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 $\pm 180°$. 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 1D crease patterns, and reconstruction of a sequence of simple folds. Indeed, we prove that a 1D 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 [10])

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

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

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

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

‡MIT Laboratory for Computer Science, 200 Technology Square, Cambridge, MA 02139, USA, {edemaine, mdemaine}@mit.edu.

§Computer Science, Oregon State University, 102 Dearborn Hall, Corvallis, OR 97331-3202, USA, saurabh@cs.orst.edu.

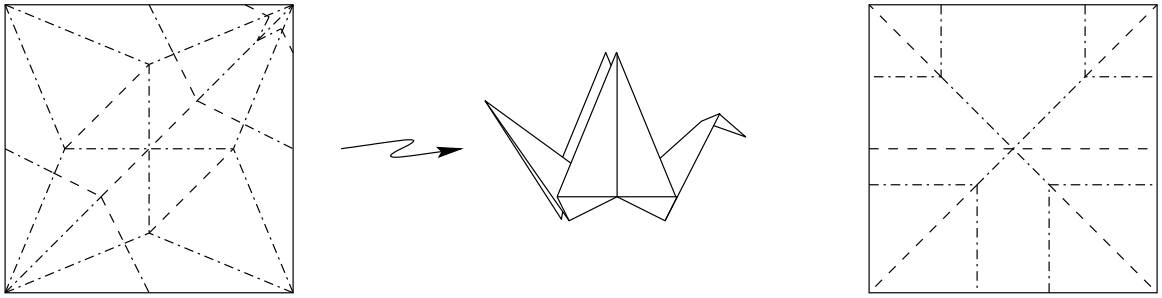


Figure 1: Sample crease patterns. Left: the classic crane. Right: pattern of Hull [13], which cannot be folded flat, for any mountain-valley assignment.

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 [2]. 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. Figure 2 shows an example of a simple folding. 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.

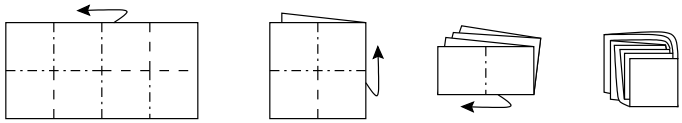


Figure 2: Folding a 2×4 map via a sequence of 3 simple folds.

Unsurprisingly, 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. In particular, there is no uniformly mountain or valley segment that could serve as the first simple fold. Also, the hardness gadgets of [2] require nonsimple folds which allow the paper to curve during folding [6]. Thus, the complexity of general flat foldability has no direct connection to simple foldability.

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 is 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 "1D" case to distinguish it from the general 2D problem. In contrast to the 2D problem, we show that 1D flat foldability is equivalent to 1D 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 whether a given structure can be manufactured using a given machine. (See references cited below.) While origamists can develop particular skill in performing nonsimple 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 algorithmic understanding of the several manufacturing problems that arise, e.g., in making three-dimensional cardboard and sheet-metal structures.

Related Work. Our problems are related to the classic combinatorics questions of *map folding* [10, 21]. These questions ask for the *number* of different flat foldings of a particular crease pattern, namely an $m \times n$ rectangular grid, either by a sequence of simple folds or by a general flat folding. Two foldings are usually considered "different" in this context if they differ in the total order of the faces in the folding. These questions have been studied extensively [10, 21], particularly in the one-dimensional $(1 \times n)$ case [7, 17, 20, 27], but remain largely unsolved. In contrast with these combinatorial questions, we study the algorithmic complexity of the decision problems, and for more general crease patterns.

The mathematical and algorithmic problems arising in the study of flat origami have been examined by several researchers, e.g., Hull [13], Justin [14], Kawasaki [16], and Lang [18]. Of particular relevance to our work is the paper by Bern and Hayes [2], which shows that the general problem of deciding flat foldability of a crease pattern is strongly NP-hard. Demaine et al. [5] 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 [24, 25]. 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.

There has been quite a bit of work on the related problems of manufacturability of sheet metal parts (see e.g. [28]) and folding cartons (see e.g. [19]). Existing CAD/CAM techniques (including BendCad and PART-S) rely on worst-case exponential-time state space searches (using the A* algorithm). In general, the problem of bend sequence generation is a challenging (and provably hard [1]) coordinated motion planning problem. For example, Lu and Akella [19] utilize a novel configuration-space formulation of the folding sequence problem for folding cartons using fixtures; their search, however, is still worst-case exponential time. Our work differs from the prior work on sheet metal and cardboard bending in that the structures we are folding are ultimately "flat" in their folded states (all bend angles in the input crease pattern are $\pm 180°$, according to a mountain-valley assignment that is part of the input crease pattern). Also, we are concerned only with the feasibility of the motion of the (stiff) material that is being folded—does it collide with itself during the folding motion? We are not addressing here the issues of reachability by the tools that perform the folding. As we show, even with the restrictions that come with the problems we study, there is a rich mathematical and algorithmic theory of foldability.

Summary of Our Results. We develop a variety of new algorithmic results (see Table 1):

(1) We analyze the 1D 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 1D 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.

			Model of folding						
Dim.	**Paper**	**Creases**	All-layers simple folds		Some-layers simple folds		One-layer simple folds		General flat folding
1D			$O(n)$ rand. $O(n \lg n')$ det.	$\not\equiv$	$O(n)$	$\equiv$	$O(n)$	$\equiv$	$O(n)$
2D	Rect	Ortho	$O(n)$ rand. $O(n \lg n')$ det.	$\not\equiv$	$O(n)$	$\not\equiv$	$O(n)$	$\not\equiv$	Open [8]
2D or	Ortho Square	Ortho Ortho + 45°	Weakly NP-complete		Weakly NP-complete		Weakly NP-complete		Open
2D	Square	General							Strongly NP-hard [2]

Table 1: Summary of the complexities of deciding flat foldability by various models of simple folds, and by general flat foldings. The symbols $\equiv$ and $\not\equiv$ denote equivalences and nonequivalences between certain models. The abbreviations "rand." and "det." denote randomized and deterministic algorithms, respectively.

(3) We give an algorithm for deciding simple foldability of orthogonal crease patterns on a rectangular piece of paper[1] (the "map folding problem"), in the one-, some-, and all-layers cases, based on and with the same running times as our 1D results.

(4) We prove that it is (weakly) NP-complete to decide simple 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 simple foldability of a square piece of paper with a crease pattern that includes *diagonal* creases (angled at 45°), in addition to axis-parallel creases.

(6) We show that it is (weakly) NP-complete to decide simple foldability of an orthogonal piece of paper having a crease pattern for which no mountain-valley assignment is given.

Note that our hardness results do not strengthen those of [2], because deciding simple foldability is different from deciding flat foldability.

2 Definitions

We are concerned with foldings in one and two dimensions, although several of our definitions and results extend to higher dimensions. A one-dimensional piece of paper is a (line) *segment* in $\mathbb{R}^1$. A two-dimensional piece of paper is a (connected) *polygon* in $\mathbb{R}^2$, possibly with holes. In both cases, the paper is folded through one dimension higher than the object; thus, segments are folded through $\mathbb{R}^2$ and polygons are folded through $\mathbb{R}^3$. *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 cross itself, although multiple layers of paper may touch. See Figure 3.

[1] Throughout this paper, the notions of "orthogonal" and "rectangular" implicitly require axis-parallelism with a common set of axes.

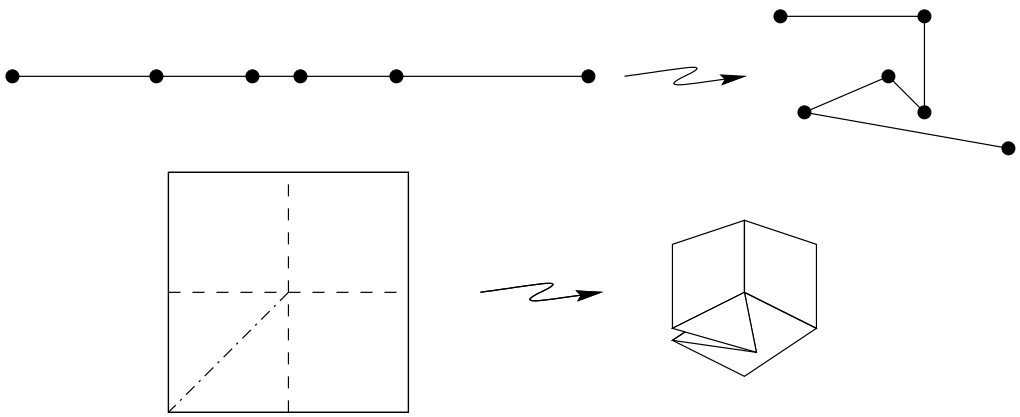


Figure 3: Sample nonflat foldings in one and two dimensions.

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 $\mathbb{R}^1$, and a flat folding of a polygon lies in $\mathbb{R}^2$. In reality, there can be multiple layers of paper at a point, so the folding really occupies a finite number of infinitesimally close copies of $\mathbb{R}^1$ or $\mathbb{R}^2$. See Figure 4. 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, 13, 18]. For each pair of facets of the crease pattern that fold to overlapping polygons, this partial order must specify which facet is layered above the other.

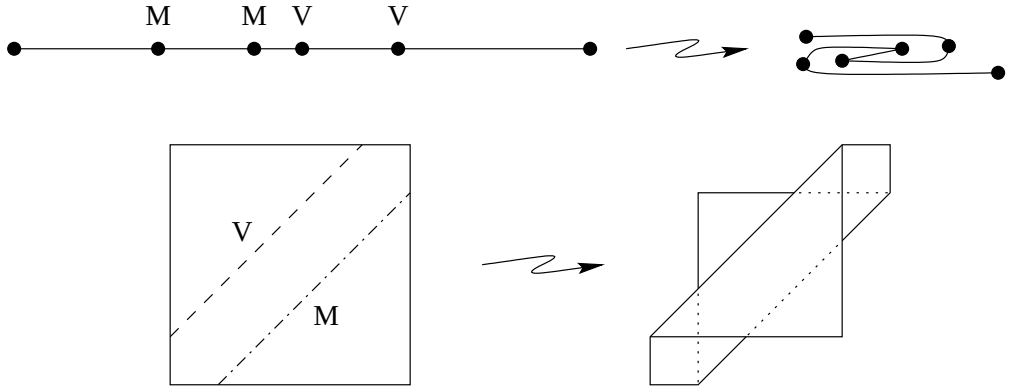


Figure 4: Sample flat foldings in one and two dimensions. Mountains and valleys are denoted by M's and V's, respectively.

If we orient the piece of paper to have a top and bottom side, we can talk about the *direction* of a 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 4. Together, a crease pattern and a mountain-valley assignment form a *mountain-valley pattern*.

This paper is concerned with the following general 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. We distinguish three types of simple folds: one-layer, all-layers, and some-layers. Refer to Figure 5.

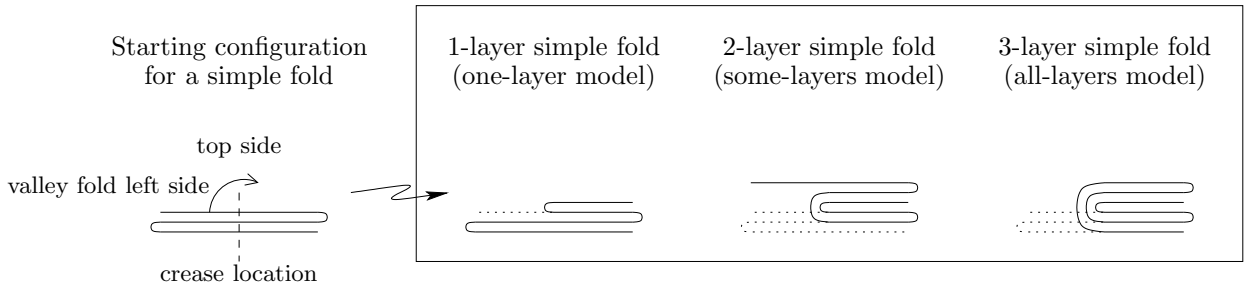


Figure 5: Illustration of a simple fold in 1D, which is specified by a crease location, a notion of the "top side", how many of the top layers are folded (here ranging from 1 to 3), whether the fold is mountain or valley, and whether the left or right side is folded.

We begin by defining in 1D the most general type of simple fold, the some-layers simple fold. A *some-layers simple fold* is specified by (1) an orientation of the folded piece of paper to specify a *top side*, (2) an externally visible crease (point) on the top side of the folded piece of paper, (3) the number ℓ of layers to be folded, and (4) the orientation of the fold, mountain or valley, relative to which side is the top. Such a fold newly creases the piece of paper at ℓ points: at the specified crease on the topmost layer and at the $\ell - 1$ creases (points) immediately below. If we locally color the piece of paper near the new creases, blue to the left of the creases and red to the right of the creases, and propagate this coloring, we should obtain a partition of the piece of paper into two (not necessarily connected) components. If we find a conflict that some paper should be colored simultaneously red and blue, the simple fold is not valid. Otherwise, the execution of the simple fold corresponds to continuously rotating the blue portion of paper by 180° around the crease point, either clockwise or counterclockwise according to whether the fold is valley or mountain. During this rotation, both the red and blue portions of the paper remain rigid. If the paper self-intersects during this rotation, the simple fold is invalid.

Some-layers simple folds are most general in the sense that any number ℓ of layers can be folded at once. A *one-layer simple fold* is the special case in which $\ell = 1$. An *all-layers simple fold* is the special case in which ℓ is the entire number of layers coinciding at the specified crease point.

In 2D, the situation is more complicated because the number of layers folded can vary along the crease segment. See Figure 6 for an example. Thus, a some-layers simple fold must specify the desired number of layers for each portion of the crease, for a specified subdivision of the crease segment into portions. We construct the new creases that result from this fold by copying each portion of the crease to the specified number of layers below the topmost layer. Then, as before, we color the piece of paper, verify that this coloring is consistent (in particular, verifying that the assignment of layers was valid), and rotate the blue portion of paper, barring self-intersection.

In this more general setting, a one-layer simple fold is the special case of folding only one layer along the entire crease. An all-layers simple fold is the special case of folding all layers for each portion of the crease. The simple fold in Figure 6 is an example that cannot be made a one-layer simple fold, and indeed, cannot be modified to use any smaller number of layers at any point. This fact can be verified by attempting to fold such a piece of paper in practice, or by checking that the resulting red-blue coloring is invalid.

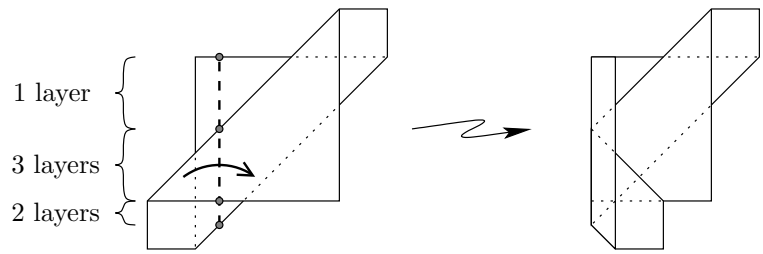
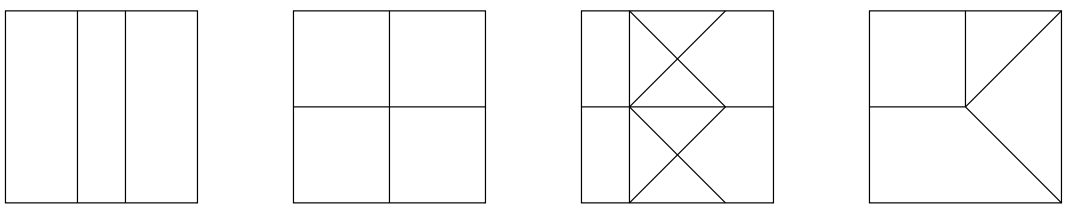


Figure 6: Illustration of a simple fold in 2D (starting from Figure 4, right), where we must specify the number of folded layers for each portion of the crease, because this number can vary along a single fold.

Figure 7 further illustrates the differences among the three models of simple folding, and their limitations with respect to general flat folding, by giving examples of crease patterns that can be folded with one model but not the others. Of course, flat foldability by one-layer simple folds or by all-layers simple folds implies flat foldability by some-layers simple folds, which in turn implies flat foldability in general. The examples in the figure prove that no other general implications hold.



flat-foldable by one-layer simple folds but not flat-foldable by all-layers simple folds

flat-foldable by all-layers simple folds but not flat-foldable by one-layer simple folds

flat-foldable by some-layers simple folds but not flat-foldable by one-layer and/or all-layers simple folds

flat-foldable by general origami folding but not flat-foldable by any simple folds

Figure 7: Crease patterns illustrating the unique power of each model of simple folding, and the limitations compared to general flat folding.

3 1D: One-Layer and Some-Layers

This section is concerned with the 1D 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 8.

More formally, let $c_1, \ldots, c_n$ 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.

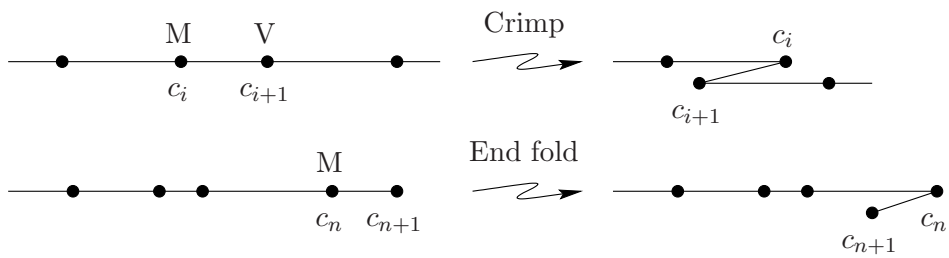


Figure 8: The two local operations for one-dimensional one-layer folds.

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 1D mountain-valley pattern. We claim further that an operation exists for any pattern satisfying a certain "mingling property". Specifically, a 1D mountain-valley pattern is called *mingling* if for every sequence $c_i, c_{i+1}, \ldots, c_j$ 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 1D mountain-valley pattern is mingling.*

Proof: Consider a flat folding of a mountain-valley pattern, and let $c_i, \ldots, c_j$ be consecutive creases with the same direction. The portion $c_{i-1}, \ldots, c_{j+1}$ of the segment can be in one of three configurations (see Figure 9):

1. The portion forms a "spiral" with (c_{i-1}, c_i) being the outermost edge of the spiral, and (c_j, c_{j+1}) being the innermost; or
2. The portion forms a "spiral" with (c_j, c_{j+1}) being the outermost edge of the spiral, and (c_{i-1}, c_j) being the innermost; or
3. The portion forms two "spirals" connected by a common outermost edge and with (c_{i-1}, c_i) and (c_j, c_{j+1}) being the two innermost edge.

Now if $|c_{i-1} - c_i| > |c_i - c_{i+1}|$, then (c_{i-1}, c_i) cannot be the innermost edge of a 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) cannot be the innermost edge of the spiral. Because in all three configurations above we must have at least one of (c_{i-1}, c_i) and (c_j, c_{j+1}) as innermost, we cannot have both inequalities true. □

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

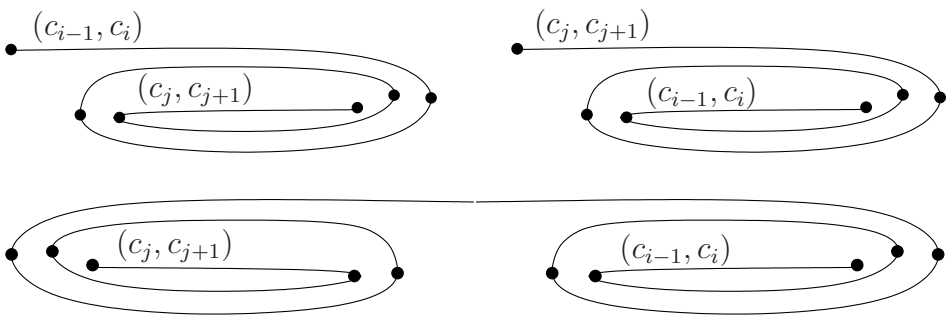


Figure 9: The innermost edge of a spiral cannot be longer than the adjacent edge, in contrast to the outermost edge which can be arbitrarily long.

Proof: Let i be maximum such that $c_1, \ldots, c_i$ 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 $c_i, \ldots, c_j$ 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 $c_{j+1}, \ldots, c_k$ 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. □

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 10. 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.

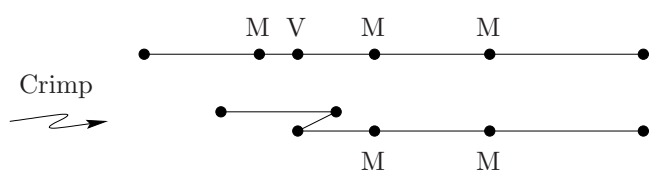


Figure 10: A mingling mountain-valley pattern that when crimped is no longer mingling and hence not flat-foldable. Indeed, the original mountain-valley pattern is not flat-foldable.

Lemma 3.3 *Folding a foldable end preserves flat 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. □

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, such as the one in Figure 11 (left).

We orient our view to 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 11. 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}). □

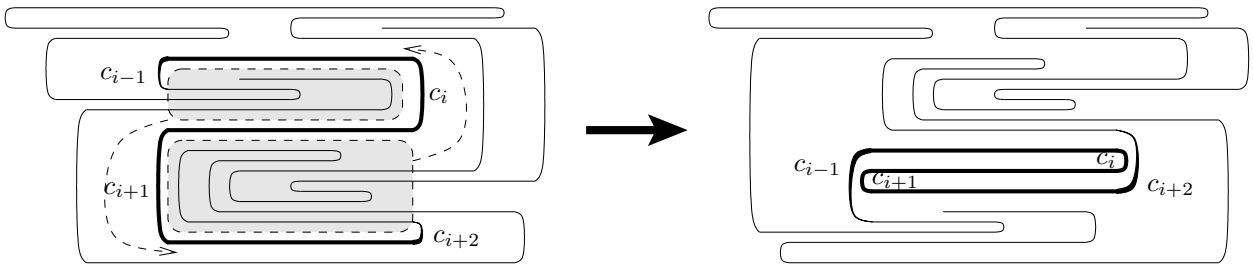


Figure 11: Moving layers of paper out of the zig-zag formed by a crimp (c_i, c_{i+1}), highlighted in bold.

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

Theorem 3.5 *Any flat-foldable 1D 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. □

A particularly interesting consequence of this theorem is the following connection to general flat foldability:

Corollary 3.6 *The following are equivalent for a 1D 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.*

One-dimensional flat foldability has been studied extensively in the combinatorial context [7, 17, 20, 27], but primarily for the simple crease pattern in which the distances between consecutive creases are identical. A structure similar to ours, in particular highlighting the importance of crimps, is hinted at by Justin [14, Section 6.1], though it is not followed through algorithmically.

Finally, we show that Theorem 3.5 leads to a simple linear-time algorithm:

Theorem 3.7 *The 1D 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 3.5, if the list ever becomes empty, it is because the mountain-valley pattern is not flat-foldable. □

4 1D: All-Layers Simple Folds

The 1D 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 [4].) 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 $\ell_0\ d_1\ \ell_1\ d_2 \cdots d_{n-1}\ \ell_{n-1}\ d_n\ \ell_n$, where each $d_i \in \{M, V\}$ specifies the direction of the ith crease c_i, and each ℓ_i is a positive rational number specifying the distance between adjacent creases c_i and c_{i+1}. We call each d_i and ℓ_i a *symbol* of the string. It will be helpful to introduce some more uniform notation for symbols. For a string S of length $N = 2n + 1$, we denote the ith symbol by $S[i]$, where $1 \le i \le N$.

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 impossible to fold later. To formalize this condition, we define the *complement* of symbols in the string: $\mathrm{comp}(\ell_i) = \ell_i$, $\mathrm{comp}(M) = V$, and $\mathrm{comp}(V) = M$. For each even index i, at which $S[i] = d_{i/2} \in \{M, V\}$, we define the *fold at position* i to be the all-layers simple fold of the corresponding crease $c_{i/2}$. We call this fold *allowable* if $S[i - x] = \mathrm{comp}(S[i + x])$ for all $1 \le x \le \min(i - 1, N - i)$, except that $S[1]$ and $S[N]$ (the end lengths) 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. □

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 the fold at position 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 4.2 *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 [9]. 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 deterministic algorithm for integer sorting [26] 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. The fold at position i of S is allowable 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 less than or equal 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 [12, 23]. 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 jth 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. □

4.2 Randomized Algorithm

In this section we describe a simple randomized algorithm that solves the 1D 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 [22] 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 [15]. We treat a substring as the base-n' representation of an integer, where we use $0, \ldots, n' - 1$ 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).

Because an exact match is not required on the last length for a fold to be allowable, both fingerprints on either side exclude 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. Thus, given the appropriate fingerprints, we can check whether a fold is allowable in $O(1)$ time.

By choosing the prime p randomly from the range $2, \ldots, n^3$, the probability that this algorithm makes a mistake after at most n folds is $O((\log n)/n)$; see [15]. More generally, if we choose p from the range $2, \ldots, n^c$, 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.

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

Theorem 4.3 *The 1D all-layers simple-fold problem can be solved in $O(n)$ time, both in expectation and with high probability, on a machine supporting random numbers and hashing of the input lengths.*

5 Orthogonal Simple Folds in 2D

In this section, we generalize our results for 1D simple folds to *orthogonal* 2D 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. 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, 13]. Hence, the crease pattern is a grid of creases (a *map*), although the space between grid lines may vary. Edmonds [8] observed that orthogonal 2D mountain-valley patterns may be flat-foldable but not by simple folds; see Figure 12 for two examples. Recall from Section 3 that the opposite holds in 1D: one-layer and some-layers folds are equivalent to general flat-foldability. In this section we simultaneously handle some-layers and all-layers simple folds; with one-layer simple folds, only 1D maps are foldable.

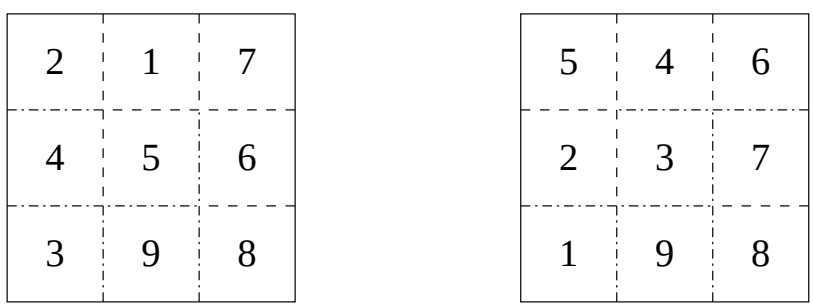


Figure 12: Two maps that cannot be folded by simple folds, but can be folded flat. (These are challenging puzzles.) The numbering indicates the overlap order of faces.

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 $n = n_1 n_2$.

In a simply foldable mountain-valley pattern, there must be at least one crease line, all the way across the paper, that is entirely valley or mountain; otherwise, the pattern would not permit any simple folds. Furthermore, all such crease lines must be parallel; otherwise, the vertex of intersection between two crossing crease lines would not be locally flat-foldable. Without loss of generality, assume that these crease lines are horizontal, and let $\mathcal{H}$ denote the set of them.

We claim that all crease lines in $\mathcal{H}$ must be folded before any other crease. This is so because (1) folding along any vertical crease line v will lead to a mismatch of creases at the intersection

of v with any unfolded elements of $\mathcal{H}$ and (2) horizontal crease lines not in $\mathcal{H}$ are not entirely mountain or valley and hence cannot be folded before some vertical fold is made. Thus we have a corresponding 1D problem (with some- or all-layers folds) to solve with the added necessary condition that the non-$\mathcal{H}$ folds must match up appropriately after all the folds in $\mathcal{H}$ are made. (The time spent checking this necessary condition can be attributed to the non-$\mathcal{H}$ folds that vanish after every fold.) Because $\mathcal{H}$ contains at least one fold, performing the $\mathcal{H}$ folds (strictly) reduces the size of the problem, and we continue. The base case consists of just horizontal or vertical folds, which corresponds to a 1D problem. In summary we have

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

To find $\mathcal{H}$ 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 $\mathcal{H}$. To do this we traverse all the creases that will vanish after a fold and decrement the corresponding numbers. The cost of this traversal is attributed to the vanishing creases. 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 $\mathcal{H}$ in the next step. Thus, we obtain

Theorem 5.2 *Some-layers simple folding of an orthogonal crease pattern on a rectangular piece of paper can be solved in deterministic linear time. All-layers simple folding in the same situation can be solved in randomized linear time, or deterministic linear time plus the time required to sort the edge lengths.*

This theorem easily generalizes to higher dimensions, with a running time linear in $n = n_1 n_2 \cdots n_d$ plus possibly the time required to sort the edge lengths.

6 Hardness of Simple Folds in 2D

In this section we prove that the problem of deciding whether a 2D 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, which is (weakly) NP-hard [11]: given a set X of n integers $a_1, a_2, \ldots, a_n$ 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 $a_1 \in S$.

We transform an instance of the PARTITION problem into an orthogonal 2D crease pattern on a orthogonal polygon, as shown in Figure 13. All creases are valleys. There is a staircase of width ε, where $0 < \varepsilon < 2/3$, with one step of length a_i corresponding to each element a_i in X. In addition, there are two final steps of length L and $2L$, where L is chosen greater than $A/2$. The total width W_1 of the staircase is chosen to be less then the width W_2 of the frame attached to the staircase.

The main mechanism in the reduction is formed by the vertical creases v_0 and v_1. Basically, the first time we fold one of these two creases, the staircase must fit within the frame, or else the second of these two creases is blocked. Then when we fold the other of these two creases, the staircase exits the frame, enabling us to fold the remaining creases in the staircase.

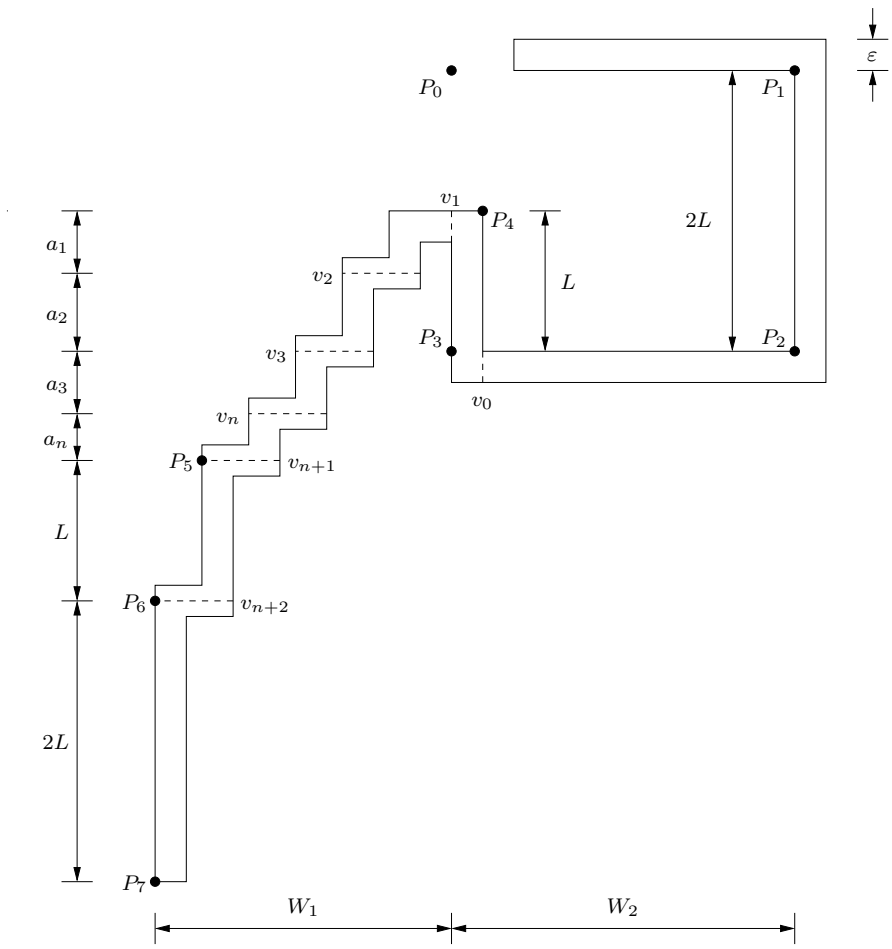


Figure 13: Hardness reduction from PARTITION problem.

Lemma 6.1 *If the* PARTITION *instance has a solution, then the crease pattern in Figure 13 is simply foldable.*

Proof: For $2 \leq i \leq n$, valley fold 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 $a_1, \ldots, a_n$, we travel in the $-y$ direction for elements that belong to S and in the $+y$ direction for elements that belong to $\bar{S}$. Because 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. Because $L > A/2$, the steps corresponding to a_i's are confined to remain in between the y coordinates of points P_1 and P_2. Because P_5 has the same y-coordinate as P_4 and because 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 valley fold v_{n+2}. Because the 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 situation is illustrated in Figure 14.

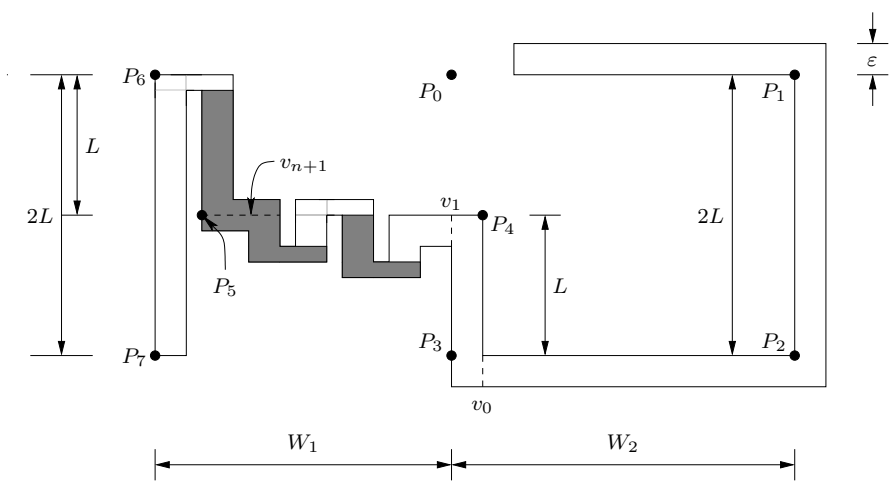


Figure 14: Semi-folded staircase confined between y coordinates of P_1 and P_2. The top side of the paper is drawn white and the other side is drawn gray.

Now valley fold v_1. Because $W_2 > W_1$, 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 valley fold 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, valley fold all of the remaining unfolded creases in the staircase. This can be done because the rectangular frame is now on the left of P_4 and all steps of the staircase are on the right of P_4. □

Lemma 6.2 *If the crease pattern in Figure 13 is simply foldable, there is 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. Because 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.

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 the $+y$ direction. Thus there is a solution to the PARTITION instance, if the crease pattern in Figure 13 is foldable. □

Lemmas 6.1 and 6.2 imply the following theorem.

Theorem 6.3 *The problem of deciding simple foldability of an orthogonal piece of paper with an orthogonal mountain-valley pattern is (weakly) NP-complete, for all-layers, some-layers, and one-layer simple folds.*

Even on a rectangular piece of paper it is hard to decide foldability if, besides axis-parallel, there are creases in diagonal directions (45 degrees with respect to the axes):

Theorem 6.4 *It is (weakly) NP-complete to decide the simple foldability of an (axis-parallel) square sheet of paper with a mountain-valley pattern having axis-parallel creases and creases at the diagonal angles of 45 degrees with respect to the axes, for both all-layers and some-layers simple folds.*

Proof: We transform an instance of the PARTITION problem to a 2D 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 15. By making several such turns we can get the strip into the shape of the initial paper as in Figure 13, 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.) Call these folds $d_1, d_2, d_3, \ldots$. Immediately after each d_i we make a fold parallel to the strip to reduce its width. See Figure 16. Call these folds $r_1, r_2, r_3, \ldots$ respectively. Thus we make these folds in the following order: $d_1, r_1, d_2, r_2, \ldots$. 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*.

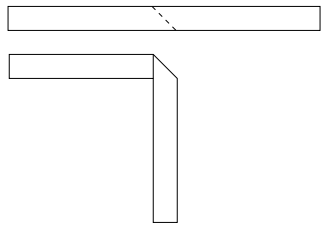


Figure 15: Turning a strip.

Finally, we create valley folds, as shown in Figure 13. We refer to these valleys as *F-folds*. After making the F-folds, we can unfold the paper to get the desired crease pattern.

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 13). We now prove the other direction: if the crease pattern can be flat folded, then the PARTITION problem has a solution.

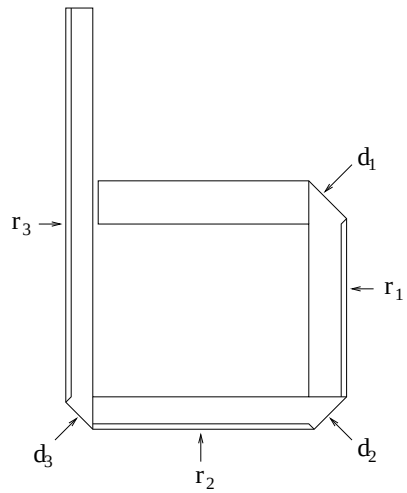


Figure 16: Illustration of first few DR-folds used in construction.

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 $d_1, r_1, d_2, r_2, \ldots$.

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 13.

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. □

The problem is open for the one-layer case.

7 No Mountain-Valley Assignments

An interesting case to consider is when the 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:

Theorem 7.1 *The problem of deciding the simple foldability of an orthogonal piece of paper with a crease pattern (without a mountain-valley assignment) is (weakly) NP-complete, for both all-layers and some-layers simple folds.*

Proof: For the all-layers case, the proof of Theorem 6.3 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.

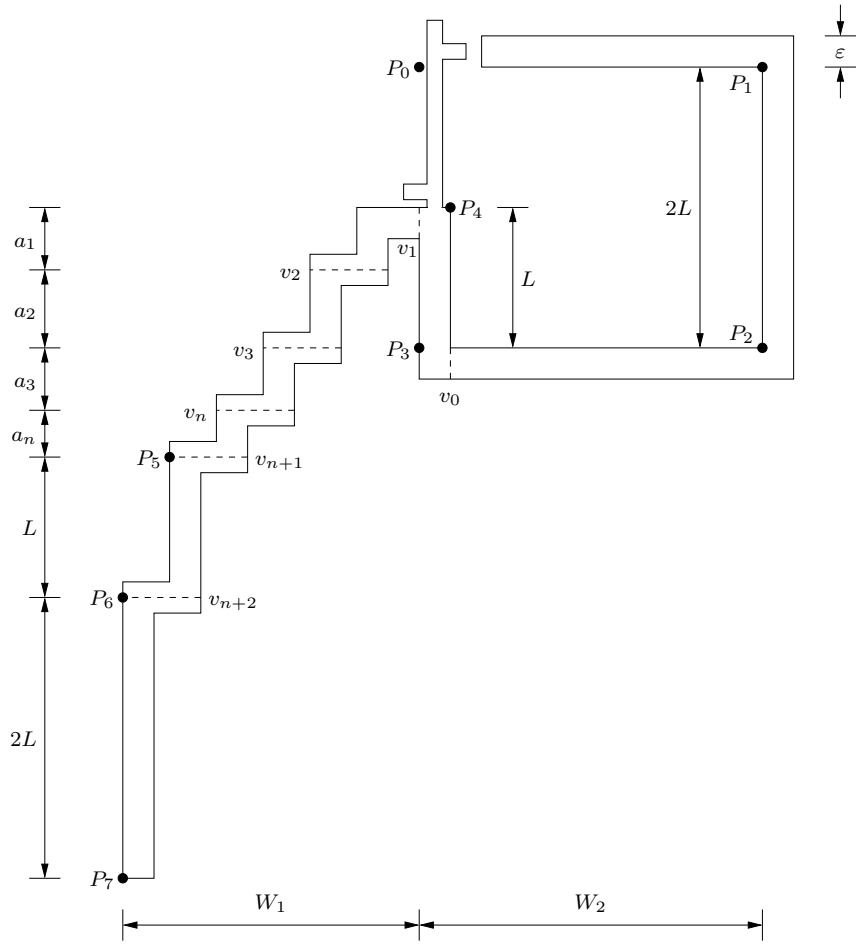


Figure 17: Hardness reduction when no mountain-valley assignment is given.

For the some-layers case the proof of Theorem 6.3 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 17, and the more detailed Figure 18. There are only two differences between this construction and the one in Figure 13. 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 18. When creating the crease pattern, these new folds are made before the folds v_0 and v_1. Each crimp consists of two folds very close to each other, changing 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 is 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 in any legal folding. This is because the second fold must be a mirror image through

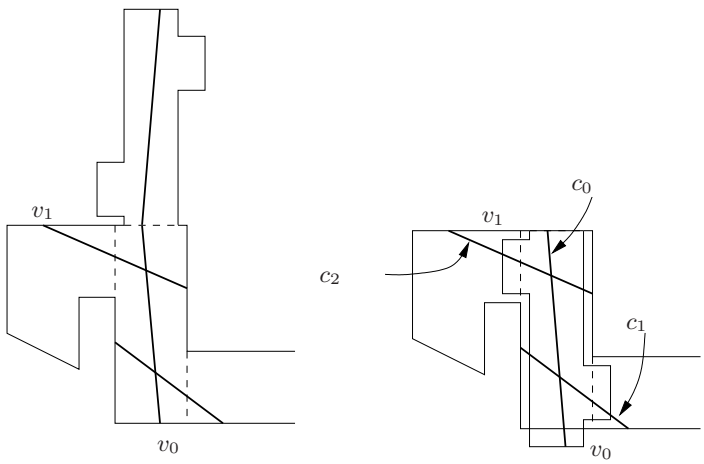


Figure 18: Interesting part of construction for hardness reduction.

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

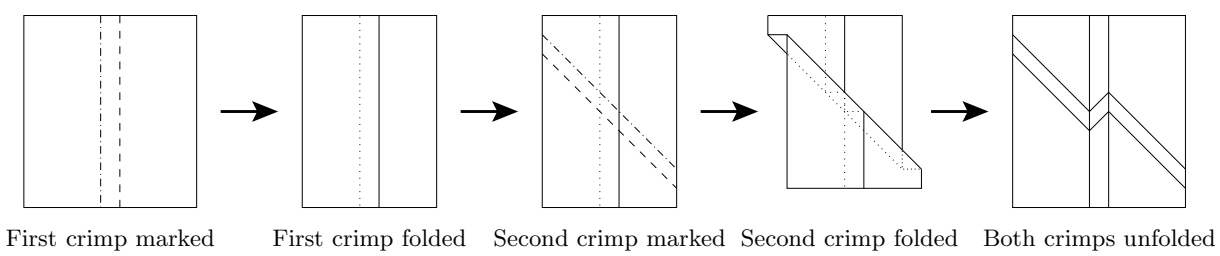


Figure 19: Crimps intersecting at an angle other than 90 degrees cannot be folded out of order.

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. □

The problem is open for the one-layer case.

8 Conclusion

We have presented efficient algorithms for deciding flat foldability of a map (rectangle with horizontal and vertical creases) via a sequence of simple folds, for any of three different restrictions

on the number of layers that can be folded at once. In the all-layers model, there may be several solution sequences of simple folds, and they can vary significantly in length; for example, in a 1D pattern that alternates mountain and valley, there is a sequence with roughly $\log n$ folds and a sequence with roughly n folds. (In contrast, the shape of the final folded state is independent of the folding process, depending only in the crease pattern.) Jeff Erickson[2] observed that there is also a polynomial-time algorithm for minimizing the length of the simple-fold sequence: the one-dimensional subproblems are forced, and in each one-dimensional subproblem, we can use dynamic programming on the $O(n^2)$ substrings of the mountain-valley pattern. This optimization problem is of course trivial for one-layer simple folds (the number of folds equals the number of creases), but it remains open for some-layers simple folds, where there is an interesting interplay between the efficiency of all-layers folds and the power of one-layer folds.

On the complexity side, we have shown that slight generalizations of the basic map-folding problem are (weakly) NP-complete. However, there still remains a gap. For example, what is the complexity of deciding simple foldability of an orthogonal crease pattern on an orthogonally convex piece of paper? Even more simply, what is the complexity of deciding simple foldability of an orthogonal crease pattern on a convex piece of paper, or even a non-axis-aligned rectangle? These variations possess the needed difficulty that making one fold may produce a piece of paper that is no longer a subset of the original piece of paper; thus, it is not clear that making a fold always makes progress. Another direction to consider is nonrectangular maps, for example, a triangular map whose faces are unit equilateral triangles (polyiamonds). We conjecture that deciding simple foldability is again (weakly) NP-complete in this context. Also, for the problems that we show to be weakly NP-hard, it remains open whether there are pseudopolynomial-time algorithms for solving simple foldability, or whether the problems are strongly NP-complete.

Our study of special cases of crease patterns may also be interesting in the context of general flat foldings. Here the goal would be to strengthen Bern and Hayes's NP-hardness result [2] to special crease patterns, or perhaps more interesting, to find special cases in which flat foldability is polynomially testable. One special case of particular interest, posed by Edmonds [8], is an $m \times n$ grid with a prescribed mountain-valley assignment. Along these lines, Justin [14] observed that even $2 \times n$ maps in which every 2×2 submap is flat-foldable may not be totally flat-foldable. Di Francesco [7] suggests that a useful algebraic structure in $1 \times n$ map folding may generalize. However, we do not even know whether testing general flat foldability is NP-hard for the cases in which testing simple foldability is NP-hard: orthogonal polygons with orthogonal creases, and rectangles with orthogonal and 45° creases. Our hardness reductions rely on the restriction to simple folds.

Acknowledgments

We thank Jack Edmonds for helpful discussions which inspired this research, in particular for posing some of the problems addressed here. We also thank Joseph O'Rourke and an anonymous referee for extensive comments which greatly improved this paper. 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

[2]Personal communication, March 2001.

Sun Microsystems.

Addendum

We recently learned that Calinescu, Karloff, and Thorup [3] independently discovered linear-time algorithms for some-layers simple foldability in the 1D and 2D orthogonal cases.

References

[1] Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. An algorithmic study of manufacturing paperclips and other folded structures. *Computational Geometry: Theory and Applications*, 25:117–138, 2003.

[2] Marshall Bern and Barry Hayes. The complexity of flat origami. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 175–183, Atlanta, January 1996.

[3] Gruia Calinescu, Howard Karloff, and Mikkel Thorup. Manuscript, 2000.

[4] Pierluigi Crescenzi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5(3), 1998.

[5] Erik D. Demaine, Martin L. Demaine, and Joseph S. B. Mitchell. Folding flat silhouettes and wrapping polyhedral packages: New results in computational origami. *Computational Geometry: Theory and Applications*, 16(1):3–21, 2000.

[6] Erik D. Demaine and Joseph S. B. Mitchell. Reaching folded states of a rectangular piece of paper. In *Proceedings of the 13th Canadian Conference on Computational Geometry*, pages 73–75, Waterloo, Canada, August 2001. `http://compgeo.math.uwaterloo.ca/~cccg01/proceedings/short/eddemaine-33029.ps`.

[7] P. Di Francesco. Folding and coloring problems in mathematics and physics. *Bulletin of the American Mathematical Societ*, 37(3):251–307, July 2000.

[8] Jack Edmonds. Personal communication, August 1997.

[9] Martin Farach. Optimal suffix tree construction with large alphabets. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 137–143, Miami Beach, Florida, October 1997.

[10] Martin Gardner. The combinatorics of paper folding. In *Wheels, Life and Other Mathematical Amusements*, chapter 7, pages 60–73. W. H. Freeman and Company, 1983.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[12] Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, May 1984.

[13] Thomas Hull. On the mathematics of flat origamis. *Congressus Numerantium*, 100:215–224, 1994.

[14] Jacques Justin. Towards a mathematical theory of origami. In Koryo Miura, editor, *Proceedings of the 2nd International Meeting of Origami Science and Scientific Origami*, pages 15–29, Otsu, Japan, November–December 1994.

[15] Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, March 1987.

[16] Toshikazu Kawasaki. On the relation between mountain-creases and valley-creases of a flat origami. In H. Huzita, editor, *Proceedings of the 1st International Meeting of Origami Science and Technology*, pages 229–237, Ferrara, Italy, December 1989. An unabridged Japanese version appeared in *Sasebo College of Technology Report*, 27:153–157, 1990.

[17] John E. Koehler. Folding a strip of stamps. *Journal of Combinatorial Theory*, 5:135–152, 1968. Minor errata appear in Series A, 19(3):367, 1975.

[18] Robert J. Lang. A computational algorithm for origami design. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 98–105, Philadelphia, PA, May 1996.

[19] L. Lu and S. Akella. Folding cartons with fixtures: A motion planning approach. *IEEE Transactions on Robotics and Automation*, 16(4):346–356, August 2000.

[20] W. F. Lunnon. A map-folding problem. *Mathematics of Computation*, 22(101):193–199, 1968.

[21] W. F. Lunnon. Multi-dimensional map-folding. *The Computer Journal*, 14(1):75–80, February 1971.

[22] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*, chapter 8.4, pages 213–221. Cambridge University Press, 1995.

[23] Baruch Schieber and Uzi Vishkin. On finding lowest common ancestors: Simplification and parallelization. *SIAM Journal on Computing*, 17(6):1253–1262, December 1988.

[24] John S. Smith. Origami profiles. *British Origami*, 58, June 1976.

[25] John S. Smith. *Pureland Origami 1, 2, and 3*. British Origami Society. Booklets 14, 29, and 43, 1980, 1988, and 1993.

[26] Mikkel Thorup. Faster deterministic sorting and priority queues in linear space. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 550–555, San Francisco, California, January 1998.

[27] Jacques Touchard. Contribution à l'étude du problème des timbres poste (Contribution to the study of the problem of postage stamps). *Canadian Journal of Mathematics*, 2:385–398, 1950.

[28] Cheng-Hua Wang. *Manufacturability-driven decomposition of sheet metal*. PhD thesis, Carnegie Mellon University, Robotics Institute, September 1997. Technical report CMU-RI-TR-97-35.