

When Can You Fold a Map?

Esther M. Arkin*

Michael A. Bender†

Erik D. Demaine‡

Martin L. Demaine‡

Joseph S. B. Mitchell*

Saurabh Sethia†

Steven S. Skiena†

Abstract

We explore the following problem: given a collection of creases on a piece of paper, each assigned a folding direction of mountain or valley, is there a flat folding by a sequence of simple folds? There are several models of simple folds; the simplest *one-layer simple fold* rotates a portion of paper about a crease in the paper by $\pm 180^\circ$. We first consider the analogous questions in one dimension lower—bending a segment into a flat object—which lead to interesting problems on strings. We develop efficient algorithms for the recognition of simply foldable 1-D crease patterns, and reconstruction of a sequence of simple folds. Indeed, we prove that a 1-D crease pattern is flat foldable by any means precisely if it is by a sequence of one-layer simple folds.

Next we explore simple foldability in two dimensions, and find a surprising contrast: “map” folding and variants are polynomial, but slight generalizations are NP-complete. Specifically, we develop a linear-time algorithm for deciding foldability of an orthogonal crease pattern on a rectangular piece of paper, and prove that it is (weakly) NP-complete to decide foldability of (1) an orthogonal crease pattern on a orthogonal piece of paper, (2) a crease pattern of axis-parallel and diagonal (45-degree) creases on a square piece of paper, and (3) crease patterns without a mountain/valley assignment.

1 Introduction

The easiest way to refold a road map is differently.

— Jones’s Rule of the Road (M. Gardner [6])

Perhaps the best-studied problem in origami mathematics is the characterization of flat-foldable

*Department of Applied Mathematics and Statistics, SUNY, Stony Brook, NY 11794-3600, email: {estie, jsbm}@ams.sunysb.edu.

†Department of Computer Science, SUNY, Stony Brook, NY 11794-4400, email: {bender, saurabh, skiena}@cs.sunysb.edu.

‡Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, email: {eddemaine, mldemaine}@uwaterloo.ca.

crease patterns. A crease pattern is a straight-edge embedding of a graph on a polygonal piece of paper; a flat folding must fold along all of the edges of the graph, but no more. For example, two crease patterns are shown in Figure 1. The first one folds flat into a classic origami crane, whereas the second one cannot be folded flat (unless the paper is allowed to pass through itself), even though every vertex can be “locally” flat folded.

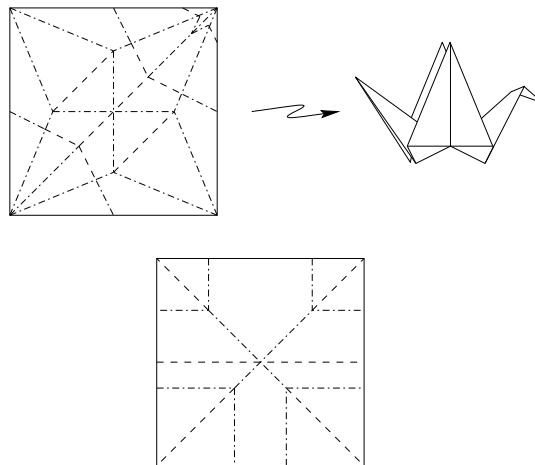


Figure 1: Sample crease patterns. Top: the classic crane. Bottom: pattern of Hull [9], which cannot be folded flat, for any mountain-valley assignment.

The algorithmic version of this problem is to determine whether a given crease pattern is flat foldable. The crease pattern may also have a direction of “mountain” or “valley” assigned to each crease, which restricts the way in which the crease can be folded. (Our figures adhere to the standard origami convention that valleys are drawn as dashed lines and mountains are drawn as dot-dashed lines.)

It is known that the general problem of deciding flat foldability of a crease pattern is NP-hard (Bern and Hayes [2], SODA’96). In this paper, we consider the important and very natural case of recognizing crease patterns that arise as the result of flat foldings using *simple foldings*. In this model, a flat folding is made by a sequence of *simple folds*, each of which folds one or more layers of paper along a single line

segment. As we define in Section 2, there are different types of simple folds (termed “one-layer,” “some-layers,” and “all-layers”), depending on how many layers of paper are required or allowed to be folded along a crease.

Note that not every flat folding can be achieved by a simple folding. For example, the crane in Figure 1 (top) cannot be made by a simple folding. Also, the hardness gadgets of [2] require non-simple folds.

The problem we study in this paper is that of determining whether a given crease pattern (usually with specified mountain and valley assignments) can be folded flat by a sequence of simple folds, and if so, to construct such a sequence of folds.

Several of our results are based on the special case in which the creases in the piece of paper are all parallel to one another. This case can be seen to be equivalent to a *one-dimensional* folding problem of folding a line segment (“paper”) according to a set of prescribed crease points (possibly labeled “mountain” or “valley”). We will therefore refer to this special case, which has a rich structure of its own, as the “1-D” case to distinguish it from the general 2-D problem. In contrast to the 2-D problem, we show that 1-D flat foldability is equivalent to 1-D simple foldability.

Motivation. In addition to its inherent interest in the mathematics of origami, our study is motivated by applications in sheet metal and paper product manufacturing, where one is interested in determining if a given structure can be manufactured using a specified creasing machine (e.g., a “pan brake”), which is typically restricted to performing simple folds. While origamists can develop particular skill in performing non-simple folds to make beautiful artwork, practical problems of manufacturing with sheet goods require simple and constrained folding operations. Our goal is to develop a first suite of results that may be helpful towards a fuller understanding of the several manufacturing problems that arise, e.g., in making three-dimensional cardboard and sheet-metal structures.

Related Work. Our problem is related to the classic combinatorics question of *map folding* [14]. This question asks for the *number* of foldings of a particular crease pattern, namely an $m \times n$ rectangular grid, by a sequence of simple folds. See also the discus-

sion in Gardner’s book [6]. In contrast with this combinatorics question, we study the algorithmic complexity of the decision problem, also in some more general instances of crease patterns.

The mathematical and algorithmic problems arising in the study of flat origami have been examined by several researchers, e.g., Hull [9], Justin [10], Kawasaki [12], and Lang [13]. Of particular relevance to our work is the SODA’96 paper by Bern and Hayes [2], in which the general problem of deciding flat foldability of a crease pattern is shown to be NP-hard. In SoCG’99, Demaine et al. [4] used computational geometry techniques to show that any polygonal (connected) silhouette can be obtained by simple folds from a rectangular piece of paper.

Our model of simple folding is also closely related to “pureland origami,” a restriction introduced by Smith [17, 18]. Pureland folds include simple folds, but they also allow paper to be “tucked” into pockets, as well as “opened up” into three dimensions provided that no creases are made during the process.

Summary of Our Results.

(1) We analyze the 1-D one-layer and some-layers cases, giving a full characterization of flat-foldability and an $O(n)$ algorithm for deciding foldability and producing a folding sequence, if one exists.

(2) We analyze the 1-D all-layers case as a “string folding” problem. In addition to a simple $O(n^2)$ algorithm, we give an algorithm utilizing suffix trees that requires time linear in the bit complexity of the input, and a randomized algorithm with expected $O(n)$ running time.

(3) We give a linear-time algorithm for deciding foldability of orthogonal crease patterns on a rectangular paper (the “map folding problem”), in the one-, some-, and all-layers cases, based on our 1-D results.

(4) We prove that it is (weakly) NP-complete to decide foldability of an orthogonal crease pattern on a piece of paper that is more general than a rectangle: a simple orthogonal polygon.

(5) We also prove that it is (weakly) NP-complete to decide foldability of a rectangular piece of paper with a crease pattern that includes *diagonal* creases (at 45-degrees), in addition to axis-parallel creases.

(6) We show that it is (weakly) NP-complete to

decide foldability of a orthogonal piece of paper having a crease pattern for which no mountain-valley assignment is given.

2 Definitions

We are concerned with foldings in one and two dimensions. A one-dimensional piece of paper is a (line) *segment* in \mathbf{R}^1 . A two-dimensional piece of paper is a (connected) *polygon* in \mathbf{R}^2 , possibly with holes. In both cases, the paper is folded through one dimension higher than the object; thus, segments are folded through \mathbf{R}^2 and polygons are folded through \mathbf{R}^3 . *Creases* have one less dimension; thus, a crease is a point on a segment and a line segment on a polygon.

A *crease pattern* is a collection of creases on the piece of paper, no two of which intersect except at a common endpoint. A *folding* of a crease pattern is an isometric embedding of the piece of paper, bent along every crease in the crease pattern (and not bent along any segment that is not a crease). In particular, each facet of paper must be mapped to a congruent copy, the connectivity between facets must be preserved, and the paper cannot pass through itself. See Figure 2.

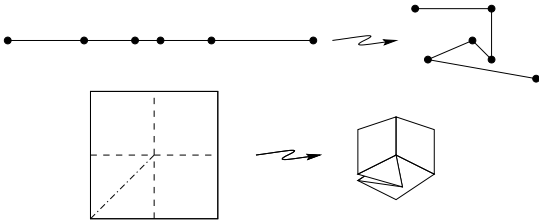


Figure 2: Sample non-flat foldings in one and two dimensions.

A *flat folding* has the additional property that it lies in the same space as the unfolded piece of paper. That is, a flat folding of a segment lies in \mathbf{R}^1 , and a flat folding of a polygon lies in \mathbf{R}^2 . In truth, there can be multiple layers of paper at a point, so the folding really occupies a finite number of infinitesimally close copies of \mathbf{R}^1 or \mathbf{R}^2 . See Figure 3. More formally, a flat folding can be specified by a function mapping the vertices to their folded positions, together with a partial order of the facets of paper that specifies their overlap order [2, 9, 13].

If we orient the piece of paper to have a top and bottom side, we can talk about the *direction* of a

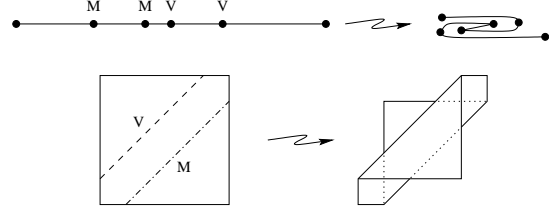


Figure 3: Sample flat foldings in one and two dimensions. Mountains and valleys are denoted by M's and V's, respectively.

crease in a flat folding. A *mountain* brings together the bottom sides of the two adjacent facets of paper, and a *valley* brings together the top sides. A *mountain-valley assignment* is a function from the creases in a crease pattern to $\{M, V\}$. This is the labeling shown in Figure 3. Together, a crease pattern and a mountain-valley assignment form a *mountain-valley pattern*.

This paper is concerned with the following generic question.

Problem: Simple Folding

Given a mountain-valley pattern, is there a simple folding satisfying the specified mountains and valleys? If so, construct such a simple folding.

There are three natural versions of this problem, depending on the type of “simple folds” allowed. In general, a *simple folding* is a sequence of simple folds. Each simple fold takes a flat-folded piece of paper, and folds it into another flat folding using additional creases. There are three types of simple folds: one-layer, all-layers, and some-layers.

A *one-layer simple fold* f is a crease on the folded piece of paper, together with a direction. If we look at the unfolded piece of paper, then f partitions it into two parts, call them A and B . Performing f corresponds to rotating A about f by $\pm 180^\circ$, where \pm depends on the fold direction, if this does not cause the paper to cross itself. This makes just a single crease, which is what we mean by folding one layer of paper.

An *all-layers simple fold* f is also a crease together with a direction. Now we consider the partition of the flat folding (instead of the unfolded piece of paper) by f into two parts, call them A and B again. Performing f corresponds to rotating (all layers of) A about f by $\pm 180^\circ$. This makes a crease

through all of the layers of paper at f . Note that this type of fold can never cause the paper to cross itself.

Finally, a *some-layers simple fold* f is the most general. It takes some of the top [bottom] layers of A , and rotates them about f by 180° [-180°], provided the paper does not cross itself.

3 1-D: One-Layer and Some-Layers

This section is concerned with the 1-D one-layer simple-fold problem. We will prove the surprising result that we only need to search for one of two local operations to perform. The two operations are called *crimps* and *end folds*, and are shown in Figure 4.

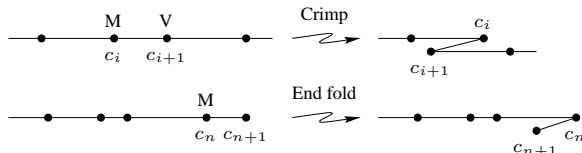


Figure 4: The two local operations for one-dimensional one-layer folds.

More formally, let c_1, \dots, c_n denote the creases on the segment, oriented so that c_i is left of c_j for $i < j$. Let c_0 [c_{n+1}] denote the left [right] end of the segment. Despite the “ c ” notation (which is used for convenience), c_0 and c_{n+1} are not considered *creases*; instead they are called the *ends*.

First, a pair (c_i, c_{i+1}) of consecutive creases is *crimpable* if c_i and c_{i+1} have opposite directions and

$$|c_{i-1} - c_i| \geq |c_i - c_{i+1}| \leq |c_{i+1} - c_{i+2}|.$$

Crimping such a pair corresponds to folding c_i and then folding c_{i+1} , using one-layer simple folds.

Second, c_0 is a *foldable end* if $|c_0 - c_1| \leq |c_1 - c_2|$, and c_{n+1} is a *foldable end* if $|c_{n-1} - c_n| \geq |c_n - c_{n+1}|$. *Folding* such an end corresponds to performing a one-layer simple fold at the nearest crease (crease c_1 for end c_0 , and crease c_n for end c_{n+1}).

We claim that one of the two local operations exists in any flat-foldable 1-D mountain-valley pattern. We claim further that an operation exists for any pattern satisfying a certain “mingling property.” Specifically, a 1-D mountain-valley pattern is called *mingling* if for every sequence c_i, c_{i+1}, \dots, c_j of consecutive creases with the same direction, either

1. $|c_{i-1} - c_i| \leq |c_i - c_{i+1}|$; or
2. $|c_{j-1} - c_j| \geq |c_j - c_{j+1}|$.

We call this the mingling property because, for maximal sequences of consecutive creases with the same direction, it says that there are folds of the opposite direction nearby. In this sense, the mountain-valley pattern is “crowded” and the mountains and valleys must “mingle” together.

First we show that mingling mountain-valley patterns include flat-foldable patterns:

Lemma 3.1 *Every flat-foldable 1-D mountain-valley pattern is mingling.*

Proof: Consider a flat folding of a mountain-valley pattern, and let c_i, \dots, c_j be consecutive creases with the same direction. The portion c_{i-1}, \dots, c_{j+1} of the segment forms a “spiral” (see Figure 5) which can be in one of two configurations:

1. (c_{i-1}, c_i) is the outermost edge of the spiral, and (c_j, c_{j+1}) is the innermost; or
2. (c_j, c_{j+1}) is the outermost edge of the spiral, and (c_{i-1}, c_i) is the innermost.

Now if $|c_{i-1} - c_i| > |c_i - c_{i+1}|$, then (c_{i-1}, c_i) must be the outermost edge of the spiral, or else (c_{i-1}, c_i) would penetrate through c_{i+1} . Similarly, if $|c_{j-1} - c_j| > |c_j - c_{j+1}|$, then (c_{j-1}, c_j) must be the outermost edge of the spiral. Therefore, we cannot have both. \square

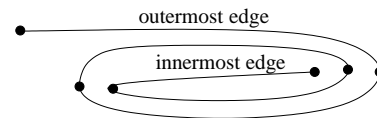


Figure 5: The innermost edge of a spiral cannot be longer than the adjacent edge, in contrast to the outermost edge which can be arbitrarily long.

Next we show that having the mingling property suffices to imply the existence of a single crimpable pair or foldable end.

Lemma 3.2 *Any mingling 1-D mountain-valley pattern has either a crimpable pair or a foldable end.*

Proof: Let i be maximum such that c_1, \dots, c_i all have the same direction. By the mingling property, either $|c_0 - c_1| \leq |c_1 - c_2|$ or $|c_{i-1} - c_i| \geq |c_i - c_{i+1}|$. In the former case, c_0 is a foldable end, so we have the desired result. A generalization of the latter case is that we have c_i, \dots, c_j all with the same orientation, and $|c_{j-1} - c_j| \geq |c_j - c_{j+1}|$. If $j = n$,

then c_{n+1} is a foldable end, so we have the desired result. Otherwise, let k be maximum such that c_{j+1}, \dots, c_k all have the same direction. By the mingling property, either $|c_j - c_{j+1}| \leq |c_{j+1} - c_{j+2}|$ or $|c_{k-1} - c_k| \geq |c_k - c_{k+1}|$. In the former case, (c_j, c_{j+1}) is a crimpable pair, so we have the desired result. In the latter case, induction applies. \square

Ideally, we could show at this point that performing either of the two local operations preserves the mingling property, and hence a mountain-valley pattern is mingling precisely if it is flat-foldable. Unfortunately this is false, as illustrated in Figure 6. Instead, we must prove that flat foldability is preserved by each of the two local operations; in other words, if we treat the folded object from a single crimp as a new segment, it is flat foldable.

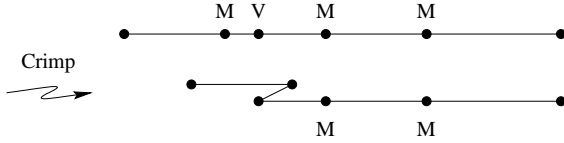


Figure 6: A mingling mountain-valley pattern that when crimped is no longer mingling and hence not flat foldable. Indeed, the original mountain-valley pattern is not flat foldable.

Lemma 3.3 *Folding a foldable end preserves foldability.*

Proof: This is obvious because folding a foldable end is equivalent to chopping off a portion of the segment. Thus, if we take a flat folding of the original pattern, remove that portion of the segment, and double up the number of layers for the adjacent portion of the segment, we have a flat folding of the new object. \square

Lemma 3.4 *Crimping a crimpable pair preserves flat foldability.*

Proof: Let (c_i, c_{i+1}) be a crimpable pair, and assume by symmetry that c_i is a mountain and c_{i+1} is a valley. Consider a flat folding F of the original segment. Let us regard the segment (c_i, c_{i+1}) as flipping over during the folding, so that the remainder of the (unfolded) segment keeps the same orientation. Thus, (c_{i-1}, c_i) is above (c_i, c_{i+1}) which is above (c_{i+1}, c_{i+2}) . Now suppose that F places some layers of paper in between (c_i, c_{i+1}) and (c_{i+1}, c_{i+2}) . Then these layers of paper can be moved to immediately

above (c_{i-1}, c_i) , because (c_{i-1}, c_i) is at least as long as (c_i, c_{i+1}) , and hence there are no barriers closer than c_i . See Figure 7. Similarly, we move material between (c_i, c_{i+1}) and (c_{i-1}, c_i) to immediately below (c_{i+1}, c_{i+2}) . In the end, we have a flat folding of the object obtained from making the crimp (c_i, c_{i+1}) . \square

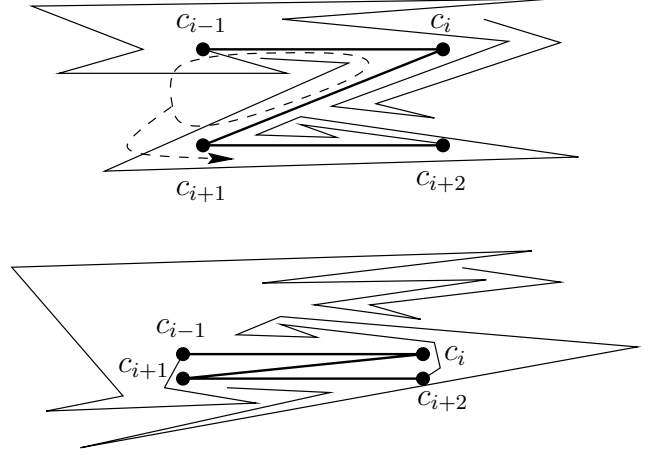


Figure 7: Moving layers of paper out of the zig-zag formed by a crimp (c_i, c_{i+1}) .

Combining all of the previous results, we have the following:

Theorem 3.1 *Any flat-foldable 1-D mountain-valley pattern can be folded by a sequence of crimps and end folds.*

Proof: By Lemma 3.1, the pattern is mingling, and hence by Lemma 3.2 we can find a crimpable pair or a foldable end. Making this fold preserves flat foldability by Lemmas 3.3 and 3.4, so by induction the result holds. \square

A particularly interesting consequence of this theorem is the following:

Corollary 3.1 *The following are equivalent for a 1-D mountain-valley pattern P :*

1. P has a flat folding.
2. P has a some-layers simple folding.
3. P has a one-layer simple folding.

Finally, let us show that Theorem 3.1 leads to a simple linear-time algorithm:

Theorem 3.2 *The 1-D one-layer and some-layers simple-fold problems can be solved in $O(n)$ worst-case time on a machine supporting arithmetic on the input lengths.*

Proof: First note that it is trivial to check in constant time whether a pair of consecutive folds form a crimp or whether an end is foldable. We begin by testing all such folds, and hence in linear time have a linked list of all possible folds at this time. We also maintain reverse pointers from each symbol in the string to the closest relevant possible fold. Now when we make a crimp or an end fold, only a constant number of previously possible folds can no longer be possible, and a constant number of previously impossible folds can be newly possible. These folds can be discovered by examining a constant-size neighborhood of the performed fold. We remove the old folds from the list of possible folds, and add the new folds to the list. Then we perform the first fold on the list, and repeat the process. By Theorem 3.1, if the list ever becomes empty, it is because the mountain-valley pattern is not flat foldable. \square

4 1-D: All-Layers Simple Folds

The 1-D all-layers simple-fold problem can be cast as an interesting “string folding” problem. (This folding problem is not to be confused with the well-known protein/string folding problem in biology [3].) The input mountain-valley pattern can be thought of as a string of lengths interspersed with mountain and valley creases. Specifically, we will assume that the input lengths are specified as integers or equivalently rational numbers. (Irrational numbers can be replaced by close rational approximations, provided the sorted order of the lengths is preserved.)

Thus, an input string is of the form $d_0 c_1 d_1 c_2 \dots c_{n-1} d_{n-1} c_n d_n$, where each $c_i \in \{M, V\}$ and each d_i is a positive rational number. We call each c_i and d_i a *symbol* of the string. It will be helpful to introduce some more uniform notation for symbols. For a string S of length n , we denote the i th symbol by $S[i]$, where $1 \leq i \leq n$.

When we make an all-layers simple fold, we cannot “cover up” a crease except with a matching crease (which when unfolded is in fact the other direction), because otherwise this crease will be im-

possible to fold later. To formalize this condition, we define the *complement* of symbols in the string: $\text{comp}(d_i) = d_i$, $\text{comp}(M) = V$, and $\text{comp}(V) = M$. Finally, we call a crease (M or V symbol) $S[i]$ *allowable* if $S[i - x] = \text{comp}(S[i + x])$ for all $1 \leq x \leq \min(i - 1, n - i)$, except that $S[1]$ and $S[n]$ (the ends) are allowed to be shorter than their complements.

Lemma 4.1 *A mountain-valley pattern can be folded by a sequence of all-layers simple folds precisely if there is an allowable fold, and the result after performing that fold has an allowable fold, and so on, until all creases of the segment have been folded.*

Proof: Performing an all-layers simple fold that is not allowable forbids us from all-layers simple folding certain creases, and hence the resulting segment cannot be completely folded after that point. Therefore, only allowable folds can be in the sequence. It remains to show that performing an allowable fold preserves foldability by a sequence of all-layers simple folds. But performing an allowable fold is equivalent to removing the smaller portion of paper to one side of the fold. Hence, it can only make more (allowable) folds possible, so the mountain-valley pattern remains foldable. \square

By Lemma 4.1, the problem of testing foldability reduces to repeatedly finding allowable folds in the string. Testing whether a fold at position i is allowable can clearly be done in $O(1 + \min\{i - 1, n - i\})$ time, by testing the boundary conditions and whether $S[i - x] = \text{comp}(S[i + x])$ for $1 \leq x \leq \min(i - 1, n - i)$. Explicitly testing all creases in this manner would yield an $O(n^2)$ -time algorithm for finding an allowable fold (if one exists). Repeating this $O(n)$ times results in a naive $O(n^3)$ algorithm for testing foldability.

This cubic bound can be improved by being a bit more careful. In $O(n^2)$ time, we can determine for each crease $S[i]$ the largest value of k for which $S[i - x] = \text{comp}(S[i + x])$ for all $1 \leq x \leq k$. Using this information it is easy to test whether a crease $S[i]$ is allowable. After making one of these allowable folds, we can in $O(n)$ time update the value of x for each crease, and hence maintain the collection of allowable folds in linear time. This gives an overall $O(n^2)$ bound, which we now proceed to improve

further.

We present two efficient algorithms for folding strings. The algorithm in Section 4.1 is based on suffix trees and runs in time linear in the bit complexity of the input. In Section 4.2, we use randomization to obtain a simpler algorithm that runs in $O(n)$ time.

4.1 Suffix-Tree Algorithm

In this section, we prove the following:

Theorem 4.1 *A string S of length n can be tested for all-layers simple foldability, in time that is dominated by that to construct a suffix tree on S .*

The difficulty with the time bound is that sorting the alphabet seems to be required. Other than the time to sort the alphabet, it is possible to construct a suffix tree in $O(n)$ time [5]. To sort the alphabet in the comparison model, $O(n \log n')$ time suffices, where n' is the number of distinct input lengths. In particular, if the input lengths are encoded in binary, then the algorithm is linear in this bit complexity. On a RAM, the current state-of-the-art algorithm for integer sorting [19] uses $O(n(\log \log n)^2)$ time and linear space.

Proof: Let S^C be the complement string of S (i.e., the complement of each letter of S), and let S^R be the reverse string of S . Position i of S represents an allowable fold precisely if the first $\min(i-2, n-i+1)$ characters of the suffix of S^R starting in the $(n-i+2)$ nd position are identical to the suffix of S^C starting in the $(i+1)$ st position, and the single endpoint of S ($S[1]$ if $i-1 < n-i$, $S[n]$ if $n-i < i$) has length \leq to its complement.

We build a single suffix tree containing all suffixes of S^C and S^R in $O(n)$ time. Further, we augment this tree with the capability to perform least-common ancestor (LCA) queries in constant time after linear preprocessing time [8, 16]. This LCA data structure enables us to return the length of the longest prefix match of two given suffixes in constant time.

To find the end-most possible fold, we can search for the longest prefix match of $S^C[i+1]$ and $S^R[n-i+2]$, where the j th fold attempt takes place at $i = (j-1)/2$ if j is odd, and $i = n+1-j/2$ if j is even. Thus the attempted folds alternate in from the left and right ends. A fold can occur at i if $S[i]$ equals M or V , and the length of the longest prefix match between $S^C[i+1]$ and $S^R[n-i+2]$

is $\min(i-1, n-i)$, or if the boundary condition above is satisfied. We then perform this first legal fold, thus reducing the length of S . We can continue our scan for the next fold by appropriately reducing the length of the necessary longest prefix match to reflect the new end of the string. Note that the suffix tree remains unchanged, and hence once one is computed, the folding process takes $O(n)$ time. \square

4.2 Randomized Algorithm

In this section we describe a simple randomized algorithm that solves the 1-D all-layers simple-fold problem in $O(n)$ time. There are two parts to the algorithm:

1. assigning labels to the input lengths so that two lengths are equal precisely if they have the same label; and
2. finding and making allowable folds.

The first part is essentially element uniqueness, and can be solved in linear expected time using hashing. For example, the dynamic hashing method described by Motwani and Raghavan [15] supports insertions and existence queries in $O(1)$ expected time. We can use this data structure as follows. For each input length, check whether it is already in the hash table. If it is not, we assign it a new unique identifier, and add it to the hash table. If it is, we use the existing unique identifier for that value (stored in the hash table). Let n' denote the number of distinct labels found in this process (or 2, whichever is larger).

For the second part, we will show that each performed fold can be found in $O(1+r)$ time, where r is the number of creases removed by the discovered fold (in other words, the minimum length to an end of the segment to be folded). However, it is possible that the algorithm makes a mistake, and that some of the reported folds are not actually possible. Fortunately, mistakes can be detected quickly, and after $O(1)$ expected iterations the pattern will be folded. (Unless of course the pattern is not flat foldable, in which case the algorithm reports this fact correctly.)

The algorithm proceeds simultaneously from both ends of the segment, so that it will find an allowable fold in time proportional to the minimum length from either end. At any point, the algorithm has

a *fingerprint* of the string traversed before reaching that point, as well as a fingerprint of the corresponding string immediately after that point (reversed and complemented). These fingerprints are maintained in $O(1)$ time per step along the segment. If the fingerprints match, then with high probability the underlying vectors also match, and we have an allowable fold. When we find such a fold (which takes $O(1+r)$ time), the creases on the short side are discarded and the two searches are restarted starting from both ends of the segment. This process is repeated until no allowable folds are found, in which case either the folding is complete (there are no creases left to perform) or the crease pattern is not foldable by a sequence of all-layers simple folds (creases remain). In the former case, the folding sequence must be double checked (again using $O(1+r)$ time per fold), and if it is incorrect, the entire process is repeated with a new randomly chosen “basis” for fingerprints.

The fingerprints are based on Karp and Rabin’s randomized string matching algorithm [11]. We treat a substring as the base- n' representation of an integer, where we use $0, \dots, n' - 1$ to denote one of the lengths, and 0 or 1 to denote a fold direction. Then the fingerprint of a substring is simply this integer modulo p for a randomly chosen prime p . This fingerprint can be updated easily in constant time. To add a symbol to the end of the string, we multiply the current fingerprint by n' , and add on the new symbol. To add a symbol to the beginning of the string (which is necessary for the reverse complement substring), we add on the new symbol times $(n')^k$ where k is the current length of the string (we maintain $(n')^k \bmod p$ throughout the computation).

By choosing the prime p randomly from the range $2, \dots, n^3$, the probability that this algorithm makes a mistake after at most n folds is $O((\log n)/n)$; see [11]. More generally, if we choose p from the range $2, \dots, n^c$, then the probability of failure is $O(c(\log n)/n^{c-2})$. Thus, with high probability, the algorithm gives a correct positive answer (it always gives correct negative answers). To obtain guaranteed correctness, we simply check the answer and repeat the entire process upon failure.

Finally, one detail remains. Because an exact match is not required on the last length for a fold to be allowable, both fingerprints on either side ex-

clude the last symbol, and we make a separate check that the length at the end is less than or equal to the length onto which it folds. This requires only $O(1)$ additional time per fingerprint test.

In conclusion, the algorithm we have presented proves the following result:

Theorem 4.2 *The 1-D all-layers simple-fold problem can be solved in $O(n)$ expected time on a machine supporting random numbers and hashing of the input lengths.*

5 Orthogonal Simple Folds in 2-D

In this section, we generalize our results for 1-D simple folds to *orthogonal* 2-D crease patterns, which consist only of horizontal and vertical folds on a rectangular piece of paper, where horizontal and vertical are defined by the sides of the rectangular paper. We handle all three cases: one-, some-, and all-layers folds.

In such a pattern, the creases must go all the way through the paper, because every vertex of a flat-foldable crease pattern has degree at least four [2, 9]. Hence, the crease pattern is a *map* or grid of creases. To know what time bounds we desire, we must first discuss encoding the input. A natural encoding of maps specifies the height of each row and the width of each column, thereby using $n_1 + n_2$ space for an $n_1 \times n_2$ grid. The mountain-valley assignment, however, requires $\Theta(n_1 n_2)$ space to specify the direction for each edge of the grid. Hence, our goal of linear time amounts to being linear in $n = n_1 n_2$.

Note that in exactly one of the two orientations, vertical or horizontal, there must be at least one crease line, all the way across the paper, that is entirely valley or mountain. (It cannot be that there is both a horizontal crease and a vertical crease all the way through the paper, since their intersection would be a vertex that is locally unfoldable.) Without loss of generality, assume it is vertical. Let the set of these vertical fold lines be V . Note that all fold lines in V must be folded before any other fold. Thus we have a corresponding 1-D problem (one-, some- or all-layer folds) with added restrictions that the non- V folds must match up appropriately after all the folds in V are made. Since V contains at least one fold, performing these folds has (strictly) reduced the size of the problem, and we continue. The base case con-

sists of just horizontal or vertical folds, which corresponds to a 1-D problem. In summary we have:

Lemma 5.1 *If a crease pattern is foldable, it remains foldable after the folds in V have been made in any feasible way considering V to be a 1-D problem and ignoring other creases.*

To find V quickly we maintain the number of mountain and valley creases for each row and column of creases. We maintain these numbers as we make folds in V . Every time the number of mountain or valley creases hits zero in a column or a row, we add the row or column to a list to be used as the new V in the next step. In summary, we have the following:

Theorem 5.1 *The problem of deciding simple foldability of an orthogonal crease pattern on a rectangular piece of paper can be solved in linear time.*

6 Hardness of Simple Folds in 2-D

In this section we prove that the problem of deciding whether a 2-D axis-parallel mountain-valley pattern can be simply folded is (weakly) NP-hard, if we allow the initial paper to be an arbitrary orthogonal polygon. We also show that it is (weakly) NP-hard to decide whether a mountain-valley pattern on a square piece of paper can be folded by some-layers simple folds, if the creases are allowed to be axis-parallel *plus* at a 45-degree angle.

Both hardness proofs are based on a reduction from an instance of PARTITION: Given a set X of n integers a_1, a_2, \dots, a_n whose sum is A , does there exist a set $S \subset X$ such that $\sum_{a \in S} a = A/2$? For convenience we define the set $\bar{S} = X \setminus S$. Also, without loss of generality, we assume that $a_1 \in S$.

The PARTITION problem is known to be (weakly) NP-hard [7].

We transform an instance of the PARTITION problem into an orthogonal 2-D crease pattern on a orthogonal polygon, as shown in Figure 8.

In the figure, all creases are valleys. There is a staircase of width ε corresponding to a_1, \dots, a_n , where $0 < \varepsilon < 2/3$. There is a step in the staircase of length a_i corresponding to each element a_i in X . L is a constant greater than $A/2$. Also $W_2 > W_1$. Also let there be a coordinate system with horizontal x -axis and vertical y -axis.

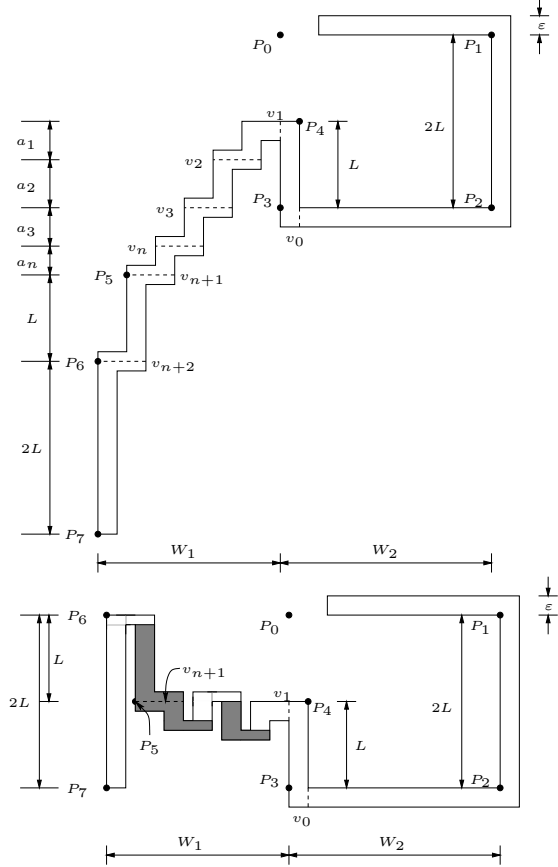


Figure 8: Top: Hardness reduction from PARTITION problem. Bottom: Semi-folded staircase confined between y coordinates of P_1 and P_2 . The top side of the paper is drawn white and the other side is drawn gray.

Lemma 6.1 *If the PARTITION instance has a solution, then the crease pattern in Figure 8 is simply foldable.*

Proof: For $2 \leq i \leq n$, fold valley v_i if exactly one of a_{i-1} and a_i is in S .

After these folds, as we travel along the steps corresponding to a_1, \dots, a_n , we travel in negative y direction for elements that belong to S and in positive y direction for elements that belong to \bar{S} .

Since the sums of elements of both S and \bar{S} are $A/2$, the point P_5 has the same y -coordinate as the point P_4 after these folds.

Also since $L > A/2$, the steps corresponding to a_i 's are confined to remain in between the y coordinates of points P_1 and P_2 .

Since P_5 has the same y -coordinate as P_4 and since the vertical distance between P_5 and P_6 is L ,

point P_6 will have the same y -coordinate as either P_1 or P_2 . Now fold the valley v_{n+2} . Since vertical distance between P_6 and P_7 is $2L$, the y -coordinate of P_7 will be same as that of P_1 or P_2 and the step between P_6 and P_7 will lie exactly between the y -coordinates of P_1 and P_2 . This is illustrated in Figure 8 (bottom).

Now fold valley v_1 . Since $W_2 > W_1$, the partly folded staircase, which currently lies between the y -coordinates of P_1 and P_2 , fits within the rectangle $P_0P_1P_2P_3$.

Now fold valley v_0 . We now have the semi-folded stairs on the right and the rectangular frame $P_0P_1P_2P_3$ on the left.

Finally, fold all of the remaining unfolded valleys in the staircase. This can be done as the rectangular frame is now on the left of P_4 and all the steps of the staircase are on the right of P_4 . \square

Lemma 6.2 *If the crease pattern in Figure 8 is simply foldable, there exists a solution to the PARTITION instance.*

Proof: If either v_0 or v_1 is folded without having the staircase confined between the y -coordinates of P_1 and P_2 , the rectangular frame $P_0P_1P_2P_3$ would intersect with the staircase and would make the other of v_0 and v_1 impossible to fold.

Hence the staircase must be brought between the y -coordinates of P_1 and P_2 before folding either v_0 or v_1 . Since the last and the second last steps of the staircase are of size $2L$ and L , respectively, point P_5 must have the same coordinate as the point P_4 when the staircase is confined between the y -coordinates of P_1 and P_2 .

Now as we travel from P_4 to P_5 along the staircase, we travel equally in positive and negative y directions along the steps corresponding to the elements of X . Hence the sum of elements along whose steps we travel in negative y direction is same as the sum of elements along whose steps we travel in positive y direction. Thus there exist a solution to the PARTITION instance, if the crease pattern in Figure 8 is foldable. \square

From Lemmas 6.1 and 6.2, the following theorem follows.

Theorem 6.1 *The problem of deciding simple foldability of a orthogonal paper with an orthogonal*

crease pattern is (weakly) NP-complete.

In the Appendix, we prove the following theorem, which shows that even on a rectangular piece of paper it is hard to decide foldability if there is even *one* crease direction other than axis-parallel:

Theorem 6.2 *It is (weakly) NP-complete to decide the foldability of an (axis-parallel) square sheet of paper with a crease pattern having axis-parallel creases and creases at the diagonal angle of 45 degrees with respect to the axes.*

7 No Mountain-Valley Assignments

An interesting case to consider is when all creases do not have mountain-valley assignment: Any crease can be folded in either direction. Even with this flexibility, we are able to show that the problem is hard (see the Appendix for the proof):

Theorem 7.1 *The problem of deciding the foldability of a orthogonal paper with a crease pattern that does not have mountain-valley assignment is (weakly) NP-complete, both for the all-layers and some-layers cases.*

The problem is open for the one-layer case.

Acknowledgments. We thank Jack Edmonds for helpful discussions which inspired this research, in particular for posing some of the problems addressed here. E. Arkin and J. Mitchell thank Mike Todd for posing algorithmic versions of the map-folder's problem in a geometry seminar at Cornell.

E. Arkin acknowledges support from the National Science Foundation (CCR-9732221) and HRL Laboratories. M. Bender acknowledges support from HRL Laboratories. J. Mitchell acknowledges support from HRL Laboratories, the National Science Foundation (CCR-9732221), NASA Ames Research Center, Northrop-Grumman Corporation, Sandia National Labs, Seagull Technology, and Sun Microsystems.

References

- [1] E.M. Arkin, S.P. Fekete, J.S.B. Mitchell, and S.S. Skiena. On the manufacturability of paper-clips and sheet metal structures. Manuscript, 1999.
- [2] M. Bern and B. Hayes. The complexity of flat origami. In *Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 175–183, 1996.
- [3] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5(3), 1998.
- [4] E.D. Demaine, M.L. Demaine, and J.S.B. Mitchell. Folding flat silhouettes and wrapping polyhedral packages: New results in computational origami. *Computational Geometry: Theory & Applications*, 16(1), pp. 3–21, 2000.

- [5] M. Farach. Optimal suffix tree construction with large alphabets. In *Proc. 38th Annual Symposium on Foundations of Computer Science*, pp. 137–143, 1997.
- [6] M. Gardner. The combinatorics of paper folding. In *Wheels, Life and Other Mathematical Amusements*, Chapter 7, pages 60–73. W. H. Freeman, New York, NY, 1983.
- [7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.
- [8] D. Harel and R.E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, May 1984.
- [9] T. Hull. On the mathematics of flat origamis. *Congressum Numerantium*, 100:215–224, 1994.
- [10] J. Justin. Towards a mathematical theory of origami. In Koryo Miura, ed., *Proc. 2nd International Meeting of Origami Science and Scientific Origami*, pages 15–29, Otsu, Japan, November–December 1994.
- [11] R.M. Karp and M.O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, March 1987.
- [12] T. Kawasaki. On the relation between mountain-creases and valley-creases of a flat origami. In H. Huzita, editor, *Proc. 1st International Meeting of Origami Science and Technology*, pp. 229–237, Italy, 1989. Full paper (Japanese) in *Sasebo College of Technology Report*, 27:153–157, 1990.
- [13] R.J. Lang. A computational algorithm for origami design. In *Proc. 12th Annual ACM Symposium on Computational Geometry*, pp. 98–105, 1996.
- [14] W.F. Lunnon. Multi-dimensional map-folding. *The Computer Journal*, 14(1):75–80, February 1971.
- [15] R. Motwani and P. Raghavan. *Randomized Algorithms*, Chapter 8.4, pp. 213–221. Cambridge University Press, 1995.
- [16] B. Schieber and U. Vishkin. On finding lowest common ancestors: Simplification and parallelization. *SIAM Journal on Computing*, 17(6):1253–1262, 1988.
- [17] J.S. Smith. Origami profiles. *British Origami*, 58, June 1976.
- [18] J.S. Smith. *Pureland Origami 1, 2, and 3*. British Origami Society. Booklets 14, 29, and 43, 1980, 1988, and 1993.
- [19] M. Thorup. Faster deterministic sorting and priority queues in linear space. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 550–555, 1998.

Appendix

Proof (Theorem 6.2): As in section 6 we reduce from PARTITION.

We transform an instance of the PARTITION problem to a 2-D crease pattern on an axis-parallel square having all creases either orthogonal (axis-parallel) or at 45 degrees to the axes.

First, we establish a set of horizontal folds, evenly spaced and alternating mountain and valley,

which result in the paper becoming a long thin rectangle (“strip”); these initial folds will be called *I-folds*.

Now, a rectangular strip can be turned by 90 degrees by making a single fold as shown in Figure 9. By making several such turns we can get the strip into the shape of the initial paper as in Figure 8, except that the corners at each turn are “shaved” by 45-degree chamfers. (A strip can readily be folded in order to avoid these chamfers; however, it is easier to describe our construction with the simpler single folds at each bend.) Let’s call these folds d_1, d_2, d_3, \dots . Immediately after each d_i we make a fold parallel to the strip to reduce its width. See Figure 10. Let’s call these folds, r_1, r_2, r_3, \dots respectively. Thus we make these folds in the following order: $d_1, r_1, d_2, r_2, \dots$. We refer to the d_i folds as the *D-folds*, the r_i folds as the *R-folds* and both kinds together as *DR-folds*.

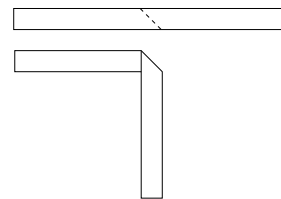


Figure 9: Turning a strip.

Finally, we create valley folds, as shown in Figure 8. We refer to these valleys as *F-folds*. After making the F-folds, we can unfold the paper to get the desired crease pattern.

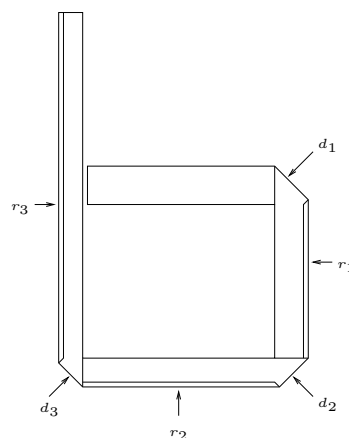


Figure 10: Illustration of first few DR-folds used in construction.

It is easy to see that if a solution to the partition problem exists, the above crease pattern can be

folded by first making the I-folds, followed by the DR-folds in the order in which they were created, followed by the F-folds (as done in Figure 8). We now prove the other direction: if the crease pattern can be flat folded, then the partition problem has a solution.

Each D-fold intersects all I-folds. And each R-fold intersects at least one D-fold. Hence none of the DR-folds can be made before all of the (initial) I-folds are made.

Because r_1 intersects d_1 and was created after folding d_1 , there is a precedence constraint that in any valid folding d_1 occurs before r_1 . Similarly r_1 occurs before d_2 and so on. Thus the DR-folds must occur in the order $d_1, r_1, d_2, r_2, \dots$

Also none of the F-folds can be made before the corresponding R-folds which it intersects are made. Thus it is guaranteed that after I-folds d_1, r_1, d_2, r_2, r_3 would be made in that order before any other folds. This puts our rectangular frame P_0, P_1, P_2, P_3 in place as in Figure 8.

Just as in proof of Lemma 6.2, to enable folds v_0 and v_1 to be made, the strip following d_3 must be folded so that it is confined between the y -coordinates of P_1 and P_2 . For this proof, this proof there is an additional constraint that no point on the strip following d_3 should have an x -coordinate differing from that of P_0 by more than W_2 . We can choose W_2 as small as we want, by reducing the width of the strip that I-folds create. In particular we can choose W_2 to be smaller than all a_i 's. Thus to meet the above constraints all the lengths of the strip which correspond to a_i 's will have to be vertical. And just as in proof of Lemma 6.2, there must exist a solution to partition problem, if all these vertical strips have to be between the y -coordinates of P_1 and P_2 . \square

Proof (Theorem 7.1): For the all-layers case, the proof of Theorem 6.1 works without mountain-valley assignments as well. This is so because the staircase must be confined as before to make both turns v_1 and v_2 in either direction. If the staircase is not confined before either of v_1 or v_2 is made in either direction, it will overlap with the frame, and, in the all-layers case, as soon as two layers of paper overlap they are “stuck” together.

For the some-layers case the proof of Theo-

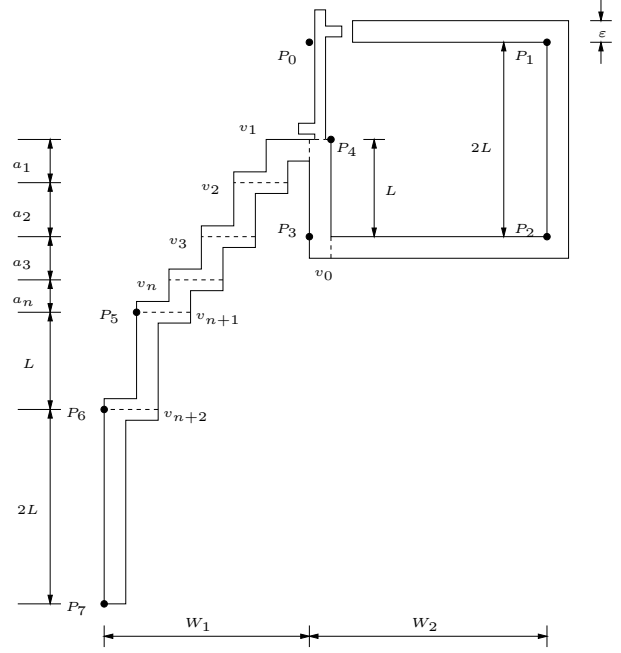


Figure 11: Hardness reduction when no mountain-valley assignment is given.

rem 6.1 does not work, as the folds v_0 and v_1 can be made in opposite directions, and so a folding exists whether or not a partition exists. We modify the construction to ensure that v_0 and v_1 must be folded in the same direction. See Figure 11, and the more detailed Figure 12. There are only two differences between this construction and the one in Figure 8. First is the extra piece of paper (*flap*) attached at the top of the staircase. Second is the addition of the fold of the flap, and three “crimps” shown in Figure 12. When creating the crease pattern, these new folds are made before the folds v_0 and v_1 . Each crimp is two folds very close to each other, and thus they change the shape of our construction only infinitesimally.

It is easy to argue that if there is a solution to the PARTITION problem, then our construction can be folded. This can be done by first folding the flap, followed by crimp c_0 , followed by crimps c_1, c_2 and then following the algorithm described in proof of Lemma 6.1.

We now prove the other direction; that is, if our construction is foldable, then there exists a solution to the PARTITION problem. We start by noting the following: Given a crease pattern in which two folds intersect at an angle other than 90 degrees, it is easy to tell which of the two folds must be folded first

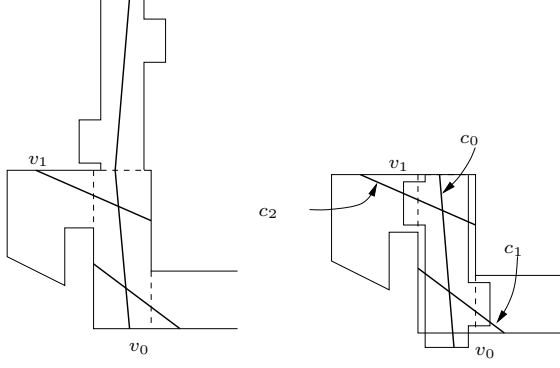


Figure 12: Interesting part of construction for hardness reduction.

in any legal folding. This is because the second fold must be a mirror image through the first fold. If the angle of intersection is not 90 degrees, then the second fold does not form a straight line in the crease pattern, but rather is two line segments reflected around the first fold. (If two creases meet at a 90-degree angle, and no mountain-valley assignment is given, then there are two possible orders of folding the two creases.) As a consequence, while constructing a crease pattern, if a crimp or a fold c_y is folded after another crimp or fold c_x and if c_y intersects c_x at any angle other than 90 degrees, then c_x must be folded before c_y can be folded in any legal folding of this crease pattern.

Figure 13 illustrates two crimps intersecting at 45 degrees and the crease pattern they create.

In our construction, from the above discussion, the flap must be folded before crimp c_0 can be folded, which in turn needs to be folded before crimps c_1 and c_2 can be folded. Further, crimps c_1 and c_2 need to be folded before folds v_0 and v_1 can respectively be folded. Thus, before either v_0 or v_1 can be folded, the flap must be folded. Once the flap is folded in either direction, v_0 and v_1 are forced to fold in the same direction. With this constraint, the rest of the proof is the same as that of Lemma 6.2. \square

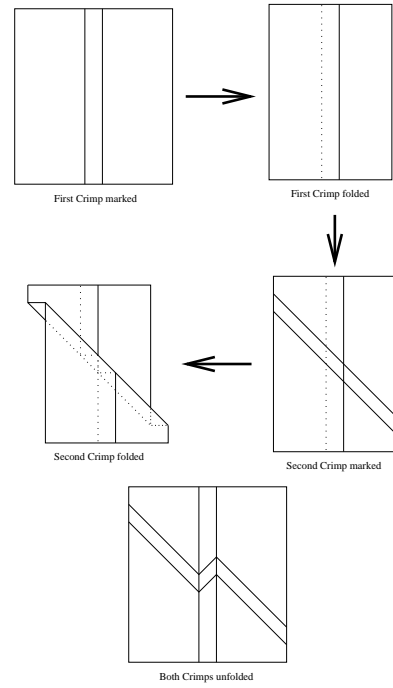


Figure 13: Crimps intersecting at angle other than 90 degrees cannot be folded out of order.