを保持すればよい点でメモリの効率が良い。さらに配列としてデータを保持しているため、各要素へのアクセスが高速である。

アルゴリズムの概要を Algorithm1 に示す。ここで、table は動的計画法に用いる表、m は 8 価頂点の個数、$s(=2^{3n-4})$ はフロンティアの最大状態数、T はタイルの集合 $(|T|=36)$ である。フロンティアの状態番号 j に関する for ループは並列処理することで高速に計算可能である。ただし、同一のインデックスに同時に書き込まないための工夫が必要で、本研究では、アルゴリズムをプログラミング言語で実装するにあたり、openMP の omp_lock_t 型を用いた。

Algorithm 1 恒等写像 e によって固定される展開図の数え上げ

table[0][0] $\leftarrow 1$ とし、他の要素を 0 で初期化
for 各頂点 $i = 0$ **to** $m - 1$ **do**
  prev $\leftarrow i \bmod 2$, next $\leftarrow (i+1) \bmod 2$
  table[next][$*$] の全要素を 0 にリセット
  for 各状態 $j = 0$ **to** $s - 1$ **do**
    for all タイル $t \in T$ **do**
      if 状態 j に $i+1$ 枚目のタイルとして t が設置可能 **then**
        $k \leftarrow \mathrm{put}(i, j, t)$ (タイル設置後の状態番号)
        table[next][k] $\leftarrow$ table[next][k] + table[prev][j]
      end if
    end for
  end for
end for
sum $\leftarrow$ table[$m \bmod 2$][$*$] の全要素の合計
return $N_{\mathrm{corner}} \cdot \mathrm{sum}$ (N_{corner}=2^4)

3.2 群作用によって固定される展開図の数え上げ

本節では恒等写像 e 以外の各群作用 $g \in D_4$ に対し、$g.x = x$ を満たす局所平坦折り可能な展開図 $x \in X_n$ の個数 $|\mathrm{fix}_{X_n}(g)|$ を数え上げる。いずれの作用においても、基本的には恒等写像 e に対する動的計画法を基本とするが、対称性に基づき以下の変更を加える。ただし、展開図、頂点、タイル等の対象 s への、群 G の元 g による作用を $g.s$ と表記する。

- 探索領域 V_s の限定
  $g \in D_4$ の作用によって 8 価頂点の集合 V を軌道分解し、各軌道の代表元を含む最小限の頂点集合 V_s のみを探索対象とする。領域外のタイルの配置は V_s 内の対応するタイルの回転または鏡映のコピーとして一意に決定される。
- 境界の整合性
  頂点 $v \in V_s$ と $u \in g.V_s$ が隣接する場合、その間の辺の割り当てが矛盾しないようにタイル設置可能性判定を行う。
- 不動点頂点制約
  $g.v = v$ を満たす 8 価頂点 v（回転中心や対称軸上の頂点）については、設置可能なタイルを $g.T = T$ を満たすタイル T に制限する。
- 係数 N_{corner}
  展開図上の四隅の辺のうち、対称性によって独立に決定できる辺の数を k としたとき、

$N_{\text{corner}} = 2^k$ とする。

3.2.1 回転 r_1 によって固定される展開図の数え上げ

探索領域は図 12 に示す通り、全体の約 1/4 の領域となる。V_s 内のタイル配置が決定されれば、中心の 8 価頂点 c に設置されるタイルは一意に定まる。

図 12 の青い辺と赤い辺の割り当てが一致する必要があることに留意し、フロンティアの情報として、通常の「更新中の行の境界情報」に加え、「探索領域 V_s の右端列の境界情報」を保持する。V_s の最終行にある 8 価頂点に対しタイル設置可能性判定を行う際、最右列の境界情報を参照する。

四隅の辺はすべて回転 r_1 で移り合うため、独立な辺は 1 つであり、$N_{\text{corner}} = 2^1$ となる。

3.2.2 回転 r_2 によって固定される展開図の数え上げ

探索領域 V_s は図 13 に示す約 1/2 の領域である。8 価頂点の個数が奇数の場合には中心の頂点 c が存在し、$r_2.T_c = T_c$ を満たす必要がある。V_s の最終行にある頂点においてタイルの設置可能性判定を行う際は、図 13 の赤い辺と青い辺の割り当てが一致することに留意する。展開図上の四隅の辺において、上側にある 2 つの辺に対して独立に折り割り当てを決定できるため $N_{\text{corner}} = 2^2$ となる。

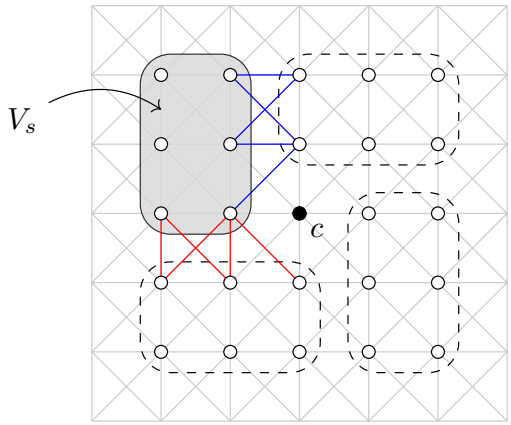


図 12: 回転写像 r_1 における探索領域 V_s

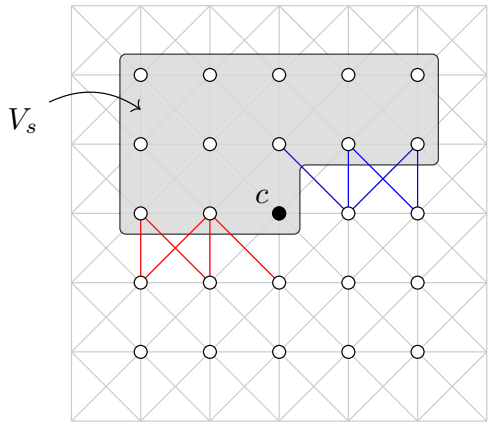
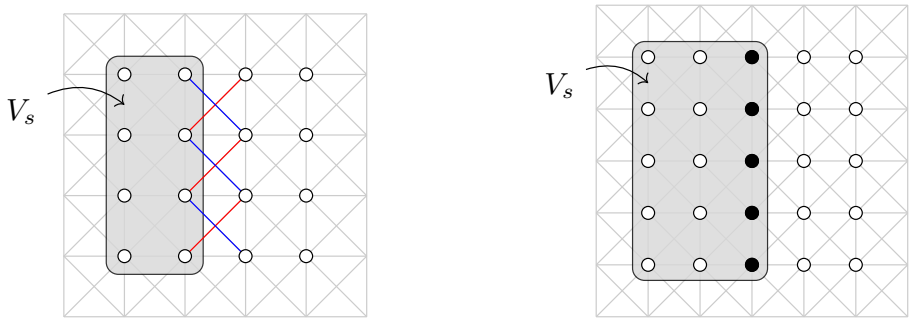


図 13: 回転写像 r_2 における探索領域 V_s

3.2.3 鏡映 s_v によって固定される展開図の数え上げ

探索領域 V_s は図 14 に示す約 1/2 の領域である。8 価頂点の個数が偶数の場合、図 14a に示す青い辺と赤い辺の割り当てが一致するようにタイルの設置可能性判定をする。8 価頂点の個数が奇数の場合、反転軸上の頂点集合 V_c (図 14b の黒丸の頂点の集合) に対し、左右対称なタイルを割り当てる。展開図上の四隅の辺において、左側の 2 つの辺が独立であるため $N_{\text{corner}} = 2^2$ となる。



(a) 8 価頂点数が偶数の場合　　(b) 8 価頂点数が奇数の場合

図 14: 鏡映写像 s_v における探索領域 V_s

3.2.4　鏡映 s_{d_1} によって固定される展開図の数え上げ

探索領域 S_v は図 15 に示す、主対角線によって分割された三角形の領域とする。主対角線上のすべての 8 価頂点に対し、設置可能なタイルを主対角線で対称なタイルに限定する。展開図上の四隅の辺において、右上の辺を除く 3 つの辺が独立であるから $N_{\text{corner}} = 2^3$ となる。

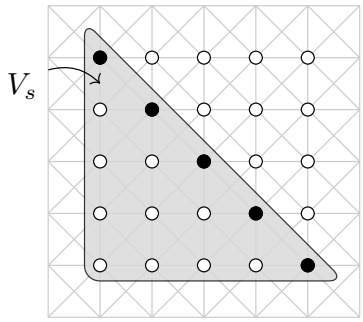


図 15: 鏡映写像 s_{d_1} における探索領域 V_s

4　実験結果

作成したプログラムはメインの処理に C++ とマルチコア並列化のために openMP を用い、一部の処理に python を用いた。また、テーブル内の各セルの値は int 型や long 型に収まらないため、中国の剰余定理を用いて小さな整数に分割したのち、最後に復元を行った。復元後には int 型や long 型に収まらない巨大な整数となるため、多倍長ライブラリの GMP を用いた。

実行環境は表 1 の通りである。作成したプログラムを実行して得られた結果を表 2 に、計算にかかった時間と仮想メモリ使用量のピーク値 (VmPeak)、および VmPeak の近似値を表 3 に示す。さらに、物理メモリ使用量のピーク値 (VmHWM)、および VmHWM の近似値を表 4 に示す。最も時間のかかった 12×12 のサイズの数え上げは、2.1 時間の計算時間がかかり、VmPeak は 478 GB、VmHWM は 476 GB であった。

表 1: 実行環境

OS	AlmaLinux 9.6
CPU	AMD EPYC 7713 64-Core Processor ×2
コア数／スレッド数	128 ／ 256
メモリ容量	503GB

表 2: 局所平坦折り可能な展開図の個数

問題のサイズ	展開図の個数
2×2	116
3×3	58530
4×4	259650300
5×5	7144660230240
6×6	1201334538620830944
7×7	1244914416834394675537920
8×8	800161402340519477457558604569 6
9×9	320446296167130689887000090453414475776
10×10	802327064019240321588478691881163181436081868 80
11×11	125925540414936510696483270546587754245391944613796225024
12×12	124153325509480170781238947409682620841377908035249645318377822617 6

山本らの研究 [14] では対称性を考慮した数え上げを行い、4×4 の大きさの問題まで数え上げられているが、本研究の結果はこれと一致している。また、榎本らの研究 [13, 4] では対称性を考慮しないかつ四隅の辺の割り当てを行わない場合の数え上げを行い、8×8 の大きさの問題まで数え上げられている。本研究で作成したプログラムの一部を用いることで同様の条件で問題を解くことができ、この結果も榎本らの結果と一致することを確認できた。

5　考察

実験結果より、ピークメモリ使用量が計算のボトルネックとなっていることがわかる。そこで、問題サイズ $n \times n$ におけるメモリ使用量のピーク値を近似的に求めることを考えたい。アルゴリズムを観察すれば、問題サイズが 1 つ大きくなるごとにフロンティア辺が 3 本増え、それに伴ってフロンティア辺の状態の総数は 8 倍となることがわかる。このことより、ピークメモリ使用量は定数 a, b を用いて $a + b \times 8^n$ で近似できると考えられる。実際に $a = 2047.72, b = 0.000006914142868$ としたものが表 3 の最右列であるが、VmPeak に対す

表 3: 計算時間と VmPeak および VmPeak の近似値

問題のサイズ	計算時間（秒）	VmPeak(MB)	$a + b \times 8^n$
2×2	0.18	2047.72	2047.73
3×3	0.21	2047.72	2047.73
4×4	0.23	2047.75	2047.75
5×5	0.22	2047.98	2047.95
6×6	0.29	2049.54	2049.54
7×7	0.50	2062.23	2062.23
8×8	2.73	2163.77	2163.73
9×9	18.09	2975.76	2975.73
10×10	117.40	9471.79	9471.73
11×11	954.68	61439.73	61439.76
12×12	7461.39	477183.74	477184.00

表 4: VmHWM および VmHWM の近似値

問題のサイズ	VmHWM(MB)	$a + b \times 8^n$
2×2	6.28	6.29
3×3	6.28	6.29
4×4	6.31	6.31
5×5	6.50	6.51
6×6	8.10	8.10
7×7	18.01	20.79
8×8	118.26	122.29
9×9	930.26	934.29
10×10	7426.26	7430.29
11×11	59392.37	59398.32
12×12	475136.31	475142.56

る精度の良い近似になっている。

物理メモリ使用量のピーク値（VmHWM）については、定数 a を $a = 6.28$ に変更し、比較した (表 4)。その結果、実測値と計算値の間にわずかな差はあるものの、近似式として機能することを確認した。

また、計算時間においても、8×8 から 12×12 の問題サイズにおいては、ひとつ小さい

サイズの問題と比べ 5 倍〜8 倍程度の計算時間がかかっている。従って、このプログラムで 13×13 のサイズの問題を計算するには約 4 TB のメモリと約 17 時間の計算時間が必要であると予想できる。

6 まとめ

本研究では、動的計画法や最小完全ハッシュ関数、バーンサイドの補題を用いて、12×12 までの局所平坦折り可能な 45 度格子展開図の総数を数え上げた。12×12 の問題では 124153325509480170781238947409682620841377908035249645318377822617 6 通りの局所平坦折り可能な展開図があることがわかった。また、これを計算するために、約 477GB のメモリと約 2 時間の計算時間が必要であった。

実際の折り紙の創作において最も欲しい情報は、折りたたんだ後の形状とその形状を得るための展開図である。ゆえに、今後の課題として、展開図の総数だけでなく、折りたたんだ後の形状と展開図の列挙が考えられる。さらに、山谷を適切に割り当てられるかどうかの判定や、大域的に平坦に折りたためるかどうかの判定についても行う必要があると考えられる。実際、4×4 の問題に対し、適切な山谷の割り当てが存在しない展開図や、大域的に平坦に折りたためない展開図が松川らによって発見されている [12]。

7 謝辞

本研究は富士通次世代コンピューティング基盤協働研究拠点の助成を受けております。

文献

[1] M. Bern and B. Hayes. The complexity of flat origami. *Proceedings of the 7th annual ACM-SIAM symposium on Discrete Algorithms*, pp. 175–183, 1996.

[2] William Burnside. Theory of groups of finite order. *Messenger of Mathematics*, Vol. 23, p. 112, 1909.

[3] Alvin Chiu, William Hoganson, Thomas C. Hull, and Sylvia Wu. Counting locally flat-foldable origami configurations via 3-coloring graphs. *Graphs and Combinatorics 37*, pp. 241–261, 2021.

[4] Y. Enomoto, Y. Kawakami, K. Seto, T. Horiyama, and J. Mitani. Counting and ZDD-based enumeration of locally flat-foldable box-pleated crease patterns on the 45-degree grid system. The 5th International Workshop on Enumeration Problems & Applications, Nov. 2022.

[5] Thomas C. Hull, Marcus Michelen, and Corrine Yap. On random locally flat-foldable origami. arXiv:2502.04279, 2025.

[6] H. Iwashita, Y. Nakazawa, J. Kawahara, T. Uno, and S. Minato. Fast computation of the number of paths in a grid graph. *The 16th Japan Conference on Discrete and Computational Geometry and Graphs*, pp. 17–19, Sep. 2013.

[7] Hiroaki Iwashita, Yoshio Nakazawa, Jun Kawahara, Takeaki Uno, and Shin-ichi Minato. Efficient computation of the number of paths in a grid graph with minimal perfect hash functions. *TCS Technical Report. No. TCS-TR-A-13-64. Hokkaido University Division of Computer Science*, 2013.

[8] Jun Kawahara, Takeru Inoue, Hiroaki Iwashita, and Shin-ichi Minato. Frontier-based search for enumerating all constrained subgraphs with compressed representation. *IEICE Transactions on Fundamentals*, Vol. E100-A, No. 9, pp. 1773–1784, September 2017.

[9] T. Kawasaki. On the relation between mountain-creases and valley-creases of a flat origami. *Proceedings of the First International Meeting of Origami Science and Technology*, pp. 229–237, 1989.

[10] Donald Ervin Knuth. *The art of computer programming: Bitwise tricks & techniques*, Vol. 4. Addison-Wesley, 2009.

[11] Robert J. Lang. Treemaker. `https://langorigami.com/article/treemaker/`. アクセス日:2025-09-04.

[12] Yoshihisa Matsukawa, Yohei Yamamoto, and Jun Mitani. Enumeration of flat-foldable crease patterns in the square/diagonal grid and their folded shapes. *Journal for Geometry and Graphics*, Vol. 21, No. 2, pp. 169–178, 2017.

[13] 榎本優大, 河上悠輝, 脊戸和寿, 堀山貴史, 三谷純. 45 度系格子パターンにおける局所平坦折り可能な展開図の数え上げと ZDD による列挙. 電子情報通信学会総合大会, DS-1-11, 2022 年 3 月.

[14] 山本陽平, 三谷純. 45° 系格子パターンから作り出される平坦折り形状の列挙. 折り紙の科学, Vol. 4, pp. 23–33, 2015.

[15] 前川淳, 笠原邦彦. ビバ! おりがみ. サンリオ, 新装版, 1989.