Complexity of pleat folding

Tsuyoshi Ito* Masashi Kiyomi† Shinji Imahori‡ Ryuhei Uehara§

Abstract

We introduce a new origami problem about *pleat foldings*. For a given assignment of n creases of mountains and valleys, we make a strip of paper well-creased according to the assignment at regular intervals. We use simple folding as a basic operation. More precisely, we assume that (1) paper has 0 thickness and some layers beneath a crease can be folded simultaneously, (2) each folded state is flat, and (3) the paper is rigid except at the n given creases. We also assume that each crease remembers its last folded state made at the crease. We aim to find efficient ways of folding a given mountain-valley assignment in this model. We call this problem *unit folding problem* for general patterns, and *pleat folding problem* when the mountain-valley assignment is "$MVMVMV\cdots$." The complexity is measured by the number of foldings and the cost of unfoldings is ignored. Trivially, we have an upper bound n and a lower bound $\log(n+1)$. We first give some nontrivial upper bounds: (a) any mountain-valley assignment can be made by $\lfloor \frac{n}{2} \rfloor + \lceil \log(n+1) \rceil$ foldings, and (b) a pleat folding can be made by $O(n^\epsilon)$ foldings for any $\epsilon > 0$. Next, we also give a nontrivial lower bound: (c) almost all mountain-valley assignments require $\Omega\left(\frac{n}{\log n}\right)$ foldings. The results (b) and (c) imply that a pleat folding is easy in the unit folding problem.

1 Introduction

Origami has recently attracted much attention as mathematics and as theoretical computer science [3]. In the computational origami, the best studied problem is the characterization of flat foldable crease patterns. When the paper and crease pattern are orthogonal, the problem is called the map folding problem, and the problem has been well studied by Arkin et al. [2]. Roughly speaking, (1) the problem in 1D can be solved in linear time; it can be determined in $O(n)$ time whether or not a given strip of paper with n creases is flat foldable, and (2) the problem in 2D is NP-complete; to determine whether or not a given zig-zag form paper with n creases is flat foldable is NP-complete (see [2] for the details).

In this paper, we deal with a simpler problem. The input of our problem is a strip of paper of length $n+1$ with n desired creases, and we assume that these creases are placed at regular intervals on the paper. That is, we assume that the paper corresponds to an interval $[0..n+1]$, n creases are placed at each integer point i with $1 \leq i \leq n$, and each integer point is assigned to M (mountain) or V (valley) which is the target folded state. We aim to fold the paper and make it "well-creased" according to the assignment. More precisely, each integer point *memorizes* its last folded state (M or V) made at the point, and we have to make the creases' folded states fit the given assignment.

A typical example is *pleat folding* that is one of basic tools for origami design (Figure 1; the angel is made from just one square paper without any cutting). The mountain-valley assignment for the pleat folding is in the form "$MVMV\cdots$." From the industrial point of view, pleat folding has many applications including clothes, curtains, etc.

We employ *simple folding* defined in [2] (see also [3, Section 14.1]). More precisely, our origami model is as follows; the paper has 0 thickness, some layers beneath a crease can be folded simultaneously, each folded state is flat, and the paper is rigid except at given n creases. Clearly, this folding problem has a trivial solution (or upper bound of the number of foldings); we can fold any mountain-valley assignment by just folding n creases independently. On the other hand, we have a trivial lower bound of the number of foldings; we have to fold at least $\lceil \log_2(n+1) \rceil$ times to make n creases (by folding repeatedly at the center point of the paper).

For a given string s of length n in $\{M,V\}^n$, the *unit folding problem* is to obtain the well-creased paper following the mountain-valley assignment s. We call it *pleat folding problem* if the mountain-valley assignment is "$MVMV\cdots$." The *complexity* is measured by the number of foldings, and the number of unfoldings is ignored. We first show the following nontrivial upper bound of the number of foldings to solve any unit folding problem:

Theorem 1 *The unit folding problem of n creases can be solved by $\lfloor \frac{1}{2}n \rfloor + \lceil \log(n+1) \rceil$ foldings for any*

*School of Computer Science, McGill University, tsuyoshi@cs.mcgill.ca

†School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), mkiyomi@jaist.ac.jp

‡Graduate School of Information Science and Technology, The University of Tokyo, imahori@mist.i.u-tokyo.ac.jp

§School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), uehara@jaist.ac.jp

Figure 1: An "angel" designed and folded by Takashi Hojyo on the cover of [1]. (`http://www.origami.gr.jp/Magazine/Index/103-108.html`)

mountain-valley assignment[1].

One may think that the pleat folding problem is the most difficult in the unit folding problems. However, this is not the case. We show an efficient way to obtain the pleat foldings.

Theorem 2 *For any positive real number $\epsilon > 0$, the pleat folding problem of n creases can be solved by $O(n^\epsilon)$ foldings for sufficiently large n.*

On the other hand, we can show a lower bound of the general unit folding problem.

Theorem 3 *For the unit folding problem, almost all mountain-valley assignments with n creases require $\Omega\left(\frac{n}{\log n}\right)$ foldings.*

Theorems 2 and 3 imply that the pleat folding problem is not the most difficult unit folding problem.

In origami, "unfolding" is much easier than "folding." Hence we ignore the cost of unfoldings in our model. However, even if we count the number of unfoldings in addition to foldings, all the results in this paper hold with small changes within constant factors.

[1]In this paper, we assume the base of logarithm is two unless specified otherwise.

2 Models of foldings and unfoldings

The input to a unit folding problem consists of a string of length n over an alphabet $\{M, V\}$. The ith letter (M or V) corresponds to the *ith assignment* at the ith crease which is placed at the integer point i in the interval $[0..n+1]$. In general, we have several ways when we fold the paper at the point i. The following simple fold models are proposed by Arkin et al. [2] (see also [3, p. 225]):

One-layer simple fold model: We always fold the top layer of paper.

All-layers simple fold model: We simultaneously fold all layers of paper under the crease point.

Some-layers simple fold model: We fold some layers beneath the crease point. (Note that we have to valley fold inside consecutive layers.)

We measure the complexity of a folding algorithm by the number of simple foldings made by it, and ignore the cost of unfoldings. In the one-layer simple fold model, we always fold exactly one crease. Thus, we cannot improve the trivial upper bound n for the unit folding problem, and hence we have nothing to do. Therefore, hereafter, we only consider the all-layers simple folding and some-layers simple folding. We additionally introduce the following three unfolding models, which define the allowable unfolding operations.

All-unfold model: Once we decide to unfold, the paper is unfolded completely. That is, we have the paper corresponding to the interval $[0..n+1]$ again.

Reverse-unfold model: We can rewind any number of the last folding steps as far as we can. This model is useful to consider some upper and lower bounds.

General-unfold model: For a folded state s, we can obtain another folded state t by the one general unfolding operation if s can be obtained from t by consecutive some-layers simple foldings.

The models will be used to discuss both of the upper and lower bounds of the algorithms. Of course, the general-unfold model is the most natural and the strongest model. This paper mainly studies the first two unfold models, which are natural restrictions of the general-unfold model.

Without loss of generality, we assume that the leftmost point of any folded paper is always placed at point 0. Each point on the paper has its own label i that is given by the initial position in $[1..n]$.

A folding algorithm will be called *end-free* if it can be applied on the paper with n creases even if we extend both endpoints of the paper to infinity. Precisely, the paper can be corresponds to an interval $[-N..N]$ for any large $N \gg n$. This property will be required when we use the algorithm recursively.

3 Upper bounds

We first prove Theorem 1, which is the basic tool of this paper.

Proof. (of Theorem 1) To simplify the proof, we first suppose that $n = 2^k - 1$. The strategy is simple; (1) fold the paper in half k times, (2) unfold all, and (3) fix each crease if it is not folded correctly.

On step (1), the folding direction is determined by the majority at the point. That is, we see the point i, which is the center of the current paper, check the all directions of the desired creases on the point i, and take majority. (We note that half creases are turned over from the original direction, but fixing this is easy; the creases are alternately turned over and that can be checked from 1 to n.) Hence step (1) makes at least half of creases to be folded correctly. Thus in step (3), the number of foldings is bounded above by $\lfloor \frac{n}{2} \rfloor$ (the rounding comes from the first folding that always succeeds). Therefore we have the theorem.

When $n \neq 2^k - 1$, we just add an imaginary paper to make the length of the paper $n' = 2^k - 1$ with $\min_k n < 2^k - 1$, and apply the algorithm on this paper of length n'. □

We note that (a) the algorithm in the proof is end-free, (b) it works in the all-layers simple fold model and hence some-layers simple fold model, and (c) it works in all the unfold models we described since all unfoldings in steps (2) and (3) can be done by all-unfoldings.

We call a unit folding problem *mountain folding problem* if the mountain-valley assignment is "$MMM\cdots$." We next show a simple proposition which gives us a strong intuition to consider the pleat folding.

Proposition 4 *Let A' be any algorithm for the mountain folding problem in any model, and $f_0(n)$ be the number of foldings by A' to make n mountain foldings. Then we can solve the pleat folding problem in the model with $f_0(\lfloor n/2 \rfloor) + f_0(\lceil n/2 \rceil)$ foldings.*

Proof. For the mountain-valley assignment "$MVMV\cdots$," we first fold all $f_0(\lceil n/2 \rceil)$ mountains by A'. Next we reverse the paper and fold all $f_0(\lfloor n/2 \rfloor)$ valleys by A' again. □

Now we turn to the upper bounds of the number of foldings for solving the pleat folding problem. First, we focus on the all-layers simple fold and reverse-unfold (or all-unfold) model. In this weak model, we show a weaker upper bound. By Proposition 4, we can focus on the mountain folding problem instead of the pleat folding problem. This will help us to have an intuition.

Lemma 5 *The mountain folding problem of n creases can be solved by $O(n^{\log \phi}) \sim O(n^{0.69})$ foldings in the all-layers simple fold and all-unfold model, where $\phi = \frac{1+\sqrt{5}}{2}$.*

Proof. To simplify the proof, we suppose that $n = 2^k$. The basic strategy is similar to the algorithm in the proof of Theorem 1. In step (0), we fold the paper at point 1 and make the length of the paper 2^k. The next two steps (1) and (2) are the same; we fold the paper in half k times and unfold completely. Now we focus on the mountain-valley pattern made by the two steps (1) and (2). First, we fold the paper in half, and obtain the correct crease at the center point. Second, we fold the paper in half, and obtain one correct crease, and one incorrect crease. Third, we fold the paper in half, and obtain two correct creases, and two incorrect creases. In ith folding with $i > 1$, we obtain 2^{i-2} correct creases and 2^{i-2} incorrect creases that appear on the paper alternately. The 2^{i-2} incorrect creases are placed at regular intervals. Thus, we can use our end-free algorithm recursively on these 2^{i-2} creases in step (3). Let $f_1(n)$ be the number of the foldings of this recursive algorithm. It is easy to see that $f_1(1) = 1$, $f_1(2) = 2$, $f_1(3) = 3$, and $f_1(4) = 4$. By the above observation, we have

$$f_1(2^k) = 1+k+f_1(2^{k-2})+f_1(2^{k-3})+\cdots+f_1(2)+f_1(1).$$

By the fact that $f_1(2^k) - f_1(2^{k-1})$ equals $f_1(2^{k-2})+1$, we have the following relationship.

$$(f_1(2^k) + 1) = (f_1(2^{k-1}) + 1) + (f_1(2^{k-2}) + 1).$$

Thus $(f_1(2^k) + 1)$ gives the Fibonacci numbers for k. (We define the Fibonacci numbers F_i by $F_0 = 0$, $F_1 = 1$ and $F_i = F_{i-1}+F_{i-2}$ for $i > 1$. It is well known that $F_i = \frac{\phi^n - (-\phi)^{-n}}{\sqrt{5}} = O(\phi^n)$, where $\phi = \frac{1+\sqrt{5}}{2}$.) Using initial conditions $f_1(2^0) = 1$, $f_1(2^1) = 2$, and $f_1(2^2) = 4$, we have $f_1(2^k)+1 = F_{k+3}$. Hence we have $f_1(n) = f_1(2^k) = O(\phi^k) = O\left(\phi^{\log n}\right) = O\left(n^{\log \phi}\right)$, which completes the proof. □

Next, we show that the number of foldings can be reduced by using the stronger model.

Lemma 6 *For any fixed $\epsilon > 0$, the mountain folding problem of n creases can be solved by $O(n^\epsilon)$ foldings for sufficiently large n in the some-layers simple fold and reverse-unfold models.*

Proof. In the algorithm of the proof of Lemma 5, "folding in half" is the basic operation. We change

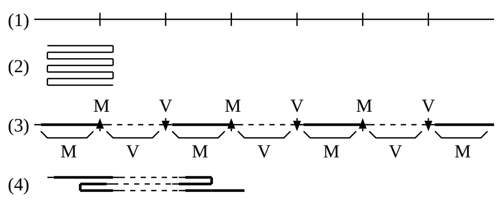

Figure 2: Mountain folding by zig-zag method

this to "folding in zig-zag form of p segments" (Figure 2). Roughly speaking, the algorithm first folds the paper into zig-zag form of p segments of the same length by making $p-1$ alternating foldings at regular intervals (in Figure 2(1) and (2) for $p=7$). It recursively calls itself with the folded paper of length n/p. After the recursive call, it unfolds at the folded $p-1$ crease points. Then around $p/2$ segments have been mountain folded, and $p/2$ segments have been valley folded (in Figure 2(3), thick lines and dashed lines indicate the segments of the mountain and valley foldings, respectively). The algorithm then piles up the valley segments exactly by folding at the center points of the mountain segments (see Figure 2(4)). The algorithm again calls itself recursively with the local area of length n/p that consists of all and only the valley segments. By the recursive call, half of the valley segments are corrected, and still half of them are not corrected. (The creases of the center points of the mountain segments are fixed here.) Hence the algorithm again piles up the valley segments and recursively calls itself again. The algorithm repeats this process and finally obtain the desired mountain folded state.

The algorithm runs in the some-layers simple fold and reverse-unfold model, and the correctness of the algorithm is easy. Hence we analyze its complexity. To simplify the proof, we suppose that $n=p^k-1$ and $p=2^q-1$. (We let q be an arbitrary positive integer, and hence p is odd and n is even.) Let $f_2(n)$ be the number of foldings made by the algorithm. By a careful analysis, we have $f_2(0)=0$, $f_2(x)\le x$ and

$$f_2(n)=q\cdot f\left(\frac{n+1}{p}-1\right)+p-2(q-2)+\frac{3(p+1)}{2}\sum_{\ell=2}^{q-1}\frac{1}{2^{\ell-1}}.$$

By a simple calculation, we obtain

$$\begin{aligned} f_2(n) &< q\cdot f\left(\frac{n+1}{p}-1\right)+\frac{5p+11}{2} \\ &< (5p+11)q^{k-2}+p-1=O(p\cdot q^{k-2}). \end{aligned}$$

Since $q=\log_2(p+1)$ and $k=\log_p(n+1)$, we have $p\cdot q^{k-2}\sim p\cdot n^{\frac{\log\log p}{\log p}}$. Hence, letting p be a sufficiently large constant, we have $\frac{\log\log p}{\log p}<\epsilon$ and $p\cdot q^{k-2}=O(n^\epsilon)$, which completes the proof. □

By Proposition 4 and Lemma 6, we have Theorem 2.

4 Lower bounds

We have to clarify the unfold model to state a lower bound. In fact, Theorem 3 holds for both of the all-unfold model and the reverse-unfold model, but we have no results about general-unfold model.

Lemma 7 *All but $o(2^n)$ mountain-valley assignments of n creases require $\Omega\left(\frac{n}{\log n}\right)$ foldings in all of the three folding models and both of the all-unfold model and the reverse-unfold model.*

Proof. The lemma is obtained by a simple counting argument. Since the some-layers simple fold model is at least as powerful as the one-layer and the all-layers simple fold models, we only consider the some-layers simple fold model. Similarly, we only discuss the reverse-unfold model. Suppose that we make at most k foldings. At each folding, there are two choices for the direction of folding (mountain or valley). There are at most n choices for the set of positions of the folding by considering the bottom-most layer of a valley folding or the top-most layer of a mountain folding. There are at most $n+1$ choices for how many unfolding steps we rewind just after the last folding. In total, the number of the possible mountain-valley assignments made by at most k foldings is bounded by $\sum_{i=0}^{k}(2n(n+1))^i\le(2n(n+1)+1)^k<(2(n+1)^2)^k$. If we let $k=\lfloor\frac{n}{4(1+\log_2(n+1))}\rfloor$, then $(2(n+1)^2)^k<2^{n/2}=o(2^n)$. Therefore all but $o(2^n)$ mountain-valley assignments require at least $\lfloor\frac{n}{4(1+\log_2(n+1))}\rfloor=\Omega\left(\frac{n}{\log n}\right)$ foldings in the reverse-unfold model (hence also in the all-unfold model). □

Lemma 7 implies Theorem 3. Lemma 7 also implies that if we choose a mountain-valley assignment of n creases uniformly at random, the expected number of foldings required to make it is $\Omega\left(\frac{n}{\log n}\right)$ in the all-unfold model and the reverse-unfold model.

5 Conclusion

There are several open problems.

By Theorem 3, a random mountain-valley assignment requires $\Omega\left(\frac{n}{\log n}\right)$ foldings with high probability. Finding a specific assignment which requires $\Omega\left(\frac{n}{\log n}\right)$ foldings will give some new insight on which assignments are easy to fold and which are not. Is there any mountain-valley assignment that requires $\Omega(n)$ foldings?

Lemma 7 deals with general mountain-valley assignments. When we focus on the pleat folding problem, is there a nontrivial lower bound for this specific problem? Especially, can we make pleat foldings in poly-log foldings?

From the practical point of view, developing better algorithm for the pleat folding problem is interesting, especially, in general-unfold model. Up to now, we have no idea for dealing with general-unfold model.

Acknowledgments

The authors thank to Takashi Hojyo who gives a permission to use his "angel" for our presentation of this research. The authors are grateful to Erik Demaine, Joseph O'Rourke, and Ron Graham for their helpful comment and suggestion at CCCG 2008.

References

[1] *Origami Tanteidan*, Number 107. Japan Origami Academic Society, 2008.

[2] E. Arkin, M. Bender, E. Demaine, M. Demaine, J. Mitchell, S. Sethia, and S. Skiena. When Can You Fold a Map? *Computational Geometry: Theory and Applications*, 29(1):23–46, 2004.

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