Electronic Preprint for Journal of Information Processing Vol.28

Regular Paper

Valid Orderings of Layers When Simple-Folding a Map

Yiyang Jia[1,a)] Jun Mitani[1,b)] Ryuhei Uehara[2,c)]

Received: November 12, 2019, Accepted: June 1, 2020

Abstract: In this work, we consider a decision problem on whether an overlapping order of all the squares of an $m \times n$ map is valid under a particular context called simple folding. This problem belongs to the field of map folding. It is a variation of both the decision problem of valid orders of a $1 \times n$ map under the context of simple folds and the decision problem of valid orders of an $m \times n$ map under the context of general folds. We provide a method and its corresponding linear-time algorithm to solve this problem, which is based on the construction of a directed graph representing the adjacency relations among squares.

Keywords: map folding problem, overlapping order, $m \times n$ map, linear-time algorithm

1. Introduction

This research is in the field of computational origami, in which *flat folding* is a significant topic among diverse problems. It concerns whether a given crease pattern can be flat folded or not. For the simplest case, i.e., when there is only one vertex in the given crease pattern, the satisfiability for the pattern to be flat folded is called *local flat-foldability*. Two crucial conditions about it are listed below.

Condition 1 (Kawasaki [1], Justin [2]): For a flat-foldable vertex, the alternate angles between its adjacent creases sum up to π.

Condition 2 (Maekawa [3], Justin [2]): For a flat-foldable vertex, the numbers of its related creases assigned to be mountains and valleys differ by ± 2.

Whereas a linear-time algorithm on deciding the local flat-foldability of an arbitrary planar crease pattern exists, it is NP-hard to decide if all the vertices can be flat folded simultaneously without self-intersection, i.e., the *global flat-foldability*. Both of the conclusions are introduced in Ref. [4].

Concerning flat-folding, some simplifications have been applied to the general problem to find simpler solutions for some limited cases. The map folding problem is a typical one that specializes the crease patterns. It limits the paper to a rectangle of size $m \times n$ and limits the crease pattern to an embedded $m \times n$ grid pattern. This problem was first presented by Jack Edmonds in 1997 with also a mapping from every non-boundary edge to a set $\{M, V\}$, namely the *Mountain-Valley assignment* (*MV assignment*) as the input. The desired output is the answer of if the given map can be folded into a square of size 1×1 on the plane or not [5]. Although it simplifies the pattern, analyzing the global flat-foldability still seems intricate. It is known so far that the flat-foldability of a $2 \times n$ map needs $O(n^9)$ time to decide [6]. A superficially simple but not flat-foldable example is shown in **Fig. 1**. This pattern is locally flat-foldable for each vertex, but it is not globally flat-foldable.

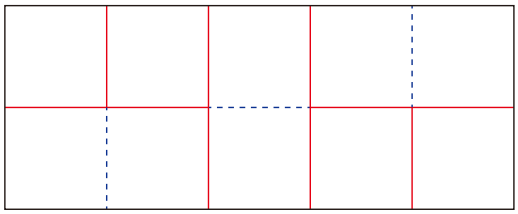


Fig. 1 A map of size 2×5 which cannot be flat-folded (The red segments indicate mountains whereas blue segments indicate valleys).

To reduce the difficulty, some further simplifications are applied to the map folding. We can classify them into three types according to the problem settings. The first type limits the shape or the size of the map, e.g. limits the value of m or n to be a constant. The second type limits the type of the folds, e.g., limits the folding operations to *simple folds*. The third type limits the configuration of the folded state, e.g., limits the ordering of layers in the folded state.

Arkin et al. [5] and Morgan [6] investigated the first type. They studied the maps of size $1 \times n$ and $2 \times n$, respectively. They focused on the flat-foldability of some given crease patterns, the accessibility to some desired states and the complexity to find a folding which leads to a certain flat-folded state.

A typical one of the second type is limiting the folds to simple folds [5] whose definition will be given in Section 2. The book [7] concludes the results in Refs. [4] and [5]. It also describes a linear-time algorithm to determine whether an MV assignment can be simple-folded or not. Another extension named *simple unfold* is introduced in Ref. [8]. Their research concerns $1 \times n$ MV-assigned maps and concludes that all the achievable flat-folded states by general folding are also achievable by only simple folds. Our research employs the result of their study and discusses it in Section 4. Furthermore, a related problem on the

1 University of Tsukuba, Tsukuba, Ibaraki 305–8577, Japan
2 School of Information Science, JAIST, Nomi, Ishikawa 923–1292, Japan
a) yiyangjia@cgg.cs.tsukuba.ac.jp
b) mitani@cs.tsukuba.ac.jp
c) uehara@jaist.ac.jp

topic of the $1 \times n$ map folding is solved in Ref. [9].

The research by Nishat [10] belongs to the third type. They explored the validity of the orders of layers in the final folded state of an $m \times n$ map. Although their method of checking and enumerating the flat-folded states follows an intuitive way, their exponential time algorithm still implicates a great likelihood of the non-existence of polynomial time algorithms.

Our research congregates the latter two types of limitations, which can be viewed as further research of Ref. [10] by limiting the general folds to simple folds. Our purpose also differs from the work in Ref. [7]. Their research concentrates on the simple-foldability of a given MV assignment regardless of how many valid flat-folded states can be induced from it; whereas our research focuses on the validity of a certain given flat-folded state. The aim is to figure out the potential relationship between simple folds and the final flat-folded state, about whether or not a given ordering of the layers in a non-assigned map indicates a valid flat-folded state achievable by only simple folds. More specifically, the input is an overlapping order O of the $m \times n$ squares of the map in the final flat state. The output is a decision on the validity of O, i.e., according to whether O is achievable only by simple folds, the algorithm outputs *true* or *false*. We also discuss the computational complexity of this decision question. Theorem 1.1 concludes our main result.

Theorem 1.1. *Given an ordering of all the squares representing a flat folded state of an $m \times n$ map, its validity under the context of simple folds can be decided in $O(mn)$ time.*

2. Terminology

In this section, we present the terminology for a clear description.

A map $M_{m,n}$ is a rectangular sheet of paper with $m \times n$ congruent squares arrayed in m rows and n columns. The two sides of the sheet are differentiated as the *front side* and the *back side*. Its *crease pattern* is specified as a grid pattern consisting of all the edges of the squares other than the ones on the boundary of the map. These edges are called *creases* and their non-boundary endpoints are called *vertices* (following the terminology in Ref. [5]). A mapping from the set of creases to the set $\{M, V\}$ is defined as the *Mountain-Valley assignment* (*MV assignment*). The MV assignment models the way to fold every crease along either of the two possible directions, i.e., a *Mountain Fold* ("M", denoted by a red solid-line segment) or a *Valley Fold* ("V", denoted by a blue dashed-line segment). Each vertex has degree 4, and it is known that this vertex is locally flat-foldable if and only if three of them are assigned the same and the remaining one is assigned differently.

$M_{m,n}$ is assumed to be at the first quadrant of a right-handed Cartesian coordinate whose unit length is equal to the side length of the squares. The bottom row of $M_{m,n}$ is aligned with the x-axis from the origin. We use $s_{i,j}$ $(0 \le i < n, 0 \le j < m)$ to refer to the square whose lower-left vertex is located at (i, j) before any fold. Both the pair $s_{i,j}$, $s_{i+1,j}$ and the pair $s_{i,j}$, $s_{i,j+1}$ are called *neighbour* squares. Without loss of generality, $s_{0,0}$ is supposed to always face the front side up during the whole folding process.

The notations for creases are defined as follows. $h_{i,j}$ refers to a horizontal crease whose left endpoint is located at (i, j) and $v_{i,j}$ refers to a vertical crease whose bottom end is located at (i, j). Then, we use h_a to indicate the set of all the creases located on $y = a, 1 \le a \le m-1$ and v_b to indicate the set of all the creases on $x = b, 1 \le b \le n-1$ and call such h_as and v_bs *crease lines*. The notations so far are illustrated by the example of 6×8 size in **Fig. 2**.

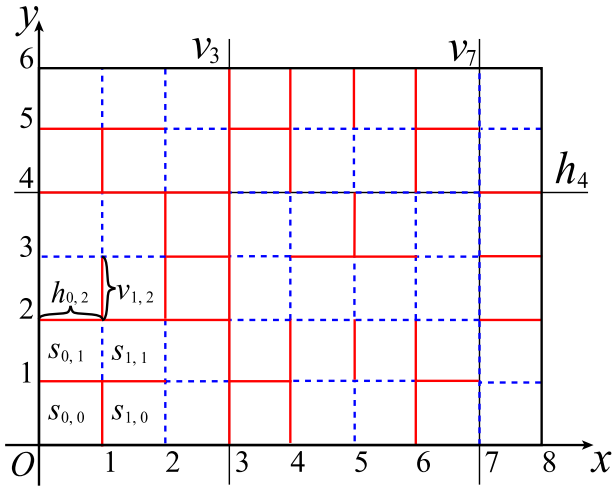


Fig. 2 An example of 6×8 map.

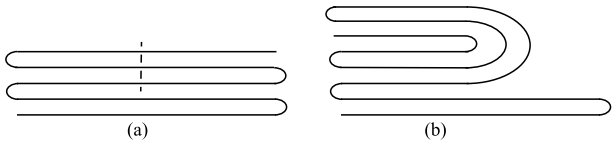


Fig. 3 An example of some-layers simple fold.

The following definitions are about the foldings. *Simple folds* focused in this research are exactly the same with the *some-layers simple fold* model in Ref. [5]. An illustration is given in **Fig. 3**. It is distinguished from the general folds by limiting that (1) the states before and after a simple fold are both flat and (2) only some continuously adjacent layers whose surfaces touch each other in pairs are folded along a single crease line. They mean that a simple fold is along a line and avoids interference on other layers. We do not concern other models of simple folds in this research. A *folding sequence* is a continuous reconfiguration of the map from its initial state R_0 (a single sheet of paper) to a target flat state R_t of the 1×1 size without tearing, stretching or self-penetrating. A *simple folding sequence* is a folding sequence composed of some ordered simple folds. For convenience, a simple folding sequence is briefly called a *simple folding* in the following descriptions. We note that every partly flat-folded state during the folding is always in the shape of a rectangle. An *overlapping order* is a list of the layers ordered from bottom to top in a flat state, which is represented by a fixed sequence of the elements in $\{s_{i,j} | 0 \le i < n \text{ and } 0 \le j < m\}$. In R_t, each square forms a single layer.

In any partly or completely flat-folded state, we say a pair of squares whose surfaces touch each other are *adjacent*. When a pair of squares s_{a_1,b_1} and s_{a_2,b_2} are adjacent and s_{a_1,b_1} is supposed to be folded below s_{a_2,b_2} in R_t, the *adjacency relation* between them is indicated by a tuple $(s_{a_1,b_1}, s_{a_2,b_2})$. As an extension, the

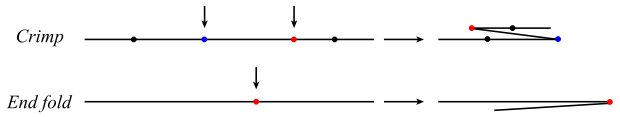


Fig. 4 The crimp and the end-fold.

adjacency relation among a set of squares $\{s_1, s_2, \ldots, s_i\}$ in which all the pair of s_a and s_{a+1} have adjacent relation (s_a, s_{a+1}) is indicated by the tuple $(s_1, s_2, \ldots, s_i)$.

A *valid ordering* is a feasible overlapping order by folding a map to the 1×1 size with only the simple folds considered in this paper.

An earlier study [5] defines two kinds of simple folds for a one-dimensional map. They are respectively called *end-folds* and *crimps*, which describe the folds illustrated in **Fig. 4**. An end-fold is a folding of either the first or the last crease along the one-dimensional map whereas a crimp is a folding of a pair of adjacent creases with different labels. We use the same terms since a single simple fold of a two-dimensional map can be seen as a simple fold of a one-dimensional map if we view the folding along a crease line in the two-dimensional map as along a crease point in the one-dimensional map. A simple folding applied on $M_{m,n}$ is composed of at most $m + n - 2$ crimps and end folds. To simplify the description, we consider a crimp as one folding operation.

We formalize the problem *The Valid Total Order Using Simple Folds* (*VTOS*) as below:

When an overlapping order O of all the squares of $M_{m,n}$ is given, is O a valid ordering reachable by a simple folding starting from R_0?

3. Outline

The procedure proposed in this paper to decide the validity of O is specified as follows: (1) Compute the MV assignment from O and check if it is everywhere locally flat-foldable (Section 4.1); if not, we return false. Otherwise, we proceed to the next step. (2) Check a necessary condition of the MV assignment to be flat-foldable with only simple folds; if this condition is not satisfied, we immediately return false. Otherwise, we obtain an incomplete order P of the simple folds (Section 4.2.1), and we proceed to the next step. (3) Construct a directed graph G to describe the adjacency relations of the squares which can be obtained from P (Section 4.3). (4) Traverse O while adding the remaining edges to G and checking if O follows the construction of G (Section 4.3). Note that in the above steps, (2) and (3) can be concluded at the same time and only part of the edges in G are assigned before applying (4). This procedure can be realized in linear time.

In Section 4, we first present the method to obtain the MV assignment and its corresponding incomplete order of simple folds. Then, we introduce how to construct a directed graph G of adjacency relations based on the known information so far. Finally, we complete our algorithm with the check of the consistency between O and G. Based on these steps, the algorithm description is given in Section 5. We provide some possible future research in Section 6.

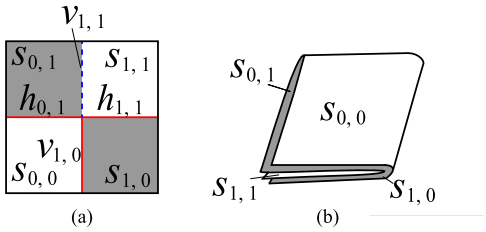


Fig. 5 The checkerboard pattern of a 2×2 map and its corresponding folded state.

4. Decision on the Validity of the Given Order

4.1 Computing the MV Assignment

We first traverse O to figure out the MV assignment of the map. Note that O may correspond to an MV assignment not everywhere locally flat-foldable. If it is not everywhere locally flat-foldable, we immediately return false. The check of local flat-foldability is concluded in $O(mn)$ time by checking the assignment of the creases around every vertex [4]. The MV assignment can be obtained by traversing O once since the label of any crease is uniquely decided by the overlapping relation between its two incident squares. Thus, the MV assignment can be clarified in $O(mn)$ time because O has $m \times n$ elements. The computation is detailed as follows.

By the reason that $s_{0,0}$ is fixed to face front side up, $s_{i,j}$ faces front side up (down) when $i + j$ is even (odd) in R_t. This property is described in Ref. [10] by embedding a checkerboard pattern into the map.

Consider an arbitrary $s_{i,j}$ facing front side up in the final folded state, if it is positioned below its neighbour square, the crease between them must be labeled V. If it is positioned over its neighbour square, the crease must be labeled M. Following this way, the labels of all the creases can be figured out through a traverse of O. A simple example of a 2×2 map is illustrated in **Fig. 5**. Given the overlapping order $(s_{1,0}, s_{1,1}, s_{0,1}, s_{0,0})$ of the folded state in (b), where $s_{0,0}$ is positioned above $s_{1,0}$ and $s_{0,1}$ makes it clear that $h_{0,1}$ and $v_{1,0}$ are labeled Ms. Since $s_{1,1}$ also faces its front side up and is positioned below $s_{0,1}$ but above $s_{1,0}$, $h_{1,1}$ is labeled M and $v_{1,1}$ is labeled V.

4.2 The Incomplete Order of Simple Folds

In this section, we first describe the way to construct an incomplete order of simple folds when the MV assignment is simple-foldable. Then, we introduce some notations based on the incomplete order.

4.2.1 Construction of the Incomplete Order P

After the MV assignment of $M_{m,n}$ is obtained by the method described in Section 4.1, next we decide whether the MV assignment is simple-foldable. If it is simple-foldable, we construct an incomplete order of simple folds $P = (p_0, p_1, \ldots, p_{t-1})$ with $t \leq m + n - 1$ according to this MV assignment, which satisfies that (1) each p_k is either a maximal set of some h_as or a maximal set of some v_bs perpendicular to those involved in p_{k+1} and (2) the crease lines in p_k are folded directly after p_{k-1} and before p_{k+1}. By this indication, a p_k uniquely describes the fold along one crease line or the folds along multiple parallel crease

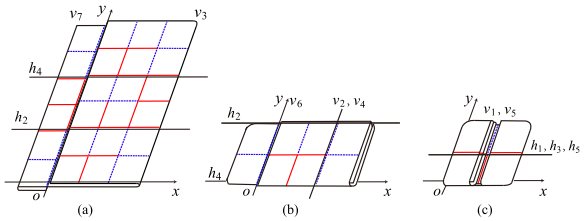


Fig. 6 Obtaining the partial order based on a known MV assignment via a simple folding.

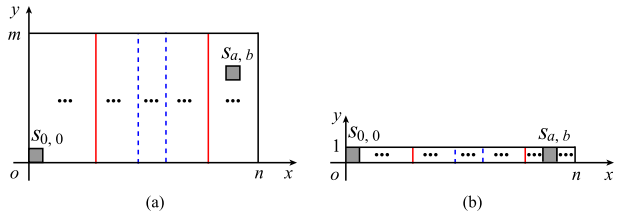


Fig. 7 The uniformity of the positions in an $m \times n$ map and a $1 \times n$ map.

lines. Without loss of generality, we assume that p_{2i} consists of h_as and p_{2i+1} consists of v_bs. Since there exist $m-1$ horizontal crease lines and $n-1$ vertical crease lines in an $m \times n$ map, $|p_0|+|p_2|+\ldots = m-1$ and $|p_1|+|p_3|+\ldots = n-1$. When the MV assignment is not simple-foldable, we conclude that O is invalid and end the process.

Roughly speaking, the method to construct P uses the one introduced in Section 14.2 in Ref. [7] where their concern is the simple-foldability of a given MV assignment. According to their description, the computation costs $O(mn)$ time.

When we approach the construction of P, an important observation is that a partly simple-folded state of $M_{m,n}$ can be viewed as a new map. At least one of the crease lines of the new map should have all creases labeled the same, or it would be impossible to apply the next simple fold.

Note that the inner order of any p_k does not affect the shape of its folded state but only affects the resulting ordering of the involved squares in such layers. This means that $M_{m,n}$ can always be folded to the shape of the largest rectangle divided by the folds. If $M_{m,n}$ is flat-foldable under the context of simple folds, the computation of P should end as $M_{m,n}$ is reduced to a map of size 1×1.

The details of the construction are omitted here because it is the same with the computation in Section 14.2 in Ref. [7]. For a better understanding, we provide an example of the process in the rest of this section. The succeeding steps are described in the following sections.

For the instance in Fig. 2, the folding consists of 5 steps. **Figure 6** illustrates the first three steps in (a), (b) and (c), respectively. In the first step, the map is folded along v_3 and v_7, i.e., $p_0 = \{v_3, v_7\}$. The check of their reasonability is implemented first locally on the pairs of creases folding to the same position, then upon the global map to check the self-intersection. In detail, first pairs of creases in the two areas of size 6×3 adjacent to v_3 and the two columns adjacent to v_7 are respectively verified to match each other; then we check if we can sequence v_3 and v_7 to achieve a feasible folding. In this case, the answer is yes since they comprise a crimp. Similarly, in the second step we fold the crimp involved in $p_1 = \{h_2, h_4\}$. Note that different inner orders of p_1 induce different overlapping orders, meaning that a given MV assignment may give more than one overlapping order; The folding in the third step is recorded by $p_2 = \{v_2, v_4, v_6\}$, which involves two end folds and no crimp since v_2, v_4 are located on the same line segment. The omitted following two steps consist of $p_3 = \{h_1, h_3, h_5\}$ and $p_4 = \{v_1, v_5\}$. In conclusion, this example can be flat folded via a simple folding with $P = (p_0, p_1, p_2, p_3, p_4)$ along exactly $6+8-2 = 12$ crease lines.

4.2.2 Notations and Some Theoretical Basis

In this section, we introduce some notations and theoretical basis. The simple folding starts from the initial state R_0. Corresponding to the simple folding sequence $P = (p_0, p_1, \ldots, p_{t-1})$, the sequence of partly flat-folded states is indicated by $\mathscr{R} = (R_0, R_1, R_2, \ldots, R_t)$ where each R_{k+1} indicates the flat-folded state directly after folding p_k. As denoted in Section 2, the last element R_t represents the final flat-folded state. For a precise description, we assume that $s_{0,0}$ is located at $(0,0)$ in every element of $\mathscr{R}$.

We use a 3-tuple $g(j,x,y)$ for $j,x,y \in Z$, which is called a *group*, as a description of the squares located at a certain coordinate (x,y) in R_j. Specifically, we can describe R_0 by the set of $g(0,x,y)$s assigned as follows. For every (x,y) with $0 \le x < n, 0 \le y < m$, let $g(0,x,y) = \{s_{x,y}\}$. For other coordinates (x,y)s where no square locates initially, let $g(0,x,y) = \emptyset$. Based on the abovementioned assumption, $s_{0,0}$ belongs to every $g(j,0,0)$.

Unless any exception is noted, in the following descriptions, p_k refers to an arbitrary element in P which folds the map from corresponding R_k to R_{k+1} and (x,y) refers to an arbitrary coordinate. For convenience, the following descriptions about groups concern only the non-empty ones.

$g(k+1,x,y)$ is defined for the sake of properly indicating the squares supposed to be positioned at (x,y) in R_{k+1}. Its precise definition will be given later in this section. We first provide the theoretical basis and proof so that it can be well-defined. For the squares involved in $g(k+1,x,y)$, we have Observation 4.1 and Lemma 4.2 as follows.

Observation 4.1. *The squares involved in the same rows (columns) in R_k are never separated into different rows (columns) in any R_u where $u > k$.*

Lemma 4.2. *The feasible inner order of p_k affects the squares involved in $g(k+1,x,y)$ only on the aspect of their overlapping order. The squares involved in $g(k+1,x,y)$ are decided, regardless of the inner order of p_k.*

Proof. We use an illustration in **Fig. 7** (a) to explain the correctness. Without loss of generality, assume that the crease lines in p_k are vertical. We consider a square $s_{a,b}$ in $g(k,x',y')$ and aim to find its corresponding $g(k+1,x,y)$.

It is clear that $y' = y = b$ holds for $s_{a,b}$. We focus on the value of x for $s_{a,b}$, which can be computed using a one-dimensional model illustrated in Fig. 7 (b). The folding is composed of a sequence of crimps and end folds [5]. Corollary 12.1.5 of Ref. [7] clarifies that the value of x for $s_{a,b}$ is certain after folding all the crease lines in p_k. The same analysis can be applied to the case when the crease lines in p_k are horizontal. Therefore, by the recursion of k, the squares involved in any $g(k+1,x,y)$ are certain. Lemma 4.2 is proven. □

Based on Lemma 4.2, we can conclude that the set of squares located at (x, y) in R_{k+1} is comprised of the set of squares located at some certain (x', y')s in R_k. So far, we can define $g(k+1, x, y)$ as $g(k+1, x, y) = \{g(k, x', y') \mid g(k, x', y')$ is supposed to be folded to (x, y) by $p_k\}$. Namely, $g(k+1, x, y)$ is a set with each element as a $g(k, x', y')$ supposed to be folded to (x, y) by p_k.

For $g(k+1, x, y)$, if the overlapping order of its elements is uniquely decided in R_{k+1}, then we say that the inner order of $g(k+1, x, y)$ is *clear*, otherwise we say that the inner order of $g(k+1, x, y)$ is *unclear*. Furthermore, the definition of *adjacent relation* is extended to a pair of groups. If the uppermost square in $g(k, x'_1, y'_1)$ touches the lowermost square in $g(k, x'_2, y'_2)$ in R_u where $u > k$, we say that they are *adjacent* and denote them by $(g(k, x'_1, y'_1), g(k, x'_2, y'_2))$ or $(g(k, x'_2, y'_2), g(k, x'_1, y'_1))$ according to which side they face up in R_t (the parity of their coordinates).

The following lemma is about the adjacency relations.

Lemma 4.3. *Any pair of adjacent squares in R_k are still adjacent in the final state R_t even though the adjacency relation between them may be converse.*

Proof. A pair of adjacent squares in R_k are always folded together by the subsequent folds. The definition of simple folds ensures that no other layers can be inserted between them. Thus a pair of adjacent squares in R_k are also adjacent in R_{k+1}. The proof of their adjacency in R_t is concluded by the recursion relation of $g(k+1, x, y)$. Their overlapping order in R_t is decided by their coordinates in R_0, which can be checked using a checkerboard pattern as mentioned in Section 4.1. The proof of Lemma 4.3 is concluded. □

Now we introduce another type of relations among groups. The crease lines of p_k divide R_k into multiple rectangles. We view each of them as a single layer in R_{k+1} and denote the set of these rectangles as $L_k = \{l(k, i)\}$ with $i = 0, 1, \ldots, |p_k|\}$. We consider a $g(k+1, x, y)$ composed of $g(k, x', y')$s from different $l(k, i)$s. Once the inner order of a $g(k+1, x, y)$ becomes clear, the overlapping order of these $l(k, i)$s is also decided in R_{k+1}. Specially, if $g(k+1, x_2, y_2)$ is composed of $g(k, x'_2, y'_2)$s from these $l(k, i)$s, then the inner order of $g(k+1, x_2, y_2)$ also becomes clear. For such $g(k, x', y')$ and $g(k, x'_2, y'_2)$ related by the same $l(k, i)$s, we say that $g(k, x', y')$ and $g(k, x'_2, y'_2)$ are *associated* and call the relations among them *interdependencies*.

4.3 The Directed Graph G and the Validity Checking

We have already introduced the construction of the incomplete order P representing the partial order of simple folds. P only provides partial information to decide the validity of O. The reason is that O should be verified by checking all the adjacency relations, whereas not all the adjacency relations can be decided only by P. To solve this problem, in this section, we introduce a directed graph $G = \{V, E\}$. G is constructed according to P and O. Each edge represents the adjacency relation (in R_t) between a pair of squares. Thus when G is concluded, E should include $m \times n - 1$ directed edges which induce a Hamiltonian path. The validity of O is checked by referencing G. For convenience to indicate the different states of G, when constructing G, each of its partly specified states achieved directly after p_k is denoted by $G_{k+1} = \{V_{k+1}, E_{k+1}\}$, which corresponds to all the possible R_{k+1}s. In other words, every G_{k+1} represents a certain middle state of G, corresponding to the adjacency relations decided by P and involved in R_{k+1}.

Initially, $G_0 = \{V_0, E_0\}$ is constructed as a graph with $V_0 = \{s_{i,j} \mid 0 \le i < n, 0 \le j < m\}$ and $E_0 = \emptyset$. Each square $s_{i,j}$ is abstracted as a node $s_{i,j}$ of G_0.

Corresponding to the final state R_t, G_t is obtained when all the elements of P are considered. Note that G_t does not represent the final state of G. In the next step we traverse O and complete G according to O. If G indicates a valid state reachable by simple folds and O is verified to be consistent with it, we claim that O is valid.

The adjacency relations decided by p_k are indicated in G by assigning edges to G_k. As mentioned before, the adjacency relation of two squares known to be adjacent in R_t can be figured out by a parity check on their initial coordinates. Thus in the following, we only consider the adjacency relations in R_t when assigning the edges. In every G_k, a directed edge from $s_{a,b}$ to $s_{c,d}$ represents the adjacency relation $(s_{a,b}, s_{c,d})$ in R_t.

G is achieved through three phases. In the first two phases we only consider P without touching O. Every G_{k+1} is constructed from G_k according to p_k. G_t is obtained before traversing O.

In the first phase, we consider the adjacency relations inside the groups whose inner orders are clear. Their adjacency relations are assigned as edges. In the second phase, we record the interdependencies among groups whose inner orders are unclear. The method is indexing the elements of associated groups consistently. In the last phase, we construct G referencing G_t and O by assigning the edges according to the recorded interdependencies and check the consistency between O and G to determine the validity.

These phases are detailed in Sections 4.3.1, 4.3.2 and 4.3.3 respectively.

4.3.1 Construction of the Adjacency Relation Graph G from P

In this phase, we concern $g(k+1, x, y)$s whose inner orders are clear. Note that $g(k, x', y')$s comprising $g(k+1, x, y)$ can have unclear inner orders. The adjacency relations involved in such $g(k+1, x, y)$s are assigned as edges in E.

Corresponding to each p_k, the $g(k, x', y')$s comprising $g(k+1, x, y)$ are certain. Furthermore, whether the inner order of $g(k+1, x, y)$ is clear or not can also be decided with the same method mentioned in Section 14.2 in Ref. [7]. We conclude that there are three possible cases where the inner order of $g(k+1, x, y)$ is clear. (1) $|p_k| = 1$. (2) The order of folds in a certain p_k is uniquely decided. (3) $g(k, x', y')$s relocated by the folds in p_k do not overlap each other.

To properly assign the edges to G corresponding to all the adjacency relations involved in $g(k+1, x, y)$, we handle $g(k, x', y')$ with an unclear inner order in the following manner. For every $g(k, x', y')$ whose inner order is unclear, we use two new types of nodes called a *top pseudo node* and a *bottom pseudo node*, denoted by $g_t(k, x', y')$ and $g_b(k, x', y')$ to represent its uppermost and lowermost element in R_t (the overlapping order may be converse to R_k), respectively. Then, the adjacency relations between $g(k, x', y')$ and the two groups upper and lower adjacent to

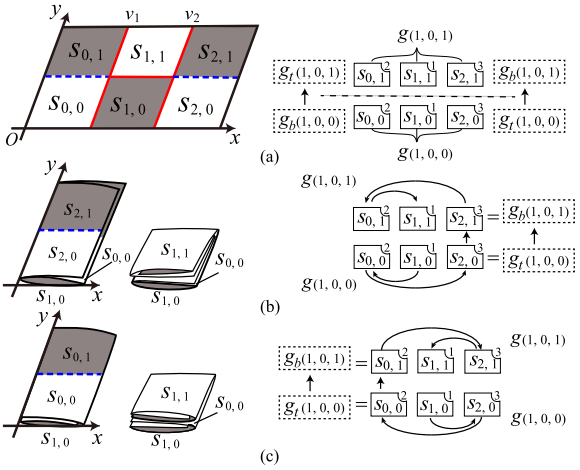


Fig. 8 The interdependencies among groups. The numbers on the upper-right corner of nodes indicate their indices.

$g(k, x', y')$ in R_{k+1} can be specified in G by assigning an incoming edge to $g_b(k, x', y')$ and an outgoing edge from $g_t(k, x', y')$. In this manner, we can assign all the edges between the corresponding adjacent nodes in G.

If the inner order of $g(k+1, x, y)$ is unclear, this edge assigning phase is skipped. Only the two pseudo nodes $g_t(k+1, x, y)$ and $g_b(k+1, x, y)$ are added to G for the subsequent handlings. All the pseudo nodes will be removed in the third phase once the inner orders of their groups become clear.

Along with the construction, G_t includes all the adjacency relations involved in the groups with clear inner orders. Other elements in E are specified during the operations in the third phase based on O, which will be detailed in Section 4.3.3.

We explain the process using the simple example shown on the left-hand side of **Fig. 8** (a) with its corresponding G initially specified as six square nodes $s_{i,j}$ $(i = 0, 1, 2, j = 0, 1)$, as illustrated in the right figure of (a). Initially, $g(0, i, j) = \{s_{i,j}\}$. The inner order of $p_0 = \{v_1, v_2\}$ remains unclear. That means corresponding to R_1 (of size 1×2), $g(1, 0, 0)$ and $g(1, 0, 1)$ are specified as $g(1, 0, 0) = \{g(0, 0, 0), g(0, 1, 0), g(0, 2, 0)\}$ and $g(1, 0, 1) = \{g(0, 0, 1), g(0, 1, 1), g(0, 2, 1)\}$. The inner orders of both groups are unclear. We add the top pseudo nodes $g_t(1, 0, i)$ and the bottom pseudo nodes $g_b(1, 0, i)$ to represent $g(1, 0, i)$ with $i = 0, 1$, respectively. $p_1 = \{h_1\}$ folds the map to R_t and decides that $g(1, 0, 1)$ is upper adjacent to $g(1, 0, 0)$ in R_t since $s_{0,0}$ faces front side up. Correspondingly, we add an edge from $g_t(1, 0, 0)$ to $g_b(1, 0, 1)$.

4.3.2 The Specification of the Interdependencies among Groups Based on *P*

In this section, we give the method to index the nodes involved in the associated groups with unclear inner orders. Even though there are multiple cases of the inner orders in these groups, the simple folds still restrict the consistency of the inner orders of these associated groups.

To indicate the interdependencies as introduced in Section 4.2.2, we assign indices to the $g(k, x', y')$s which comprise a $g(k+1, x, y)$ with an unclear inner order. We index $g(k, x', y')$ by i if it is in the rectangle $l(k, i)$ (introduced in Section 4.2.2) before folding p_k. The details are as follows.

We choose an arbitrary feasible order of p_k by the method in Section 14.2 of Ref. [7]. Each rectangle divided by the crease lines in p_k is viewed as a single segment along the axis perpendicular to the crease lines. A feasible overlapping order of these rectangles can be obtained. We index the rectangles from bottom to top to obtain $L_k = \{l(k, i) | i = 0, 1, \ldots, |p_k|\}$. According to where $g(k, x', y')$ is relocated by p_k, the index of $g(k, x', y')$ is specified as i.

Using these indices, even if the inner order of p_k is arranged differently, in any R_t reachable by simple folds, the adjacency relations among associated groups are consistent with each other. The way to assign the remaining edges according to the indices and O is specified in the third phase (Section 4.3.3).

As an example, we show the indexing for the map illustrated in Fig. 8. Corresponding to the feasible state shown on the left side of (b) where p_0 is sequenced as (v_1, v_2), the figure on the right side of (b) illustrates the indexing in G. In detail, the elements in $g(1, 0, 0)$ are ordered by an ordered set $(g(0, 1, 0), g(0, 0, 0), g(0, 2, 0))$. The elements in $g(1, 0, 1)$ are ordered by an ordered set $(g(0, 1, 1), g(0, 0, 1), g(0, 2, 1))$. In both groups, the ordinal number of each element represents its index.

4.3.3 The Completion of Adjacency Relation Graph by Traversing *O*

As mentioned before, the edges corresponding to the unclear inner orders of groups are still unspecified after the first two phases. In the third phase, G is concluded from G_t by concluding the edge assigning corresponding to every group with an unclear inner order. Meanwhile, the check for the validity of O is also implemented. Along with the construction of G, $g_t(k+1, x, y)$ and $g_b(k+1, x, y)$ are removed from G when the inner order of $g(k+1, x, y)$ becomes clear. Moreover, the edges incident to $g_t(k+1, x, y)$ and $g_b(k+1, x, y)$ are replaced by the edges incident to its uppermost and lowermost elements. Therefore, as above-mentioned, as the final state of G, a Hamiltonian path with exactly $m \times n - 1$ directed edges should be induced.

In this step, O is traversed from the first element to the last one. For every adjacent pair of squares $(s_{u,v}, s_{u',v'})$ in O whose adjacency relation is valid, there are two cases (1) and (2) of G. Case (1) corresponds to the case where the adjacency relation between $s_{u,v}$ and $s_{u',v'}$ is already decided and specified in G. It can be further divided to two subcases (1a) and (1b). Case (1a) is that there exists a corresponding directed edge between $s_{u,v}$ and $s_{u',v'}$. Case (1b) is that a directed edge is assigned from $s_{u,v}$ to a group which includes $s_{u',v'}$ and with its inner order still totally unclear at this point. In such case, we specify $s_{u',v'}$ as the lowermost element of this group. There exists no other case because based on our edge assigning process, before $(s_{u,v}, s_{u',v'})$ is traversed, we have already specified the uppermost element of the group including $s_{u,v}$ as $s_{u,v}$.

Case (2) corresponds to the interdependencies among groups. As introduced before, the adjacency relation between a pair of $g(k, x', y')$s of the same $g(k+1, x, y)$ uniquely indicates both the MV assignment and the ordinal number of a certain crease line in p_k, which clarifies the adjacency relation between corresponding rectangles $l(k, i_1)$ and $l(k, i_2)$. In other words, this adjacency relation decides the adjacency relations between all

the pairs of $g(k, a', b')$s in $l(k, i_1)$ and $l(k, i_2)$ belonging to the same $g(k+1, a, b)$. When specifying G in the second phase, all the interdependencies among the unclear adjacency relations are recorded by the indices. Therefore, we view every adjacency relation in O which is unspecified in G as new information. We add edges between the nodes corresponding to this pair of $g(k, x', y')$s and all their interdependent pairs $g(k, a, b)$s. The directions of edges are assigned consistently with R_t. In this manner, the interdependent adjacency relations are represented by the newly added edges in all the associated groups. Whether O follows the relations added in other groups or not is also checked along the traverse. An example is given at the end of this section.

During the traverse, if (1) O is always in accordance to G and (2) the inner order of every group is valid under the context of simple folds, we can claim the validity of O. For (2), whether the inner order follows the simple folds or not can be verified with the method provided in Ref. [8]. Their main conclusion is that any valid flat-folded state of a $1 \times n$ map can be unfolded to R_0 by a sequence F of simple folds and unfolds with $|F| \leq 2n$. On account of the interdependencies among the groups, each time we only have to apply the unfolds to an L_k if the inner order of $g(k, x', y')$ is unclear. Only the edge assignment in one group is checked since it can represent all its associated groups.

Once an edge is added to E, the adjacency relation represented by it is unchangeable during the after folding by Lemma 4.3. Correspondingly, we can claim that a valid R_t should respect all the adjacency relations indicated by G. The invalid orders are definitely ruled out by our specification since we strictly follow the restrictions of simple folds. In addition, any valid order corresponding to a reasonable G is accepted by our method. Hence, the correctness of our method follows.

We use the example in Fig. 8 introduced in the last section to simulate the check in the third phase. Two possible foldings and their corresponding edge assignments of G are illustrated in Fig. 8 (b) and (c). Specifically, Fig. 8 (b) shows the case when the first pair traversed in O is $(s_{1,0}, s_{0,0})$. This means p_0 is sequenced as (v_1, v_2). First, an edge is assigned from $s_{1,0}$ to $s_{0,0}$, namely the nodes indexed 1 and 2 in $g(1, 0, 0)$. Then, we assign an edge to their interdependent nodes in $g(1, 0, 1)$ (with the same indices) from $s_{0,1}$ to $s_{1,1}$ (the direction is reversed because of the parity). The next pair traversed in O should be $(s_{0,0}, s_{2,0})$ because $g(1, 0, 0)$ is supposed to be folded under $g(1, 0, 1)$. Otherwise, O is invalid. The corresponding edge assignment is from $s_{0,0}$(indexed 2) to $s_{2,0}$ (indexed 3) in $g(1, 0, 0)$ and from $s_{2,1}$(indexed 3) to $s_{0,1}$ (indexed 2) in $g(1, 0, 1)$. In addition, as $s_{1,0}$ and $s_{2,0}$ are clarified as the lowermost and the uppermost elements of $g(1, 0, 0)$, the edges incident to $g_t(1, 0, 0)$ and $g_b(1, 0, 0)$ are reassigned to $s_{2,0}$ and $s_{1,0}$, respectively. Then $g_t(1, 0, 0)$ and $g_b(1, 0, 0)$ are removed from G. At this point, all the edges in E are assigned. Then we traverse the remaining elements in O to check if their adjacency relations follow G. Another possible folding is when the first pair traversed in O is $(s_{1,0}, s_{2,0})$, corresponding to the case that p_0 is sequenced as (v_2, v_1) illustrated in (c). The edges assigned to the nodes in $g(1, 0, 0)$ include the one between $(s_{1,0}, s_{2,0})$ and the other between $(s_{2,0}, s_{0,0})$. At the same time, the edges incident to the nodes with the same indices in $g(1, 0, 1)$ are assigned. With the directions conversed, the edges assigned to the nodes in $g(1, 0, 1)$ are specified as one between $(s_{0,1}, s_{2,1})$ and one between $(s_{2,1}, s_{1,1})$. The edges reassignment is shown on the right side of (b) and (c).

5. Algorithm Description

Algorithm 1 describes the mentioned process. We use the coordinates of the squares in R_0 to represent them in the input. In this description, the ith member in O is indicated by O_i. From the flow of the algorithm, we can conclude that it can be accomplished in polynomial time.

Next we discuss that Algorithm 1 can be realized in $O(mn)$ time. Note that to achieve a linear-time realization, the coordinates of $s_{0,0}$ are not standardized to $(0, 0)$ in each step any more.

For the first two steps, the MV assignment can be obtained by traversing O once in $O(mn)$ time and the computation of the incomplete order P for simple folds costs $O(mn)$ time by analyzing the labels of all the creases only once (the result in [7]). Next, we consider the time complexity of the three phases introduced in Section 4.3. Since every group represents the squares positioned the same when folding a certain p_k, they can be figured out along with the traverse of P and cost the same time as simple folding a map, i.e., $O(mn)$ time referencing the method in Ref. [7]. Furthermore, this process can simultaneously figure out

Input : A sequence O of the ordering of layers from bottom to top // indicating $m \times n$ numbered squares
Output: A boolean value // the validity of O under the context of simple folds
Begin
  initialization
  $P \leftarrow \emptyset$ // the incomplete order sequence
  $G \leftarrow$ A directed graph with $m \times n$ nodes
  Specify the MV assignment by traversing O
  // Section 4.1. $O(mn)$
  if *the MV assignment is not locally flat-foldable* **then**
    return *false* // The map cannot be flat-folded. $O(mn)$
  end
  Construct P as an incomplete order by referring the MV assignment // Section 4.2. $O(mn)$
  if *the construction failed* **then**
    return *false* // The map cannot be folded by simple folds.
  end
  foreach p_k *in* P **do**
    // Partly specify the directed graph G (Section 4.3.1 and 4.3.2 $O(mn)$)
    if p_k *uniquely decides the adjacency relation between two squares or groups already exist* **then**
      Add edges representing the corresponding adjacency relations after folding p_k to G
    else
      Make group nodes based on their positions after the folding according to p_k // The nodes correspond to the sets $g(k+1, x, y)$s for the admissible xs and ys, and this operation is done only if the order of p_k is unclear
      For each group, add a top pseudo node and a bottom pseudo node to G
      Index the elements of each group from bottom to top according to an arbitrary flat folded state after folding p_k
    end
  end

```
Delete the isolated pseudo nodes in G
for i = 0 to m × n − 1 do
    if O_i is not in any group S then
        if O_{i+1} is in a group g and the next adjacent node of O_i is
          the bottom pseudo node of g then
            assign all the incident edges of the bottom pseudo
              node to O_{i+1} and then remove this pseudo node
              from G
            next i
        else
            if the next adjacent node of O_i is O_{i+1} then
                next i
            else
                return false
            end
        end
    else
        if O_{i+1} and O_i are in the same group then
            if O_i is not adjacent to O_{i+1} in G then
                assign the edge from O_i to O_{i+1} and the edges
                  between the same-indexed nodes in other
                  groups // the direction is decided by the side
                  the squares face up
                next i
            end
        else
            if O_{i+1} is in another group then
                // It means that all the edges in the group
                  including O_i are specified since squares of the
                  same group are adjacent in the final order;
                  otherwise O is invalid
                if O_i is in a group whose arrangement in G is
                  invalid under the context of simple folds then
                    return false // checked with the method in
                      Ref. [8] O(max{m, n})
                end
                assign all the incident edges of the top pseudo
                  node of the group containing O_i and then
                  remove this pseudo node from G
                assign all the incident edges of the bottom
                  pseudo node of the group containing O_{i+1} to
                  O_{i+1}, and then remove this pseudo node from G
                next i
            else
                return false // O_{i+1} is not in any group
            end
        end
    end
end
return true
end
```

Algorithm 1: A linear-time algorithm for *VTOS*.

whether the inner orders of the groups are clear or not. We now discuss the time complexity when constructing G. To represent the groups with unclear inner orders, there would be no more than $2 \times m \times n$ pseudo nodes added and thus there are $O(mn)$ nodes in any state of G. The edge-assigning of adjacency relations in the first phase can be implemented along with the traverse of P and involves no more than $O(mn)$ single edge additions. Thus, this phase costs $O(mn)$ time using a simple BFS algorithm. In the second phase, the indexing of elements of groups with unclear inner orders essentially equals to arranging a reasonable order for the crease lines in p_ks, which cost the same time as arranging P, i.e., $O(mn)$ time.

In the third phase, for all the adjacency relations in O, whether they are in accordance with G or not can be concluded by traversing O once, which costs $O(mn)$ time. The construction of G is also completed during this traversing. There remains only the time consumption of checking the edge assignments in groups with unclear inner orders. The orders of these groups are clarified by the overlapping orders in O and are required to be checked for the simple-foldability. This check is applied to the corresponding L_ks by seeing them as one-dimensional maps and using the method in [8]. According to their result, the process costs linear time to the factor of the number of elements. Thus, the total consumption of all this kind of checks during the whole process costs $O(m + n)$ time since there are $m + n - 2$ crease lines.

In conclusion, the realization of Algorithm 1 costs $O(mn)$ time.

6. Conclusion and Future Work

The core of our study is about this problem: for a given $m \times n$ map with an ordering of its squares, can it be folded to the given ordering by simple folds? Superficially, the given overlapping order only indicates an unverified folded state. Whereas under the context of simple folds, the total order of all the folding operations can be obtained from the overlapping order. Our strategy can be briefly concluded as: first compute a partial order on the creases set, which consists of sets of creases not sequenced. Then on occasion of the folds involved in each set, establish a mapping between the squares and their positions to achieve a directed graph on the adjacency relations. This mapping is based on the consistency between their positions and the folds no matter how the order of the folds is arranged. Finally, the check on the validity of the given order is concluded by referencing and complementing the directed graph whereas checking the uniformity between the directed graph and the given order.

Among all the possible directions for further research, we would like to emphasize the following one. When the input is changed to some partial orders, the problem may become far more complicated. It means that we have to decide a reasonable simple folding sequence from some incomplete information on the folded state. The standpoint is that a total order leads to only one case whereas a partial order may give rise to more cases. Thus we view the research on deciding whether the partial-order version is in class P as a compelling direction.

References

[1] Kawasaki, T.: On the relation between mountain-creases and valley-creases on a flat origami, *Proc. 2nd International Meeting of Origami Science and Scientific Origami*, Huzita, H. (Ed.), *Origami Science and Technology*, pp.229–237 (1989).

[2] Justin, J.: Towards a mathematical theory of origami, Miura, K. (Ed.), *Proc. 2nd International Meeting of Origami Science and Scientific Origami*, pp.15–29 (1997).

[3] Kasahara, K. and Takahama, T.: *Origami for the Connoisseur*, Japan Publications Inc. (1998).

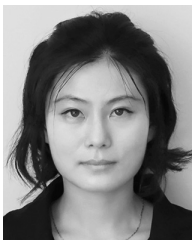
[4] Bern, M. and Hayes, B.: The Complexity of Flat Origami, *Ann. ACM-SIAM Symposium on Discrete Algorithms*, pp.175–183, ACM (1996).

[5] Arkin, E.M., Bender, M.A., Demaine, E.D., Demaine, M.L., Mitchell, J.S., Sethia, S. and Skiena, S.S.: When Can You Fold a Map, *Computational Geometry: Theory and Applications*, Vol.29, No.1, pp.23–46 (2002).

[6] Morgan, T., Thomas, D., et al.: Map folding, PhD Thesis, MIT (2012).

[7] Demaine, E.D. and O'Rourke, J.: *Geometric folding algorithms*, Cambridge University Press Cambridge (2007).

[8] Uehara, R.: Stamp foldings with a given mountain-valley assignment, *Origami 5*, pp.599–612, AK Peters/CRC Press (2016).
[9] Umesato, T., Saitoh, T., Uehara, R. and Ito, H.: Complexity of the stamp folding problem, *International Conference on Combinatorial Optimization and Applications*, pp.311–321, Springer (2011).
[10] Nishat, R.: Map folding, PhD Thesis, Bangladesh University of Engineering and Technology (2013).



Yiyang Jia is a Ph.D. student at Graduate School of Systems and Information Engineering, University of Tsukuba. Her research mainly focuses on computational origami, especially map folding problems.

Jun Mitani received his Ph.D. in Engineering from the University of Tokyo in 2004. He has been a professor of University of Tsukuba since April 2015. His research interests center in computer graphics, especially geometric modeling techniques. He studies geometry of curved origami as well as interactive design interfaces.

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.