JAIST Repository
https://dspace.jaist.ac.jp/

Title	Efficient Enumeration of Flat-Foldable Single Vertex Crease Patterns
Author(s)	Ouchi, Koji; Uehara, Ryuhei
Citation	IEICE TRANSACTIONS on Information and Systems, E102-D(3): 416-422
Issue Date	2019-03-01
Type	Journal Article
Text version	publisher
URL	http://hdl.handle.net/10119/16094
Rights	Copyright (C)2019 IEICE. Koji Ouchi and Ryuhei Uehara, IEICE TRANSACTIONS on Information and Systems, E102-D(3), 2019, 416-422. http://www.ieice.org/jpn/trans_online/
Description	

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

IEICE TRANS. INF. & SYST., VOL.E102–D, NO.3 MARCH 2019

PAPER Special Section on Foundations of Computer Science — Algorithm, Theory of Computation, and their Applications —

Efficient Enumeration of Flat-Foldable Single Vertex Crease Patterns*

Koji OUCHI[†a)], *Nonmember and* **Ryuhei UEHARA**[†b)], *Member*

SUMMARY We investigate enumeration of distinct flat-foldable crease patterns under the following assumptions: positive integer n is given; every pattern is composed of n lines incident to the center of a sheet of paper; every angle between adjacent lines is equal to $2\pi/n$; every line is assigned one of "mountain," "valley," and "flat (or consequently unfolded)"; 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.

key words: *computational origami, enumeration algorithm, flat foldability, Kawasaki theorem, Maekawa theorem*

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 22.5°, 15° or some other specific angle which divides 360°. A nontrivial example, which was designed by the first author, is shown in Fig. 1. It is based on a unit angle of 15°. Once origamists fix the unit angle as $(360/n)°$ for suitable positive integer n, their designs are restricted to one between quite real shapes and abstract shapes, which is the next matter in art.

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

A crease pattern is said to be *flat-foldable* if and only if there exists an assignment of mountain and valley to each crease line so that after folding along the assignment, we can have a flat folded sheet of paper without penetrating itself. We here note that the ordering of the layers of paper is not given, and it is not easy to compute it even if the assignment is given. When a mountain or valley assignment to every crease line is given, the flat-foldability can be computed in linear time [1], [2]. In fact, the algorithm also gives us the ordering of the layers at the same time. However, its rigorous proof is not so simple, which is the main topic of Chapter 12 in [2]. 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 that does not contain a specification of whether folds are mountains or valleys is very different from the case that a mountain and valley assignment is given. Hull investigated this problem from the viewpoint of counting [3]. Precisely, he considered the number of flat-foldable assignments of mountain and valley to a given crease pattern of n lines which were incident to the single vertex. In [3], 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 n, and we restrict ourselves to the single vertex folding of unit angle $(360/n)°$. In order to investigate our problem, we assign one of three labels — "mountain," "valley," and "flat" — to each of n creases. When a crease line is labeled "flat," this crease line is not folded in the final folded state. In this way, we can deal with the single vertex crease patterns of unit angle equal to $(360/n)°$, which is more realistic situation from the viewpoint of origami design.

Our aim is to enumerate all distinct flat-foldable assignments of the three labels to n creases. In other words, our algorithm eventually enumerates all flat-foldable crease patterns with labels of "mountain" and "valley" of unit angle $(360/n)°$. We consider the sheet of paper is a disk, the vertex is at the center of the disk, and two 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 crease patterns under this assumption.

For flat-foldability of a given crease pattern, there

Manuscript received March 20, 2018.
Manuscript revised July 26, 2018.
Manuscript publicized October 31, 2018.

[†]The authors are with the School of Information Science, Japan Advanced Institute of Science and Technology, Nomi-shi, 923–1292 Japan.

[*]A preliminary version was presented at WALCOM 2017.

a) E-mail: k-ouchi@jaist.ac.jp
b) E-mail: uehara@jaist.ac.jp

DOI: 10.1587/transinf.2018FCP0004

Copyright © 2019 The Institute of Electronics, Information and Communication Engineers

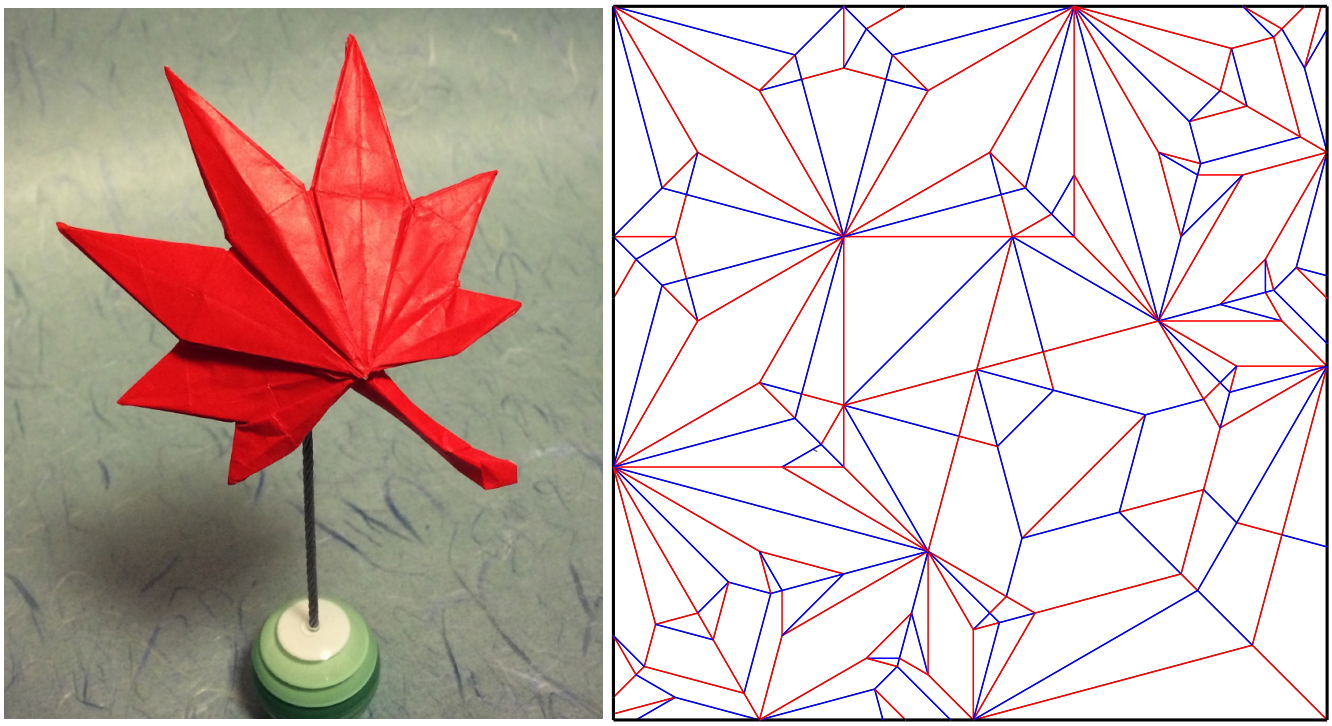

Fig. 1 "Maple leaf" designed and folded by the first author (left). Its crease pattern is based on 15° unit angle (right).

are two well-known theorems in the area of computational origami, which are called the Kawasaki Theorem and the Maekawa Theorem (see [2, Chapter 12] for further details):

Theorem 1 (The Kawasaki Theorem): Let θ_i be an angle between the ith and the $(i+1)$st crease lines. We assume that either mountain or valley is assigned to each crease line. A single-vertex crease pattern 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 θ_{2i+1} is equal to the sum of the even angles θ_{2i}, or equivalently, either sum is equal to 180°: $\theta_1+\theta_3+\cdots+\theta_{n'-1} = \theta_2+\theta_4+\cdots+\theta_{n'} = 180°$.

We note that the Kawasaki Theorem gives a necessary and sufficient condition for flat-foldability, however, mountain-valley assignments are not given. That is, we have to compute foldable assignments for foldable crease pattern satisfying the Kawasaki Theorem. In order to compute a flat-foldable assignment, we can use the Maekawa Theorem:

Theorem 2 (The Maekawa Theorem): We assume that a single-vertex crease pattern defined by angles $\theta_1+\theta_2+\cdots+\theta_{n'} = 360°$ is flat-foldable under a mountain and valley assignment (that is, no "flat" is assigned). Then the number of mountains and the number of valleys differ by ±2.

We again 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., [4]–[6]. Using techniques that follow above properties of origami, we construct an enumeration algorithm for flat-foldable crease patterns for given n, where each angle between two crease lines is a multiple of $(360/n)°$. As far as the authors know, this is the first algorithm for the realistic computational origami problem. As a result, we succeeded to enumerate flat-foldable crease patterns up to $n = 38$ in a reasonable time.

2. Preliminaries and Outline of Algorithm

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

(1) Assign "crease" or "flat" to each of n crease lines incident to a single vertex so that the Kawasaki Theorem is satisfied.
(2) For each "crease", assign "mountain" or "valley" so that the Maekawa Theorem is satisfied.
(3) Output the pattern if this 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 up to rotation and reflection in each step.

A simple example is given in Fig. 2. For $n = 8$, we first generate all possible crease lines in phase 1 which is described in a binary string (in the figure, we only show one, but there are exponentially many). Here "0" and "1" denote "crease" and "flat" respectively. Therefore, for a string 00011011, we have four crease lines in the shape in Fig. 2.

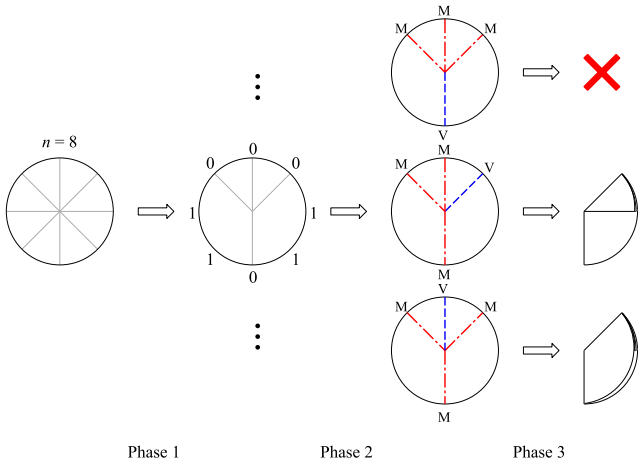

Fig. 2 Simple example for $n = 8$.

In phase 2, we assign mountain (M) or valley (V) to each crease line. In phase 3, we check whether each crease pattern with M/V assignments 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.

3. Description of Algorithm

Now we describe more details in each phase.

3.1 Phase 1: Assignment of "crease"/"flat"

In phase 1, we are given n crease 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 n, without loss of generality, we assume that n 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 consider 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 n. For bracelets, we have an optimal enumeration algorithm [7]:

Theorem 3 (Sawada2001): Bracelets of length n can be enumerated in constant amortized time.

That is, the algorithm in [7] runs in a time proportional to $B(n)$ which denotes the number of bracelets of length n.

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

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

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

3.2 Phase 1: Satisfying the Kawasaki Theorem

After assigning "crease" or "flat" to each crease, we have to check whether these crease lines satisfy 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 θ_i is $k \times \frac{360}{n}^\circ$ for given even n. That is, θ_i consists of k unit angles. Now we regard θ_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 n, 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 n beads such that (1) the necklace consists of $n/2$ white beads and $n/2$ black beads, and (2) the number of runs[†] of white beads (and hence black beads) is n'. This notion is investigated as "balanced twills on n harnesses" in [8] and listed in OEIS as A006840. Then the number is given as follows:

Theorem 4 (Hoskins and Street 1982): The number of distinct balanced twills on $n = 2k'$ harnesses is

$$B'(2k') = \frac{1}{8k'}\left\{\begin{array}{l} \displaystyle\sum_{\substack{d \text{ divides } n \\ d=2e}} \phi\left(\frac{k'}{e}\right)\binom{2e}{e} \\ \displaystyle + \sum_{d \text{ divides } k'} \phi\left(\frac{2k'}{d}\right)2^d \\ \displaystyle + 2k'\binom{2\lfloor k'/2 \rfloor}{\lfloor k'/2 \rfloor} + k2^{k'} \end{array}\right\}. \tag{2}$$

We note that Eq. (2) just gives us the numbers for each n, 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 [7]. 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 θ_i is the angle between the ith and $(i+1)$st creases as defined in Theorem 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

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

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 5: For a given even number n, phase 1 can be done in $O(B(n))$ time, where $B(n)$ is the number of bracelets of length n.

Furthermore, we can prune the search tree with the following theorem:

Theorem 6: If a given single vertex crease pattern 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 $1 \leq m \leq n' - 1$.

3.3 Phase 2: Assignment of "mountain"/"valley"

In this phase, we inherit a binary string of length n 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 Fig. 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 Ms and the number of Vs should differ by 2. To avoid symmetry case, we can assume that (the number of Ms)−(the number of Vs)= 2. Thus the next step is described as follows:

(2b) For the resulting string over $\{0, 1, 2, \ldots, n-1\}$, assign all possible Ms and Vs to each 0 such that the number of Ms is 2 larger than the number of Vs. For example, for the string 01010303, we obtain the set of strings $\{V1M1M3M3, M1V1M3M3, M1M1V3M3, M1M1M3V3\}$.

For a string s generated by step 2a, which describes an unassigned crease pattern, we can have equivalent 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 $V1M1M3M3$ is a crease pattern which is the mirror image of a crease pattern $M1M1V3M3$, hence we consider they are equivalent. (In Fig. 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 $\{M, V, 1, 2, \ldots, n-1\}$ 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 lexicographically 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 [9], [10] (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 [11]. 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 $\{M, V, 1, 2, \ldots, n-1\}$ 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 $O(n)$ 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 7: For a given crease pattern from phase 1 based on n unit angles, we can generate all distinct assignments of mountain and valley that satisfy the Maekawa Theorem in $O(nC(n))$ time with space linear in the product of n and the number of such assignments, where $C(n)$ is $\binom{n}{n/2-1}$.

Proof. The number of lines in the crease pattern is at most n, and the number of Ms is 2 larger than the number of Vs. Thus, the number of strings s' over $\{M, V, 1, 2, \ldots\}$ of length at most n with the constraint for the number of Ms and Vs is at most $\binom{n}{n/2-1}$. 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 n and the number of desired assignments. ∎

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 [2, Chap 12] . Therefore, we can perform this phase in linear time. Roughly, the algorithm is simple; it finds a local minimal angle, folds two creases on the boundary of the small fan-shape, 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 [2, page 204] , the rigorous proof is first done by Demaine and O'Rourke in [2, Chap 12] .

We obtain the following obvious upper bound of the number of the outputs in this phase by integration of the observations in Sects. 3.2 and 3.3:

Theorem 8: For a given even number n, the number of distinct flat-foldable mountain and valley assignments with unit angle $(360/n)^\circ$ is $O\left(B'(n)\binom{n}{n/2-1}\right)$ where $B'(n)$ is the number of distinct balanced twills on n harnesses (see Eq. (2)).

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 9: For a given even number n, all distinct flat-foldable mountain and valley assignments with unit angle $(360/n)^\circ$ can be done in $O\left(nB(n)\binom{n}{n/2-1}\right)$ time with $O\left(n\binom{n}{n/2-1}\right)$ space, where $B(n)$ is the number of bracelets of length n (see Eq. (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.

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 and passes the outputs one by one to worker processes. Each worker process runs Phase 2 and 3 for given "crease"/"flat" assignment, and outputs the flat-foldable single vertex crease patterns.

4. Experimental Results

As shown in Theorem 8, the upper bound of the number of distinct flat-foldable mountain and valley assignments is exponential if $(360/n)^\circ$ unit angle is introduced. Exact values for each n are difficult to estimate theoretically. Therefore, we here show experimental results. The program is written in C++ using its default STL library. We use 96 nodes (576 cores, 12TB memory) of a supercomputer SGI UV3000. The computation time is at most 5 days for each n.

4.1 The Number of Crease Patterns

The exact numbers of distinct patterns obtained at each phase are shown in Table 1 and Fig. 3. As mentioned in Sect. 3.2, the result of phase 1, which enumerates "crease"/"flat" assignments 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.

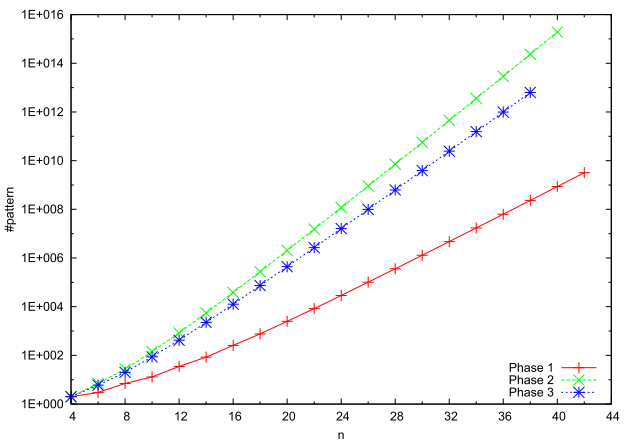

Fig. 3 The number of enumerated patterns. The number of lines in a pattern is even number from 2 to n.

Table 1 The number of enumerated patterns. The number of lines in a pattern is even number from 2 to n.

n	Phase 1	Phase 2	Phase 3
2	1	1	1
4	2	2	2
6	3	7	6
8	7	27	20
10	13	143	87
12	35	837	420
14	85	5529	2254
16	257	38305	12676
18	765	276441	73819
20	2518	2042990	438795
22	8359	15396071	2649555
24	28968	117761000	16188915
26	101340	912100793	99888892
28	361270	7139581543	621428188
30	1297879	56400579759	3893646748
32	4707969	449129924559	24548337096
34	17179435	3601920245329	155622071065
36	63068876	29069099909934	991375878185
38	232615771	235928559206883	6343073841027
40	861725794	1924593128183050	-
42	3204236779	-	-

Table 2 Distribution of the patterns obtained at phase 1.

n	#line 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

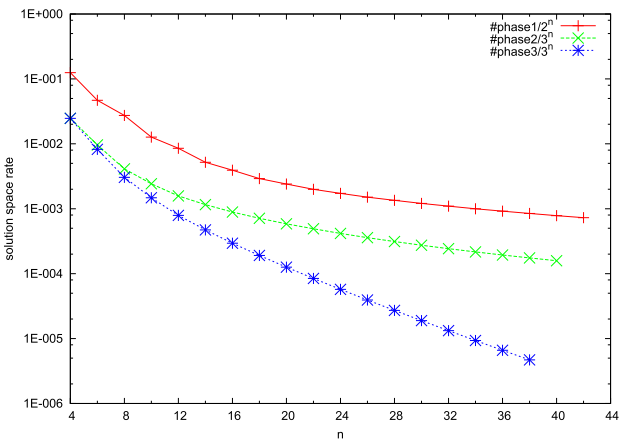

Fig. 4 The rate of solutions against possible patterns at each phase.

4.2 Solution Space

We measure the rate of the number of solutions against that of possible patterns at each phase (see Table 3 and Fig. 4), which suggests how difficult the problems are. We can see that the solution spaces are very sparse at each phase. There

Table 3 #solution/#possible at each phase.

n	#Phase1/2^n	#Phase2/3^n	#Phase3/3^n
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	-
42	0.000728559	-	-

are 2^n possible "crease"/"flat" assignments at phase 1. Only about 4.7% is the solution for phase 1 if $n = 6$. It decreases significantly and gets less than 1% for $n \geq 12$. The rates at phase 2 and phase 3 are against 3^n 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., 2.5% at phase 2 when $n = 6$. Such rate at every phase seems to be exponential to n according to Fig. 4.

5. Concluding Remarks

We develop the first algorithm for enumerating distinct flat-foldable single vertex crease patterns. We also experimentally show how many such patterns there are, which is done the first time as well. Improving the algorithm and investigating further for the counting problems are the future work. For example, rather than Sawada's algorithm in Theorem 3, enumeration of the sequences stated in Theorem 4 may directly improve the running time of our algorithm drastically.

We also examine the rates in each phase; experimentally, they seem to decrease exponentially. Nevertheless, we conjecture that there are exponentially many flat-foldable crease patterns. Showing theoretical lower and upper bounds also remains open.

Acknowledgments

We would like to thank Yota Otachi for his fruitful discussions and comments. This work is partially supported by MEXT/JSPS Kakenhi Grant Number 26330009 and 24106004, and JAIST Research Grant 2017 (Houga).

References

[1] M. Bern and B. Hayes, "The complexity of flat origami," Proc. 7th Ann. ACM-SIAM Symp. on Discrete Algorithms, pp.175–183, ACM, 1996.

[2] E.D. Demaine and J. O'Rourke, Geometric Folding Algorithms: Linkages, Origami, Polyhedra, Cambridge University Press, 2007.
[3] T. Hull, "Counting mountain-valley assignments for flat folds," Ars Combinatoria, vol.67, pp.175–187, 2003.
[4] D. Avis and K. Fukuda, "Reverse search for enumeration," Discrete Applied Mathematics, vol.65, no.1-3, pp.21–46, 1996.
[5] M.J. Zaki, "Efficiently mining frequent trees in a forest," Proc. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.71–80, ACM, 2002.
[6] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "An efficient algorithm for enumerating closed patterns in transaction databases," International Conference on Discovery Science, Lecture Notes in Computer Science, vol.3245, pp.16–31, Springer, 2004.
[7] J. Sawada, "Generating bracelets in constant amortized time," SIAM J. Comput., vol.31, no.1, pp.259–268, 2001.
[8] W.D. Hoskins and A.P. Street, "Twills on a given number of harnesses," J. Austral. Math. Soc. (Series A), vol.33, no.1, pp.1–15, 1982.
[9] R. De La Briandais, "File searching using variable length keys," IRE-AIEE-ACM '59, Western Joint Computer Conference, pp.295–298, ACM, 1959.
[10] E. Fredkin, "Trie memory," Commun. ACM, vol.3, no.9, pp.490–499, 1960.
[11] K.S. Booth, "Lexicographically least circular substrings," Information Processing Letters, vol.10, no.4-5, pp.240–242, 1980.

Koji Ouchi is a doctoral student in School of Information Science, Japan Advanced Institute of Science and Technology (JAIST). He received B.E. and M.E. degrees from Hokkaido University, Japan, in 2010 and 2012, respectively. He was an engineer in SEC Corporation Ltd. during 2012–2016. He entered JAIST in 2016. His research interests include computational origami, enumeration algorithms, and machine learning.

Ryuhei Uehara is a professor in School of Information Science, Japan Advanced Institute of Science and Technology (JAIST). He received B.E., M.E., and Ph.D. degrees from University of Electro-Communications, Japan, in 1989, 1991, and 1998, respectively. He was a researcher in CANON Inc. during 1991–1993. In 1993, he joined Tokyo Woman's Christian University as an assistant professor. He was a lecturer during 1998–2001, and an associate professor during 2001–2004 at Komazawa University. He moved to JAIST in 2004. His research interests include computational complexity, algorithms and data structures, and graph algorithms. Especially, he is engrossed in computational origami, games and puzzles from the viewpoints of theoretical computer science. He is a member of EATCS, and vice chair of EATCS Japan Chapter.