Algorithmic Folding Complexity*

Jean Cardinal[1], Erik D. Demaine[2], Martin L. Demaine[2], Shinji Imahori[3], Tsuyoshi Ito[4], Masashi Kiyomi[5], Stefan Langerman[1], Ryuhei Uehara[5], Takeaki Uno[6]

[1] Département d'Informatique, Université Libre de Bruxelles, CP 212, B-1050 Brussels, Belgium. e-mail: {jcardin,stefan.langerman}@ulb.ac.be

[2] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. e-mail: {edemaine,mdemaine}@mit.edu

[3] Department of Computational Science and Engineering, Graduate School of Engineering, Nagoya University, Nagoya 464-8603, Japan. e-mail: imahori@na.cse.nagoya-u.ac.jp

[4] Institute for Quantum Computing and School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. e-mail: tsuyoshi@iqc.ca

[5] School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan. e-mail: {mkiyomi,uehara}@jaist.ac.jp

[6] National Institute of Informatics, Chiyoda-ku, Tokyo 101-8430, Japan. e-mail: uno@nii.ac.jp

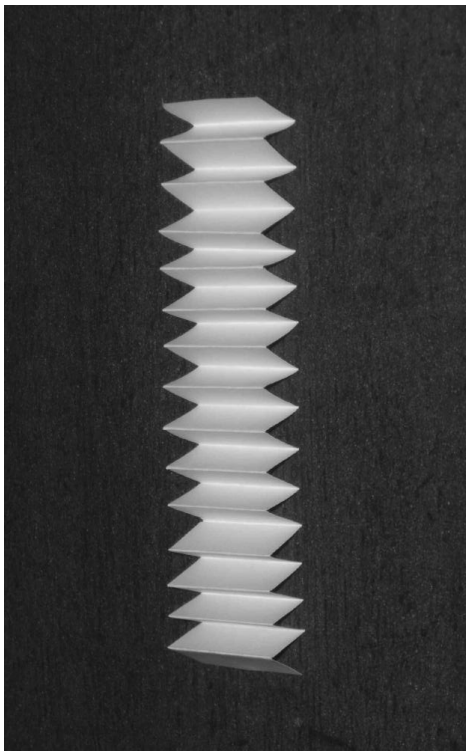
Abstract. How do we most quickly fold a paper strip (modeled as a line) to obtain a desired mountain-valley pattern of equidistant creases (viewed as a binary string)? Define the *folding complexity* of a mountain-valley string as the minimum number of simple folds required to construct it. We first show that the folding complexity of a length-n *uniform* string (all mountains or all valleys), and hence of a length-n *pleat* (alternating mountain/valley), is $O(\lg^2 n)$. We also show that a lower bound of the complexity of the problems is $\Omega(\lg^2 n / \lg\lg n)$. Next we show that almost all mountain-valley patterns require $\Omega(n/\lg n)$ folds, which means that the uniform and pleat foldings are relatively easy problems. We also give a general algorithm for folding an arbitrary sequence of length n in $O(n/\lg n)$ folds, meeting the lower bound up to a constant factor.

Key words. Algorithm design, Origami

1. Introduction

What is the best way to fold an origami model? Origamists around the world struggle with this problem daily, searching for clever, more accurate, or faster folding sequences and techniques. Many advanced origami models require substantial *precreasing* of a prescribed mountain-valley pattern (getting each crease folded slightly in the correct direction), and then folding all the creases at once. For example, in Brian Chan's instructional video for folding the MIT seal *Mens et Manus* in "three easy steps" [4], he spends about three hours precreasing, then three hours folding those creases, and then four hours of artistic folding. The precreasing component is particularly tedious, leading us to a natural algorithmic

* Parts of this paper were presented at European Workshop on Computational Geometry (EuroCG 2009) [9] and International Symposium on Algorithms and Computation (ISAAC 2009) [3]. An extended abstract of [3] was presented at Japan Conference on Computational Geometry and Graphs (JCCGG 2009).



(a) How fast can we fold this?

(b) An origami angel with many pleats, folded by Takashi Hojyo (reproduced with his kind permission).

Fig. 1. Pleats.

problem of *optimal precreasing*: what is the fastest way to precrease a given mountain-valley pattern? Although the standard method of "fold one crease, unfold, repeat" is usually the most accurate, it might be possible to fold the paper along some of the desired creases to bring several other desired creases into alignment, and thereby precrease them all at once.

We focus here on a simple kind of one-dimensional precreasing, where the piece of paper is a long rectangular strip, which can be abstracted into a line segment, and the creases uniformly subdivide the strip. A mountain-valley string is then simply a string over the alphabet $\{M, V\}$ (M for mountain, V for valley). Of particular interest in origami is the *pleat*, which alternates $MVMVMV\cdots$; see Fig. 1.

Related work. About n different mountain-valley strings of length n can be folded using the absolute minimum number of folds, $\lceil \lg(n+1) \rceil$. These strings are called *paper folding sequences* and have been studied much earlier [1,5,12]. Among the paper folding sequences, the simplest one generates the *dragon curve*, which was discussed by Heighway in the 1960s (see [8] for further details).

Model. We follow the folding and unfolding model based on the simple-fold model of Arkin et al. [2] (see also [6, p. 225]).

The paper strip is a one-dimensional line with creases at every integer position. We are allowed to fold only at those positions, possibly many times, and the direction of a crease

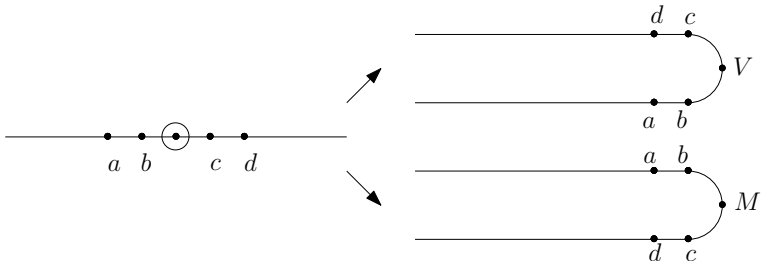


Fig. 2. Folding a mountain or a valley.

(in $\{M, V\}$) at the end of the algorithm is the one that was folded last. The paper is rigid except the integer positions. The goal is to achieve a particular string of Ms and Vs at the end. Folding has the effect of superimposing the layers of paper on the right and left of the crease; see Fig. 2. The paper has zero thickness, and thus an arbitrary number of layers can be folded simultaneously. Naturally, when many layers are folded, every other layer is reversed, and the direction of the crease for these layers is flipped. Finally we can, at any step, unfold the paper at zero cost, in the reverse order in which it was folded, and this does not change the directions of the creases. The complexity of the algorithm is the number of folds. (We do not count unfold operations, though doing so would increase the cost by only a constant factor since the number of unfolds is bounded above by the number of folds.)

Several variants of this model arise from varying the allowed folding and unfolding operations. We do not consider a model where we can only fold one layer at a time, since in this model we obviously need n folds for any pattern. We distinguish between two models for the folding operation.

All-layers fold model: We simultaneously fold all layers of paper under the crease point.
Some-layers fold model: We simultaneously fold the k successive layers immediately beneath the crease point, for any desired number k.

We also introduce the following three alternatives for the allowed unfolding operations.

All-unfold model: Once we decide to unfold, the paper is unfolded completely.
Reverse-unfold model: We can rewind any number of the last folds as far as we want.
General-unfold model: For a folded state s, we can obtain another folded state t by one general unfolding operation, provided s can be obtained from t by consecutive some-layers simple foldings.

The general-unfold model is the most realistic, but our algorithms require only the power of the reverse-unfold model.

A folding algorithm is *end-free* if it can be applied to an infinite paper strip, viewed as a line instead of a line segment, with n equally spaced creases along it, without creasing anywhere else. We discuss end-free algorithms only, which is crucial for allowing us to apply an algorithm recursively.

Our results. In this paper, we develop surprisingly efficient algorithms for precreasing a mountain-valley string, especially the pleat.

First, we show how to fold a uniform mountain-valley string $MMM\cdots$ of n mountain creases using just $O(\lg^2 n)$ simple fold operations. By executing this algorithm twice, we obtain the same bound for pleat. This folding is exponentially faster than the standard

folding. From a complexity-theoretic perspective, this is the first polynomial-time algorithm for pleat folding, because the only input is the number n. We also show a lower bound $\Omega(\lg^2 n/\lg\lg n)$ for these strings. Hence the $O(\lg^2 n)$ folding algorithm is quite close to the lower bound.

Second, we show how to fold an arbitrary mountain-valley string of n creases using just $O(n/\lg n)$ folds. This algorithm is the first to beat the straightforward $n-1$ upper bound by more than a constant factor, and is asymptotically optimal up to a constant factor; we show that almost all mountain-valley strings require $\Theta(n/\lg n)$ foldings. We effectively exploit that every string has some redundancy in it, similar to how the Lempel-Ziv algorithm can compress any string into $O(n/\lg n)$ block pointers [10,11].

Unfortunately, our algorithms are not about to revolutionize pleat folding or other practical paper precreasing, because they assume ideal zero-thickness paper. In reality, folding more than a few layers of paper leads to some inaccuracy in the creases, called *creep* in origami circles, and our algorithms require folding through $\Theta(n)$ layers. Nonetheless, our results lead the way for the development of practical algorithms that limit the number k of layers that can be folded through simultaneously, with speed increasing as k grows.

From an information-theoretic perspective, paper folding offers an intriguing new definition of the algorithmic complexity of a binary string. The *folding complexity* of a mountain-valley string is the minimum number of folds needed to be constructed. Similar to how Kolmogorov complexity compresses a string down to instructions for a Turing machine, folding complexity compresses a string down to instructions for an origamist. Unlike Kolmogorov complexity, however, folding complexity is computable, though its exact computational complexity remains open except for a trivial upper bound of EXPTIME. We lack a specific (deterministic) string whose folding complexity is asymptotically the maximum possible. (The pleat was an early candidate, now known to be far from the worst case.) Nonetheless, our results shed some light on the structure of this new measure.

2. Folding Pleat

Every mountain-valley pattern with n creases requires at least $\lceil\lg(n+1)\rceil$ folds, just to get all the creases folded in some direction. The purpose of this section is to show that the pleat pattern, while seemingly difficult to fold, has fairly close upper and lower bounds. But first we need to show that the lower bound can be met for any n (not just of the form 2^k-1):

Lemma 1. *A string of n equally spaced creases (unspecified as mountain or valley) can be folded (and not unfolded) using at most $1+\lceil\lg(n+1)\rceil$ simple folds in the end-free all-layers fold model.*

Proof. If the number of segments between creases, $n+1$, is 2^k for an integer k, then the folding is the obvious one: fold at the middle crease, k times. Otherwise, $n+1=2^k+r$ for some integers k and $0\le r<2^k$. Valley-folding at the rth crease would place the initial r segments on top of the remaining 2^k segments, at which point we could apply the power-of-2 algorithm. But to make the folding end-free, we first mountain-fold at the $\lfloor r/2\rfloor$th crease, so that these two creases form a zig-zag and the power-of-2 algorithm creases only where desired. □

Theorem 1. *The folding complexity of a uniform string and of a pleat of length n in the all-layers fold, all-unfold model is at most $\frac{3}{2}\lg^2 n$.*

Proof. Given an algorithm for folding a uniform string, we can apply it twice to obtain the pleat sequence. We therefore concentrate on finding an algorithm for folding the uniform sequence of n mountains. To simplify, we assume that $n = 2^k - 1$ for some positive integer k. That is, when we repeat folding at the middle crease $k-1$ times, we have a paper of length 1. In the following, we use some symbols to describe crease states; "M", "V", and "x" mean the mountain-folded, valley-folded, and not folded creases, respectively. These folded creases are viewed from the top of the paper, and if the creases are piled, papers of even depths are reversed, and hence their folded direction are flipped. After folding, "[" and "]" mean the left and right edges of the paper, that are folded creases. The folded creases on the edges are denoted by "+" after unfolding. They are mountain or valley folded; we do not mind the states and the algorithm fixes all creases labeled "+". For the symbol "M", "M^i" means i repetitions of it. The algorithm consists of three steps:

Step 0. Repeat folding at the middle crease $k-3$ times. After that, we have the pattern "[xxx]" of length 4 on the paper. Then mountain-fold three times and obtain "[MMM]". Next, unfold the paper and obtain the sequence "+MMM+VVV+MMM+VVV+MMM+VVV+$\cdots$."

Step 1. In order to pile all patterns "VVV" at the same place, repeat folding $k-3$ times at the middle crease in the "MMM" pattern which is the closest one to the center of the paper. (Note that this step requires $k-3$ foldings; since the "MMM" pattern closest to the center of the paper is not on the center exactly. Hence the algorithm first folds $k-4$ times, and the last one folding is done to fix the length to 8.) After that, we have the pattern "[M+VVV+M]" of length 8. Then mountain-fold five times at each place with label "V" or "+", and obtain "[MMMMMMM]." Unfold the paper and obtain the sequence "VM+MMMMMMM+MVVVVVM+MMMMMMM+MVVVVVM+M$\cdots$."

Step 2. Repeat Step 1; precisely, (1) pile all patterns "VVV$\cdots$V" at the same place by repeatedly folding at the middle crease in the pattern "MMM$\cdots$M" that is the closest to the center, and (2) mountain-fold at each place with label "V" or "+", and unfold the paper.

Step 1 is repeatedly performed for each $i = 2, 3, 4, \ldots, k-2$. Finally, when $i = k-2$, the number of consecutive patterns of "V" becomes one, fix it, and algorithm halts.

We regard Step 0 as the first phase. Then, in each phase $i = 1, 2, \ldots, k-2$, the algorithm first piles the consecutive "V"s pattern by $(k-i-2+1)$ (or $k-i-2$ when $i = 1, k-2$) foldings at center of the consecutive mountain pattern, and next folds $2i+1$ times to fix the creases labeled by "V" and "+". (Note that, among them, $2i-1$ creases (at irregular intervals) have label "V" and two outermost creases have label "+." For example, the pattern after piling is

$$\text{“}[\mathsf{M}^{31} + \mathsf{M}^{15}\mathsf{V}\mathsf{M}^{7}\mathsf{V}\mathsf{M}\mathsf{M}\mathsf{M}\mathsf{V}\mathsf{M}\mathsf{V}\mathsf{V}\mathsf{V}\mathsf{V}\mathsf{V}\mathsf{M}\mathsf{V}\mathsf{M}\mathsf{M}\mathsf{M}\mathsf{V}\mathsf{M}^{7}\mathsf{V}\mathsf{M}^{15} + \mathsf{M}^{31}]\text{”}$$

when $i = 6$.) Hence the total number of foldings is equal to $\sum_{i=1}^{k-2}((k-i-2+1)+2i+1)-2 = \sum_{i=1}^{k-2}(k+i)-2 = k(k-2)+(k-2)(k-1)/2-2 < \frac{3}{2}\lg^2 n$. □

We next turn to the lower bound for the folding complexity of a uniform string and of a pleat of length n, which is very close to the upper bound in Theorem 1. We first show the lower bound of a uniform string.

Theorem 2. *The folding complexity of a uniform string of length n in the some-layers fold, all-unfold model is at least* $\lg^2 n/4 \lg\lg n - o(\lg^2 n/4 \lg\lg n)$.

Proof. For each n, we suppose that an optimal algorithm folds $f(n)$ times to make n mountains. Then, by the proof of Theorem 1, $f(n) < 2\lg^2 n$. If $f(n_1) > f(n_2)$ for some pair of $n_1 < n_2$, regarding the paper of length n_1 as of length n_2 with virtual paper, we can improve $f(n_1)$ to $f(n_2)$, which is a contradiction. Hence $f(n)$ is a monotone non-decreasing function.

By using the argument of expected value, since the algorithm only folds $f(n)$ times to make n creases, it folds $n/f(n)$ creases at once at least one time. To pile the $n/f(n)$ creases, by Lemma 1, at least

$$\lg(n/f(n)) = \lg n - \lg f(n) > \lg n - \lg(2\lg^2 n) = \lg n - 2\lg\lg n - 1 \tag{1}$$

foldings are required. On the other hand, when the algorithm folds $n/f(n)$ creases at once, $n/2f(n)$ creases are valley-folded.

In the all-unfold model, we have to unfold to deal with these $n/2f(n)$ valley-folded creases. After unfolding, to mountain-fold these $n/2f(n)$ valleys, the algorithm has to fold $f(n/2f(n)) \geq f(n/4\lg^2 n)$ times since $f(n)$ is a monotone non-decreasing function, and $f(n) < 2\lg^2 n$. (We note that these valleys are no longer uniform, but this does not decrease the number of foldings and hence the algorithm still has to fold at least $f(n/2f(n))$ times.) Hence the algorithm has to pile $(n/2f(n))/f(n/4\lg^2 n)$ creases, and

$$\lg \frac{n/2f(n)}{f(n/4\lg^2 n)} > \lg \frac{n}{4\lg^2 n \cdot 2(\lg \frac{n}{4\lg^2 n})^2} > \lg n - 4\lg\lg n - 3 \tag{2}$$

foldings are required to pile these creases.

When the algorithm folds $n/2f(n)f(n/4\lg^2 n)$ creases at once, again we have $n/4f(n)f(n/4\lg^2 n)$ valley-folded creases. We can repeat the same argument, and $\lg n - 6\lg\lg n - 5$ foldings are required.

Repeating this argument, at ith iteration, $\lg n - 2i(\lg\lg n + 1) + 1$ foldings are required. This argument terminates when $\lg n - 2i(\lg\lg n + 1) + 1 \leq 1$, or consequently, $i = \lceil \frac{\lg n}{2\lg\lg n} \rceil$. Hence, we have

$$\sum_{i=1}^{\lceil \frac{\lg n}{2\lg\lg n} \rceil} \lg n - 2i(\lg\lg n + 1) + 1 \leq \frac{\lg n}{2\lg\lg n}\left(\frac{\lg n}{2} - \lg\lg n - \frac{\lg n}{2\lg\lg n}\right),$$

which completes the proof. □

We show here a simple proposition:

Proposition 1. *Two creases i and j can be piled at the same point if and only if $i - j$ is even.*

Proof. Imagine that the creases 1 and k come to the same position by the folding at the position i for $1 < i < k$. When we fold the position $i + 1$ instead of i, 3 and k come to the same position. (This fact may not be trivial; some people feel that 2 and k come to the same position, and a classic quiz is based on this mistake.) Hence 2 and k cannot be folded at once. Taking $i = 1, 2, \ldots$, we can see that k can be folded with the point 1 if and only if k is odd. This fact implies that we can only fold the set of integers at once only if they have the same parity. □

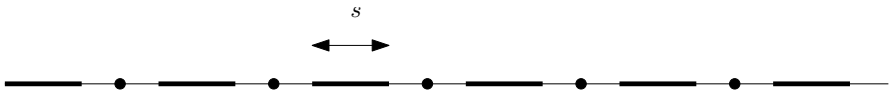


Fig. 3. Partitioning the sequences in odd-length chunks. Note that every pair of chunks is separated by an odd number of creases.

Combining Theorem 2 and Proposition 1, we have the lower bound of a pleat of length n:

Corollary 1. *The folding complexity of a pleat string of length n in the some-layers fold, all-unfold model is at least* $\lg^2 n/4 \lg\lg n - o(\lg^2 n/4 \lg\lg n)$.

Proof. By Proposition 1, each folding has its own parity in the proof of Theorem 2. Hence, replacing each "mountain-fold" at each even parity by "valley-fold", the lower bound of a uniform string is also valid for a pleat string. □

3. Folding Arbitrary Sequences

We show here that the folding complexity of a random sequence is $\Omega(n/\lg n)$ with high probability.

Theorem 3. *The folding complexity of a random sequence of length n in the some-layers fold, reverse-unfold model is at least*

$$\frac{n}{3+\lg n}$$

with high probability.

Proof. We use a counting argument. Suppose that we make k folds (and k unfolds). We note that, if a sequence of length n is obtained by less than k folds, we can also obtain the sequence using k folds exactly.

At each fold, 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. Thus we have $(2n)^k$ possibilities here.

At each unfolding operation, we may rewind some folding operations. We will rewind k folding operations in total. Moreover, for any step, the number of rewind operations so far cannot exceed the number of folding operations. Thus, there are at most $C(k)$ choices for the unfolding operations, where $C(k)$ is the kth Catalan number. Note that for large k, $C(k) < 4^k$.

Overall, the number of possible sequence of length n obtained by in at most k folds is bounded by $(2n)^k C(k)$. If we let $k = n/(3+\lg n)$, then, for sufficiently large values of n, $(2n)^k C(k) < (2n)^k 4^k = (8n)^k = 2^n$.

Therefore, the number of possible sequences of length n by at most k folding (and unfolding) operations is less than the number of all sequences of length n. □

We give an upper bound that matches this lower bound up to a constant factor. Given an arbitrary sequence of length n, and an odd number $s \geq 3$, we divide the sequence into *chunks* of size s, each pair of successive chunks being separated by one crease. Note

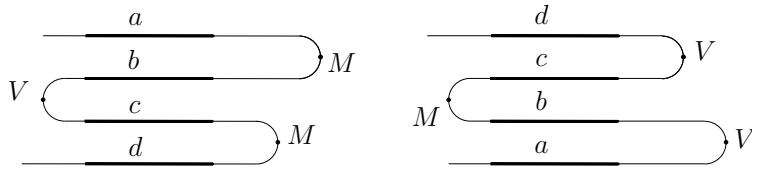


Fig. 4. Two ways to match chunks.

that, because s is odd, every pair of chunks is separated by an odd number of creases (see Fig. 3). This will allow us to align any subset of chunks by folding them on top of each other.

Suppose there are at most k distinct patterns among such chunks, that is, the subsequence of creases in any chunk belongs to a set of at most k distinct sequences. We denote by $f(n, k, s)$ the worst-case complexity of folding such a sequence in the some-layers fold, reverse-unfold model.

Lemma 2.

$$f(n, k, s) \leq 4n/s + ks \lg n.$$

Proof. Consider one kind of chunk, and suppose it appears t times. We can fold a zig-zag to match all the t chunks, so that we can fold them together. Note that this is always possible, because the distance between any two chunks is odd. Also note that folding the zig-zag involves some-layers folding operations. The number of required folds for the zig-zag is exactly $t - 1$.

When the chunks are folded, we have to fix the creases that may have been destroyed by the zig-zag. The zig-zag can be folded in at least two different ways (see Fig. 4). By choosing one of them, we can ensure that at most one half of the zig-zag creases destroy previously folded creases. Hence the zig-zag costs at most $t - 1 + t/2 \leq 3t/2$ folds: $t - 1$ folds to create it, then $t/2$ folds to fix the creases that have been destroyed.

The folding of the chunks themselves costs s folds. However, because they are mapped in alternating directions, only half of them are correctly folded. To fix this, we can recurse on the remaining half. The overall cost is therefore at most

$$(3t/2 + s) + (3t/4 + s) + (3t/8 + s) + \cdots + (1 + s) < 3t + s \lg(3t/2).$$

Now suppose that there are t_i occurrences of the chunk of type i. Note that $\sum_i t_i = n/s$. Thus repeating the steps above for each of the k kinds of chunks, we can fold all the chunks in

$$\sum_{i=1}^{k} \left(3t_i + s \lg \frac{3t_i}{2} \right) \leq 3n/s + ks \lg n$$

folds. Finally, we have to take care of the remaining creases separating the chunks. There are n/s of them. Folding them separately, we obtain

$$f(n, k, s) \leq 4n/s + ks \lg n.$$

□

This directly yields a $\Theta(n/\lg n)$ upper bound for folding arbitrary sequences.

Theorem 4. *The folding complexity of a binary sequence of length n in the some-layers fold, reverse-unfold model is at most*

$$(4+\varepsilon)\frac{n}{\lg n} + o\left(\frac{n}{\lg n}\right),$$

for any $\varepsilon > 0$.

Proof. Pick $s = (1-\varepsilon)\lg n$, and $k = 2^s = n^{1-\varepsilon}$ in the formula of Lemma 2, with $\varepsilon := \varepsilon/(4+\varepsilon)$. This yields:

$$\frac{4n}{(1-\varepsilon)\lg n} + (1-\varepsilon)n^{1-\varepsilon}\lg^2 n = (4+\varepsilon)\frac{n}{\lg n} + o\left(\frac{n}{\lg n}\right).$$

□

Note that the upper and lower bounds are within a factor 4 of each other. It remains open whether the same upper bound is possible in the all-layers fold model.

Acknowledgments

This work was put forwarded at the WAFOL '09 workshop in Brussels. We thank all the other participants, as well as Guy Louchard, for useful discussions. The eighth author is supported by Grant-in-Aid for Scientific Research (Challenging Exploratory Research) from Japan Society for the Promotion of Science.

References

1. Jean-Paul Allouche. Sur la complexité des suites infinies. *Bull. Belg. Math. Soc.*, 1:133–143, 1994.

2. Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Martin L. Demaine, Joseph S. B. Mitchell, Saurabh Sethia, and Steven S. Skiena. When can you fold a map? *Comput. Geom. Theory Appl.*, 29(1):23–46, 2004.

3. Jean Cardinal, Erik D. Demaine, Martin L. Demaine, Shinji Imahori, Stefan Langerman, and Ryuhei Uehara. Algorithmic Folding Complexity In *20th International Symposium on Algorithms and Computation (ISAAC 2009)* Lecture Notes in Computer Science Vol. 5856, 452–461, Springer-Verlag, 2009. (Extended abstract was presented at *7th Japan Conference on Computational Geometry and Graphs (JCCGG 2009)*, 143–144, 2009.)

4. chosotec. The making of Mens et Manus (in origami), vol. 1. `http://techtv.mit.edu/collections/chosetec/videos/361-the-making-of-mens-et-manus-in-origami-vol-1`, March 2007.

5. Michel Dekking, Michel Mendès France, Alf van der Poorten. Folds! *Math. Intell.*, 4:130–138, 173–181, 190–195, 1982.

6. Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms.* Cambridge University Press, 2007.

7. Erik D. Demaine and Joseph O'Rourke. Open problems from CCCG 2008. In *Proc. 21st Canadian Conference on Computational Geometry (CCCG'09)*, to appear, 2009.

8. Martin Gardner. Mathematical games. *Scientific American*, 216, 3 (March 1967), 124–125; 216, 4 (April 1967), 118–120; 217, 1 (July 1967), 115.

9. Tsuyoshi Ito, Masashi Kiyomi, Shinji Imahori, and Ryuhei Uehara. Complexity of pleat folding. In *Proc. 25th Workshop on Computational Geometry (EuroCG'09)*, 53–56, 2009.

10. Jacob Ziv and Abraham Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

11. Jacob Ziv and Abraham Lempel. Compression of Individual Sequences via Variable-Rate Coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

12. M. Mendès France and A. J. van der Poorten. Arithmetic and analytic properties of paper folding sequences. *Bull. Austr. Math. Soc.*, 24:123–131, 1981.