

MIT Open Access Articles

Algorithmic folding complexity

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: Cardinal, Jean et al. "Algorithmic Folding Complexity." Algorithms and Computation. Springer Berlin / Heidelberg, 2009. 452-461. (Lecture notes in computer science, v. 5878.) Copyright © 2009, Springer

As Published: http://dx.doi.org/10.1007/978-3-642-10631-6_47

Publisher: Springer

Persistent URL: <http://hdl.handle.net/1721.1/62218>

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike 3.0



Algorithmic Folding Complexity

Jean Cardinal¹, Erik D. Demaine², Martin L. Demaine², Shinji Imahori³,
Stefan Langerman^{1*}, and Ryuhei Uehara⁴

¹ Université Libre de Bruxelles (ULB)

B-1050 Brussels, Belgium

{jcardin,slanger}@ulb.ac.be

² MIT

Cambridge, MA 02139, USA

{edemaine,mdemaine}@mit.edu

³ University of Tokyo

Tokyo 113-8656, Japan

imahori@mist.i.u-tokyo.ac.jp

⁴ Japan Advanced Institute of Science and Technology (JAIST)

Ishikawa 923-1292, Japan

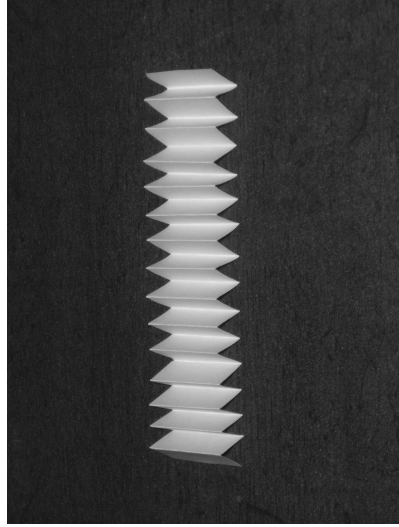
uehara@jaist.ac.jp

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 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 polylogarithmic in n . We also show that the maximum possible folding complexity of any string of length n is $O(n/\lg n)$, meeting a previously known lower bound.

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 his instructional video for folding the MIT seal *Mens et Manus* in “three easy steps” [3], Brian Chan 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 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

* Maître de recherches du F.R.S.-FNRS



(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.

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 pattern is then simply a binary string over the alphabet $\{M, V\}$ (M for mountain, V for valley), which we call a *mountain-valley string*. Of particular interest in origami is the *pleat*, which alternates $MVMVMV \dots$; see Figure 1.

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 $MMM \dots$ of n mountain creases using just $O(\lg^{1+\sqrt{2}} n)$ simple fold operations. These operations fold only along desired creases, and the last direction that each crease gets folded is mountain. By combining two executions of this algorithm, we obtain the same bound for pleats. This folding is exponentially faster than both the standard folding and the best known folding of $O(n^\varepsilon)$ folds [7]. From a complexity-theoretic perspective, this is the first polynomial-time algorithm for pleat folding, because the only input is the number n .

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 [7]. We effectively exploit that every string has some redundancy in it, similar to how Lempel–Ziv can compress any string into $O(n/\lg n)$ block pointers.

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-theory perspective, paper folding offers an intriguing new definition of the algorithmic complexity of a binary string. The *folding complexity* [7] of a mountain–valley string is the minimum number of folds needed to construct it. 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 (between P and EXPTIME) remains open. 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.

Related work. Uehara [6] posed the problem we tackle here in August 2008. In March 2009, Ito, Kiyomi, Imahori, and Uehara [7] formalized the problem and made some partial progress. On the positive side, they showed how to fold any mountain–valley string using $\lfloor n/2 \rfloor + \lceil \lg(n+1) \rceil$ folds, a bound we improve on by a logarithmic factor; and they showed how to fold the uniform string and hence a pleat using $O(n^\varepsilon)$ folds, for any $\varepsilon > 0$.⁵ On the negative side, they showed that almost every mountain–valley string requires $\Omega(n/\lg n)$ folds using an information-theoretic argument. We tighten this lower bound to prove that a lead constant factor of 1 suffices, reasonably close to our asymptotically matching upper bound which has a lead constant factor of $4 + \varepsilon$, for any $\varepsilon > 0$.

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 [8, 4, 1].

Model. We follow the folding and unfolding model of [7], which in turn is based on the simple-fold model of Arkin et al. [2] (see also [5, 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 (in $\{M, V\}$) at the end of the algorithm is the one that

⁵ A somewhat more careful analysis shows that the same algorithm uses $2^{O(\sqrt{\lg n \lg \lg n})}$ folds.

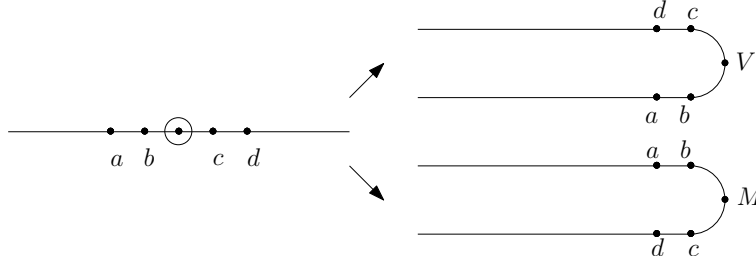


Fig. 2. Folding a mountain or a valley.

was folded last. The goal is to achieve a particular string of M s and V s at the end. Folding has the effect of superimposing the layers of paper on the right and left of the crease; see Figure 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.)

Several variants of this model arise from varying the allowed folding and unfolding operations. Clearly, a fold can be made simultaneously on several layers of the folded strip, allowing a form of parallelism. (Without this, we 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 .

For upper bounds, we concentrate on the more-restrictive all-layers fold model, while for lower bounds, we use the more-flexible some-layers fold model. 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, so we focus entirely on that 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 (effectively ignoring any surrounding creases).

2 Folding Pleats

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 [7]. The purpose of this section is to show that the pleat pattern, while seemingly difficult to fold, has a fairly close upper bound. 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 $0 \leq r < 2^k$. Valley-folding at the r th 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, reverse-unfold model is $O(\lg^c n)$, where $c = 1 + \sqrt{2}$.*

Proof. Given an algorithm for folding a uniform sequence of n mountains, we can apply it twice to obtain the pleat sequence. We therefore concentrate on finding an algorithm for folding the uniform sequence.

Let $T(n)$ be a monotone upper bound on the number of required folds. Let $w := 2^k - 1$ for some value k . We suppose that $w < \sqrt{n}$. The exact value of w will be determined later.

The algorithm is decomposed into three steps.

Step 1. We apply Lemma 1 to fold the strip into $\lfloor n/w \rfloor$ layers of width w , at a cost of $\lg(n/w) + O(1)$ folds. Then we recurse on these layers, at a cost $T(w)$. After unfolding, we get sequences of w mountains separated by $w+2$ other creases (because half of the layers were reversed). The rest of the algorithm will work only with these creases; there are a final $n \bmod w$ creases which we will deal with at the end of the algorithm simply by recursing. The overall cost of this step (both preprocessing and postprocessing) is at most

$$\lg \frac{n}{w} + O(1) + 2T(w) = 2T(w) + O(\lg n). \quad (1)$$

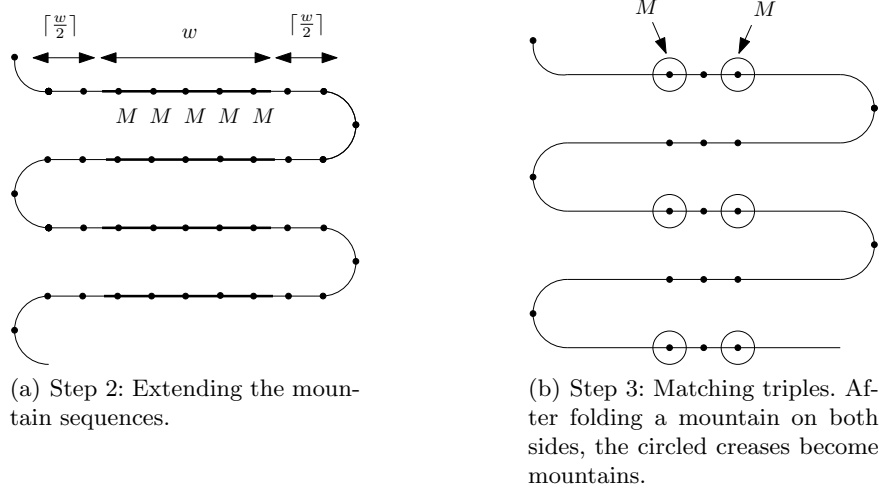


Fig. 3. Folding pleats.

Step 2. We apply Lemma 1 to a scaled version of the creases in order to align the mountain sequences (possible because the gap between the sequences, $w + 2$, is odd). See Figure 3(a), where the lemma's folding is depicted as a zig-zag for simplicity. This folding costs $\lg \frac{n}{2w} + O(1)$ folds. The crease sequences within the layers are composed of three subsequences, the middle one being the aligned mountains, of length w , and the two others being sequences of length $\lfloor (w + 2)/2 \rfloor = \lceil w/2 \rceil = 2^{k-1}$. We call the two others *side* sequences.

We recurse again on the side sequences, folding mountains on the right, and valleys on the left. (Here we use the end-free property.) This folding costs $2T(\lceil w/2 \rceil)$ folds, and has the effect of extending the lengths of the mountain sequences. Thus, after these recursive calls, we have mountain sequences of length $\lceil 3w/2 \rceil$ separated by sequences of length $2^{k-1} + 1$.

Next, we unfold everything and iterate these operations of aligning mountain sequences and recursing on both side sequences. This is possible because the distance between the mountain sequences remains odd. The width remains $2w$, so folding still costs $\lg \frac{n}{2w} + O(1)$, but the length of the side sequences decreases by a factor of 2 at every iteration. We iterate until the side sequences have length 1. At that point, we have mountain sequences of length $2w - 3$ separated by triples of “junk” creases. Overall, the cost of this step is:

$$\lg w \cdot \left(\lg \frac{n}{2w} + O(1) \right) + 2 \sum_i T \left(\left\lfloor \frac{w}{2^i} \right\rfloor \right) \leq \lg w \cdot \left(\lg \frac{n}{2w} + 2T(w) \right) + O(\lg n). \quad (2)$$

Step 3. At the end of the previous step, we are left with sequences of $2w - 3$ mountains separated by three junk creases. We can again apply Lemma 1 to align the junk triples, (possible because the gap between the triples, $2w - 3$, is

odd). This folding costs $\lg \frac{n}{2^i w} + O(1)$ folds. Considering the aligned triples, we now fold a mountain on the left and right creases of the triples; see Figure 3(b). (More precisely, we fold a mountain on the left, then unfold once, then fold a mountain on the right, then unfold everything.) This procedure costs two folds, and creates mountains on the left and right creases of the layers that are not reversed. Hence, for half the triples, there is now only one junk crease left. We iterate the above steps on the remaining triples only, reducing by a factor of 2 the number of remaining triples at every step. This procedure remains possible because the remaining triples are separated by an odd number of creases, and the middle crease lies within one of the original triples. The overall cost is $\sum_i (\lg \frac{n}{2^i w} + O(1))$.

When finished, we are left with all the middle creases of the original triples. By a simple scaling, this is a subproblem of size n/w that can be solved recursively. We therefore incur a final, additional cost of $T(n/w)$. Overall, the complexity of Step 3 is bounded by

$$T\left(\frac{n}{w}\right) + \sum_i \left(\lg \frac{n}{2^i w} + O(1)\right) \leq T\left(\frac{n}{w}\right) + \lg^2 \frac{n}{w} + O(\lg n). \quad (3)$$

Analysis. Summing the complexities of the three previous steps, we get

$$T(n) \leq (2 \lg w + 2)T(w) + \lg \frac{n}{w} \left(\lg w + \lg \frac{n}{w}\right) + T\left(\frac{n}{w}\right) + O(\lg n) \quad (4)$$

$$= (2 \lg w + 2)T(w) + O(\lg^2 n) + T\left(\frac{n}{w}\right). \quad (5)$$

We suppose that $T(n) < \lg^c n$ for some constant $c > 1$. We find a value for w such that the first term is also $O(\lg^2 n)$:

$$(2 \lg w + 2)T(w) < \lg^2 n \quad (6)$$

$$2 \lg^{c+1} w + 2 \lg^c w < \lg^2 n \quad (7)$$

$$w < 2^{b \cdot \lg^{\frac{2}{c+1}} n} \quad (8)$$

for some constant $b > 1$. The recurrence relation for $T(n)$ now becomes:

$$T(n) \leq T\left(\frac{n}{2^{b \cdot \lg^{\frac{2}{c+1}} n}}\right) + O(\lg^2 n). \quad (9)$$

This solves to:

$$T(n) = O(\lg^{\frac{3c+1}{c+1}} n). \quad (10)$$

(The proof is omitted in this version of the paper.)

But we made the hypothesis that $T(n) = O(\lg^c n)$, yielding

$$\frac{3c+1}{c+1} = c \quad (11)$$

$$c = 1 + \sqrt{2}. \quad (12)$$

□

3 Folding Arbitrary Sequences

Ito, Kiyomi, Imahori, and Uehara [7] proved that the folding complexity of a random sequence is $\Omega(n/\lg n)$ with high probability. We now refine this result.

Theorem 2 ([7]). *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 k th 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 . \square



Fig. 4. Partitioning the sequences in odd-length chunks. Note that every pair of chunks is separated by an odd number of creases.

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 that, because s is odd, every pair of chunks is separated by an odd number of creases (see Figure 4). 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.

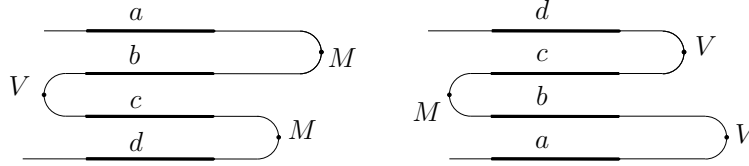


Fig. 5. Two ways to match chunks.

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 Figure 5). 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. But 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 3. *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 initiated at the WAFOL'09 workshop in Brussels. We thank all the other participants, as well as Guy Louchard, for useful discussions.

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. Brian Chan. 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.
4. Michel Dekking, Michel Mendès France, Alf van der Poorten. Folds! *Math. Intell.*, 4:130–138, 173–181, 190–195, 1982.
5. Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms*. Cambridge University Press, 2007.
6. 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.
7. 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.
8. 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.