Doctoral thesis

# Research on Flat-Foldable Single-Vertex Crease Patterns

by

Koji Ouchi

Supervisor: Ryuhei Uehara

**Graduate School of Advanced Science and Technology**
**Japan Advanced Institute of Science and Technology**
**[Information Science]**

March, 2020

# Abstract

This paper aims to help origami designers by providing methods and knowledge related to a simple origami structure called *flat-foldable single-vertex crease pattern*. A *crease pattern* is the set of all given *creases*. A crease is a line on a sheet of paper, which can be labeled as "mountain" or "valley". Such labeling is called *mountain-valley assignment*, or MV assignment. *MV-assigned crease pattern* denotes a crease pattern with an MV assignment. A sheet of paper with an MV-assigned crease pattern is *flat-foldable* if it can be transformed from the completely unfolded state into the flat state that all creases are completely folded without penetration. In applications, a material is often desired to be flat-foldable in order to store the material in a compact room. A *single-vertex crease pattern* (SVCP for short) is a crease pattern whose all creases are incident to the center of the sheet of paper. A deep insight of SVCP must contribute to development of both basics and applications of origami because SVCPs are basic units that form an origami structure.

A decision problem whether a given crease pattern is flat-foldable or not was studied by Bern and Hayes in 1996. There are several theorems related to flat-foldable SVCP: for example, the Kawasaki Theorem, the Maekawa Theorem, and the Big-Little-Big Lemma. A *forcing set* is one of the promising properties in origami applications. A forcing set is a subset of a given flat-foldable crease pattern $C$. If the creases in the forcing set are folded according to given MV assignment $\mu$, the all other creases in $C$ are also folded according to $\mu$. In an application called self-folding origami, a thin material folds into an intended shape by rotating the planes around creases according to the label mountain or valley assigned on the creases. The cost of such an application can be reduced if it is enough to put actuators on a subset of creases. Such an optimization problem can be modeled as a *minimum forcing set problem*. The minimum forcing set problem supposes us to find a forcing set with the minimum number of creases. The input of this problem is a flat-foldable MV-assigned crease pattern $(C, \mu)$. Damian et al. proposed an algorithm for finding a minimum forcing set for arbitrary 1D origami in 2015. Ballinger et al. developed an algorithm for Miura-ori in 2015. The minimum forcing set for arbitrary 2D origami may be important in origami applications. However, there is no algorithm for such case so far.

In this paper, we propose an algorithm for finding a forcing set of flat-foldable MV-assigned SVCP, which might help us to construct an algorithm for arbitrary 2D origami. Our algorithm, which runs in $O(n^2)$ time where $n$ is the number of given creases, is a variant of the algorithm by Damian et al. Furthermore, we show that the number of creases in the minimum forcing set for SVCP is $n/2$ or $n/2 + 1$. The proof for the size of minimum forcing set is by considering a situation that we repeatedly crimp consecutive creases forming a minimal angle with different assignments. Roughly speaking, such size is $n/2$ if the number of remaining creases after crimp repetition is two, and otherwise it is $n/2 + 1$.

It is also interesting to know how many flat-foldable MV-assigned crease patterns there are. In the case of SVCP, the tight upper and lower bounds on such count has been shown by Hull in 2003. However, enumeration of flat-foldable crease patterns has not been studied actively, although it is relative to counting. This paper tackles an efficient enumeration of flat-foldable MV-assigned SVCPs. Such enumeration provides us concrete examples of MV-assigned SVCPs, which must be helpful to construct a new origami structure. In this enumeration, let a positive even number $q$ be an input, and let the angle between two adjacent creases be a multiple of unit angle $(360/q)^\circ$. Our algorithm reduces symmetrically duplicate patterns up to rotation and reflection. As far as the author knows, MV-assigned SVCP enumeration introducing the unit angle and reduction of symmetry in this paper is the first trial in the world. The author notes that the problem condition in this paper is different from that for the upper and lower bounds on counting by Hull. Our enumeration algorithm is composed of three phases: (1) enumerate crease patterns of at most $q$ creases satisfying the Kawasaki Theorem; (2) enumerate MV assignments on the crease patterns obtained in (1) satisfying the Maekawa Theorem; (3) test flat foldability of the obtained MV-assigned SVCPs. The phase (1) can be done in parallel to (2) and (3) with master-worker model: the master process computes the phase (1); a worker process computes the phase (2) and (3) for an SVCP given by the master process. In experiment, our algorithm enumerates approximately $4.07 \times 10^{13}$ flat-foldable MV-assigned SVCPs for $q = 40$ in 34 hours using a supercomputer.

This paper contributes to the development of origami by proposing algorithms for two problems: minimum forcing set for MV-assigned SVCP and enumeration of flat-foldable MV-assigned SVCPs with unit angle. The result of minimum forcing set for MV-assigned SVCP must help investigation of minimum forcing set for arbitrary 2D origami. The enumeration provides us examples of flat-foldable MV-assigned SVCPs and reveals that the number of flat-foldable SVCPs and that of flat-foldable MV-assigned SVCPs are numerous.

**Keywords**: computational origami, flat foldability, forcing set, enumeration

# Acknowledgments

The author greatly appreciates the constant encouragement and the kind guidance of his supervisor Professor Ryuhei Uehara of Japan Advanced Institute of Science and Technology during this work. The author would like to thank Assistant Professor Giovanni Viglietta of Japan Advanced Institute of Science and Technology for his helpful discussions and suggestions. The author would like to thank Associate Professor Yota Otachi of Kumamoto University for his helpful discussions and suggestions.

The author is grateful to all who have affected or suggested his areas of research. The author devotes his sincere thanks and appreciation to all of them and his colleagues.

# Contents

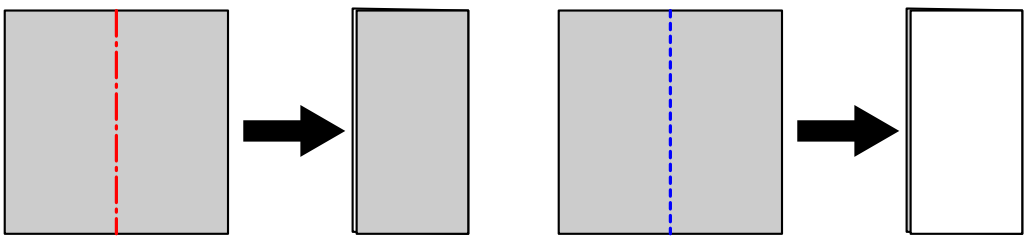
# Chapter 1

# Introduction

## 1.1 Background

From the viewpoint of industry, folding is a promising way to make a thing compact with flexibility to easily expand, e.g., for solar panels of artificial satellites. Such an origami structure can be defined by a *crease pattern* and a *mountain-valley assignment* (or *MV assignment* shortly). A crease pattern $C$ is a set of *creases* that are lines to be folded on the sheet of paper. MV assignment $\mu$ is a function that gives a label "mountain" (M) or "valley" (V) to each crease, that is, MV assignment tells us the direction (upward or downward) of rotation of paper around a given crease. Figure 1.1 shows a mountain fold and a valley fold. We call $(C, \mu)$ an *MV-assigned crease pattern*.

Origami can be classified by dimension and characterizations. One



(a) Mountain fold. (b) Valley fold.

Figure 1.1: Example of mountain fold and valley fold.

dimensional origami (1D origami) is defined as folding a strip of paper with creases orthogonal to the longer edges of the strip. We can abstract 1D origami by a line segment for the strip and points on the segment for the creases (see Figure 1.2). *Stamp folding* is a 1D origami such that the distances between consecutive creases are equal.

Two dimensional origami (2D origami) has creases with arbitrary directions on a piece of paper and the folded shape is flat. Flat-foldable *single-vertex crease pattern* (SVCP) is a 2D origami whose creases are incident to the center of a sheet of paper to be folded. We consider the sheet of paper for SVCP is a disk. Figure 1.3 is an example of flat-foldable MV-assigned SVCP. *Miura-ori* is a class of 2D origami, whose crease pattern is made of two dimensional array of congruent parallelograms as shown in Figure 1.4.

Three dimensional origami (3D origami) is considered to have non-flat folded shape. Its crease pattern is usually on a flat piece of paper.

In applications of origami, a sheet of paper, or a thin material, is often desirable to satisfy *flat foldability* in order to make the size of the material small. Flat foldability means whether a crease pattern or an MV-assigned crease pattern can be folded into a flat shape without penetrating itself if all creases are folded completely. There are some pieces of software to help designers' tough investigation of origami. Mitani developed ORIPA[1] [21, 22], which is software for drawing MV-assigned crease pattern, evaluating its flat foldability, and estimating the state of the folded sheet of paper. Origamizer[2] [25] by Tachi creates an MV-assigned crease pattern which can be folded into the input 3D model. Although the software help designers very well, testing whether a crease pattern is flat-foldable or not is substantially difficult: Bern and Hayes showed that such test on a given MV-assigned crease pattern is NP-hard [4].

In application called self-folding origami [11, 14, 18, 19], it is preferable to use fewer actuators that implement motion of planes around creases to

[1] http://mitani.cs.tsukuba.ac.jp/oripa/

[2] http://origami.c.u-tokyo.ac.jp/~tachi/software/


Figure 1.2: An example of 1D origami.

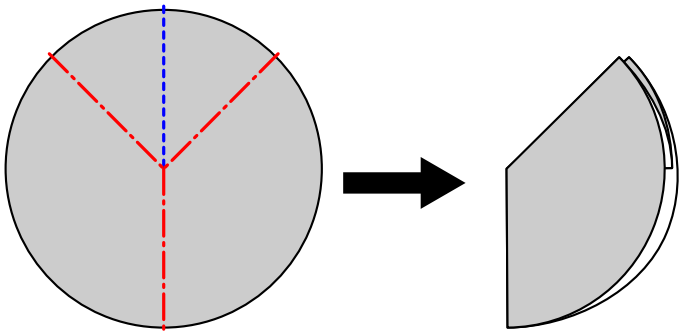


Figure 1.3: An example of flat-foldable MV-assigned SVCP.

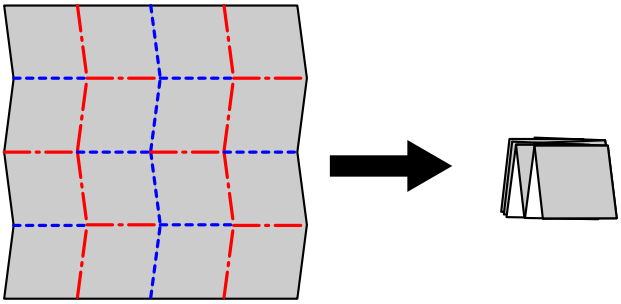


Figure 1.4: An example of Mimura-ori.

reduce the production cost, which leads to *minimum forcing set* problem. A *forcing set* is a subset of given flat-foldable creases which forces other creases to follow the given MV assignment when folding. A minimum forcing set is a forcing set with the minimum number of creases. Finding a minimum forcing set of arbitrary 2D origami is an open problem.

In design of origami, designers often introduce a constraint that an angle between creases incident to a vertex is only multiples of 22.5°, 15° or some other specific *unit angle* which divides 360°. This restriction reduces the choice of creases which is a part of difficulty of designing, however, such designing is still tough work. In addition, designers often aim to obtain a flat-folded shape. Therefore, designers may want a catalog of components of flat-foldable crease patterns to make their design easier, which requires a kind of enumeration.

This paper focuses on flat-foldable SVCP. We can consider that SVCP is a basic component of a crease pattern because a crease pattern is a set of SVCPs. Therefore, investigating SVCPs may contribute to study and design of 2D origami. We also denote an SVCP with MV assignment by an *MV-assigned SVCP*. Although SVCP is a fundamental and important component of origami, there are few pieces of previous research compared to complicated 2D and 3D origami applications. This paper studies two topics; one is minimum forcing set for SVCP, and the other is enumeration of flat-foldable MV-assigned SVCPs with unit angle. These must give us insight for further development of 2D origami studies and applications.

## 1.2 Contents of Thesis

Chapter 2 gives an overview of related work in computational origami: flat foldability, forcing set, and enumeration and counting. Flat foldablility is a main component of our research. Our key theorems and lemma on flat foldability are the Kawasaki Theorem, the Maekawa Theorem, and the Big-Little-Big Lemma. Forcing set is studied for 1D origami [6] and Miura-ori [3]. Matsukawa et al. studied an enumeration of flat-foldable crease pattern and possible folded shapes in $45^\circ$ system with $4 \times 4$ grid [20]. Mitani developed an enumeration of all possible flat-folded states after folding [21, 22]. Counting MV assignment is actively studied by Hull. He derived the tight upper and lower bound on the MV assignment for flat-foldable SVCP. Ginepro and Hull also counted the MV assignments for Miura-ori of $h \times w$ grid.

In Chapter 3, we consider minimum forcing sets for SVCP. Minimum forcing set problem is a model for saving resource in self-folding origami: the number of actuators for self folding can be reduced to the size of the minimum forcing set for a given MV-assigned crease pattern. We analyze the size of minimum forcing set for SVCP and propose an algorithm for finding a minimum forcing set for a given MV-assigned SVCP.

In Chapter 4, we propose an algorithm for enumerating the all distinct flat-foldable MV-assigned SVCPs under an assumption that every angle between two creases is multiple of $(360/q)^\circ$ for given $q$. The algorithm considers that crease patterns and MV-assigned crease patterns are equivalent if they are equal up to rotation and reflection. The result of enumeration can be seen as knowledge for designing origami because SVCP is a component of an origami model. The number of enumerated patterns is highly exponential, which implies how difficult SVCP origami is.

Chapter 5 notes the results and future work.

# Chapter 2

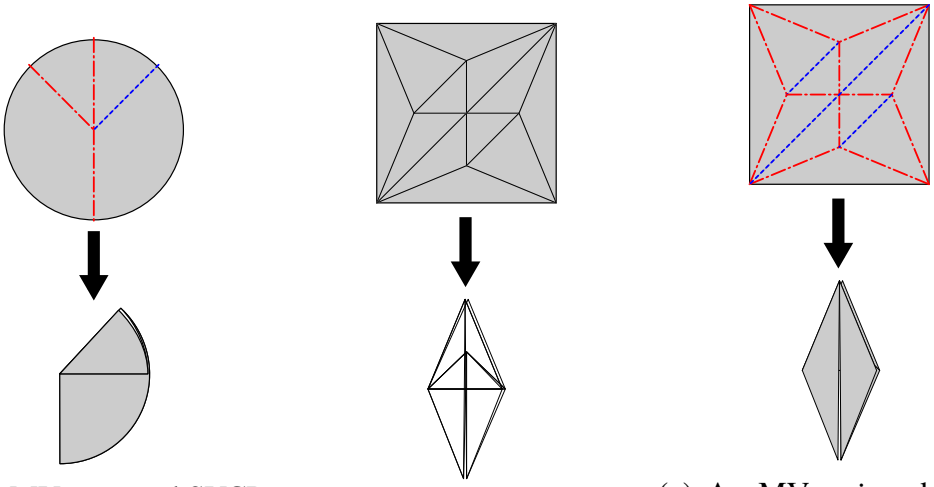
# Related Work

## 2.1 Flat Foldability of Origami

This paper entirely works on flat-foldable crease pattern. A crease pattern or an MV-assigned crease pattern is flat-foldable if it can be folded into a flat shape without penetrating itself when all creases are folded completely. Great work on flat foldability was done by Bern and Hayes [4]. They showed the hardness of flat foldability in 3 situations: an MV-assigned SVCP (Figure 2.1a), an arbitrary crease pattern (Figure 2.1b), and an arbitrary MV-assigned crease pattern (Figure 2.1c).

The test for an MV-assigned SVCP takes linear time with respect to the number of creases. If only a crease pattern is given, we can test the existence of MV assignments such that the area around each vertex is flat-foldable. This problem is called *local flat foldability* problem. The test of local flat foldability can be done in linear time as well, and so is generating a valid assignment [8]. However, computing flat foldability is NP-hard even if we are given a valid MV-assigned crease pattern. The proposed algorithm in Chapter 4 uses the test of flat foldability for MV-assigned SVCP in order to detect flat-foldable ones in the enumeration.

Assuming that given $n$ creases are indexed clockwisely and the index starts from 1, there are several conditions related to flat foldability of MV-assigned SVCP as follows:



(a) An MV-assigned SVCP. (b) A crease pattern. (c) An MV-assigned crease pattern.

Figure 2.1: Three situations for flat foldability. We can obtain a wire frame of folded shape in (b).

**Theorem 2.1 (The Kawasaki Theorem [16])** *Let $\theta_i$ be an angle between the ith and the $(i+1)$st creases. We assume that either mountain or valley is assigned to each crease. An SVCP defined by angles $\theta_1+\theta_2+\cdots+\theta_n = 360°$ is flat-foldable if and only if n is even and the sum of the odd angles $\theta_{2i+1}$ is equal to the sum of the even angles $\theta_{2i}$, or equivalently, either sum is equal to $180°$: $\theta_1 + \theta_3 + \cdots + \theta_{n-1} = \theta_2 + \theta_4 + \cdots + \theta_n = 180°$.*

**Theorem 2.2 (The Maekawa Theorem [8, Chapter 12] )** *We assume that a given MV-assigned SVCP defined by angles $\theta_1 + \theta_2 + \cdots + \theta_n = 360°$ is flat-foldable. Then the number of mountains and the number of valleys differ by $\pm 2$.*

**Lemma 2.1 (Big-Little-Big Lemma [15, 16])** *If an angle $\theta_i$ is strictly minimal, that is, $\theta_{i-1} > \theta_i < \theta_{i+1}$ holds, then the creases forming $\theta_i$ have assignment different from each other in any flat-foldable MV-assigned crease pattern.*

**Lemma 2.2 (Generalized Big-Little-Big Lemma (Theorem 4 of [13]))** *If a sequence of k angles $\theta_i, \theta_{i+1}, \ldots, \theta_{i+k-1}$ is strictly minimal, that is, $\theta_{i-1} > \theta_i = \theta_{i+1} = \cdots = \theta_{i+k-1} < \theta_{i+k}$ holds, then the difference of the*

*number of mountains and the number of valleys on the creases between the angles $\theta_{i-1}, \theta_i \ldots, \theta_{i+k}$ is zero for odd $k$ and is one for even $k$.*

Note that the Kawasaki Theorem is a necessary and sufficient condition on angles between creases and others are necessary conditions. The Maekawa Theorem and the Big-Little-Big Lemma are components of proof in Chapter 3. These conditions are used to reduce the computation time of the enumeration algorithm in Chapter 4.

## 2.2 Forcing Sets

Forcing set is a new topic in computational origami. A forcing set $F$ is a subset of a given flat-foldable crease pattern $C$. If the creases in the forcing set are folded according to given MV assignment $\mu$, the all creases in $C \setminus F$ are also folded according to $\mu$ and no other way to be folded. Figure 2.2 depicts the concept of minimum forcing set for a flat-foldable MV-assigned SVCP.

In an application called self-folding origami [11, 14, 18, 19], a thin material folds into an intended shape by rotating the planes around creases according to the label mountain or valley assigned on the creases. The cost of such an application can be reduced if it is enough to put actuators on a subset of creases. Such an optimization problem can be modeled as a *minimum forcing set* problem. The minimum forcing set problem supposes us to find a forcing set with the minimum number of creases. The input of this problem is a flat-foldable MV-assigned crease pattern $(C, \mu)$. Table 2.1 shows the current progress of studies on minimum forcing set.

Damian et al. proposed an algorithm for finding a minimum forcing set of arbitrary 1D origami in 2015 [6]. Outline of their algorithm is as follows: repeat crimping minimal distance sequence of creases to construct a forest whose nodes represents the sequences, where the sequence of a parent node includes a crease surviving the crimp on the sequence of a child node; traverse the forest in preorder manner to decide the forcing

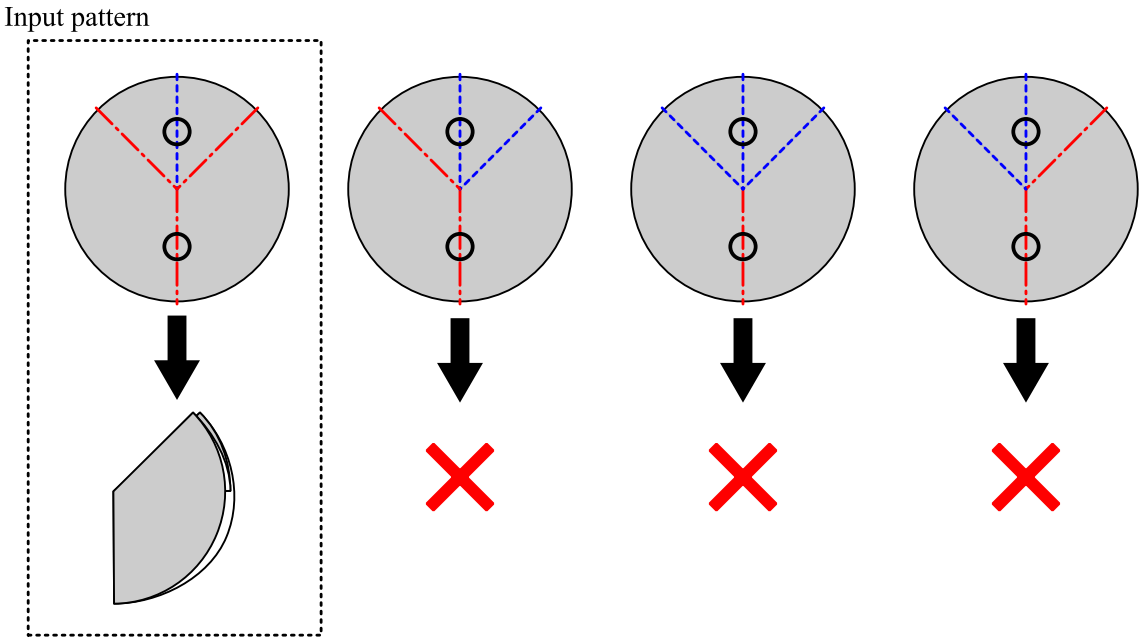



Figure 2.2: An example of a minimum forcing set for the flat-foldable MV-assigned SVCP in Figure 1.3. Each crease with a small circle is the crease of the forcing set. Given MV assignment is a unique foldable assignment if we try all possible MV assignments on not forcing creases.

Table 2.1: An overview of studies on minimum forcing set. Dim. means dimension, and MFS means minimum forcing set.

| Dim. | Constraint | Finding MFS | Bounds for the size of MFS | Paper |
|---|---|---|---|---|
| 1D | Arbitrary | $O(n)$ time | Open | [6] |
| 2D | Arbitrary | Open | Open | — |
| 2D | Miura-ori | $O(h^2w^2)$ time for $h \times w$ grid | Lower: $h + w - 2$, Upper: $\lceil hw/2 \rceil$ | [3] |
| 2D | Single-vertex | $O(n^2)$ time | $n/2$ or $n/2 + 1$ | Our result |

creases.

Ballinger et al. developed an algorithm for finding a minimum forcing set of 2D Miura-ori in 2015 [3]. To describe the problem, they used equivalence between locally flat-foldable Miura-ori MV assignments and 3-vertex colorings of a grid graph with one vertex pre-colored shown in [10]. They showed tight bounds of the number of the creases in the minimum forcing set, and proposed a method for testing whether a given set is forcing or not as well. The minimum forcing set for arbitrary 2D origami may be important in origami applications. However, algorithm for such case has

Table 2.2: An overview of studies on enumeration and counting of origami.

| Target | Enumeration | Count | Bounds for count |
|---|---|---|---|
| Stamp folding | [24] | [17] (by formula) | [26] |
| MV assignment for Miura-ori | Open | [10] (by formula) | Open |
| 45° system ($4 \times 4$ grid) | [20] | [20] (by enuemration) | Open |
| MV-assigned SVCP, with unit angle, excluding symmetrical duplications | Our result | Our result (by enumeration) | Open |
| MV assignment for SVCP, without unit angle, including symmetrical duplications | — | — | [13] |

not been developed.

In this paper, we propose an algorithm for finding a minimum forcing set of flat-foldable MV-assigned SVCP by converting the algorithm in [6], which might help us to construct an algorithm for arbitrary 2D origami. We also show the size of a minimum forcing set for SVCP is $n/2$ or $n/2+1$. As far as the author knows, our work is the first trial for minimum forcing set of SVCP.

## 2.3 Enumeration and Counting of Origami

Enumeration is an emerging topic in computational origami due to a rise of calculation speed of computers. Mitani has implemented in ORIPA [21, 22] an enumeration of all possible flat-folded states after folding input MV-assigned crease pattern. If we focus on the size of the output of enumeration, the problem can be seen as a counting problem. Table 2.2 summarizes the current progress of studies on enumeration and a count of origami.

Hull showed tight upper and lower bounds on the count of MV assignments for flat-foldable SVCP [13]. The situation is similar to our enumeration of flat-foldable MV-assigned SVCP in Chapter 4, but Hull did not remove duplications up to rotation and reflection. Therefore Hull's result cannot be applied to the analysis of our enumeration.

Koehler studied counting of stamp folding which is a kind of 1D origami [17]. Stamp folding considers to fold a strip of square or rect-

angular stamps which are separated by creases, and we are supposed to fold along the all creases without penetration. He derived complicated formulas to compute the counting and listed such numbers by computer for $N \leq 16$ where $N$ is the number of stamps. Sawada and Li developed a constant amortized time algorithm to generate stamp foldings in [24]. The numbers of stamp folding is known for $N \leq 45$ so far, listed as A000136 in OEIS[1]. Uehara showed that the lower bound of the count is $\Omega(3.065^N)$ and the upper bound is $O(4^N)$ [26].

Ginepro and Hull counted the number of locally flat-foldable MV assignments on Miura-ori of $h \times w$ grid where $h$ and $w$ are given [10]. They derived recurrence and closed formula for $w \leq 5$ and arbitrary $h$, and computed the numbers of MV assignments for $w \leq 5$ and $h \leq 8$. The computed numbers fit with existing integer sequence of the number of ways of 3-coloring the vertices of a grid graph with one vertex pre-colored (A078099 in OEIS), where a grid graph is a dual of the given Miura-ori crease pattern regarded as a graph. They showed such correspondence occurs for any $h$ and $w$.

Matsukawa et al. enumerated flat-foldable crease patterns and possible folded shapes in 45° system with 4×4 grid [20]. They did not count the order of layers in a folded shape. Their enumeration removes duplications up to rotation and reflection. It is shown that there are 259,650,300 flat-foldable crease patterns and 13,452 folded shapes in such situation.

Contribution of this paper in this field is that the author enumerates and counts flat-foldable MV-assigned SVCPs introducing the concept of unit angle for the first time in the world. The count in Chapter 4 is done by enumeration. Deriving the closed formula of the count is future work. Upper and lower bounds for the count is an open problem as well.

[1]The On-line Encyclopedia of Integer Sequences: http://oeis.org/

# Chapter 3

# Minimum Forcing Sets for Single-Vertex Crease Pattern

This chapter is based on the author's paper published as [OU19b]. We propose an algorithm for finding a minimum forcing set of SVCP. A forcing set is a subset of given creases that forces all other creases to fold according to the given labels. Our algorithm is a modification of an existing algorithm for 1D origami [6]. We show that the size of a minimum forcing set of an SVCP is $n/2$ or $n/2 + 1$ where $n$ is the number of the creases in the SVCP.

## 3.1 Introduction

In an origami application called self-folding origami, a thin material folds into an intended shape by rotating the planes around creases according to the label mountain or valley assigned on the creases [11, 14, 18, 19]. The cost of such an application can be reduced if it is enough to put actuators on a subset of creases. Finding such a subset of creases can be modeled as a forcing set problem.

Forcing set problem is a new topic in computational origami, which

was considered in [1, 3, 6]. Especially, minimum forcing set for flat foldability was studied for 1D origami [6] and 2D Miura-ori [3]. In a forcing set problem for flat foldability, a flat-foldable MV-assigned crease pattern $(C, \mu)$ is given. A forcing set $F$ is a subset of $C$ where $c \in F$ is assigned the value $\mu(c)$, and $F$ makes the other creases $c' \in C \setminus F$ to be assigned the value $\mu(c')$: $F$ is not a forcing set if $c'$ can have the assignment opposite to $\mu(c')$ to make the given crease pattern fold flat. A forcing set $F$ is called minimum if there is no other forcing set with size less than $|F|$.

This chapter focuses on minimum forcing sets for flat-foldable SVCP. If $|C|$ is two, we are to fold the sheet of paper in half, and it is obvious that the size of the minimum forcing set is one. To simplify calculation of index circulation, we assume the index of creases starts with 0 in this chapter. An SVCP is a sequence of creases $C = (c_0, c_1, \ldots, c_{n-1})$ which are put clockwisely on the disk incident to the center. $\theta_i$ denotes the clockwise angle from $c_i$ to $c_{i+1 \bmod n}$ (see Figure 3.1).

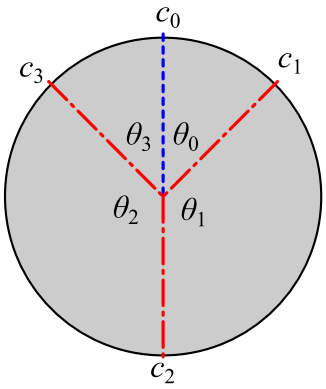



Figure 3.1: Notation of creases and angles.

In this chapter, we develop an algorithm to find a minimum forcing set of a given flat-foldable MV-assigned SVCP in $O(n^2)$ time. As far as the author knows, our algorithm is the first one for finding a minimal forcing set of flat-foldable MV-assigned SVCP, even though SVCP is an important component of origami. Our algorithm is based on that for 1D origami in [6] because the structure of SVCP is similar to 1D origami if we regard it as a ring by cutting away the inner space of the sheet of paper: the creases reduce to points on the ring, and the sheet of paper becomes 1D origami

if we cut the ring at some point. We also reveal that the size of $F$ is $n/2$ or $n/2 + 1$. Precisely, $|F|$ is $n/2$ if the SVCP is of generic angles, which is a case that the angles to be operated always differ. In the case when all the angles in the SVCP are equal, $|F|$ is $n/2$ if $n = 2$, otherwise $|F|$ is $n/2 + 1$. For a general SVCP, which does not have any constraints, the size of $F$ is $n/2 + 1$ if the crease pattern can be reduced to an SVCP of equal angles with size four or more by repeatedly crimping consecutive two creases $(c_i, c_{i+1 \bmod k})$ with different MV assignment where $\theta_i$ is minimal, otherwise $|F| = n/2$.

## 3.2 Preliminaries

This section introduces some terminology and preliminary results following [6]. Throughout the chapter we work with a flat-foldable MV-assigned crease pattern $(C, \mu)$, where $C = (c_0, c_1, \ldots, c_{n-1})$ is a flat-foldable SVCP and $\mu$ is a flat-foldable MV assignment.

### 3.2.1 Crimpable Sequences [6]

We slightly change the definition of crimpable sequence to fit the assumption that $C$ is circular. A *crimpable* sequence in SVCP is composed of consecutive creases where the angles between the creases are equal, with the property that the two angles adjacent to the left and right end of the sequence are strictly larger than the equal angles. Formally, for integers $0 \leq i < n$ and $0 < k < n$, a sequence of consecutive creases $(c_i, c_{i+1 \bmod n}, \ldots, c_{i+k \bmod n})$ is crimpable if $\theta_i = \theta_{i+1 \bmod n} = \cdots = \theta_{i+k-1 \bmod n}$ and $\theta_{i-1 \bmod n} > \theta_i < \theta_{i+k \bmod n}$. Figure 3.2 shows an example. We note that we have to take a mod on the index for circulation. Thus we may have $(i - 1) \bmod n = (i + k) \bmod n$.

A *monocrimp* operation is defined as a fold about a single pair of consecutive creases of opposite MV parity in a crimpable sequence. (See Figure 3.3.)

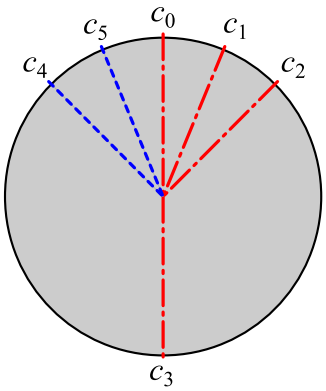



Figure 3.2: $(c_4, c_5, c_0, c_1, c_2)$ is a crimpable sequence.

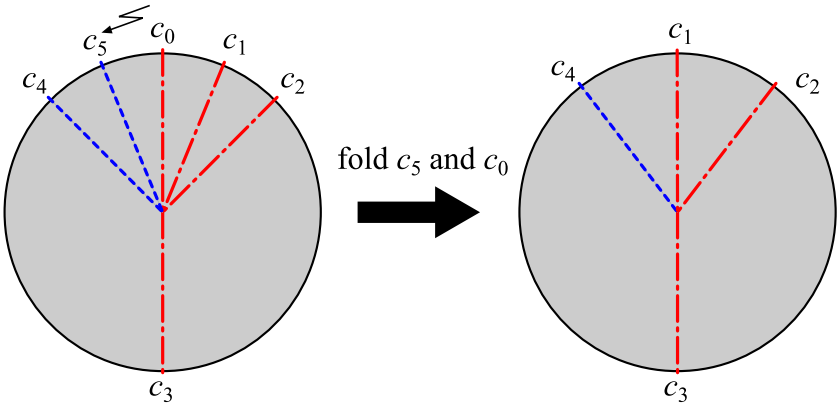



Figure 3.3: We can monocrimp $(c_5, c_0)$ because they are in a crimpable sequence $(c_4, c_5, c_0, c_1, c_2)$ and their MV assignment are different. A monocrimp makes the sheet of paper conic. We describe such a conic crease pattern by an image captured from the upside of the cone.

A *crimp* operation is a set of monocrimps repeatedly conducted on a crimpable sequence while the sequence is crimpable (Figure 3.4). In our proofs for the minimum size of a forcing set, we characterize such size by considering the conditions for flat foldability on a given SVCP while repeating a crimp operation to fold the SVCP flat. A crimp on an MV-assigned SVCP on a disk changes the disk to a cone and further crimps make the cone sharper. Such change of the shape does not affect flat foldability [8, Chapter 12]. The following theorem described with crimpable sequence is equivalent to Lemma 2.2, which will be needed in Section 3.5.

**Theorem 3.1 (Theorem 1 from [6])** *Let $\alpha$ be a crimpable sequence in a*

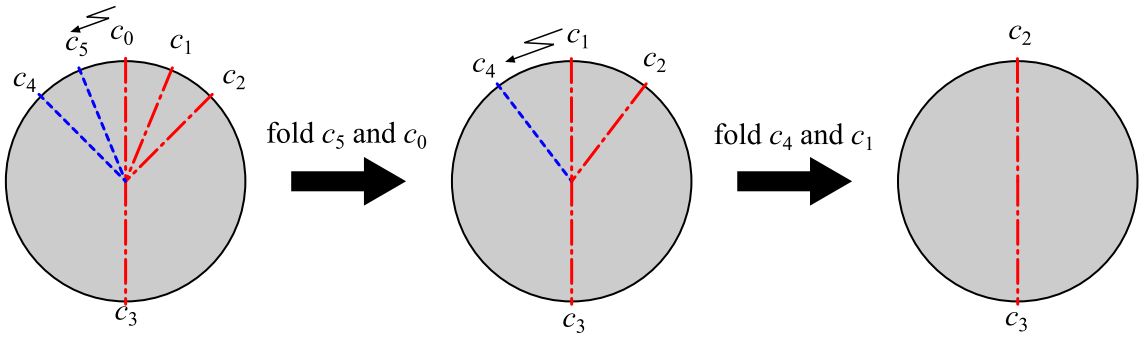



Figure 3.4: An example of a crimp operation. After monocrimping $(c_5, c_0)$, new MV pair $(c_4, c_1)$ appears and are folded by another monocrimp as a part of the crimp operation.

*flat-foldable MV-assigned SVCP. The difference in the number of M and V assignments for the creases in $\alpha$ is zero (one) if $\alpha$ has an even (odd) number of creases.*

In the case of a crimpable sequence $\alpha$ of odd length, we say that the crease remaining after a crimp operation on $\alpha$ *survives* the crimp. We note that the surviving crease in $\alpha$ is with *majority assignment* in $\alpha$ ([6, Observation 1]). Majority assignment denotes the assignment M or V which is major in a sequence or a set of creases.

### 3.2.2 End Creases [6]

*End creases* are the remains after *exhaustive crimps*. Exhaustive crimps mean repeating a crimp operation until there is no crimpable sequence (for example, see Figure 3.5). The following lemma holds for SVCP.

**Lemma 3.1** *The end creases of an flat-foldable SVCP form a flat-foldable SVCP of equal angles.*

To prove this lemma, we need the Maekawa Theorem (Theorem 2.2) and the following lemma:

**Lemma 3.2 (Corollary 12.2.11 from [8])** *An SVCP of equal angles is flat-foldable iff $|\#M - \#V| = 2$.*

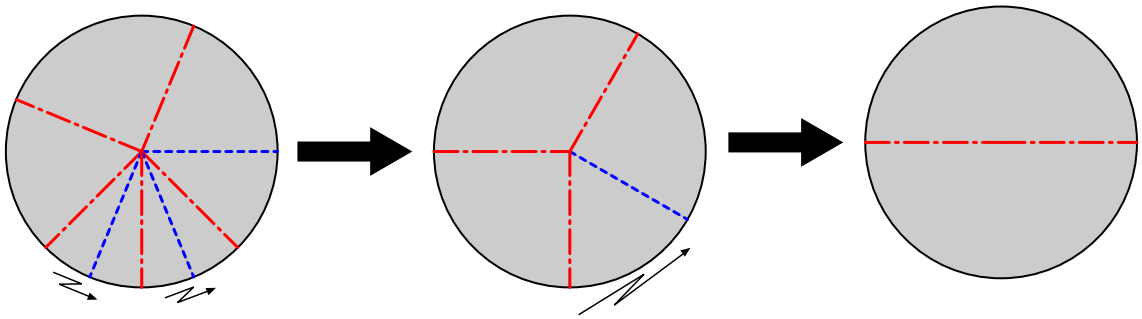


Figure 3.5: An example of exhaustive crimps. SVCP becomes equal angles by the exhaustive crimps.

Details about the Maekawa Theorem can be found in [8, Chapter 12]. Now let us prove Lemma 3.1.

Proof. We will make exhaustive crimps, that is, we will repeat crimps while processed $C$ satisfies $\theta_{i-1 \bmod n} > \theta_i = \theta_{i+1 \bmod n} = \cdots = \theta_{i+k-1 \bmod n} < \theta_{i+k \bmod n}$ for some $i$ and $k$. After this repetition, the crease pattern becomes one that consists of all equal angles as shown in Figure 3.5.

The original foldable $(C, \mu)$ satisfies the equation in Lemma 3.2 by the Maekawa Theorem. A monocrimp does not change the difference between the number of Ms and the number of Vs. Therefore after crimping all crimpable sequences in $(C, \mu)$, the crease pattern satisfies $|\#\mathrm{M} - \#\mathrm{V}| = 2$. By Lemma 3.2, the obtained SVCP of equal angles is flat-foldable. ∎

## 3.3 The Size of a Minimum Forcing Set of an SVCP

This section is devoted to proof of the theoretical minimum size of forcing sets.

### 3.3.1 SVCP of Generic Angles

In this section, let a given SVCP be of generic angles, that is, consecutive angles to be crimped always differ. Formally, SVCP is of generic angles if $\theta_i - \theta_{i+1} + \theta_{i+2} - \theta_{i+3} + \cdots + \theta_{j-1} \neq \theta_j - \theta_{j+1} + \theta_{j+2} - \theta_{j+3} + \cdots + \theta_{k-1}$

holds for any $i$, $j$, and $k$ where the length of each sequence is odd. (This definition is from [8, Subsection 12.2.2].) First we show the existence of $F$ with size $n/2$, then we prove that $F$ with size $n/2 - 1$ or less does not exist.

**Lemma 3.3** *There is a forcing set of an SVCP of generic angles, whose size is $n/2$.*

Proof. First we use a contradiction in order to show that there are always consecutive three angles which satisfy Lemma 2.1. Assume there are consecutive different angles $\theta_0, \theta_1, \ldots, \theta_{n-1}$, and any consecutive three of them do not satisfy Lemma 2.1. Then, for example, we can assume $\theta_0 > \theta_1 > \theta_2$. By the condition and assumption, $\theta_1 > \theta_2 > \theta_3$ holds. Similarly, $\theta_0 > \theta_1 > \theta_2 > \theta_3 > \theta_4 > \cdots$ holds and the sequence monotonically decreases. However, $\theta_{n-1} > \theta_0$ could never happen, which is a contradiction. Thus, we can always apply Lemma 2.1.

Applying Lemma 2.1 on $(\theta_{i-1 \bmod n}, \theta_i, \theta_{i+1 \bmod n})$ repeatedly, we can fold flat the sheet of paper. Let $(c_i, c_{i+1 \bmod n})$ be the pair of creases between the three angles. If we determine the assignment on one of $(c_i, c_{i+1 \bmod n})$, the assignment on the other of the pair is also determined. Hence we can make a forcing set by picking a crease in each pair as an element of the forcing set. We have $n/2$ such pairs because generic angles are the worst case of the number of such pairs. Therefore the size of the forcing set is $n/2$. ∎

**Lemma 3.4** *There is no forcing set of an SVCP of generic angles whose size is less than $n/2$.*

Proof. The proof is by contradiction. Assume a forcing set $F$ with size $n/2 - 1$ or less exists.

We monocrimp $(\theta_{i-1 \bmod n}, \theta_i, \theta_{i+1 \bmod n})$ according to Lemma 2.1. Every pair $(c_i, c_{i+1 \bmod n})$ is isolated from other pairs and there are $n/2$ pairs, thus every crease appears in a pair only once. Because $|F| < n/2$, there is an index $i$ such that both in $(c_i, c_{i+1 \bmod n})$ are not in $F$. This contradicts the

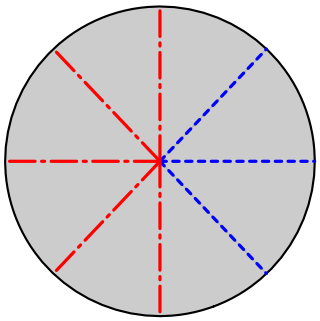


Figure 3.6: An example of flat-foldable equal-angle SVCP.

definition of $F$ because the sheet of paper folds flat in the following two cases: we assign $(M, V)$ on $(c_i, c_{i+1 \bmod n})$, or $(V, M)$ on $(c_i, c_{i+1 \bmod n})$. ∎

By Lemma 3.3 and Lemma 3.4, we obtain the following theorem.

**Theorem 3.2** *The size of a minimum forcing set for SVCP of generic angles is* $n/2$.

### 3.3.2 SVCP of Equal Angles

In this section, let a given SVCP be of equal angles, or *equal-angle* SVCP. (An example of equal-angle SVCP is in Figure 3.6.) Hence $\theta_i = \theta_{i+1 \bmod n}$ holds for any integer $i$ where $0 \leq i < n$.

**Lemma 3.5** *There is a forcing set of an equal-angle SVCP whose size is* $n/2 + 1$ *if* $n \geq 4$. *Furthermore, the forcing set is composed of all creases with majority assignment.*

Proof. Assume that $F$ consists of all majority M creases (thus all V creases are not in $F$). If $F$ is not a forcing set then we can choose some crease in $C \setminus F$ to be M, contradicting Lemma 3.2. ∎

**Lemma 3.6** *There is no forcing set of an equal-angle SVCP whose size is less than* $n/2 + 1$ *if* $n \geq 4$.

Proof. We prove it by contradiction. Assume $F$ is a forcing set of an equal-angle SVCP, whose size is $n/2$ or less. Then there may be a pair of

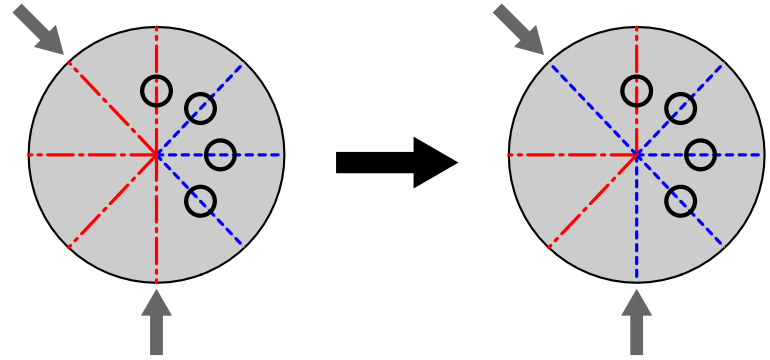


Figure 3.7: $F$ is not forcing: two Ms can be inverted to fold flat.

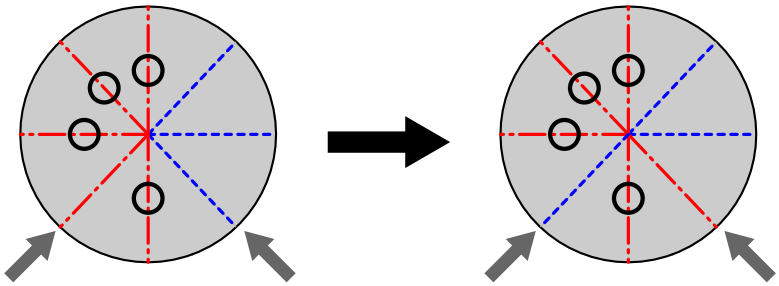


Figure 3.8: $F$ is not forcing: one M and one V can be inverted to fold flat.

an M crease and a V crease which are not in $F$ (Let M be the majority in the crease pattern). We denote such pair by $p$. We note that the creases in $p$ do not have to be consecutive.

If all V creases are in $F$, $p$ does not exist. In this case, we can invert the assignment of a pair of M creases in $C \setminus F$ to Vs (Figure 3.7), where the pair is not necessary to be consecutive. This operation holds Lemma 3.2, a contradiction.

Otherwise we can swap the MV assignment in $p$, and the resulting SVCP is flat-foldable by Lemma 3.2 (Figure 3.8). This is a contradiction to our assumption that $F$ is forcing. ∎

**Theorem 3.3** *Assume that a given SVCP is of equal angles. If the number of creases in the SVCP is two, then the minimum forcing set consists of one crease. Otherwise the size of the minimum forcing set of the SVCP is* $n/2 + 1$. *By Iverson's convention, it can be described as* $n/2 + [n \geq 4]$.

Proof. It is obvious if the number of creases in an equal-angle SVCP is two.

Lemma 3.5 and Lemma 3.6 imply that $n/2 + 1$ is the minimum size of $F$ if $n \geq 4$. ∎

### 3.3.3 General SVCP

Here we consider that a given SVCP has no constraints.

**Theorem 3.4** *Let $m$ be the number of monocrimps performed until the given SVCP becomes a flat-foldable equal-angle SVCP (cf. Lemma 3.1). $F$ denotes a minimum forcing set of the given SVCP. Then $|F| = n/2+[n-2m \geq 4]$.*

Proof. As the case of generic angles, we crimp the creases in crimpable sequences as many as possible. For each monocrimp, one of the creases in the pair must be in $F$. Such monocrimps contribute to $m$ elements in $F$.

After monocrimping $m$ times, the crease pattern has become a flat-foldable equal-angle SVCP (cf. Lemma 3.1). This equal-angle SVCP is composed of $n - 2m$ creases because two creases are consumed per one monocrimp. By Theorem 3.3, the size of a minimum forcing set of the equal-angle SVCP is $(n - 2m)/2 + [n - 2m \geq 4]$.

The minimum size of $F$ is the sum of the sizes of the two sets of forcing creases obtained above. This is because the sets do not have intersection and both are minimum. Thus, $|F| = m + (n - 2m)/2 + [n - 2m \geq 4] = n/2 + [n - 2m \geq 4]$. ∎

## 3.4 Constructing a Minimum Forcing Set

This section describes how to obtain a minimum forcing set for a given flat-foldable SVCP. We regard SVCP as a kind of 1D origami by the following way: cut away the inner space of the sheet of paper to make the sheet of paper a ring; the creases reduce to points on the ring, and the ring becomes 1D origami by cutting the ring at some point. Hence we can see that flat-foldable SVCP is a 1D origami with a constraint that the two end points of

the paper segment locate at the same point in the folded shape. Another difference between flat-foldable SVCP and ordinary 1D origami is that the end creases of flat-foldable SVCP are always equal angle (or equidistant from the viewpoint of 1D origami). Such conditions suggest a need to consider Lemma 3.5 in this section.

### 3.4.1 Crimp Forest Construction

We convert the crimp forest algorithm in [6] to an algorithm for SVCP by allowing circulation of the index of creases when finding a crimpable sequence. The converted algorithm is shown in Algorithm 1. A circulating crimpable sequence $(c_i, c_{i+1}, \ldots, c_0, c_1, \ldots, c_k)$ may occur when the algorithm finds and crimps crimpable sequences, but it does not change the behavior of the other parts of the algorithm. The algorithm constructs a forest in bottom-up manner. The edges are added if the sequence in the parent node includes the crease surviving the crimp on the sequence in child node. Figure 3.9 depicts an example of a crimp forest.

**Algorithm 1:** CrimpForestSVCP($C$, $\mu$)

```
Initialize W ← ∅
while C has a crimpable sequence do
    Let s be the crimpable sequence in C with the smallest starting
      index. // modified from [6].
    create a node v corresponding to s, and add v to W.
    Make v the parent of each root node in W whose crimpable
     sequence has a surviving crease that is in s.
    Apply the crimp operation to s.
    Update C to be the resulting crease pattern.
end
return W
```

A straightforward implementation of Algorithm 1 takes $O(n^2)$ time because a naive way to find a crimpable sequence takes $O(n)$ time: start searching from $c_0$ clockwisely; skip monotonically nonincreasing angles; stop at the right side crease $c_r$ which satisfies $\theta_{r-1 \bmod n} < \theta_r$; counterclock-

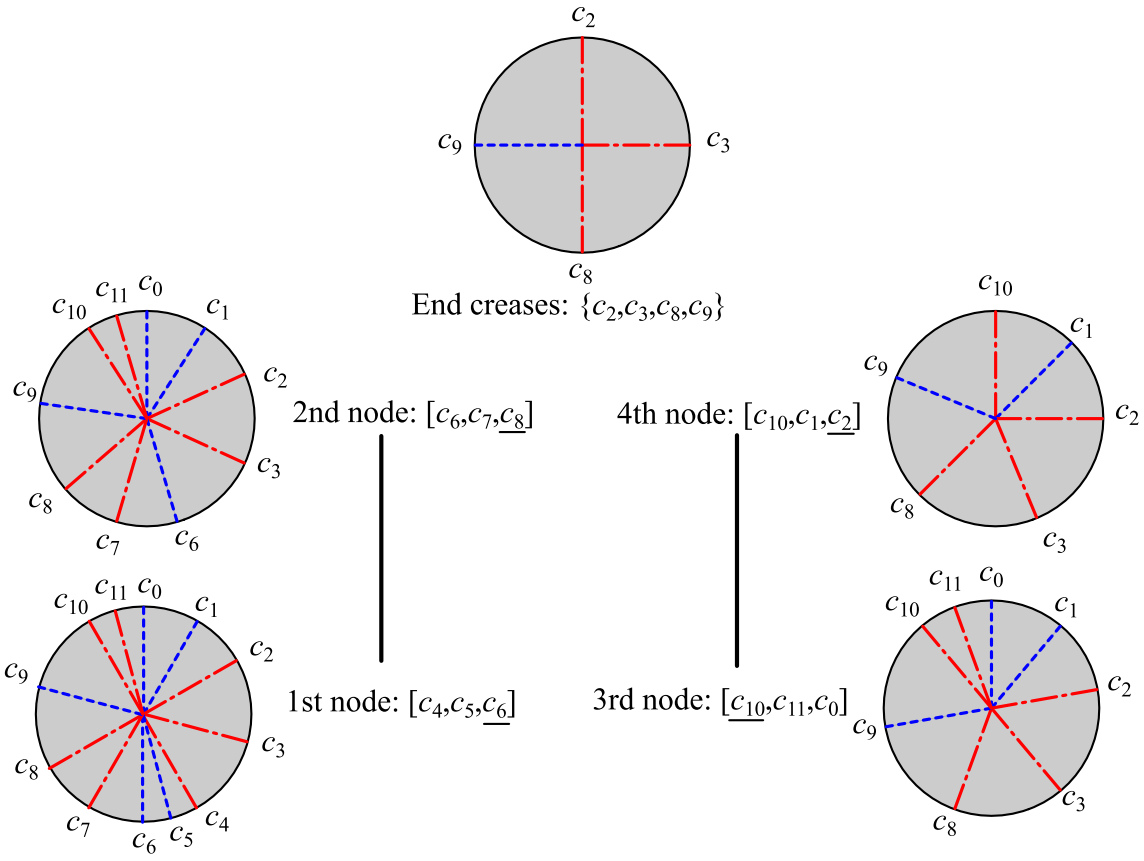



Figure 3.9: An example of a crimp forest construction. $[\cdot]$ is a crimpable sequence to be crimped. The surviving creases are underlined. Inclusion of a surviving crease is presented as an edge of a tree.

wisely from $c_r$, search the left side crease $c_l$ which satisfies $\theta_{l-1 \bmod n} > \theta_l$; other operations can be done in constant time; since the algorithm loops at most $n$ times, the time complexity of the algorithm is $O(n^2)$.

The following lemma describing the properties of crimp forest holds for SVCP as well.

**Lemma 3.7 (Lemma 4 from [6])** *Given a crease pattern $C$ and two foldable MV assignments $\mu_1$ and $\mu_2$, let $W_1$ and $W_2$ be the crimp forests corresponding to $(C, \mu_1)$ and $(C, \mu_2)$, respectively. Then the following properties hold:*

*(1). $W_1$ and $W_2$ are structually identical.*

*(2). Corresponding nodes in $W_1$ and $W_2$ have crimpable sequences of the same size and the same interval angles between adjacent creases.*

*(3). Creases involved for the first time in a crimpable sequence at a node in $W_1$ have the same position in the crimpable sequence at the corresponding node in $W_2$.*

### 3.4.2 Forcing Set Algorithm

We convert the forcing set algorithm in [6] by three modifications: switch CrimpForest($C$, $\mu$) to CrimpForestSVCP($C$, $\mu$); initialize $F$ to the majority of end creases according to Lemma 3.5 instead of all end creases; remove one crease from $F$ if $|F| = 2$ in the initialization according to Theorem 3.3. See Algorithm 2 for the detail.

**Algorithm 2:** ForcingSetSVCP($C$, $\mu$)

```
Initialize W to the output generated by CrimpForestSVCP(C, μ)
    // modified from [6].
Initialize F to the all creases with majority assignment in end
    creases that remain after running CrimpForestSVCP(C, μ)
    // modified from [6].
if |F| = 2 then // added to [6]
    Remove one crease from F.
end
foreach tree T ∈ W do
    foreach node v in a preorder traversal of T do
        if v's crimpable sequence has even length then
            Add to F all creases from v's crimpable sequence having
             M assignment.
        else if the surviving crease from v's crimpable sequence is
         already in F then
            Add to F all creases from v's crimpable sequence having
             the majority MV assignment.
        else
            Add to F all creases from v's crimpable sequence having
             the minority MV assignment.
        end
    end
end
```

The preorder traversal takes $O(n)$ time because each node is visited only once and the sum of lengths of the sequences in the nodes is $n$. Thus the main factor of computation time is CrimpForestSVCP, which takes $O(n^2)$ time.

We need the following lemma for the proof in Section 3.5:

**Lemma 3.8 (Lemma 6 from [6])** *Let $(C, \mu_1)$ be a foldable MV-assigned crease pattern, and let $F$ be the forcing set generated by Algorithm 2 with input $(C, \mu_1)$. Let $(C, \mu_2)$ be a foldable pattern such that $\mu_2$ agrees with $\mu_1$ on the forcing set $F$, that is, $\mu_2(c) = \mu_1(c)$ for $c \in F$. Let $T_1$ and $T_2$ be two structurally equivalent trees generated by the forcing set algorithm $(C, \mu_1)$ and $(C, \mu_2)$, respectively. If a crease $c$ in a crimpable sequence $\alpha_1 \in T_1$ is in $F$, then a crease (not necessarily $c$) with the same MV assignment occurs in the corresponding crimpable sequence $\alpha_2 \in T_2$, in the same position as in $\alpha_1$.*

## 3.5 Proof of Correctness

This section proves that $F$ created by Algorithm 2 is forcing and minimum. The proof is almost the same as [6] because Damian et al. use local properties of crimpable sequence and abstract properties of crimp forest, which are not affected by the change from 1D to SVCP. In this section, we organize the proof in [6] to follow and prove the different points.

Assume that there exists a different foldable MV assignment $\mu_2$ for $C$ such that $\mu_2(c) = \mu(c)$ for $c \in F$. For symmetry, let $\mu_1 = \mu$. We obtain $W_1$ and $W_2$ by running ForcingSetSVCP with input $(C, \mu_1)$ and $(C, \mu_2)$, respectively. As stated in Lemma 3.7, $W_1$ and $W_2$ are structurally identical. Let corresponding nodes $v_1 \in T_1$ and $v_2 \in T_2$ be a pair of maximal depth in the trees whose assignments differ. We call two crimpable sequences $\alpha_1$ and $\alpha_2$ *similar* if they have the same size, the same MV assignment read from left to right, and same interval angles.

The proof in [6] for forcing property is by contradiction with a case analysis as follows:

1. $v_1$ and $v_2$ are dissimilar. Let $l$ be the length of the crimpable sequences corresponding to $v_1$ and $v_2$.
   (a) $l$ is even.
   (b) $l$ is odd.
      i. The creases of $v_1$ with majority MV assignment are in $F$.
      ii. The creases of $v_1$ with minority MV assignment are in $F$.
2. All corresponding nodes in $W_1$ and $W_2$ have similar crimpable sequences.

Case 1a leads to a contradiction as shown in [6]. In this case, $v_1$ and $v_2$ are root nodes in $T_1$ and $T_2$ because they do not have surviving crease. $l/2$ creases of $v_1$ are put into $F$ with M assignment by the algorithm. The creases with M assignment in $F$ have a copy in $v_2$ by Lemma 3.8, and remaining creases must have V assignment by Theorem 3.1. Then $v_1$ and $v_2$ are similar, which is a contradiction to the assumption that $v_1$ and $v_2$ are dissimilar.

Case 1(b)i also contradicts as shown in [6]. By Lemma 3.8, the creases with majority MV assignment of $v_1$ have a copy in $v_2$ with the same MV assignment and located in the same positions. All other creases in $v_1$, $v_2$ must have the opposite assignment by Theorem 3.1. Thus $v_1$ and $v_2$ are similar, a contradiction.

The difference is in Case 1(b)ii and Case 2. In Case 1(b)ii, we have two new cases due to the second and third steps of Algorithm 2:

A. The survivor of the root node is in $F$. (Hence the majority in the root node are in $F$.)

B. The survivor of the root node is not in $F$. (Hence the minority in the root node are in $F$.)

The proof for Case A is the same as the proof of Case 1(b)ii shown in [6]. Assume without loss of generality that the minority assignment of $v_1$ is M. In Case B, we must encounter a node with majority assignment V, or an equal number of M and V assignment. Assume to the contrary that we encounter nodes with majority M assignments only. At the root node $r_1$ of $T_1$, V creases are selected as a part of $F$ since we assume surviving crease is not in $F$. Similarly, on each node from $r_1$ to $v_1$, V creases are selected as elements of $F$, which contradicts the assumption that the minority M creases of $v_1$ are in $F$. Let $w'_1$ be the first node encountered on the path from $v_1$ to $r_1$ of $T_1$ having majority assignment V or an equal number of M and V assignments. As addressed in [6], the differences in $v_1$, $v_2$'s crimpable sequences must be in first-time creases.

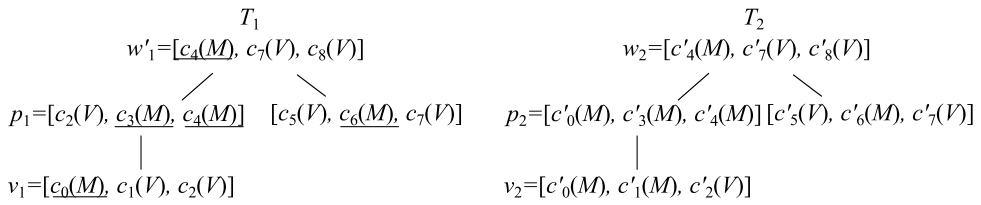



Figure 3.10: The case that first-time creases are different between $v_1$ and $v_2$, and $w'_1$ have majority assignment V. Underlined are the creases in $F$.

Let $p_1$ be the parent node of $v_1$. In Case B, if $p_1 \neq w'_1$, $p_1$'s majority are M (by definition of $w'_1$) and its creases with M should be in $F$ (otherwise it contradicts the assumption that minority of $v_1$ are in $F$). The difference of first-time creases in $v_1$ and $v_2$ causes a contradiction of Theorem 3.1 on $p_2$ in the following cases: (1) $w'_1$ has majority assignment V; (2) $w'_1$ has equal M and V assignments. Figure 3.10 shows the first case. Assume $c_1$ and $c'_1$ in Figure 3.10 are first-time creases and $c'_0$ survives a crimp operation. Then $c_3$ and $c_4$ are copied to $c'_3$ and $c'_4$ by Lemma 3.8. This contradicts Theorem 3.1 on $p_2$.

In Case 2, the end creases form a flat-foldable equal-angle SVCP (cf. Lemma 3.1). Because ForcingSetSVCP puts all majority creases of the equal-angle SVCP into $F$, the remains of the creases in the equal-angle SVCP are forced to be with minority assignment, and it is not possible that

$\mu_1$ and $\mu_2$ differ on the equal-angle SVCP. It follows $\mu_1 = \mu_2$, and therefore $F$ is a forcing set.

We have shown that the theoretical minimum size of $F$ is $n/2+[n-2m \geq 4]$ where $m$ is the number of monocrimps performed in exhaustive crimps (cf. Theorem 3.4). Here we show how $F$ yields $n/2 + [n - 2m \geq 4]$ creases by ForcingSetSVCP. The creases from the end equal-angle SVCP added to $F$ in the second and third steps in the algorithm contributes to $(n-2m)/2+[n-2m \geq 4]$ creases. The same argument as [6] can be applied for the crimped creases: corresponding to each crimpable sequence $\alpha$ with size $l$, the forcing set algorithm adds to $F$ precisely $\lfloor l/2 \rfloor$ creases; summing up over all crimp performed by the algorithm, we get $m$ creases contributed to $F$.

## 3.6 Conclusion

This is the first attempt to generate and analyze a minimum forcing set of flat-foldable SVCP. We have developed an algorithm[1] to find a minimum forcing set of flat-foldable SVCP in $O(n^2)$ time by converting existing algorithm for 1D origami. We proved that the size of such forcing set is $n/2$ or $n/2 + 1$. The proof of the correctness of the algorithm was done with the framework used in a proof in prior research for 1D origami. Minimum forcing set of arbitrary 2D origami is attractive as future work.

[1]The implementation is at https://ouchi-koji.visualstudio.com/_git/ForcingSet.

# Chapter 4

# Efficient Enumeration of Flat-Foldable Single-Vertex Crease Patterns

This chapter is based on the author's papers published as [OU17, OU19a]. We investigate enumeration of distinct flat-foldable MV-assigned SVCP under the following assumptions: positive integer $q$ is given; every pattern is composed of $q$ *potential lines* incident to the center of a sheet of paper; every angle between adjacent potential lines is equal to $(360/q)^\circ$; every potential line is classified to "crease" or "flat," and "crease" lines are then assigned with "mountain" or "valley"; MV-assigned crease patterns are considered to be equivalent if they are equal up to rotation and reflection. In this natural problem, we can use two well-known theorems for flat foldability: the Kawasaki Theorem and the Maekawa Theorem in computational origami. Unfortunately, however, they are not enough to characterize all flat-foldable crease patterns. Therefore, so far, we have to enumerate and check flat foldability one by one using computer. In this study, we develop the first algorithm for the above stated problem by combining these results in a nontrivial way and show its analysis of efficiency.

## 4.1 Introduction

Recent origami is a kind of art, and origamists around the world struggle with their problems; what is the best way to fold an origami model? One of these problems is the issue of a unit of angle that appears in the origami model. Some origamists restrict themselves to use only multiples of a unit angle which divides 360°, e.g., 22.5°, 15°, and so on. A nontrivial example, which was designed by the author, is shown in Figure 4.1. It is based on a unit angle of 15°. Once origamists fix the unit angle as $(360/q)^\circ$ for suitable positive integer $q$, their designs are restricted to one between quite real shapes and abstract shapes, which is the next matter in art.

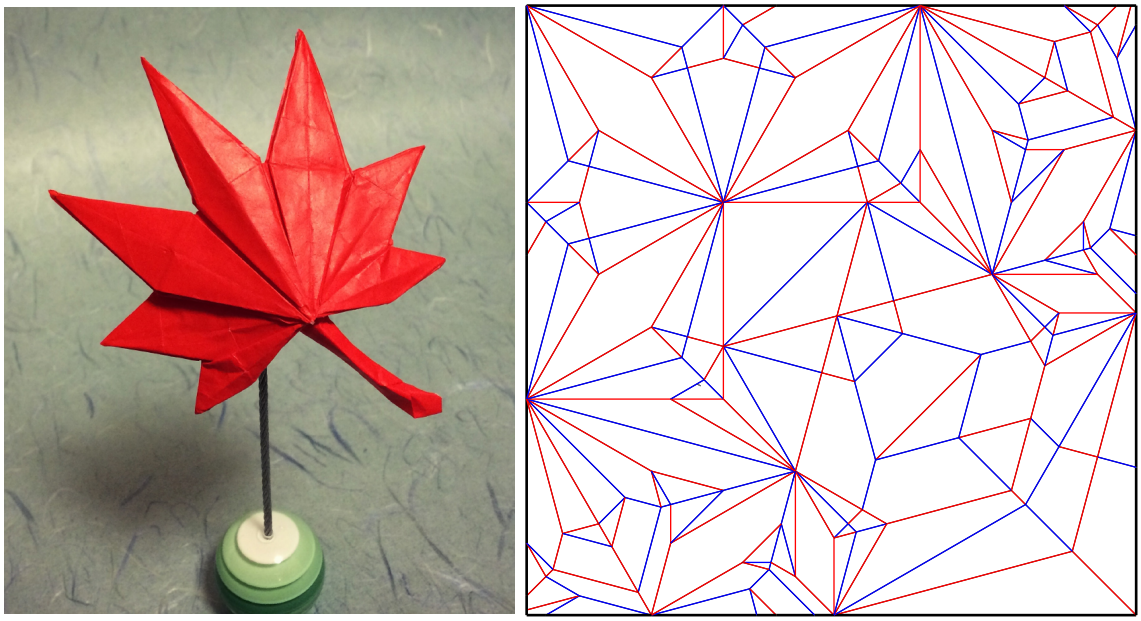


Figure 4.1: "Maple leaf" designed and folded by the author (left). Its crease pattern is based on 15° unit angle (right).

When we are given a positive integer $q$, we face a computational origami problem which is interesting from the viewpoints of discrete mathematics and algorithms. We consider the simplest origami model which is a kind of MV-assigned SVCPs; all creases are incident to the single vertex at the center of origami, and each angle between two creases is a multiple of $(360/q)^\circ$. We are concerned with only flat-foldable MV-assigned crease patterns.

We note that the ordering of the layers of paper is not given, and it is not easy to compute it even if an MV assignment is given. The flat foldability of MV-assigned SVCP can be computed in linear time [4, 8]. In fact, the algorithm also gives us the ordering of the layers in the same time. However, its rigorous proof is not so simple, which is the main topic of Chapter 12 in [8]. Roughly speaking, the algorithm repeatedly folds and glues the locally smallest angle in each step. In other words, we have no mathematical characterization for this problem, and we have to check one by one.

The problem of computing a folding for a crease pattern is very different from the case of MV-assigned crease pattern. Hull investigated this problem from the viewpoint of counting [13]. Precisely, he considered the number of flat-foldable MV assignments to a given crease pattern of $n$ creases which were incident to the single vertex. In [13], he gave tight lower and upper bounds. These bounds are given in two extreme situations; one is given in the case that all $n$ angles are different, and the other is given in the case that all $n$ angles are equal to each other. From the viewpoint of origami design, we are interested in the case between these two extreme situations. To deal with reasonable situations between extreme ones, we slightly modify the input of the problem. The input of our problem is a positive integer $q$, and we restrict ourselves to the single vertex folding of unit angle $(360/q)^\circ$. We place *$q$ potential lines* incident to the vertex with unit angle $(360/q)^\circ$. A potential line may eventually become a crease. In order to investigate our problem, we first give a label "crease" or "flat" to each potential line to generate crease patterns, then assign "mountain" or "valley" to each "crease" lines to generate MV-assigned crease patterns. When a potential line is labeled "flat," this line is not folded in the final folded state. Let us call a potential line with a label "crease" a *crease*. In this way, we can deal with the SVCPs and MV-assigned SVCPs of unit angle equal to $(360/q)^\circ$, which is more realistic situation from the viewpoint of origami design.

Our aim is to enumerate all distinct flat-foldable assignments of "mountain," "valley," and "flat" labels on $q$ potential lines. In other words, our

algorithm eventually enumerates all flat-foldable MV-assigned SVCPs of unit angle $(360/q)^\circ$. We consider the sheet of paper is a disk, the vertex is at the center of the disk, and two crease patterns (or MV-assigned crease patterns) are considered to be equivalent if they can be equal up to rotation and reflection (i.e., including turning over and exchanging all mountains and valleys). Our algorithm enumerates all distinct MV-assigned SVCPs under this assumption.

For flat foldability of a given MV-assigned SVCP, there are two well-known theorems in the area of computational origami, which are called the Kawasaki Theorem and the Maekawa Theorem (Theorem 2.1 and Theorem 2.2). We note that the Kawasaki Theorem gives a necessary and sufficient condition for flat foldability, however, MV assignments are not given. That is, we have to compute foldable MV assignments for foldable SVCP satisfying the Kawasaki Theorem. In order to compute a flat-foldable MV assignment, we can use the Maekawa Theorem. Note that the Maekawa Theorem is a necessary but not sufficient condition.

In the last decades, enumeration algorithms have been well investigated, and many efficient enumeration algorithms have been given, e.g., [2, 28, 27]. Using techniques that follow above properties of origami, we construct an enumeration algorithm for flat-foldable MV-assigned crease patterns for given $q$, where each angle between two crease lines is a multiple of $(360/q)^\circ$. As far as the author knows, this is the first algorithm for the realistic computational origami problem. As a result, we succeeded to enumerate flat-foldable crease patterns up to $q = 40$ in a reasonable time.

## 4.2 Outline of Algorithm

Based on the Kawasaki Theorem and the Maekawa Theorem, for a given $q$, we can design the outline of our enumeration algorithm as follows:

(1) Assign "crease" or "flat" to each of $q$ potential lines incident to the single vertex so that the Kawasaki Theorem is satisfied. The result

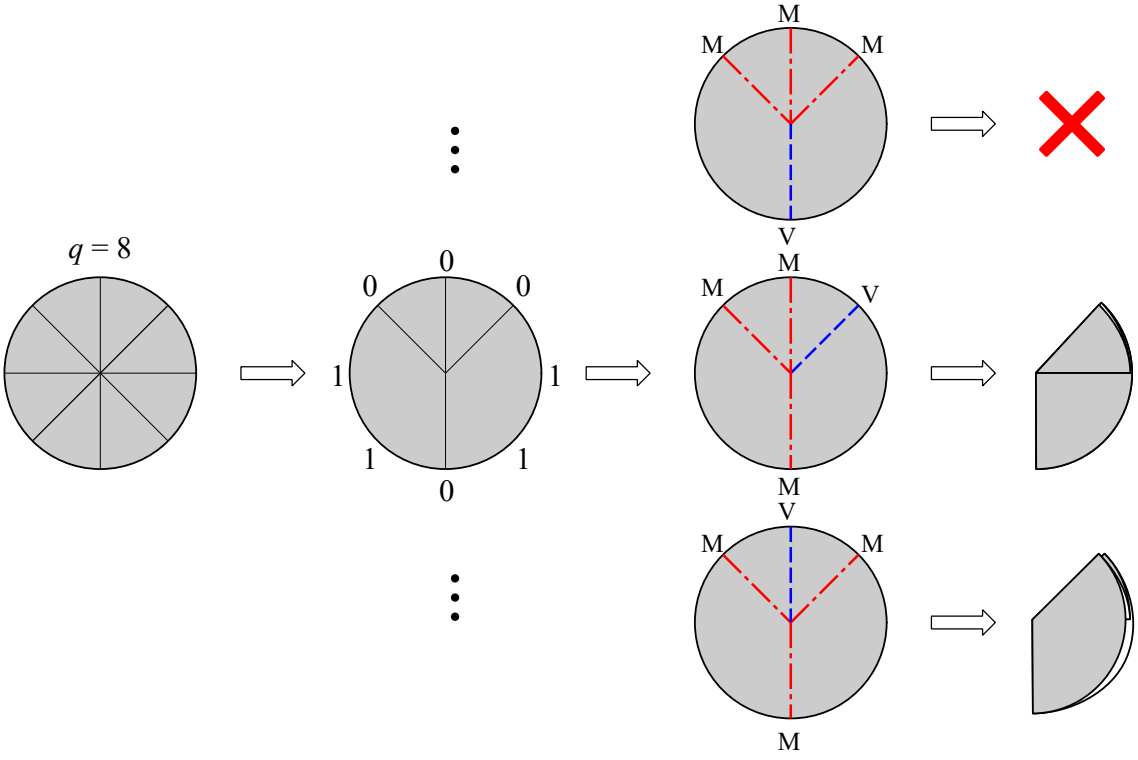



Figure 4.2: Simple example for $q = 8$.

of the assignment is a flat-foldable crease pattern.

(2) For each crease, assign "mountain" or "valley" so that the Maekawa Theorem is satisfied.

(3) Output the pattern if this MV-assigned crease pattern is flat-foldable.

Essentially, the outline consists of two different kinds of enumeration problems in phases 1 and 2, and flat foldability checking in phase 3. We note that the algorithm reduces equivalent crease patterns in phase 1 and MV-assigned crease patterns in phase 2 up to rotation and reflection.

A simple example is given in Figure 4.2. For $q = 8$, we first generate all possible crease patterns in phase 1 which is described in a binary string (in the figure, we show only one, but there are exponentially many). Here "0" and "1" denote "crease" and "flat" respectively. Therefore, for a string 00011011, we have four creases in the shape in Figure 4.2. In phase 2, we assign mountain (M) or valley (V) to each crease. In phase 3, we check

whether each obtained MV-assigned crease pattern is flat-foldable or not, and output the pattern if it is flat-foldable.

We have different issue for each phase. Especially in phases 1 and 2, we have to consider two different problems of symmetry (to reduce redundant output) and enumeration.

## 4.3 Description of Algorithm

Now we describe more details in each phase.

### 4.3.1 Phase 1: Assignment of "crease"/"flat"

In phase 1, we are given $q$ potential lines, and we have to assign "crease" or "flat" to them so that the assignment satisfies the Kawasaki Theorem. Since the crease pattern cannot be flat-folded for odd number $q$, without loss of generality, we assume that $q$ is even hereafter.

In this phase, we describe "crease" by 0 and "flat" by 1, and consider a binary string. Then it is easy to see that, before checking the Kawasaki Theorem, we have to generate all binary strings over $\Sigma = \{0, 1\}$ efficiently reducing equivalent rotations and reflections. To solve this problem, we introduce the bracelet problem, which is a classic and basic problem in combinatorics. A *bracelet* is an equivalence class of strings, taking all rotations and reversals as equivalent. This is a special case of a *necklace* whose equivalence is rotation only. In this paper, let the word *bracelet* also denote the lexicographically smallest string of the equivalence class and so does *necklace*. It is easy to observe that our problem is now enumeration of binary bracelet of length $q$. For bracelets, we have an optimal enumeration algorithm [23]:

**Theorem 4.1 (Sawada 2001)** *Bracelets of length q can be enumerated in constant amortized time.*

That is, the algorithm in [23] runs in a time proportional to $B(q)$ which denotes the number of bracelets of length $q$.

We note that the values of the function $B(q)$ are listed in the OEIS (The On-line Encyclopedia of Integer Sequences; http://oeis.org/) as A000029, and it is given as

$$B(q) = \sum_{d \text{ divides } q} \frac{2^{q/d}\phi(d)}{2q} + 2^{q/2-1} + 2^{q/2-2} \tag{4.1}$$

for an even number $q$, where $\phi()$ is Euler's totient function.

### 4.3.2 Phase 1: Satisfying the Kawasaki Theorem

After assigning "crease" or "flat" to each potential line, we have to check whether the obtained crease pattern satisfies the Kawasaki Theorem or not. The Kawasaki Theorem states that the alternating sum of angles should be equal to 0. This notion corresponds to a kind of necklace in a nontrivial way as follows. We first observe that each angle $\theta_i$ is $k \times \frac{360^\circ}{q}$ for given even $q$. That is, $\theta_i$ consists of $k$ unit angles. Now we regard $\theta_i$ as the integer $k$, and we consider $\theta_1, \theta_3, \ldots$ as "white", and $\theta_2, \theta_4, \ldots$ as "black". Then the total number of beads is $q$, and the Kawasaki Theorem states that the number of black beads is equal to the number of white beads. Precisely, each sequence of $n$ creases satisfying the Kawasaki Theorem corresponds to a necklace with $q$ beads such that (1) the necklace consists of $q/2$ white beads and $q/2$ black beads, and (2) the number of runs[1] of white beads (and hence black beads) is $n$. This notion is investigated as "balanced twills on $q$ harnesses" in [12] and listed in OEIS as A006840. Then the number is given as follows:

**Theorem 4.2 (Hoskins and Street 1982)** *The number of distinct balanced*

[1] A *run* is a maximal sequence of beads of the same color.

*twills on $q = 2k'$ harnesses is*

$$B'(2k') = \frac{1}{8k'} \left\{ \begin{array}{l} \displaystyle\sum_{\substack{d \text{ divides } q \\ d=2e}} \phi\left(\frac{k'}{e}\right)\binom{2e}{e} \\ \displaystyle + \sum_{d \text{ divides } k'} \phi\left(\frac{2k'}{d}\right) 2^d \\ + 2k'\binom{2\lfloor k'/2 \rfloor}{\lfloor k'/2 \rfloor} + k'2^{k'} \end{array} \right\}. \tag{4.2}$$

We note that Equation 4.2 just gives us the numbers for each $q$, and no concrete sets of creases. Therefore, we have to enumerate them by ourselves. A straightforward approach is to insert a test of the Kawasaki Theorem into Sawada's algorithm [23]. The test computes $\sum_{i=1}^{n}(-1)^i\theta_i$ and checks whether the value is 0 or not. Note that $n$ is the number of creases, and $\theta_i$ is the angle between the $i$th and $(i+1)$st creases as defined in Theorem 2.1.

Fortunately, the test can be amortized if the straightforward approach is applied. Sawada's algorithm is a recursive function that always determines the letters in a string sequentially from smaller index to larger index. That means, if the algorithm sets "crease" in a recursive call, we can compute the angle between the new "crease" line and the prior (and adjacent) "crease" line. It takes just constant time if the last index of "crease" line is passed to the recursive call. The obtained angle is used to calculate $\sum_{i=1}^{m}(-1)^i\theta_i$ where $m$ is the index of the currently last "crease". The alternative sum can be updated in constant time by passing to the next call the current value and either + or − to be used. When the recursive call comes to output, the Kawasaki Theorem holds if the alternative sum including the angle between the last "crease" and the first "crease" is 0.

Now we have the following theorem:

**Theorem 4.3** *For a given even number $q$, phase 1 can be done in $O(B(q))$ time, where $B(q)$ is the number of bracelets of length $q$.*

Furthermore, we can prune the search tree with the following corollary derived from the Kawasaki Theorem:

**Corollary 4.1** *If a given SVCP is flat-foldable,* $\left|\sum_{i=1}^{m}(-1)^i\theta_i\right| \leq \sum_{j=m+1}^{n}\theta_j$ *holds for any integer* $m$ *where* $1 \leq m \leq n-1$.

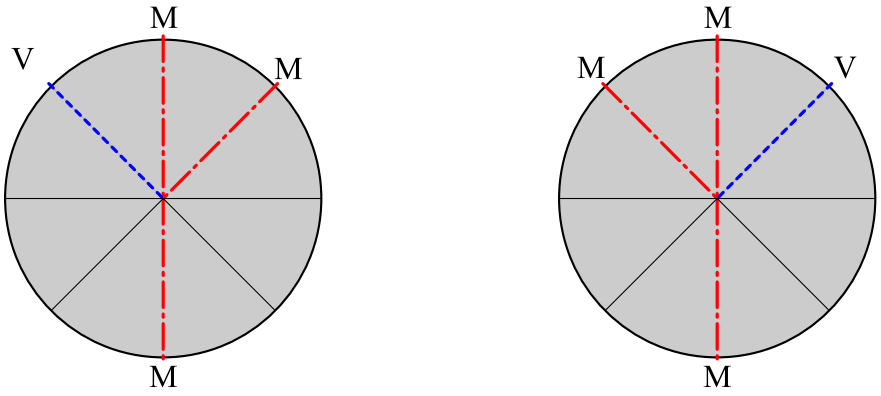
### 4.3.3 Phase 2: Assignment of "mountain"/"valley"

In this phase, we inherit a binary string of length $q$ from the phase 1, which describes "crease" (=0) or "flat" (=1). We note that the binary string is the lexicographically smallest one among rotations and reversals. Then we translate it to a set of other strings that represent the assignments of "mountain" and "valley" and the angles between adjacent creases. The first step can be described as follows:

(2a) For each adjacent pair of 0s, replace 1s between them by the number of 1s plus 1. For example, the string 00011011 in Figure 4.2 is replaced by $0\underline{1}0\underline{1}0\underline{3}0\underline{3}$, where the positive (underlined) numbers describe the number of unit angles there.

Then we assign mountain (= $M$) and valley (= $V$) to each 0, but here we only consider the assignments that satisfy the Maekawa Theorem. The Maekawa Theorem says that the number of $M$s and the number of $V$s should differ by 2. To avoid symmetry case, we can assume that (the number of $M$s)−(the number of $V$s)= 2. Thus the next step is described as follows:

(2b) For the resulting string over $\{0, 1, 2, \ldots, q-1\}$, assign all possible $M$s and $V$s to each 0 such that the number of $M$s is 2 larger than the number of $V$s. For example, for the string 01010303, we obtain the set of strings $\{V1M1M3M3, M1V1M3M3, M1M1V3M3, M1M1M3V3\}$. We note that we can prune the search tree by Lemma 2.2: If $k$ creases $(c_i, c_{i+1}, \ldots, c_{i+k-1})$ form a minimal equal angle sequence, i.e., $\theta_{i-1} > \theta_i = \theta_{i+1} = \cdots = \theta_{i+k-2} < \theta_{i+k-1}$ holds, the number of majority assignments on the $k$ creases is $\lceil k/2 \rceil$.




(a) V1M1M3M3 (b) M1M1V3M3

Figure 4.3: An example of possible mirror image on MV assignment. The letters at even indices differ but the letters at odd indices are equal between (a) and (b). (Assume that the index starts from 0.)

For a string $s$ generated by step 2a, which describes a crease pattern, we can have equivalent MV-assigned crease patterns. Precisely, if some rotation(s) or reversal(s) of $s$ is (are) equal to $s$, the result of step 2b may contain equivalent assigned crease patterns. For example, in the set of strings $\{V1M1M3M3,\ M1V1M3M3,\ M1M1V3M3,\ M1M1M3V3\}$, we can observe that $V1M1M3M3$ (Figure 4.3a) is a crease pattern which is the mirror image of a crease pattern $M1M1V3M3$ (Figure 4.3b), hence we consider they are equivalent. (In Figure 4.2, after phase 2, the crease pattern at the center has its mirror image, and it should be omitted.) To avoid such equivalent patterns, we perform the following:

(2c) For the resulting string $s'$ over $\{M, V, 1, 2, \ldots, q-1\}$ after step 2b, generate the lexicographically smallest string among rotations and reversals of $s'$, which we call $s'_{small}$, and store all $s'_{small}$. $s'$ is discarded if $s'_{small}$ has been already obtained. Note that $M < V < 1 < 2 < \cdots$.

In this process, we take a caching strategy to detect duplications; For every $s'$, we generate and store a representative of the bracelet equivalence class to which $s'$ belongs, and we refer to the representatives generated so far to check whether we have obtained an equivalent of $s'$ or not. The string $s'_{small}$ can be one of such representatives because the lexicograph-

ically smallest string is easy to be generated and unique among rotations and reversals. Because of the exponential number of strings to be cached, we use a trie [7, 9] (a.k.a. prefix tree) that is a space-efficient data structure for storing many strings. The reason to store $s'_{small}$ is that some assignments can be unique but not the lexicographically smallest. For example, assume that preprocessed "crease"/"flat" assignment "010101010202" is generated by phase 2b, which is the smallest among its equivalents. Then "V1M1M1M1V2M2" is a distinct crease pattern on it. However, the equivalent smallest string is "M1M1M1V2M2V1" which should be generated from discarded "010101020201."

To generate $s'_{small}$, we use Booth's least circular string algorithm [5]. It is a linear time algorithm to find the smallest string among rotations of a given string. Note that the algorithm doesn't care about reversals. Precisely, Booth's algorithm finds the *right index* of the lexicographically smallest string for a given circular string of length $n$ in linear time. The *right index* is the start index of a circular string that may be larger than (or on the "right" side of) the original start index 0, which is a conventional description in the field of string algorithms. To deal with both rotation and reversal, the step 2c can be implemented as follows:

(2c-1) For the resulting string $s'$ over $\{M, V, 1, 2, \ldots, q-1\}$ after step 2b, let $s'^R$ is the reverse string of $s'$. Prepare an empty trie.

(2c-2) Using Booth's algorithm, find the right index $i$ of a circular string $s'$ such that the string starting from the index $i$ is the lexicographically smallest string among all rotations of $s'$. If $i$ is not the first letter in $s'$, we discard this $s'$ since it is redundant.

(2c-3) Similarly, find the right index $j$ of the lexicographically smallest string among all rotations of $s'^R$. The index $j$ gives the smallest string among the equivalents of reversals.

(2c-4) Select the smaller string as $s'_{small}$ from the result of (2c-2) and (2c-3): rotation of $s'$ starting from $i$ and rotation of $s'^R$ starting from $j$.

If $s'_{small}$ is already in the trie, discard $s'$. Otherwise append $s'_{small}$ to the trie and $s'$ goes to phase 3 to be processed.

This test takes $O(q)$ time because the steps don't contain loops and recursions, but it runs linear time subroutines just constant times, which are Booth's algorithm, string comparison, and operations on a trie. Summarizing, we have the following theorem:

**Theorem 4.4** *For a given crease pattern from phase 1 based on q unit angles, we can generate all distinct MV assignments that satisfy the Maekawa Theorem in $O(qC(q))$ time with space linear in the product of q and the number of such assignments, where $C(q)$ is $\binom{q}{q/2-1}$.*

Proof. The number of creases in the crease pattern is at most $q$, and the number of $M$s is 2 larger than the number of $V$s. Thus, the number of strings $s'$ over $\{M, V, 1, 2, \ldots\}$ with the constraint for the number of $M$s and $V$s is at most $\binom{q}{q/2-1}$. Other management can be done in linear time, which implies the time complexity in the theorem. The space complexity is linear in the maximum number of nodes in the trie used in the algorithm, which can be suppressed by the product of $2q$ (the maximum length of $s'$) and the number of desired assignments. ∎

### Non-Caching Strategy

We can remove duplications of MV assignment without storing the representative patterns, which takes $O(q^2)$ time for the test but may be practically faster than the caching strategy. Duplications in phase 2 can be generated by rotating (or reflecting) an MV-assigned SVCP so that the "crease"/"flat" assignment does not change. Let us call such a rotation and a reflection an *MV rotation* and an *MV reflection*, respectively. We swap step 2b and 2c for the following step:

(2b') For a string $s$ obtained by step 2a, assign all possible $M$s and $V$s avoiding the MV rotations and MV reflections such that the assignments satisfy the Maekawa Theorem. We conduct a depth first search

that determines $M$ or $V$ on even indices of $s$. (We assume that the index of $s$ starts from 0 in this section.) The search can be seen as generating strings of length $n$ over $\{M, V\}$ where $n$ is the number of creases.

By a property of a depth first search for generating binary strings of fixed length, we can generate the strings for MV assignments with no duplications as follows:

(2b'-1) Copy $s$ to a string $s'$ and initialize the MV assignment on $s'$ by $M$. We are assigning $V$s to the letters at even indices of $s'$ from the start of $s'$ to the end of $s'$ by depth first search. The underlying search tree for $s = 01010101$ is shown in Figure 4.4.

(2b'-2) Assume that we have determined the MV assignment on the first $k$ creases of $s'$ by the depth first search. Let $p$ be such a prefix. The length of $p$ is $2k - 1$.

(2b'-3) Compare $p$ with the MV rotations and MV reflections of $s'$ in lexicographic order. If $p$ is smaller than one of the MV rotations and MV reflections of $s'$, then $s'$ is a duplication because such an MV rotation/MV reflection has been already searched. Figure 4.5 shows how the algorithm prunes the search tree for $s = 01010101$.

(2b'-4) If $s'$ satisfies the Maekawa Theorem, $s'$ goes to phase 3 to be processed. We can prune the search if we have assigned $n/2 - 1$ $V$s to $s'$ where $n$ is the number of creases.

For efficient computation in step 2b'-3, we construct prior to the search a function $f(i)$ that tells us the original index over $s'$ of $i$th letter in an MV rotation/MV reflection. For example, if we consider an MV rotation that shifts 2 letters clockwisely like $V1M1M1M1$ to $M1V1M1M1$, then $f(i) = (i + 2n - 2) \bmod 2n$ where $n$ is the number of creases. The comparison of the prefix and an MV rotation/MV reflection is reduced to a comparison of

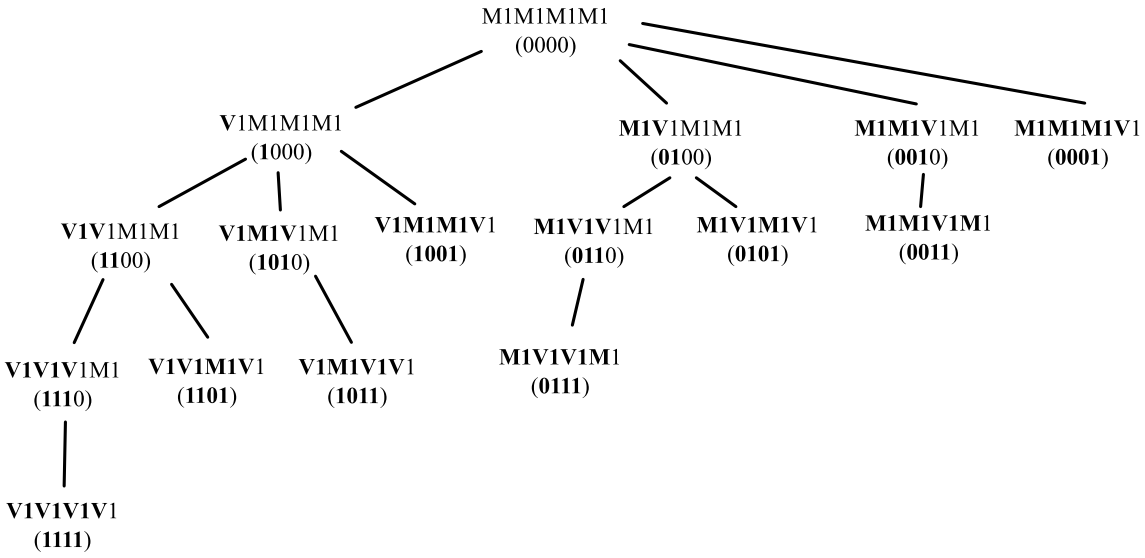



Figure 4.4: The underlying search tree for $s = 01010101$ with binary string expression of MV assignment. The bold prefix corresponds to $p$.

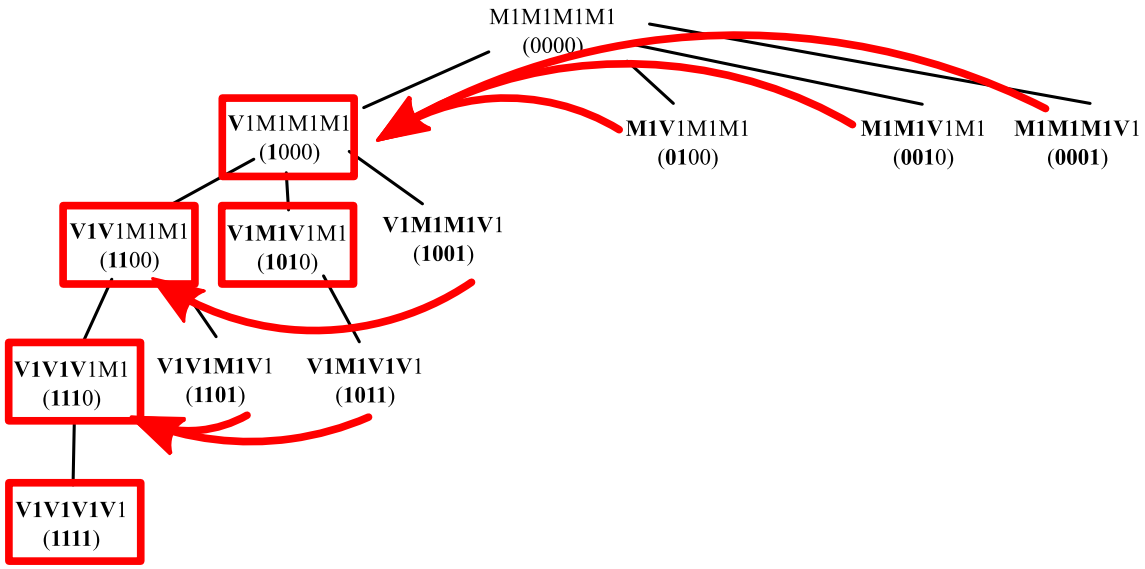



Figure 4.5: The search tree for $s = 01010101$ pruned by MV rotation. An arrow means an MV rotation of $s'$ to the representative string (the largest string among the equivalents). Pruning by the Maekawa Theorem and MV reflection are omitted to simplify the explanation in this figure.

each letter at index $i$ and at $f(i)$ of $s'$ for $i = 0, 2, \ldots, 2k-2$. (We need to construct such functions for all possible MV rotations and MV reflections.) This technique eliminates memory allocations for explicit construction of rotations and reflections. Note that we can prune the search tree further by Lemma 2.2 as done in step 2b.

Since the number of rotations and reflections is $O(q)$ and string comparison takes $O(q)$, the test takes $O(q^2)$ time. But the search for MV assignments can be faster than that with caching strategy because few "crease"/"flat" assignments have MV rotation or MV reflection and there is less memory access than caching.

### 4.3.4 Phase 3: Test of Flat Foldability

In this phase, we check if the resulting string $s'$ over $\{M, V, 1, 2, \ldots\}$ is flat-foldable or not. For this problem, Demaine and O'Rourke give a linear time algorithm [8, Chapter 12]. Therefore, we can perform this phase in linear time. Roughly, the algorithm is simple; it finds a local minimal angle whose boundary creases have opposite MV assignment, folds the boundary, glues it, and repeats until all creases are folded. However, the proof of the correctness of this algorithm is not easy; as mentioned at the footnote in [8, page 204], the rigorous proof is first done by Demaine and O'Rourke in [8, Chapter 12].

We obtain the following obvious upper bound of the number of the outputs in this phase by integration of the observations in Sections 4.3.2 and 4.3.3:

**Theorem 4.5** *For a given even number $q$, the number of distinct flat-foldable MV-assigned crease patterns with unit angle* $(360/q)^\circ$ *is* $O\left(B'(q)\binom{q}{q/2-1}\right)$ *where $B'(q)$ is the number of distinct balanced twills on $q$ harnesses (see Equation 4.2).*

### 4.3.5 Analysis of Algorithm

The correctness of our algorithm relies on the algorithms used in each phase as described above. Here we consider its time complexity and space complexity of computing all outputs. Our main theorem is the following:

**Theorem 4.6** *For a given even number $q$, enumeration of all distinct flat-foldable MV-assigned crease patterns with unit angle* $(360/q)^\circ$ *can be done in* $O\left(qB(q)\binom{q}{q/2-1}\right)$ *time with* $O\left(q\binom{q}{q/2-1}\right)$ *space, where* $B(q)$ *is the number of bracelets of length* $q$ *(see Equation 4.1).*

We note that the order of space complexity may be far from strict one because the actual required space for the computation depends on the behavior of the trie used in phase 2.

### 4.3.6 Parallel Processing

Our algorithm can be easily parallelized because each output of phase 1 is consumed by phase 2 and there is no other relation between the two phases. We implement the parallel processing as a master-worker model. The master process runs phase 1, that is, a search to generate crease patterns satisfying the Kawasaki Theorem. When the master process finds a valid crease pattern, it passes the crease pattern to a waiting worker process. The worker process that was given the crease pattern runs phase 2 (a search to generate MV assignment) and phase 3 (a test of flat foldability), and outputs the flat-foldable MV-assigned crease patterns.

## 4.4 Experimental Results

As shown in Theorem 4.5, the upper bound of the number of distinct flat-foldable MV-assigned crease patterns is exponential if $(360/q)^\circ$ unit angle is introduced. Exact values for each $q$ are difficult to estimate theoretically. Therefore, we here show experimental results: the number of the

enumerated patterns, the rate of the enumerated patterns against the number of possible patterns, and the computation time for the enumeration. The program is written in C++ using its default STL library and MPI[2].

The computation for enumeration and counting was done with 384 nodes (13824 CPU cores) for at most 1.5 days for each $q$ on a supercomputer Cray XC40. We compared the computation time in four scales on Cray XC40:

- single core for at most 4 days for each $q$,
- 32 nodes (1152 CPU cores) for at most 4 days for each $q$,
- 128 nodes (4608 CPU cores) for at most 2 days for each $q$,
- 384 nodes (13824 CPU cores) for at most 1.5 days for each $q$.

By an experiment, we found that non-caching strategy in phase 2 is faster than the caching strategy. In addition, the caching strategy with single core failed to compute for $q = 24$ or more because of excess of memory. Hence we took the non-caching strategy for the experiments.

### 4.4.1 The Number of Crease Patterns

The exact numbers of distinct patterns obtained at each phase are shown in Table 4.1 and Figure 4.6. As mentioned in Section 4.3.2, the result of phase 1, which enumerates SVCPs satisfying the Kawasaki Theorem, coincides with the sequence listed in OEIS as A006840. The counting results at the other phases are different from any existing sequences in OEIS, that is, we find totally new sequences in this study.

### 4.4.2 Solution Space

We measure the rate of the number of solutions against that of possible patterns at each phase (see Table 4.3 and Figure 4.7), which suggests how

[2]The implementation is at https://ouchi-koji.visualstudio.com/_git/FlapCPEnum?version=GBparallelization.

Table 4.1: The number of enumerated patterns. The number of creases in a pattern is even number from 2 to $q$. Pruning by Lemma 2.2 is not applied in phase 2. The rightmost column is the number of outputs in phase 3 per the output in phase 1, which can be seen as the average size of a task for a worker process in parallelization.

| $q$ | Phase 1 | Phase 2 | Phase 3 | Phase 3/Phase 1 |
|---|---|---|---|---|
| 2 | 1 | 1 | 1 | 1.00 |
| 4 | 2 | 2 | 2 | 1.00 |
| 6 | 3 | 7 | 6 | 2.00 |
| 8 | 7 | 27 | 20 | 2.86 |
| 10 | 13 | 143 | 87 | 6.69 |
| 12 | 35 | 837 | 420 | 12.00 |
| 14 | 85 | 5529 | 2254 | 26.52 |
| 16 | 257 | 38305 | 12676 | 49.32 |
| 18 | 765 | 276441 | 73819 | 96.50 |
| 20 | 2518 | 2042990 | 438795 | 174.26 |
| 22 | 8359 | 15396071 | 2649555 | 316.97 |
| 24 | 28968 | 117761000 | 16188915 | 558.86 |
| 26 | 101340 | 912100793 | 99888892 | 985.68 |
| 28 | 361270 | 7139581543 | 621428188 | 1720.12 |
| 30 | 1297879 | 56400579759 | 3893646748 | 3000.01 |
| 32 | 4707969 | 449129924559 | 24548337096 | 5214.21 |
| 34 | 17179435 | 3601920245329 | 155622071065 | 9058.63 |
| 36 | 63068876 | 29069099909934 | 991375878185 | 15718.94 |
| 38 | 232615771 | 235928559206883 | 6343073841027 | 27268.46 |
| 40 | 861725794 | 1924593128183050 | 40744074042024 | 47281.95 |
| 42 | 3204236779 | - | - | - |

difficult the problems are. We can see that the solution spaces are very sparse at each phase. There are $2^q$ possible "crease"/"flat" assignments at phase 1. Only approximately 4.7% is the solution for phase 1 if $q = 6$. It decreases significantly and gets less than 1% for $q \geq 12$. The rates at phase 2 and phase 3 are against $3^q$ since we consider "mountain"/"valley"/"flat" assignments at these phases. These two rates tend to decrease similarly to that of phase 1 and are much smaller, e.g., 1.0% at phase 2 when $q = 6$. Such rate at every phase seems to be exponential to $q$ according to Figure 4.7.

Table 4.2: Distribution of the patterns obtained at phase 1.

| $q$ | #crease of each pattern | | | | | | | | | | sum |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | |
| 4 | 1 | 1 | | | | | | | | | 2 |
| 6 | 1 | 1 | 1 | | | | | | | | 3 |
| 8 | 1 | 3 | 2 | 1 | | | | | | | 7 |
| 10 | 1 | 3 | 6 | 2 | 1 | | | | | | 13 |
| 12 | 1 | 6 | 13 | 11 | 3 | 1 | | | | | 35 |
| 14 | 1 | 6 | 26 | 30 | 18 | 3 | 1 | | | | 85 |
| 16 | 1 | 10 | 46 | 93 | 74 | 28 | 4 | 1 | | | 257 |
| 18 | 1 | 10 | 79 | 210 | 275 | 145 | 40 | 4 | 1 | | 765 |
| 20 | 1 | 15 | 124 | 479 | 841 | 716 | 280 | 56 | 5 | 1 | 2518 |

### 4.4.3 Computation Time

We compared the computation time in four scales as mentioned in Section 4.4. Table 4.4 shows the computation time with non-caching strategy for each scale and $q \geq 24$. Since the average size of a task for a worker process in parallelization gets exponentially larger as $q$ grows (see the rightmost column in Table 4.1), the program becomes more parallelized for large $q$; see the graph in Figure 4.8 of speedup rate against single core computation.

## 4.5 Conclusion

This chapter describes the first trial in the world to enumerate flat-foldable MV-assigned SVCPs with unit angles $(360/q)^\circ$ for given $q$. We have constructed an enumeration algorithm which caches representatives among rotations and reflections of MV-assigned SVCPs to avoid duplications. An enumeration algorithm without caching is also proposed. In experiment, the algorithm without caching was faster than that with caching although the caching algorithm is theoretically faster. The experiment showed that there are numerous flat-foldable SVCPs. The effect of parallelization of the program is significant: for example, the computation with 13824 CPU

Table 4.3: #solution/#possible at each phase.

| $q$ | #Phase1/$2^q$ | #Phase2/$3^q$ | #Phase3/$3^q$ |
|---|---|---|---|
| 4 | 0.125 | 0.024691358 | 0.024691358 |
| 6 | 0.046875 | 0.009602195 | 0.008230453 |
| 8 | 0.02734375 | 0.004115226 | 0.003048316 |
| 10 | 0.012695313 | 0.002421718 | 0.001473353 |
| 12 | 0.008544922 | 0.001574963 | 0.000790304 |
| 14 | 0.005187988 | 0.001155977 | 0.000471255 |
| 16 | 0.003921509 | 0.000889847 | 0.000294471 |
| 18 | 0.002918243 | 0.000713543 | 0.000190540 |
| 20 | 0.002401352 | 0.000585924 | 0.000125845 |
| 22 | 0.001992941 | 0.000490617 | 8.44317E-05 |
| 24 | 0.001726627 | 0.000416957 | 5.73202E-05 |
| 26 | 0.001510084 | 0.000358831 | 3.92975E-05 |
| 28 | 0.001345836 | 0.000312088 | 2.71641E-05 |
| 30 | 0.001208744 | 0.000273934 | 1.89112E-05 |
| 32 | 0.001096159 | 0.000242377 | 1.32477E-05 |
| 34 | 0.000999975 | 0.000215979 | 9.33144E-06 |
| 36 | 0.000917773 | 0.000193672 | 6.60501E-06 |
| 38 | 0.000846251 | 0.000174652 | 4.69561E-06 |
| 40 | 0.000783735 | 0.000158303 | 3.35130E-06 |
| 42 | 0.000728559 | - | - |

cores was 717 times faster than that with single core if $q = 32$. Speedup rate of the computation with fixed number of CPU cores gets larger as $q$ grows.

Table 4.4: Computation time with non-caching strategy [sec].

| $q$ | 1 core | 1152 cores | 4608 cores | 13824 cores |
|---|---|---|---|---|
| 24 | 64.13 | 1.91 | 1.23 | 1.61 |
| 26 | 470.45 | 7.69 | 4.50 | 5.02 |
| 28 | 3514.68 | 32.27 | 17.42 | 17.90 |
| 30 | 26578.2 | 149.93 | 70.17 | 69.49 |
| 32 | 199032 | 784.81 | 297.04 | 227.36 |
| 34 | > 4days | 4595.42 | 1355.50 | 1146.13 |
| 36 | > 4days | 29194 | 6715.16 | 4842.22 |
| 38 | > 4days | 202886 | 38451.1 | 22943.4 |
| 40 | > 4days | > 4days | > 2days | 120918 |

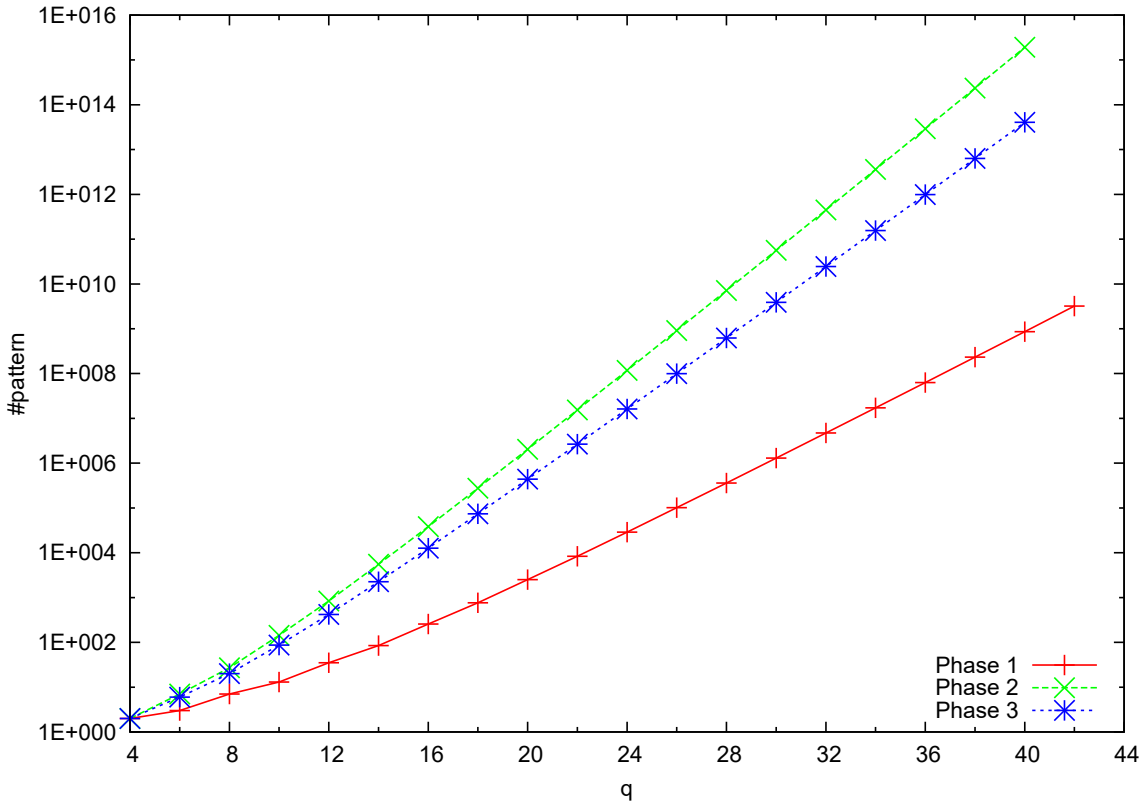



Figure 4.6: The number of enumerated patterns. The number of creases in a pattern is even number from 2 to $q$.

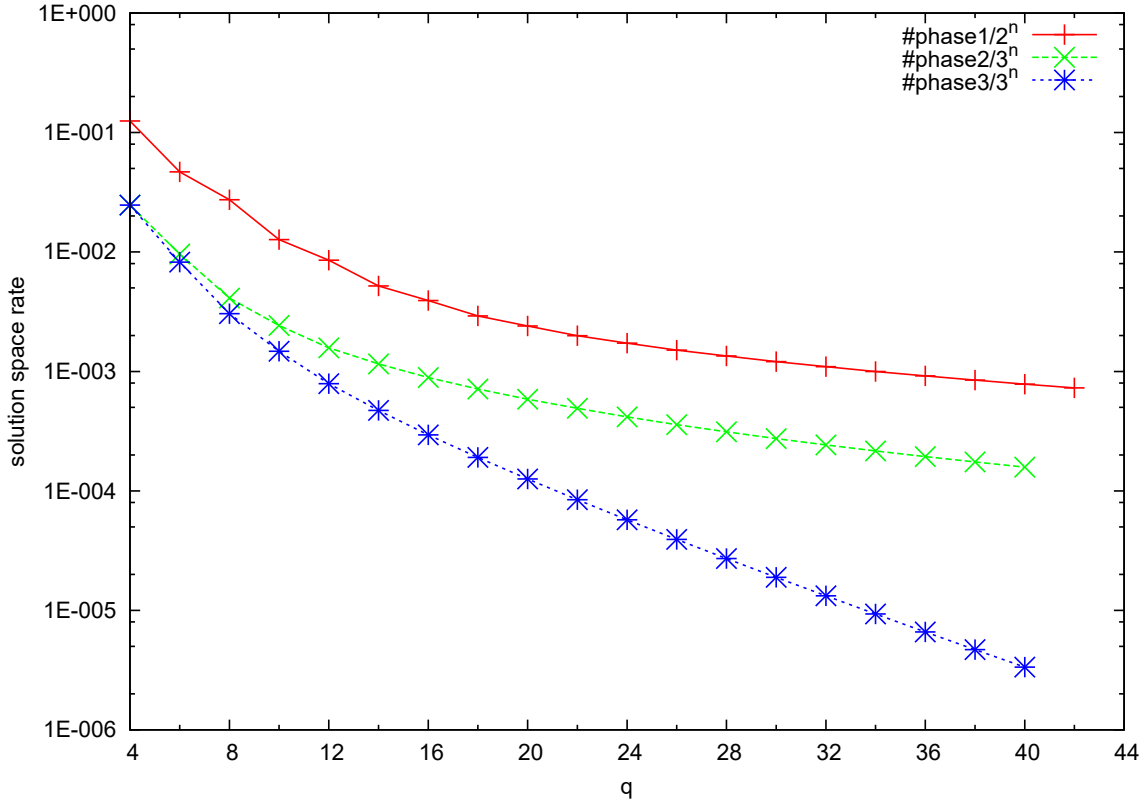



Figure 4.7: The rate of solutions against possible patterns at each phase.

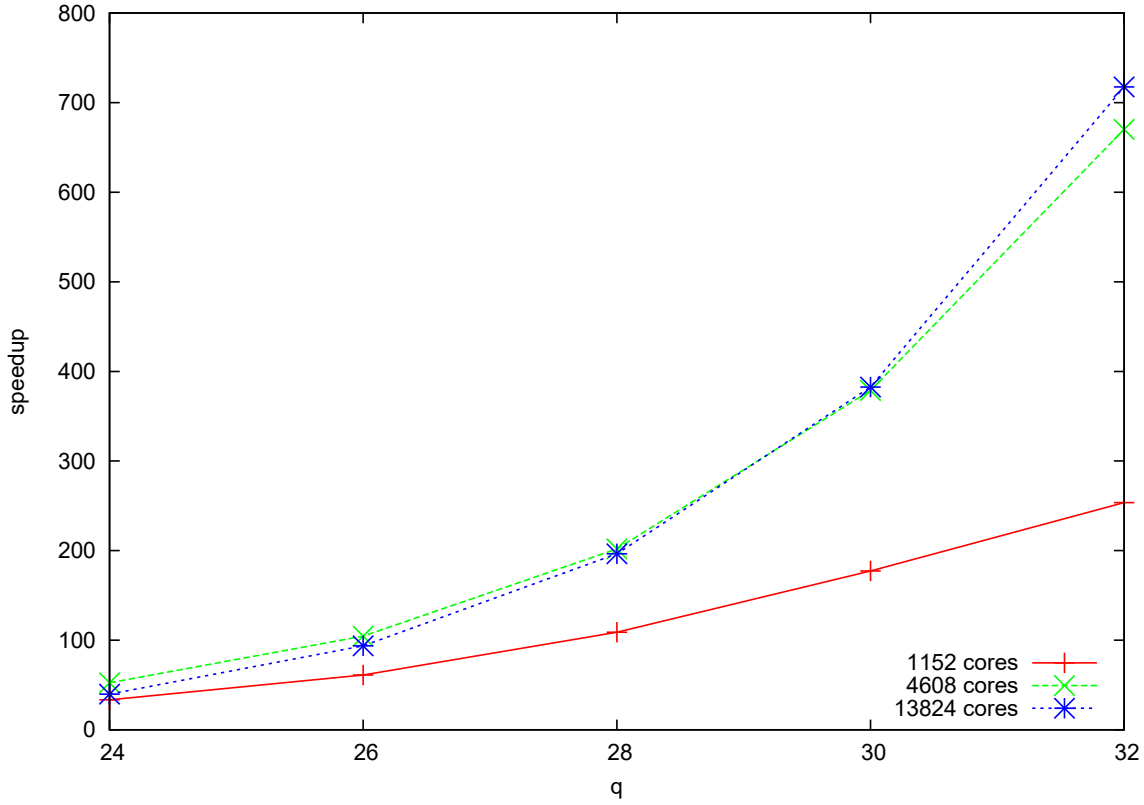



Figure 4.8: The speedup rate = (computation time using 1 core)/(computation time using multi cores).

# Chapter 5

# Conclusion

This paper contributes to basics and applications of 2D origami by focusing on SVCP that is a component of arbitrary 2D origami. SVCP is simple and old as a theme of study but still open to investigation if we consider a new general concept like forcing sets and enumeration. Our contribution can be separated into two parts.

The first contribution is the development of an algorithm for finding a minimum forcing set of a given flat-foldable MV-assigned SVCP, which may reduce the cost of implementation of origami applications. This work on SVCP is for the first time in the world as far as the author knows. The algorithm runs in $O(n^2)$ time where $n$ is the number of given creases. We have shown that the size of such forcing set is $n/2$ or $n/2+1$. Precisely, the size is $n/2$ if the number of creases remaining after crimping minimal angle sequence repeatedly is two. Otherwise the size is $n/2+1$. It is an open problem to find a minimum forcing set of arbitrary 2D origami. Considering minimum forcing sets of two- or several-vertex origami might be the first step to solve the arbitrary case. Enumeration of minimum forcing sets of a given MV-assigned crease pattern is an interesting problem as well. We believe that our result will help us to solve such open problems.

The second contribution is to enumerate distinct flat-foldable MV-assigned SVCPs with unit angles, which provides us with knowledge for designing origami. We have developed the first algorithm for enumerating

distinct flat-foldable MV-assigned SVCPs with unit angle $(360/q)^\circ$ where $q$ is a given positive even integer. The algorithm consists of three phases: generating SVCPs, generating MV-assigned SVCPs, and testing flat foldability. We have experimentally shown how many patterns in each phase there are, which is done for the first time as well. We have examined the rates of desired patterns against all possible patterns in each phase; experimentally, they seem to decrease exponentially. According to the experimental results, we conjecture that there are exponentially many flat-foldable MV-assigned SVCPs. The computation time was reduced by parallel processing. Improving the algorithm and investigating further for the counting problems are future work. For example, rather than Sawada's algorithm in Theorem 4.1, enumeration of the sequences stated in Theorem 4.2 may directly improve the running time of our algorithm drastically. Showing theoretical lower and upper bounds on counting also remains open.

# References


[1] Zachary Abel, Jason Cantarella, Erik D Demaine, David Eppstein, Thomas C Hull, Jason S Ku, Robert J Lang, and Tomohiro Tachi. Rigid origami vertices: conditions and forcing sets. *Computational geometry*, 7(1), 2016.

[2] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21–46, 1996.

[3] Brad Ballinger, Mirela Damian, David Eppstein, Robin Flatland, Jessica Ginepro, and Thomas Hull. Minimum forcing sets for Miura folding patterns. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 136–147. Society for Industrial and Applied Mathematics, 2015.

[4] Marshall Bern and Barry Hayes. The complexity of flat origami. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 175–183, 1996.

[5] Kellogg S Booth. Lexicographically least circular substrings. *Information Processing Letters*, 10(4-5):240–242, 1980.

[6] Mirela Damian, Erik Demaine, Muriel Dulieu, Robin Flatland, Hella Hoffman, Thomas C Hull, Jayson Lynch, and Suneeta Ramaswami. Minimum forcing sets for 1D origami. *arXiv preprint arXiv:1703.06373v1*, 2015.

[7] Rene De La Briandais. File searching using variable length keys. In

*Papers presented at the the March 3-5, 1959, western joint computer conference*, pages 295–298. ACM, 1959.

[8] Erik D Demaine and Joseph O' Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[9] Edward Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, 1960.

[10] Jessica Ginepro and Thomas C Hull. Counting miura-ori foldings. *Journal of Integer Sequences*, 17, 2014. Article 14.10.8.

[11] E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, Erik D. Demaine, D. Rus, and R. J. Wood. Programmable matter by folding. *Proceedings of the National Academy of Sciences*, 107(28):12441–12445, 2010.

[12] W.D. Hoskins and Anne Penfold Street. Twills on a given number of harnesses. *Journal of the Australian Mathematical Society (Series A)*, 33(01):1–15, 1982.

[13] Tom Hull. Counting mountain-valley assignments for flat folds. *Ars Combinatoria*, 67:175–187, 2003.

[14] Leonid Ionov. 3D microfabrication using stimuli-responsive self-folding polymer films. *Polymer Reviews*, 53(1):92–107, 2013.

[15] Jacques Justin. Towards a mathematical theory of origami. In *Proceedings of 2nd international meeting origami science, technology*, pages 15–29, 1994.

[16] Toshikazu Kawasaki. On the relation between mountain-creases and valley-creases of a flat origami. In *Proceedings of 1st international meeting origami science, technology*, pages 229–237, 1989.

[17] John E Koehler. Folding a strip of stamps. *Journal of Combinatorial Theory*, 5(2):135–152, 1968.

[18] Timothy G. Leong, Paul A. Lester, Travis L. Koh, Emma K. Call, and David H. Gracias. Surface tension-driven self-folding polyhedra. *Langmuir*, 23(17):8747–8751, 2007. PMID: 17608507.

[19] L. Mahadevan and S. Rica. Self-organized origami. *Science*, 307(5716):1740–1740, 2005.

[20] 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*, 21(2):169–178, 2017.

[21] Jun Mitani. Development of origami pattern editor (ORIPA) and a method for estimating a folded configuration of origami from the crease pattern. *IPSJ Journal*, 48(9):3309–3317, 2007. in Japanese.

[22] Jun Mitani. The folded shape restoration and the CG display of origami from the crease pattern. In *13th international Conference on Geometry and Graphics*, 2008.

[23] Joe Sawada. Generating bracelets in constant amortized time. *SIAM Journal on Computing*, 31(1):259–268, 2001.

[24] Joe Sawada and Roy Li. Stamp foldings, semi-meanders, and open meanders: fast generation algorithms. *the electronic journal of combinatorics*, 19(2):43, 2012.

[25] Tomohiro Tachi. 3D origami design based on tucking molecule. In *The Fourth International Conference on Origami in Science, Mathematics, and Education, R. Lang, ed., Pasadena*, pages 259–272, 2009.

[26] Ryuhei Uehara. Stamp foldings with a given mountain-valley assignment. In *Origami 5*, pages 585–592. CRC Press, 2011.

[27] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *International Conference on Discovery Science*, pages 16–31. Springer, 2004.

[28] Mohammed J Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2002.

# Publications (refereed)

[ONK11] Koji Ouchi, Atsuyoshi Nakamura, and Mineichi Kudo. Efficient construction and usefulness of hyper-rectangle greedy covers. In *2011 IEEE International Conference on Granular Computing*, pages 533–538. IEEE, 2011.

[ONK14] Koji Ouchi, Atsuyoshi Nakamura, and Mineichi Kudo. An efficient construction and application usefulness of rectangle greedy covers. *Pattern Recognition*, 47(3):1459–1468, 2014.

[OU17] Koji Ouchi and Ryuhei Uehara. Efficient enumeration of flat-foldable single vertex crease patterns. In Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen, editors, *The 11th International Conference and Workshops on Algorithms and Computation (WALCOM 2017)*, pages 19–29, Cham, 2017. Springer International Publishing.

[OU19a] Koji Ouchi and Ryuhei Uehara. Efficient enumeration of flat-foldable single vertex crease patterns. *IEICE Transactions on Information and Systems*, E102-D(3):416–422, 2019.

[OU19b] Koji Ouchi and Ryuhei Uehara. Minimum forcing sets for single-vertex crease pattern. In *31st Canadian Conference on Computational Geometry*, pages 171–176, 2019.

# Publications (unrefereed)

[OU16] Koji Ouchi and Ryuhei Uehara. Efficient enumeration of flat-foldable single-vertex crease patterns. Unrefereed poster session as an exhibition in the 29th International Conference for High Performance Computing, Networking, Storage and Analysis (SC16), 2016.

[OU18] Koji Ouchi and Ryuhei Uehara. The world record for enumeration of flat-foldable single-vertex crease patterns. Unrefereed poster session as an exhibition in the 31st International Conference for High Performance Computing, Networking, Storage and Analysis (SC18), 2018.