Minimum Forcing Sets for 1D Origami[☆]

Mirela Damian[a], Erik Demaine[b], Muriel Dulieu[c], Robin Flatland[d], Hella Hoffmann[e], Thomas C. Hull[f,*], Jayson Lynch[b], Suneeta Ramaswami[g]

[a]*Dept. of Computer Science, Villanova University, Villanova, PA 19085, USA*
[b]*CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[c]*Dept. of Computer Science and Engineering, Polytechnic Institute of NYU, Brooklyn, NY 11201, USA*
[d]*Dept. of Computer Science, Siena College, Loudonville, NY 12211, USA*
[e]*David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON Canada N2L 3G1*
[f]*Dept. of Math, Western New England University, Springfield, MA 01119, USA*
[g]*Dept. of Computer Science, Rutgers University, Camden, NJ 08102, USA*

Abstract

This paper addresses the problem of finding minimum forcing sets in origami. The origami material folds flat along straight lines called <u>creases</u> that can be labeled as mountains or valleys. A <u>forcing set</u> is a subset of creases that force all the other creases to fold according to their labels. The result is a flat folding of the origami material. In this paper we develop a linear time algorithm that finds <u>minimum</u> forcing sets in one dimensional origami.

Keywords: origami, flat folding, forcing sets

1. Introduction

In computational origami, the topic of self-folding origami, where a thin material folds itself in response to some stimulus or mechanism, has been gaining in popularity (see [1, 2, 3, 4]). The origami material can be programmed to fold along straight lines, called <u>creases</u>, by means of rotation.

As such, the self-folding process can be economized by only programming a subset of the creases to self-fold, which would then force the other, passive creases to fold as originally intended. We call this subset of creases a forcing set. Finding a forcing set of smallest size in a given crease pattern would be a useful tool for self-folding.

This paper addresses the one-dimensional (1D) variant of the forcing set problem, in which the piece of paper reduces to a horizontal line segment in $\mathbb{R}$, and creases reduce to points on the line segment. Our main result is an algorithm that finds a minimum forcing set of any given 1D folding pattern, in time linear in the number of input creases. This is the first work to consider finding forcing sets for arbitrary crease patterns. The forcing set problem was only recently introduced in the literature by Ballinger et al. [5] in a paper that considers the problem for one particular class of two-dimensional (2D) foldable patterns called Miura-ori. The authors present an algorithm for finding a minimum forcing set of a Miura-ori map in time quadratic in the total number of creases. It is worth noting that the results from [5] are specific to the Miura-ori map crease pattern and do not transfer to arbitrary crease patterns (of any dimension), which is what we investigate in this paper for the 1D case.

Although the forcing set problem was introduced recently and so there is little prior work on it, there is much related work on flat-foldable origami, which addresses the problem of determining if a given crease pattern can be folded to a flat configuration. Arkin et al. [6] give a linear time algorithm for determining if a 1D crease pattern is flat-foldable. In [7], Bern and Hayes show how to determine in linear time whether a general crease pattern has a mountain-valley assignment for which every node is locally flat-foldable, and they prove that deciding whether a crease pattern is globally flat-foldable is NP-hard. For crease patterns consisting of a regular $m \times n$ grid of squares, the complexity of deciding whether a given mountain-valley assignment can be folded flat (the map folding problem) remains open [8], although recent progress has been made on $2 \times n$ grids [9] and on testing for valid linear orderings of the faces in a flat-folding [10]. In [11], Hull gives upper and lower bounds on the number of flat-foldable mountain-valley assignments on a single-node crease pattern.

In 1D origami, the horizontal line segment representing the piece of paper is oriented, so we can talk about the top of the segment (normal pointing upward) and the bottom of the segment (normal pointing downward).

A crease is a point on the segment. A fold bends the segment $\pm 180°$ about a crease. Folds can be of two types, mountain (M) and valley (V). A mountain (valley) fold about a crease brings together the bottom (top)

sides of the left and right segments adjacent to the crease. We imagine the paper to be sticky on both sides, so that after a fold the paper will stick and cannot be undone.

A crease pattern on a segment with endpoints $c_0, c_n \in \mathbb{R}$ is a strictly increasing sequence of points $C = (c_0, c_1, ..., c_n)$. We refer to $c_1, c_2, \ldots, c_{n-1}$ as creases, and to c_0, c_n as ends. (Note that c_0 and c_n are not creases; we use similar notation for creases and ends for consistency.) For a pair of consecutive creases c_i and c_{i+1}, we will refer to $[c_i, c_{i+1}]$ as an interval.

A mountain-valley assignment (or simply an MV assignment) on the crease pattern C is a function $\mu : \{c_1, ..., c_{n-1}\} \to \{M,V\}$. Note that an MV assignment restricts the way in which each crease can be folded. A crease pattern C together with an MV assignment on C forms an MV pattern. A flat folding of an MV pattern (C, μ) is a piecewise-isometric embedding of the line segment $[c_0, c_n]$, bent along every crease c_i, for $0 < i < n$, according to the assignment $\mu(c_i)$, and not bent along any point that is not a crease. In particular, each interval $[c_i, c_{i+1}]$ of the crease pattern C must be mapped to a congruent copy, and the connectivity between intervals must be preserved. Furthermore, as the segments are folded flat in accordance with the MV assignment, they must not cross one another (although they may overlap). For a fixed crease pattern C, an MV assignment μ on C is foldable if there is a flat folding of the MV pattern (C, μ). Two creases with assignments M and V are said to have opposite MV parity. For a crease pattern C, we say that MV assignments μ_1 and μ_2 agree with each other on $F \subseteq C$ if for each crease $c \in F$, $\mu_1(c) = \mu_2(c)$.

Given an MV pattern (C, μ), we say that a subset F of C is forcing if the only foldable MV assignment on C that agrees with μ on F is μ itself. This means that, if each crease $c \in F$ is assigned the value $\mu(c)$, then each crease $c' \in C \setminus F$ must be assigned the value $\mu(c')$, in order to produce a foldable MV assignment. As an example, consider the MV pattern depicted in Figure 1a, and imagine that our crease pattern $C = (c_{i-1}, c_i, c_{i+1}, c_{i+2})$ contains only these four creases. Thus c_{i-1} and c_{i+2} are ends, and the segment folds about c_i and c_{i+1} according to the assignment $\mu(c_i) = M$ and $\mu(c_{i+1}) = V$. Assume that the two end intervals $[c_{i-1}, c_i]$ and $[c_{i+1}, c_{i+1}]$ are both strictly longer than the middle interval $[c_i, c_{i+1}]$. Then a forcing set for this example is $F = \{c_{i+1}\}$, because the only foldable assignment that agrees with $\mu(c_{i+1}) = V$ is the one that assigns c_i to be an M crease; otherwise, if c_i was also a V crease, then the bigger intervals $[c_{i-1}, c_i]$ and $[c_{i+1}, c_{i+2}]$ would both cover the smaller interval $[c_i, c_{i+1}]$ on the same side, causing a self-intersection (as depicted in Figure 1b). Similar arguments show that $F = \{c_i\}$ is also a forcing set, because it forces c_{i+1} to be a V

3

crease, otherwise the assignment would not be foldable.

A forcing set F is called <u>minimum</u> if there is no other forcing set with size less than $|F|$. In this paper we present a linear time algorithm that takes as input a flat-foldable MV pattern (C, μ) and produces as output a minimum forcing set F of C.

2. Preliminaries

This section introduces some terminology and preliminary results that will be used in constructing a minimum forcing set in Section 3 and in proving the correctness of the forcing set algorithm in Section 4.

Throughout the paper we work with a flat-foldable MV pattern (C, μ), where $C = (c_0, c_1, ..., c_n)$ is a one-dimensional crease pattern and μ is a foldable MV assignment. So whenever we talk about the MV pattern (C, μ), we assume that it is <u>flat-foldable</u>, unless otherwise specified.

2.1. Crimpable Sequences

Given a MV pattern (C, μ), our algorithm (described in Section 3) repeatedly identifies <u>crimpable</u> sequences of creases that can be used to force MV folds. A <u>crimpable</u> sequence is composed of consecutive equidistant creases, say distance d apart, with the property that the two intervals adjacent to the left and right end of the sequence are strictly longer than d. Formally, for integers $i, k > 0$, a sequence of consecutive creases $(c_i, c_{i+1}, \ldots, c_{i+k})$ is crimpable if $|c_{i+1} - c_i| = |c_{i+2} - c_{i+1}| = \ldots = |c_{i+k} - c_{i+k-1}|$ and $|c_i - c_{i-1}| > |c_{i+1} - c_i| < |c_{i+k+1} - c_{i+k}|$.
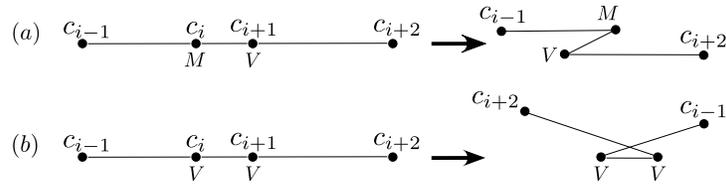

Figure 1: A monocrimp operation on a crimpable sequence (c_i, c_{i+1}). (a) c_i, c_{i+1} have opposite MV parity (b) c_i, c_{i+1} have the same MV parity; self intersection is unavoidable.

A <u>monocrimp</u> operation folds the paper about a single pair of consecutive creases of opposite MV parity in a crimpable sequence, according to the MV assignment μ. Figure 1a depicts a monocrimp operation performed on a pair of creases (c_i, c_{i+1}). The three intervals involved in the monocrimp operation will fuse together into a new interval with endpoints c_{i-1} and c_{i+2}. The result is a new crease pattern, reduced from the one before the crimp.

4

Note that the notions of top and bottom need to be redefined for the fused segment, so that the top faces upwards and the bottom faces downwards. After the monocrimp, the creases c_i and c_{i+1} will disappear and the three intervals from c_{i-1} to c_{i+2} will be replaced with an interval of length

$$c_i - c_{i-1} - (c_{i+1} - c_i) + c_{i+2} - c_{i+1} = c_{i+2} - 2c_{i+1} + 2c_i - c_{i-1}. \quad (1)$$

One important property of the monocrimp operation, which will be used in our algorithm, is captured by the following lemma.

Lemma 1. *After a monocrimp is performed on a pair of creases (c_i, c_{i+1}) in a crimpable sequence, the interval to the right of c_{i-1} (which is also the interval to the left of c_{i+2}) can get longer or remain the same size.*

PROOF. By the definition of a monocrimp, c_i and c_{i+1} belong to a crimpable sequence, therefore

$$c_i - c_{i-1} \geq c_{i+1} - c_i \leq c_{i+2} - c_{i+1} \quad (2)$$

The left (right) inequality from (2) is strict if c_i (c_{i+1}) is the first (last) crease in the crimpable sequence, otherwise the two terms on either side are equal (by the definition of a crimpable sequence). After the monocrimp both c_i and c_{i+1} disappear, and the intervals from c_{i-1} to c_{i+2} are replaced by a single interval of length $c_{i+2} - 2c_{i+1} + 2c_i - c_{i-1}$ (see (1)). Thus the interval to the right of c_{i-1} changes size from $c_i - c_{i-1}$ to

$$\begin{aligned}
c_{i+2} - 2c_{i+1} + 2c_i - c_{i-1} &= c_i - c_{i-1} + (c_{i+2} - c_{i+1}) - (c_{i+1} - c_i) \\
&\geq c_i - c_{i-1}. \quad \text{(by (2))}
\end{aligned}$$

This latter inequality follows from the definition of a crimpable pair, by which $c_{i+2} - c_{i+1} \geq c_{i+1} - c_i$. An analogous argument holds for the interval to the left of c_{i+2}. □

The following corollary follows immediately from Lemma 1.

Corollary 1. *Let $\alpha = (c_i, c_{i+1}, \ldots, c_{i+k})$ be a crimpable sequence in C. If $k > 2$, the result of a monocrimp operation performed on α is a crimpable sequence with two fewer creases.*

Note that a monocrimp operation performed on a crimpable sequence $\alpha = (c_i, c_{i+1})$ requires c_i and c_{i+1} to have opposite MV parity, for if they were both M or both V, then the bigger intervals $[c_{i-1}, c_i]$ and $[c_{i+1}, c_{i+2}]$

would both cover the smaller interval $[c_i, c_{i+1}]$ on the same side, causing a self-intersection. This leads to the key observation that, if $\mu(c_i) = M$, then it must be that $\mu(c_{i+1}) = V$, otherwise μ is not foldable. Thus we can assign one of the creases in the pair to match the assignment in μ, and the other crease will be forced to have opposite MV parity.

We will use a generalization of this property in constructing a minimum forcing set. Theorem 1 below originally appeared in an equivalent form as Theorem 4 in [11], with angles corresponding to our intervals. It quantifies the difference in the number of M and V assignments for creases in crimpable sequences of even and odd length.
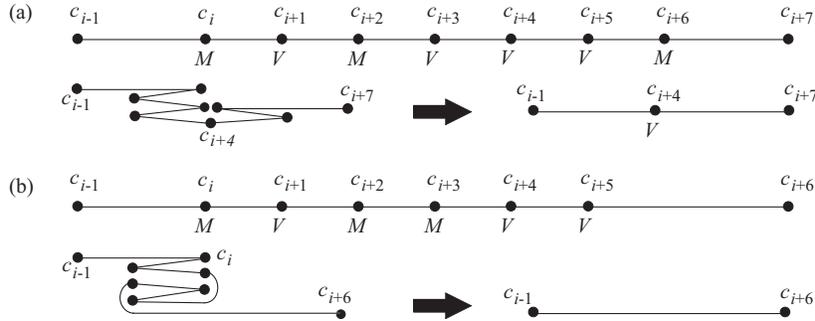

Figure 2: Crimp operation on an (a) odd number and (b) even number of creases. Below each crease pattern is the result of the crimp with folds shown (left) and with the folds fused together (right).

Theorem 1 (Theorem 4 from [11]). *Let α be a crimpable sequence in a foldable MV pattern. The difference in the number of M and V assignments for the creases in α is zero (one) if α has an even (odd) number of creases.*

This theorem allows us to define a <u>crimp</u> operation on a crimpable sequence $\alpha = (c_i, c_{i+1}, \ldots, c_{i+k})$ as an <u>exhaustive</u> sequence of monocrimps performed in α; here by <u>exhaustive</u> we mean that no monocrimp operation can be further performed on the sequence. Next we show that a crimp operation can always be performed on a crimpable sequence with two or more creases.

Lemma 2. *Let α be a crimpable sequence in a foldable crease pattern C. A crimp operation on α reduces α to a null sequence if the length of α is even, and to a single crease $c \in \alpha$ if it is odd. In the latter case, the position of c and the interval distances within the resulted crease pattern are independent of the MV assignment.*

6

PROOF. Let $\alpha = (c_i, c_{i+1}, \ldots, c_{i+k})$, for a fixed $k > 0$. By Theorem 1, among creases $c_i, c_{i+1}, \ldots, c_{i+k}$, there is an adjacent pair with opposite MV assignments. A monocrimp operation on this pair of creases eliminates the creases (thus eliminating one M and one V crease), and the result is a crimpable sequence with two fewer creases (by Corollary 1). So we can repeat this process until there are zero (if k is odd) or one (if k is even) creases remaining in α. In the latter case, the position of this crease is always between c_{i-1} and c_{i+k+1}, so it is independent of the MV assignment. As shown in the proof of Lemma 1, the lengths of the intervals in the resulted crease pattern depend only on the position of the creases in C and therefore are also independent of the MV assignment. □

In the case of a crimpable sequence α of odd length, we say that the crease obtained from α through a crimp operation survives the crimp, and all other creases from α disappear. The observation below follows immediately from the fact that a crimp operation is composed of a series of monocrimps.

Observation 1. *If a crimpable sequence α has an odd number of creases, then a crimp operation on α yields a surviving crease with the same MV assignment as the majority of the creases in α.*

The crimp operation is depicted in Figure 2a for $k = 6$, and Figure 2b for $k = 5$.

2.2. End Sequences

Our algorithm will repeatedly perform crimp operations, until no crimpable sequences are left in the crease pattern. We call the resulting crease pattern an end sequence and the creases end creases. Let λ be the sequence

Figure 3: (a) An end sequence, (b) folded flat using end folds.

of distances between consecutive creases in an end sequence E, in order from left to right. There can be no local minima in λ other than at the ends, because a local minimum would indicate the presence of a crimpable sequence. Therefore, the sequence λ starts with zero or more values that are monotonically increasing, and this is followed by zero or more values that are monotonically decreasing. The following lemma implies that no MV assignments can be forced on the creases of an end sequence, which

also implies that the forcing set of an end sequence must include all the end creases.

Lemma 3. *Any MV assignment to the creases of an end sequence is foldable.*

PROOF. We show how to fold an end sequence E with an arbitrary MV assignment using a series of *end folds*. An end fold consists of folding the first or last interval of the sequence under or over, depending on the MV assignment of the crease. To ensure that the folded interval doesn't overlap other creases, an end fold is performed only when the adjacent interval is of the same length or longer. Note that each end fold can be performed regardless of its MV assignment.

Let $\ell = [c_j, c_{j+1}]$ be the longest interval in E; in case of ties pick the leftmost interval. Because the intervals are monotonically increasing to the left of ℓ and monotonically decreasing to the right of ℓ, we can repeatedly fold the leftmost interval under (for a mountain fold) or over (for a valley fold) until ℓ becomes the leftmost interval. Similarly, because the intervals are monotonically increasing to the right of ℓ, we can do the same on the right end of E until only ℓ remains, thus producing a flat folding. See Figure 3 for an example where $\ell = [c_2, c_3]$. □

It follows from Lemma 3 that the forcing set of an end sequence must include all the end creases.

3. Constructing a Minimum Forcing Set

We describe a simple algorithm that finds a minimum forcing set F for any given foldable MV pattern (C, μ). Although the algorithm is straightforward, the proof that it finds a minimum forcing set is quite intricate. The algorithm begins by constructing a forest of trees whose nodes correspond to crimpable sequences encountered during the folding process and whose edges capture dependencies between them (subsection 3.1). Then it traverses the trees in a top-down manner to decide which creases corresponding to each node should be added to the forcing set (subsection 3.2).

3.1. Crimp Forest Construction

The CRIMPFOREST algorithm detailed in Algorithm 1 computes a *crimp forest* W corresponding to the foldable input MV pattern (C, μ). W is a collection of rooted trees, where each node corresponds to a crimpable

sequence that is encountered during the folding process. For simplicity, we will use the terms "node" and "crimpable sequence" interchangeably to refer to a node in a crimp tree. For a crease pattern C, we can easily identify its set S of crimpable sequences by scanning its interval lengths from left to right. In W, a node c is a child of another node p if the crimp operation on c has a surviving crease, and the next crimp sequence involving the surviving crease is p.

$$\textsc{CrimpForest}(C, \mu)$$

Initialize $W \leftarrow \emptyset$.
while C has a crimpable sequence **do**
    | Let s be the leftmost crimpable sequence in C.
    | Create a node v corresponding to s, and add v to W.
    | Make v the parent of each root node in W whose crimp sequence
    | has a surviving crease that is in s.
    | Apply the crimp operation to s.
    | Update C to be the resulting crease pattern.
end

Algorithm 1: CRIMPFOREST ALGORITHM.

See Figure 4 for an example of a crease pattern and corresponding crimp forest. Initially the forest W is empty. The leftmost crimpable sequence in the crease pattern from Figure 4 (bottom right) is $\alpha_1 = (c_1, c_2, c_3)$. The CRIMPFOREST algorithm processes α_1 by adding a corresponding node to the forest W and performing the crimp operation on α_1, resulting in the smaller crease pattern shown on the right, second from bottom. The crease c_1 survives the crimp operation on α_1. The leftmost crimpable sequence is now $\alpha_2 = (c_1, c_4, c_5)$. Because α_2 includes the survivor c_1 of α_1, the algorithm adds α_2 to W as a parent of α_1. After crimping α_2, the leftmost crimpable sequence is $\alpha_3 = (c_6, c_7, c_8)$, which is crimped (with c_8 surviving) and a corresponding node is added to W. The new crease pattern includes one crimpable sequence $\alpha_4 = (c_5, c_8, c_9)$. Because c_5 is the survivor of α_2 and c_8 is the survivor of α_3, the node corresponding to α_4 becomes parent of both α_2 and α_3 in W. The underlined creases from Figure 4 will be used in the description of the minimum forcing set algorithm from subsection 3.2.

The observation below follows immediately from the fact that creases at the root nodes are not involved in any further crimp operations.

Observation 2. *If a crimpable sequence α has an even number of creases (and so an equal number of M and V assignments), then α is the root of a*

9

Figure 4 (a) crimp forest:

$(c_5, \underline{c_8}, c_9)$

$(\underline{c_1}, c_4, c_5)$ $(c_6, \underline{c_7}, \underline{c_8})$

$(\underline{c_1}, \underline{c_2}, c_3)$

(a)

(b) crease patterns:

c_9 — M

c_5 (V), c_8 (M), c_9 (M)

c_1 (M), c_4 (V), c_5 (V), c_8 (M), c_9 (M)

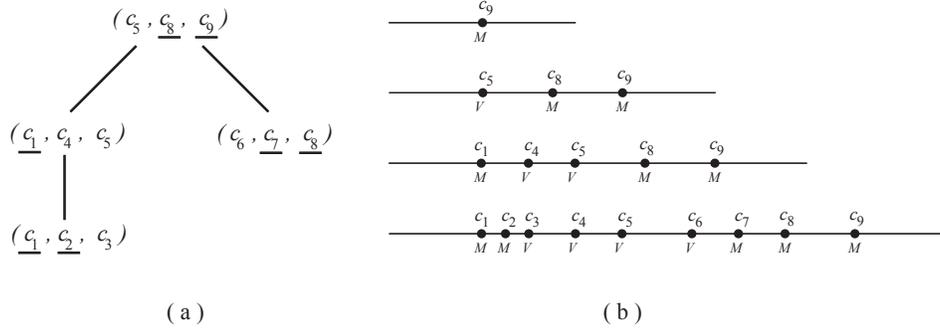c_1 (M), c_2 (M), c_3 (V), c_4 (V), c_5 (V), c_6 (V), c_7 (M), c_8 (M), c_9 (M)

(b)

Figure 4: The (a) crimp forest corresponding to the (b) crease pattern C on the lower right having interval lengths 4, 1, 1, 2, 2, 3, 2, 2, 3, 4. Each new crease pattern resulting after processing the leftmost crimpable sequence is shown above the previous pattern. The underlined creases in (a) are referred to in the algorithm from subsection 3.2.

tree in W. If the root r of a tree in W has an odd number of creases (and so unequal number of M and V assignments), then the crimp operation on r yields a surviving crease that is an end crease (in the end sequence).

We will need the following lemma in our proof of correctness in Section 4.

Lemma 4. *Given a crease pattern C and two foldable MV assignments μ_1 and μ_2, let W_1 and W_2 be the crimp forests corresponding to (C, μ_1) and (C, μ_2), respectively. Then the following properties hold:*

(1) W_1 and W_2 are structurally identical.

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

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

PROOF. By definition, a consecutive sequence $c_i, c_{i+1}, \ldots, c_{i+k}$ of creases is crimpable if the creases are equally spaced at some distance d and the intervals adjacent to the right of c_{i+k} and to the left of c_i are strictly longer than d. The definition is based solely on interval lengths, and so the identification of crimpable sequences is independent of the MV assignment.

The sequence of interval lengths resulting after a crimp operation is also independent of the MV assignment. This is because a crimp operation consists of a series of monocrimps, and after each monocrimp two adjacent creases c_i and c_{i+1} disappear and are replaced with an interval of length

10

$c_{i+2} - 2c_{i+1} + 2c_i - c_{i-1}$, as derived in equation (1) in subsection 2.1. It does not depend on the MV assignment. We note, however, that for an odd length crimpable sequence, the particular crease that survives may be different. For example, in crimpable sequence (c_1, c_2, c_3) in Figure 4a with assignment (M, M, V), crease c_1 survives; but with assignment (V, M, M), crease c_3 survives instead.

Consider now two runs of the CRIMPFOREST algorithm (Algorithm 1), one using input (C, μ_1) and the other using input (C, μ_2). In the first iteration of the while loop, both runs identify the same set of crimpable sequences in C, because identification of crimpable sequences is independent of MV assignment. In this case nodes with identical crimpable sequences of creases are added to W_1 and W_2. (We abuse our notation slightly here by letting W_1 be the forest in the current iteration of the run using input (C, μ_1), and similarly for W_2). Therefore, properties (1), (2) and (3) hold for W_1 and W_2 after the first iteration of the two runs. In addition, after performing the crimp operation on each identified crimpable sequence, the updated crease pattern C in both runs consists of the same sequence of interval lengths, because the interval sequence resulting after a crimp operation is independent of the MV assignment. As noted earlier, the particular crease surviving a crimp operation in one run may not be the same as the surviving crease in the other run. By Lemma 2, for each survivor in one run, there is a corresponding survivor in the other and they have the same position in the resulting crease sequence. This will be important later when adding edges to the forests.

Assume inductively that after the ith iteration, properties (1), (2) and (3) hold for W_1 and W_2. In addition, assume inductively that the resulting crease pattern in each run consists of the same sequence of interval lengths, and includes all creases not yet involved in any crimpable sequences in the same position (in addition to those creases surviving the crimps).

Now consider the $(i+1)$th iteration. Because the crease patterns in both runs after iteration i have the same sequence of interval lengths, the set of crimpable sequences identified in iteration $i + 1$ will also be the same. Thus each node created in iteration $i + 1$ of the run under μ_1 has a corresponding node created in the run under μ_2 with a crimpable sequence of the same size and same interval lengths. Thus property (2) holds. By the inductive hypothesis, creases involved for the first time in the crimpable sequences identified in iteration $i + 1$ have the same position in the crease patterns resulted from iteration i. In particular, they have the same position in the crimpable sequences associated with corresponding nodes in W_1 and W_2, thus establishing property (3).

11

If a node n_1 created in iteration $i + 1$ of the run under μ_1 contains a crease c_j that is a survivor of the crimpable sequence at a root node r_1 of W_1, then by the inductive hypothesis, there is a corresponding root node r_2 in W_2 whose crimpable sequence has a corresponding survivor c_ℓ. In the $(i + 1)$th iteration of the second run, a node n_2 (corresponding to n_1) is created that contains c_ℓ in its crease sequence. Therefore, edge (n_1, r_1) will be added to W_1 and edge (n_2, r_2) will be added to W_2, so the structure of the two forests remains the same. Thus property (1) holds. Because the identified crimpable sequences are in the same locations, and have the same size and same interval lengths in both runs, after applying the crimp operations in iteration $i + 1$, the resulting crease sequences have the same interval distances in both runs. By Lemma 2, survivors of these crimp operations will have the same position in the two crease sequences. This along with the inductive hypothesis implies that creases not yet involved in a crimp operation, which intersperse with survivors of previous crimp operations, have the same position in the crease patterns resulted from the two runs. This establishes the inductive hypothesis for the $(i + 1)$th iteration. $\qquad\square$

Lemma 5. *The time complexity of the* CRIMPFOREST *algorithm (Algorithm 1) is $O(n)$, where n is the number of input creases.*

PROOF. We show that the CRIMPFOREST algorithm has a simple linear-time implementation that scans the crease pattern C from left to right, repeatedly identifying the leftmost crimpable sequence, α, and then processing and crimping it as described in Algorithm 1. The only complication is that crimping α can result in a new crimpable sequence forming on its left, but we show this can be detected and handled efficiently.

We describe the details in three steps which we repeat until we reach the end of C. It is helpful here to recall that a crimpable sequence is a local minimum in the interval lengths of C.

- In step 1, we scan the intervals in C from left to right searching for the first local maximum. Specifically, we pass over zero or more intervals of monotonically increasing length until reaching the first interval, I_0, having a smaller interval immediately to its right.

- Then in step 2, we resume scanning to the right for the first local minimum, passing over one or more intervals of monotonically decreasing length until reaching the first interval having a larger interval immediately to its right. This interval marks the end of the leftmost crimpable sequence, α, in C.

- In step 3, in time linear in the number of creases in α, we can identify α's creases and the intervals I_ℓ and I_r located to its left and right, and then process α as described in the loop iteration of Algorithm 1.

If α has an odd number of creases, then I_ℓ's length is not affected by the crimp operation on α, so we go back to step 2 and resume scanning for the first local minimum starting from I_ℓ. If, on the other hand, α has an even number of creases, then I_ℓ and I_r merge into a new, longer interval I after the crimp operation, which may have produced a new crimpable sequence (i.e., local minimum) immediately to the left of I. There are three cases to consider: (a) If $I_0 \subset I$ (i.e., $I_0 = I_\ell$ merged with I_r into I), or if I_0 is immediately to the left of I and I is longer than I_0, then no new local minimum was produced, but the first local maximum may have changed; in this case, we go back to step 1 and resume scanning from I for the first local maximum (and reset I_0 if the maximum changed); (b) If I is shorter than the interval J immediately to its left, then again no new local minimum was produced and we go back to step 2, looking for the first local minimum starting from I; (c) If however I is longer than J, then J marks the end of a newly formed local minimum (because interval lengths from I_0 to J have decreasing lengths and I is longer than J); in this case, we let α be the newly formed crimpable sequence, and we go back to step 3 and process it similarly.

Using this implementation, each interval is involved in only a constant number of operations that include comparisons and crimps. This shows that the crimp forest construction can be done in time linear in the number of creases in C.

3.2. Forcing Set Algorithm

Given a foldable MV pattern (C, μ), the FORCINGSET algorithm described in Algorithm 2 computes a minimum forcing set F of C. The algorithm starts by constructing the crimp forest discussed in the previous section (subsection 3.1), then processes each tree in the forest in a top down manner. As an example, the forcing set for the crimp tree depicted in Figure 4 consists of all underlined creases in the tree. Initially the forcing set F contains the only end crease c_9. We begin a preorder traversal of the tree rooted at node $\alpha_4 = (c_5, c_8, c_9)$. Because the surviving crease c_9 of α_4 is already in F, we add to F the crease c_8, because it has the majority MV assignment (same as c_9). Moving down to $\alpha_2 = (c_1, c_4, c_5)$, the surviving crease c_5 of α_2 is not in F, so in this case we add c_1 to F because it has the minority MV assignment among all creases in α_2. The other nodes are

handled similarly. This procedure takes $O(n)$ time for a crease pattern with n creases.

FORCINGSET(C, μ)

Initialize W to the output generated by CRIMPFOREST(C, μ).
Initialize F to the set of end creases that remain after running
CRIMPFOREST(C, μ)
for each tree $T \in W$ **do**
    for each node v in a preorder traversal of T **do**
        if v's crimp sequence has even length **then**
            Add to F all creases from v's crimp sequence having M
            assignment.
        else if the surviving crease from v's crimp sequence is already
        in F **then**
            Add to F all creases from v's crimp sequence having the
            majority MV assignment.
        else
            Add to F all creases from v's crimp sequence having the
            minority MV assignment.
        end
    end
end

Algorithm 2: Forcing set algorithm.

Lemma 6 is a key ingredient in our proof of correctness (presented in Section 4).

Lemma 6. *Let (C, μ_1) be a foldable MV pattern, and let F be the forcing set generated by Algorithm 2 with input (C, μ_1). Let (C, μ_2) be a foldable pattern such that μ_2 agrees with μ_1 on the forcing set F. Let T_1 and T_2 be two structurally equivalent trees generated by the forcing set algorithm with inputs (C, μ_1) and (C, μ_2), respectively. If a crease c in a crimpable sequence $\alpha_1 \in T_1$ is in F, then a crease (not necessarily c) with the same MV assignment occurs in the corresponding crimpable sequence $\alpha_2 \in T_2$, in the same position as in α_1.*

PROOF. Our proof is by induction on the height h of the trees T_1 and T_2. The base case corresponds to $h = 1$ (each tree is a leaf node). In this case α_1 is the first crimpable sequence involving c, and the lemma follows immediately from property (3) established by Lemma 4.

The inductive hypothesis states that the lemma holds for any trees T_1 and T_2 of height i or less. If α_1 is at level i or lower in T_1, then the lemma holds by the inductive hypothesis. So consider now the case when α_1 is at level $i+1$ and contains crease $c \in F$. If c has not yet been involved in a crimp operation, then the lemma follows immediately from property (3) established by Lemma 4. Otherwise, c is a crease surviving a crimp operation performed on some child node β_1 of α_1. Because the forcing set algorithm processes T_1 top-down, the crease c is already in F at the time node β_1 is processed. As a result, all creases from β_1 with the majority assignment are added to F. By the inductive hypothesis, creases with the same MV assignment occur in the corresponding node $\beta_2 \in T_2$, in the same positions as in β_1. Because (C, μ_1) and (C, μ_2) are foldable, by Theorem 1 all other creases in β_1 and β_2 have the opposite assignment. This implies that the crease surviving the crimp operation on β_2 has the same MV assignment as the crease surviving the crimp operation on β_1, which is c. This shows that a crease with the same assignment as c occurs in α_2 as well. By Lemma 2, this crease has the same position in α_2 as c in α_1. □

4. Proof of Correctness

In this section we prove that the FORCINGSET algorithm outlined in subsection 3.2 produces a minimum forcing set F of the input MV pattern (C, μ). Call two crimpable sequences α_1 and α_2 similar if they have the same size, the same MV assignment read from left to right, and same interval lengths. Call a sequence of crimps O exhaustive if the crease pattern produced by O contains no crimpable sequences (so no more crimps could be performed on this crease pattern). Our algorithm will perform an exhaustive sequence of crimps on the crease pattern C. An important issue here is that, if several crimps could be performed on the crease pattern at one point, then we must choose a particular order for these crimps, including any new crimps that might occur as a result of the previous crimps. A natural question here is whether this ordering is important. In other words, could different orderings of crimps result in different (say, larger or smaller) forcing sets? Lemma 7 below proves that this will not be an issue.

Lemma 7. *Let O_1, O_2 be two non-identical, exhaustive sequences of crimps performed on a one-dimensional crease pattern C. Then every crimp that is in O_1 is also in O_2.*

Lemma 7 is independent of the algorithm itself, so to maintain the flow

of presentation we defer a proof of this lemma to the Appendix. Our proof of correctness uses two more fundamental lemmas.

Lemma 8 ([8]). *Any folding of a foldable 1D crease pattern can be performed by a sequence of monocrimps and end folds.*

Lemma 9. *Let (C, μ) be a foldable MV pattern, and let (a_i, a_{i+1}) be a pair of adjacent creases of opposite MV parity in a crimpable sequence of C. Let μ' be identical to μ, with the exception that $\mu'(a_i) = \mu(a_{i+1})$ and $\mu'(a_{i+1}) = \mu(a_i)$. Then μ' is a foldable MV assignment.*

PROOF. A flat folding of (C, μ') is identical to a flat folding of (C, μ), with the only exception that the overlap order of the two layers corresponding to the creases a_i and a_{i+1} gets switched.

Our main result is stated by the following theorem.

Theorem 2. *Given a flat-foldable MV pattern (C, μ), the FORCINGSET algorithm (Algorithm 2) produces a minimum forcing set F of (C, μ) in time $O(n)$, where n is the number of creases in C.*

PROOF. By Lemma 5, the CRIMPFOREST algorithm constructs the crimp forest W in linear time. The FORCINGSET algorithm then processes each crimpable sequence s corresponding to a node in W in time linear in the length of s. Since the total length of all crimpable sequences in the forest is $O(n)$, we obtain linear time complexity for the FORCINGSET algorithm.

Our proof of correctness has two parts: first we prove that F is indeed a forcing set for (C, μ), then we prove that F has a minimum size.

F is forcing. Assume to the contrary that there exists a different foldable MV assignment μ_2 for C that agrees with μ on F. For symmetry, let $\mu_1 = \mu$ and run the FORCINGSET algorithm with input (C, μ_1) to create forest W_1, then run it again with input (C, μ_2) to create forest W_2. By property (1) of Lemma 4, W_1 and W_2 are structurally equivalent.

Consider first the case when there is a pair of corresponding trees $T_1 \in W_1$ and $T_2 \in W_2$ that have at least one pair of corresponding nodes with dissimilar crimpable sequences. By Lemma 4, their crimpable sequences involve the same number of creases, have the same interval lengths, and first-time creases have the same locations in both. Therefore the sequences differ only in their MV assignments. Of all such pairs, let $v_1 \in T_1$ and $v_2 \in T_2$ be a pair of maximal depth in their trees.

16

Let ℓ be the length of the crimpable sequences corresponding to v_1 and v_2. If ℓ is even, then v_1, v_2 are root nodes in T_1, T_2 (by Observation 2). In this case, the FORCINGSET algorithm places in F the $\ell/2$ creases of v_1 with assignment M. Then by Lemma 6, there are $\ell/2$ creases in v_2 in the same positions that are also assigned M. By Theorem 1, the remaining $\ell/2$ creases in both v_1, v_2 must have V assignments. But then v_1, v_2 have similar crimpable sequences, a contradiction.

Now consider the case when ℓ is odd. If the creases of v_1 with the majority MV assignment are in F, then by Lemma 6 v_2 has creases of the same MV assignment located in the same positions. All other creases in v_1, v_2 must have the opposite assignment by Theorem 1. But then v_1, v_2 are similar, a contradiction.

$$T_1 \qquad\qquad\qquad\qquad\qquad T_2$$

$$w_1 = [c_5(V), \underline{c_8(M)}, \underline{c_9(M)}] \qquad\qquad w_2 = [c_1'(M), c_8'(M), c_9'(M)]$$

$$p_1 = [c_3(V), \underline{c_4(M)}, c_5(V)] \quad [c_6(V), \underline{c_7(M)}, \underline{c_8(M)}] \qquad p_2 = [c_1'(M), c_4'(M), c_5'(V)] \quad [c_6'(V), c_7'(M), c_8'(M)]$$

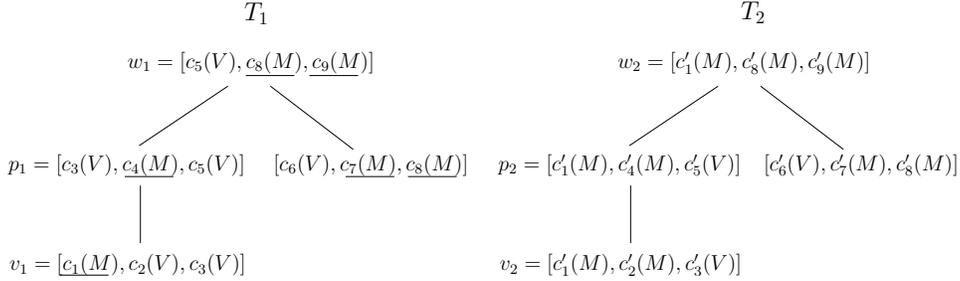$$v_1 = [\underline{c_1(M)}, c_2(V), c_3(V)] \qquad\qquad\qquad v_2 = [c_1'(M), c_2'(M), c_3'(V)]$$

Figure 5: Proof of correctness for the FORCINGSET algorithm (Algorithm 2) from subsection 3.2. Underlined are the creases included in the forcing set F.

If instead the creases of v_1 with the minority MV assignment are in F, then assume without loss of generality that the minority assignment is M. This case is depicted in Figure 5. The survivor of v_1 is not in F (because otherwise the creases of v_1 with the majority MV assignment would be in F, as indicated by the second branch of the conditional in the FORCINGSET algorithm), and it has a V assignment (because the survivor always has the majority assignment, cf. Observation 1).

Next we show that on the path from v_1 to the root r_1 of T_1, we must encounter a node with majority assignment M, or an equal number of M and V assignments. Suppose that this is not the case. Then each node u on the path from v_1 to r_1 has majority V assignments, and a crimp operation on u produces a surviving crease of type V. In particular, the crease surviving the crimp operation on r_1 is an end crease (cf. Observation 2), and therefore is in F (cf. step 2 of the forcing set algorithm). But in this case the FORCINGSET algorithm would place in F the majority V creases from r_1, and would do the same at each node on the path from r_1 down to and including node v_1

17

(because the crimp survivor at each such node would be in F at the time the node would be processed). But this contradicts our assumption that the M creases from v_1 are in F. So let w_1 be the first node encountered on the path from v_1 to r_1 having majority assignment M or an equal number of M and V assignments. (See the node labeled w_1 in Figure 5.)

By Lemma 6, if the minority M creases of v_1 are in F, then v_2 has $(\ell - 1)/2$ creases of type M located in the same positions. Therefore, any differences between v_1, v_2 must be among the other $(\ell+1)/2$ creases. Because v_1, v_2 have maximal depth, the crimpable sequences at corresponding nodes in their children's subtrees are similar, and thus their children contribute to v_1 and v_2 survivors with the same MV assignment. The differences in v_1, v_2's crimpable sequences must therefore be in first-time creases. (Note that if v_1, v_2 are leaves, then all their creases are first-time creases.)

We now show that all such first-time creases in v_2 must have a V assignment. Suppose this is not the case (as illustrated in Figure 5 where $\mu(c_2') \neq V$). Then M is the majority assignment at v_2, and the survivor of v_2 is of type M – in contrast to v_1, whose survivor is of type V. (In Figure 5, the survivor is c_1' of type M). Consider the parents p_1, p_2 of v_1, v_2, respectively. If $p_1 \neq w_1$, then p_1 has majority assignment V. In addition, its minority creases of type M are in F — for if its majority V creases were in F, then the majority V creases at v_1 would also be in F, a contradiction. By Lemma 6, each M crease in p_1 has a corresponding M crease in p_2. In addition, p_2 has an additional crease of type M propagated by v_2, making M the majority assignment at p_2. Thus p_2 produces a survivor of type M – in contrast to p_1, which has a survivor of type V. We can apply this argument at each node on the way back up the tree to the child of node w_2 (corresponding to w_1). We recycle our notation here by refering to w_2's child as p_2 and the corresponding node in T_1 as p_1.

If w_1 has equal M and V assignments, then it is the root node (see Observation 2), its M creases are in the forcing set, and corresponding M creases are also in w_2. In addition, w_2 has an additional crease of type M propagated by p_2. But then the difference in M and V creases at w_2 is greater than one, contradicting Theorem 1. If, on the other hand, w_1 has majority assignment M, then its M creases must be in F (otherwise the majority V creases at each node from p_1 down to and including v_1 would also be in F, contradicting our assumption). Therefore, w_2 has corresponding M creases. In addition, w_2 also contains an additional crease of type M that it got from p_2. But then the difference in M and V creases at w_2 is greater than one, which again contradicts Theorem 1.

We have established that corresponding first-time creases, as well as

survivors from children of v_1 and v_2, have the same MV assignment in v_1 and v_2. This implies that v_1 and v_2 are similar, contradicting our choice of dissimilar sequences.

Finally, consider the case when all corresponding nodes in W_1 and W_2 have similar crimpable sequences. Then there must be creases in (C, μ_2) not involved in any crimp operation, whose assignments differ from the corresponding creases in (C, μ_1). Such creases must be part of the end sequence of (C, μ_2). However Lemma 7 implies that end sequence creases are independent on the MV assignment. Because all end sequence creases are in F, it is not possible that μ_1 and μ_2 differ on end sequence creases, so this case is settled. It follows that μ_1 and μ_2 are identical and therefore F is a forcing set.

F is minimum. Let OPT be a forcing set of (C, μ) of minimum size. Because OPT is a forcing set, all other creases in $C \backslash$ OPT must have an MV assignment that agrees with μ. By Lemma 8, any folding of (C, μ) can be achieved by a series of monocrimps and end folds. By Lemma 7, any ordering of the crimp operations produces similar sets of crimpable sequences, and similar end sequences. This implies that each monocrimp involved in the CRIMPFOREST algorithm appears in any set of monocrimps used to produce a flat folding of (C, μ).

Let m be the number of monocrimps involved in the crimps performed by the FORCINGSET algorithm, and let e be the size of the end sequence. We show that $|$OPT$| = m + e$. First note that OPT must include at least one crease from each monocrimp; otherwise, we can switch the MV assignment for the creases involved in the monocrimp and obtain a foldable MV assignment that is different from μ (Lemma 9), contradicting the fact that OPT is a forcing set. Also by Lemma 3, OPT must include all the creases from the end sequence. The intersection between the set of creases involved in the monocrimp operations and the set of creases in the end sequence is empty, because each monocrimp removes the pair of creases involved. It follows that $|$OPT$| = m + e$.

We now show that $|F| = m + e$ as well. The end sequence added to F in the second step of the algorithm contributes the second term to this equality. We claim that, corresponding to each crimpable sequence α of size ℓ, the FORCINGSET algorithm adds to F precisely $\lfloor \ell/2 \rfloor$ creases. This is clear for the cases when α is of even length, and when the minority of the creases from α are added to F. Now note that the FORCINGSET algorithm adds to F the majority creases from α only if the survivor of α is already in F. This means that the additional contribution of α to F is again $\lfloor \ell/2 \rfloor$ (because

one of the creases from the majority is already in F). The quantity $\lfloor \ell/2 \rfloor$ is precisely the number of monocrimps involved in crimping α. Summing up over all crimps performed by the algorithm, we get the total number of creases contributed to $|F|$ by the crimpable sequences to be m as well. This proves that F is a minimum forcing set. $\qquad\square$

The proof of Theorem 2 suggests a simple way to reconstruct the original MV assignment μ, given the crease pattern C and the forcing set F produced by the FORCINGSET algorithm. First we construct a crimp forest from the sequence of distances induced by C (note that the construction uses only interval lengths and is independent on the MV assignment). By Lemma 4, the result is a crimp forest W_1 isomorphic to the crimp forest W used by the FORCINGSET to determine F. By Lemma 6, crimpable sequences for corresponding nodes in W_1 and W have the same number (and position) of creases that are in F. This implies that, for each crimpable sequence α of size ℓ corresponding to a node in W_1, at least $\lfloor \ell/2 \rfloor$ creases of the same type (either M or V) are in F (as ensured by the FORCINGSET algorithm). Then we simply assign the remaining creases in α (the ones not in F) the opposite type. Arguments similar to the ones used in the proof of Theorem 2 show that this MV assignment does not incur any conflicts for creases that occur in multiple crimpable sequences, and that it agrees with μ.

5. Conclusion

This is the first paper that addresses the problem of finding a minimum forcing set in an arbitrary one-dimensional crease pattern. While folding problems can seem simple at first, the details of even one-dimensional folding are surprisingly complex. The algorithm presented in this paper finds a minimum forcing set for an arbitrary one-dimensional crease pattern in time linear in the number of creases. The algorithm complexities arise when there are consecutive sequences of equally spaced creases, but such sequences should not be viewed as degenerate cases given that many useful origami crease patterns are tessellations. The most obvious direction for future work on forcing sets is to consider arbitrary two-dimensional crease patterns; some initial work has seen print on the Miura-ori crease pattern [12]. Finding a general algorithm for the two-dimensional case is open.

[1] E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, R. J. Wood, Programmable matter by folding, Proceedings of the National Academy of Sciences 107 (2010) 12441–12445. doi:10.1073/pnas.0914069107.

[2] L. Ionov, 3d micro fabrication using stimuli-responsive self-folding polymer films, Polymer Reviews 53 (2013) 92–107.

[3] T. G. Leong, P. A. Lester, T. L. Koh, E. K. Call, D. H. Gracias, Surface tension-driven self-folding polyhedra, Langmuir 23 (17) (2007) 8747–8751.

[4] L. Mahadevan, S. Rica, Self-organized origami,, Science 307 (5716) (2005) 1740.

[5] B. Ballinger, M. Damian, D. Eppstein, R. Flatland, J. Ginepro, T. Hull, Minimum forcing sets for Miura folding patterns, in: Proc. 26th ACM-SIAM Symposium on Discrete Algorithms (SODA '15), 2015, pp. 136–147.

[6] E. Arkin, M. A. Bender, E. D. Demaine, M. L. Demaine, J. S. B. Mitchell, S. Sethia, S. Skiena, When can you fold a map?, Comput. Geom. Theory Appl. 29 (1) (2004) 166–195. `doi:10.1016/j.comgeo.2004.03.012`.

[7] M. Bern, B. Hayes, The complexity of flat origami, in: Proc. 7th ACM-SIAM Symposium on Discrete Algorithms (SODA '96), 1996, pp. 175–183.

[8] E. D. Demaine, J. O'Rourke, Geometric Folding Algorithms: Linkages, Origami, Polyhedra, Cambridge University Press, 2007.

[9] T. Morgan, Map Folding, Master's thesis, Mass. Inst. Tech. (2012).

[10] R. I. Nishat, S. Whitesides, Map folding, in: Canadian Conf. on Comp. Geom., 2013, pp. 49–54.
URL `http://www.cccg.ca/proceedings/2013/papers/paper_49.pdf`

[11] T. Hull, Counting mountain-valley assignments for flat folds, Ars Combinatoria 67 (2003) 175–188.

[12] B. Ballinger, M. Damian, D. Eppstein, R. Flatland, J. Ginepro, T. Hull, Minimum forcing sets for miura folding patterns, in: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '15, SIAM, 2015, pp. 136–147.
URL `http://dl.acm.org/citation.cfm?id=2722129.2722140`

6. Appendix

This section is devoted to the proof of Lemma 7, which is instrumental to the proof of correctness presented in Section 4, yet it is independent of the forcing set algorithm from Section 3 and can stand as a result on its own. We begin with a few insights into folding maps, which tell us where points go when we fold sets of consecutive creases.

6.1. Folding Maps

Recall that folding a crease c_i means reflecting all points to one side of c_i (i.e., all points to the right of c_i, or all points to the left of c_i, depending on $\mu(c_i)$) about c_i. This operation (depicted in Figure 6) can be modeled
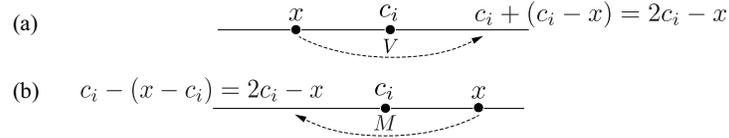
Figure 6: Reflecting point x about crease c_i (a) x left of c_i (b) x right of c_i.

by a function $\sigma_{c_i} : \mathbb{R} \to \mathbb{R}$ given by

$$\sigma_{c_i}(x) = c_i + (c_i - x) = 2c_i - x. \tag{3}$$

By composing such functions together, we can define a <u>folding map</u> that tells us where points go when we fold sets of consecutive creases. If $S = (a_1, a_2, ..., a_k)$ is a sequence of consecutive creases, then the <u>folding map σ_{S,a_1} that fixes $[a_1, a_2]$</u> (i.e., it does not fold a_1) is

$$\sigma_{S,a_1}(x) = \sigma_{a_2}\sigma_{a_3}\sigma_{a_4}\cdots\sigma_{a_i}(x) \text{ where } x \in [a_i, a_{i+1}], \ i < k.$$

Lemma 10. *Let x be an arbitrary point on a line segment, and let $S = (a_1, a_2, ..., a_i)$ be a foldable crease pattern with the property that each fold about each crease a_j, with $2 \leq j \leq i$, affects the location of x. Then*

$$\sigma_{S,a_1}(x) = 2a_2 - 2a_3 + 2a_4 - \cdots + (-1)^i 2a_i + (-1)^{i+1}x. \tag{4}$$

PROOF. The proof is by induction on the size i of S (which includes the fixed crease a_1). The base case correspond to $i = 2$ (that is, $S = (a_1, a_2)$ and the segment bends about crease a_2 only). In this case the location of x

after folding about a_2 is given by (3): $\sigma_{S,a_1}(x) = 2a_2 - x = 2a_2 + (-1)^3 x$. So the base case holds.

The induction hypothesis is that, for any sequence S of length i, the equality (4) holds. For the inductive step, consider a sequence S of length $i + 1$. Let $S = (a_1, a_2, \ldots, a_i, a_{i+1})$ of , and let $T = (a_1, a_2, \ldots, a_i)$ be the subsequence consisting of the first i elements of S.

Notice that the ordering in which the folds about the creases $a_j \in S$ are performed may affect the overlap order of the layers (intervals) in the final flat folding of S. For the purpose of our analysis, this nesting is not important; rather, it is the relative positioning of points in the folded state of S (with respect to the fixed point a_1) that affects the location of x. Arguments similar to the ones used in the proof of Lemma 2 show that any folding of the creases $a_2, a_3, \ldots, a_{i+1}$ places the points in the interval $[a_2, a_{i+1}]$ in the same positions, independent of the MV assignment. The MV assignment affects only the layering order and does not affect the positions of the creases in the folded state. This allows us to fold about the crease a_{i+1} first, which by (3) changes the position of point x to $x' = 2a_{i+1} - x$. We can now view x' as a new point, and apply the induction hypothesis to determine the new position of x' after folding about each crease in T:

$$
\begin{aligned}
\sigma_{S,a_1}(x) &= \sigma_{T,a_1}(x') \\
&= 2a_2 - 2a_3 + 2a_4 - \cdots + (-1)^i 2a_i + (-1)^{i+1} x' \\
&= 2a_2 - 2a_3 + 2a_4 - \cdots + (-1)^i 2a_i + (-1)^{i+1}(2a_{i+1} - x) \\
&= 2a_2 - 2a_3 + 2a_4 - \cdots + (-1)^i 2a_i + (-1)^{i+1} 2a_{i+1} + (-1)^{i+2} x).
\end{aligned}
$$

Thus the equality (4) holds. $\square$

Note that these folding maps do not depend on any MV assignment. They only tell us where points go after the creases are folded, regardless of the MV assignment.

6.2. Proof of Lemma 7

Lemma 7. *Let O_1, O_2 be two non-identical, exhaustive sequences of crimps performed on a one-dimensional crease pattern C. Then every crimp that is in O_1 is also in O_2.*

PROOF. Our proof is by contradiction. Assume to the contrary that there is a crimp in O_1 that is not in O_2. Let Z_1 be the first crimp in the ordered sequence O_1 that does not occur in O_2. Let C_1 be the crease sequence produced by all crimps in O_1 prior to Z_1 (and so Z_1 is a crimp to be performed

23

on C_1), and let $\alpha = (a_1, a_2, \ldots, a_k)$ be the crimpable sequence involved in the crimp Z_1. Let E_2 be the end crease sequence produced by O_2 (i.e., E_2 contains no crimpable sequences). Refer to Figure 7. We distinguish several cases, depending on whether one or more creases from the crease sequence α appear in E_2 or not.
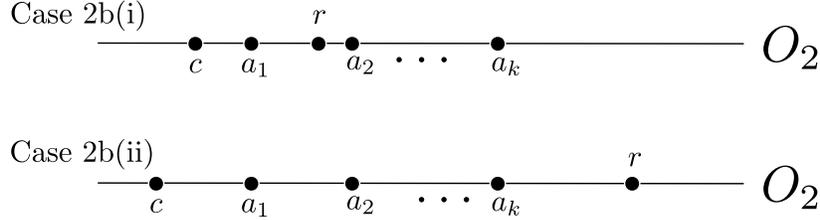

Figure 7: Case 1: all creases from α occur in E_2.

Before discussing these cases, we introduce some terminology and a new lemma that will be used in our case analysis. Define the <u>interval distance</u> of the crimpable sequence α to be $|a_2 - a_1|$ (which by definition equals $|a_{i+1} - a_i|$ for any $i = 1, \ldots, k-1$). For ease of presentation, we also define the <u>interval distance of a crimp</u> to be equal to the interval distance of the crease sequence involved in the crimp.

Lemma 11. *Let O_1 and O_2 be two sequences of crimps performed on a crease pattern $C = (a_0, a_1, \ldots, a_n)$. Let Q_1 and Q_2 be the crease sequences produced by O_1 and O_2, respectively. If a_i and a_j are adjacent creases in both Q_1 and Q_2, for some $i, j \in [0, n]$, then the distance $|a_j - a_i|$ is the same in Q_1 and Q_2.*

PROOF. Because Q_1 (Q_2) is the result of multiple crimps in O_1 (O_2), the creases a_i and a_j might not have started out in C as being adjacent. Let $d = |a_j - a_i|$ be the original distance between these two creases in C. If there are no creases between a_i and a_j in C (i.e, $j = i + 1$), then a_i and a_j have not been involved in any crimp in O_1 (O_2), therefore the distance between a_i and a_j in Q_1 (Q_2) is also d.

Suppose now that there is a sequence of creases $b_1, \ldots, b_k$ between a_i and a_j in C. Because a_i and a_j are adjacent in Q_1 (Q_2), the creases b_j, $j = 1, \ldots, k$ must have been removed by crimps in O_1 (O_2). Let $B = (a_i, b_1, \ldots, b_k)$. The folding map (4) shows us that no matter what crimps in O_1 or O_2 removed the creases b_j, the resulting distance between a_i and a_{i+1} will be

$$|\sigma_{B,a_i}(a_j) - a_i| = |2b_1 - 2b_2 + \cdots + (-1)^k 2b_k + (-1)^{k+1} a_j - a_i|.$$

24

Therefore the distance between a_i and a_j is the same in Q_1 and Q_2. $\qquad \square$

We now turn to our case analysis on the intersection between the crease sequence α and the end sequence E_2.

Case 1: All creases from α occur in E_2: $a_1, ..., a_k \in E_2$. Assume first that $a_1, ..., a_k$ are consecutive in E_2. (See Figure 7.) By Lemma 11, the distance between a_i and a_{i+1} in E_2 is equal to the distance between a_i and a_{i+1} in C_1, for $i = 1, ..., k-1$. (Recall that C_1 is the crease sequence produced by all crimps in O_1 prior to Z_1.) Because E_2 is an end sequence, we can assume without loss of generality that distances between adjacent creases from the left end of the paper up to a_1 are monotonically increasing (otherwise, we can reflect the paper about the left end and reverse the labeling of the α sequence, so that this assumption holds).

In the crease sequence E_2, let x_2 be the distance from a_1 to the crease b_2 on its left (note that b_2 may be the left end of the paper, if no other creases exist to the left of a_1). By our assumption, $x_2 \leq a_2 - a_1$. In C_1 however, the distance x_1 from a_1 to the crease b_1 on its left (which again could be the left end of the paper) is strictly greater than $a_2 - a_1$. (This is a necessary condition for the crimp Z_1 to be possible, or simply by definition of a crimp.) So the inequalities $x_1 > a_2 - a_1 \geq x_2$ hold, therefore $x_1 > x_2$. By Lemma 1, the interval to the left of a_1 in the initial crease sequence C is no longer than x_2 (because it may only increase or stay the same in size with each crimp). This implies that some crimp Z must have occurred in O_1 prior to Z_1 that lengthened the interval to the left of a_1 from some size below x_1 to size x_1. Note that the crimp Z could not occur in O_2, because the interval to the left of a_1 in E_2 is shorter than the one produced by the crimp Z. This contradicts our choice of Z_1 (because Z comes before Z_1 in the sequence O_1, and $Z \notin O_2$).

Assume now that there exists an i such that a_i and a_{i+1} are not consecutive in E_2. Let c be a crease in E_2 between a_i and a_{i+1}. Since $c \notin \alpha$, some crimp Z' must have occurred in O_1 prior to Z_1 to remove c. Now note that $Z' \notin O_2$, because $c \in E_2$ and Z' removes c. This again contradicts our choice of Z_1, because $Z' \notin O_2$ occurs before Z_1.

Case 2: There exists an i such that $a_i \notin E_2$. Let i be such that a_i is the first crease from among all creases in the sequence α to be removed by a crimp $Z_2 \in O_2$ (thus $a_i \notin E_2$). Consider the crease sequence C_2 produced by all crimps in O_2 prior to Z_2 (and so Z_2 is a crimp to be performed on C_2). Note that all creases $a_1, a_2, \ldots, a_k$ are in C_2, because a_i is the first of these creases to be removed, and C_2 is the sequence prior to removing a_i.

25

Because at least two adjacent creases are removed by a crimp, there must have been a neighbor c of a_i that was removed by Z_2 along with a_i (by the definition of a crimp).
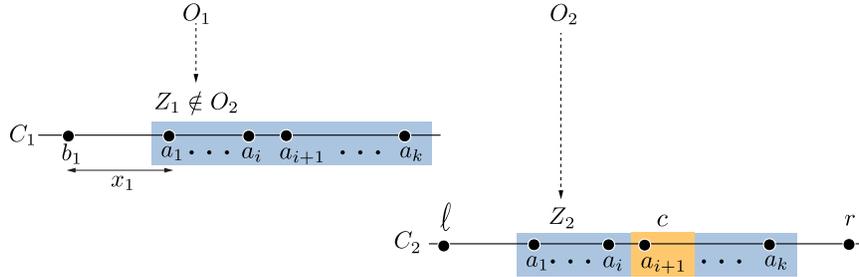

Figure 8: Case 2a: a_i and c removed by the crimp Z_2.

Case 2a: $c = a_{i-1}$ or $c = a_{i+1}$. (See Figure 8). Assume without loss of generality that $c = a_{i+1}$. By Lemma 11, the interval distance $d = a_{i+1} - a_i$ of Z_1 equals the interval distance $a_{i+1} - a_i$ of Z_2. By the definition of a crimp, the intervals immediately to the left/right of the creases involved in Z_2 must both be strictly longer than $a_{i+1} - a_i$.

Let r be the crease in C_2 immediately to the right of the creases involved in Z_2. We argue that r lies to the right of a_k. Otherwise, if $\exists j > i$ such that r lies between a_j and a_{j+1}, then

$$r - a_j \le d = a_{i+1} - a_i,$$

because by Lemma 1 the interval to the right of a_j can only increase or stay the same in size until it reaches the limit $a_{j+1} - a_j = d$ in C_1. This contradicts the requirement that $r - a_j > d$ for the crimp Z_2 to take place. Hence r lies to the right of a_k, and similarly the crease ℓ immediately to the left of the creases involved in Z_2 lies to the left of a_1. Similar arguments show that there are no creases in C_2 between a_j and a_{j+1}, for each $j = 1, 2, \ldots, k - 1$.

Next we show that there are no creases in C_2 between a_k and r, or between between ℓ and a_1. This would imply that Z_2 is equivalent to Z_1, contradicting our assumption that $Z_1 \notin O_2$.

Assume to the contrary that there exists a crease w between a_k and r in C_2. Because r lies immediately to the right of the creases involved in Z_2, we have that $w - a_k = d$ and $r - w > d$ (because Z_2 is a crimp). Note however that w is not part of the sequence α in C_1, therefore w must have been removed by a crimp $Z_w \in O_1$ prior to Z_1. The crimp Z_w cannot occur in O_2, because Z_w removes w but not a_i, and in O_2 the crease w gets removed

along with a_i by the crimp Z_2. This contradicts our choice of Z_1 (because $Z_w \notin O_2$ occurs prior to Z_1). Similar arguments show that there are no creases between ℓ and a_1 in C_2.

So the crimp $Z_2 \in O_2$ involves only $a_1, a_2, ..., a_k$ and we have that Z_1 is the same as Z_2. This contradicts our assumption that $Z_1 \notin O_2$.
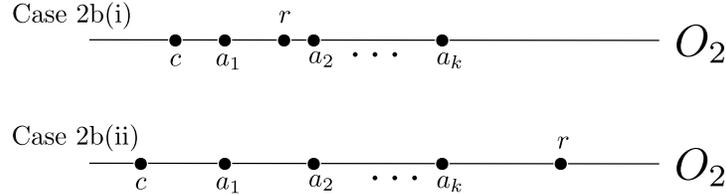
Case 2b(i)

Case 2b(ii)

Figure 9: Case 2b, depending on the location of crease r immediately to the right of the creases involved in $Z_2 \in O_2$.

Case 2b: $c \neq a_{i-1}$ and $c \neq a_{i+1}$. Assume first that $i > 1$ and $c \in (a_{i-1}, a_i)$, or $i < k$ and $c \in (a_i, a_{i+1})$. Then in O_1 some crimp Z_c prior to Z_1 must have removed c. Note that $Z_c \notin O_2$, because Z_c removes c but does not remove a_i, and in O_2 the crease c gets removed along with a_i by Z_2. This contradicts our choice of Z_1 as being the first crimp in O_1 that does not occur in O_2.

The only case left is $a_i = a_1$ and c lies left of a_1 (or the symmetric case $a_i = a_k$ and c lies right of a_k, which is similar). As in Case 2a, let r be the crease immediately to the right of the creases involved in Z_2. Then r may lie either in the interval $(a_1, a_2]$, or strictly to the right of a_2.

Assume first that $r \in (a_1, a_2]$ (see Figure 9 top). Then $r - a_1 \leq d$, because by Lemma 1 the interval immediately to the right of a_1 can only increase or stay the same in size with each crimp, and the limit $d = |a_2 - a_1|$ is reached when a_2 becomes adjacent to a_1 (without being separated by r). This along with the fact that Z_2 is a crimp implies that $a_1 - c < r - a_1 \leq d$. This means that $a_1 - c < d < a_1 - b$, where b is the crease immediately to the left of a_1 in C_1. Then c must have been removed in O_1 by a crimp Z'_c prior to Z_1. Now note that Z'_c eliminates c but does not eliminate a_1, and Z_2 eliminates both c and a_1. This implies that $Z'_c \notin O_2$. contradicting our choice of Z_1.

Assume now that r is strictly to the right of a_2 (see Figure 9 bottom). Arguments identical to those used in case 2a show that r lies strictly to the right of a_k, and no creases exist in C_2 between a_j and a_{j+1}, or between a_k and r. It follows that $a_1, a_2, ..., a_k$ are all involved in the crimp $Z_2 \in O_2$, which is identical to $Z_1 \in O_1$. This contradicts our assumption that $Z_1 \notin O_2$. $\square$