Simple Fold and Cut Problem for Line Segments
Guoxin Hu∗ Shin-ichi Nakano† Ryuhei Uehara∗ Takeaki Uno¶

Abstract
We investigate a natural variant of the fold-and-cut problem. We are given a long paper strip $P$ and $n$ line segments drawn on $P$ such that each line segment is perpendicular to the two long edges of $P$ and the distances between the line segments are not uniform. We cut all the line segments by one complete straight cut after overlapping all of them by a sequence of simple foldings. Our goal is to minimize the number of simple foldings. In this paper, we give algorithms for finding a shortest sequence of simple foldings for given $n$ line segments. We first investigate the case that the distances are almost the same. In this case, our algorithm runs in $O(n^2)$ time and $O(n^2)$ space. Next we extend the algorithm for general distances. In general case, our algorithm runs in $O(n^3)$ time and $O(n^2)$ space.

1 Introduction
Take a sheet of paper, fold it flat, and then make one complete straight cut. What shapes can the unfolded pieces have? This fold-and-cut problem was introduced formally from the viewpoint of computational geometry in 1998 [5]. It is well known that there is a universal theorem for this problem, that is, any planar graph drawn by straight lines can be fold-and-cuttable. There are two major approaches to this problem in general form (see [7] for further details). However, in any of these approaches, to cut any shape, we need quite complicated folding operations so far, and hence it is quite difficult to realize by folding robots. From this viewpoint, the fold-and-cut problem for restricted ways of folding has been investigated. Demaine et al [4] only use a simple folding as a basic operation and investigate a connected simple polygon which can be a solution of the fold-and-cut problem. There are several models of simple folding [1], and precisely, they use all-layer simple folding, which is the simplest operation among them [2]. Demaine et al [4] focus on a connected simple polygon, and it was open for disconnected polygons.

In this paper, we consider the fold-and-cut problem for disconnected polygons, which is quite complicated in general form. Therefore, we start from the simplest problem in this framework. Intuitively, we introduce a one-dimensional version of this problem. (A similar idea is introduced for some paper folding problems; see [3, 6].) We are given a set of $n$ parallel line segments on a long paper strip $P$. The given line segments are perpendicular to the two long edges of $P$, which give us the set of cut lines. A simple example is given in Figure 1(a). In this example, a paper strip $P$ is of length 23, and 13 line segments are given as shown in the figure. We have the sequence of distances $[1, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 1]$ as the input of this problem. In order to cut these line segments by one complete straight cut, we have to overlap all of these line segments on a line so that no other part overlaps on the line.

We employ the simplest folding model, which is called all-layer simple folding. Once we choose a crease line, all paper segments on the crease line are folded in the same direction. (See [1, 2] for further details of simple folding models.) From the practical viewpoint, all-layer simple folding is easier than the others, and hence there are some folding robots realizing it. Hereafter, we just use the term simple folding for simplicity.

We note that this problem always has a feasible solution achieved by a naive algorithm (Figure 1(b)) if the intervals satisfy some condition, which is discussed later: We first put crease lines halfway between two consecutive line segments, next fold along these crease lines alternately in mountain and valley folds, and obtain a pleat folding (Figure 1(c)). Then all given cut lines are on the same line, and we can cut them (and do not...
cut any other paper) with one complete straight cut. Therefore, if we perform simple folding $n - 1$ times, we obtain a valid folded state to be cut. However, we can sometimes reduce the number of simple foldings. For example, the crease line pointed by a bold arrow in Figure 1(b) can be folded first, and then the length of the paper strip is drastically reduced.

As we will see, the naive algorithm with $n - 1$ foldings always works if the minimum distance $d_{min}$ and the maximum distance $d_{max}$ satisfy $d_{max}/d_{min} \leq 2$. We say distances are *almost the same* when this condition is satisfied. (We note that the condition is sufficient, but not necessary.) However, for example, the input [10,1,1,10] does not satisfy this condition, and then the naive algorithm does not work anymore. However, even in this case, we can solve the problem: We first fold the leftmost paper segment in half 4 times. Then we have the paper strip represented by [0.625,1,1,10]. We fold again for the rightmost paper segment and obtain [0.625,1,1.0,625]. Then we can use the naive algorithm. We will see that any sequence of distances has a feasible solution for the simple fold and cut problem using this technique.

In this paper, we consider the problem for finding the optimal way of simple folding. That is, for any given set of $n$ line segments on a paper strip, our aim is finding the shortest sequence of all-layer simple foldings to overlap all line segments on a line. When the distances are almost the same, by some observations, it is not difficult to construct a straightforward algorithm that runs in $O(n^4)$ time. We will give a non-trivial efficient algorithm for solving this problem in $O(n^4)$ time and $O(n^3)$ space. Next we extend this algorithm to the problem for general case. In general case, our extended algorithm finds a shortest simple foldings in $O(n^5)$ time and $O(n^3)$ space.

## 2 Preliminaries

Let $(d_0, d_1, \ldots, d_n)$ be the input of the problem. The paper strip $P$ is of length $L = \sum_{i=0}^{n} d_i$. We regard the paper strip $P$ as a line segment of length $L$ which is placed on the interval $[0, L]$ on the $x$-coordinate. Let $\ell_0 = 0$ (and $\ell_{n+1} = L$) be the corresponding $x$-coordinate of the left (and right) endpoint of $P$, respectively. We also let $\ell_j = \sum_{i=0}^{j-1} d_i$ for $0 < j < n$. Namely, $\ell_j$ gives the $x$-coordinate of the $j$-th line segment. We sometimes call paper strip between $\ell_i$ and $\ell_{i+1}$ the $i$th paper segment.

We define the point $f_j$ as the middle point between two consecutive line segments $\ell_j$ and $\ell_{j+1}$ with $0 \leq j \leq n$ (precisely, $f_j = (\ell_j + \ell_{j+1})/2$). Except the 0th and the $n$th segment, we can assume that our algorithm always folds $P$ at some $f_j$ (otherwise, we cannot make a simple fold and cut anymore).

We here note that $\ell_0$ and $\ell_{n+1}$ are not the line segment to be cut. In a sense, they are already cut, and we do not need to fold along the line at $f_0$ and $f_n$. On the other hand, when $d_0$ and $d_n$ are quite large, we cannot use the simple pleat folding algorithm stated in the introduction. Once we fold along a line $f_i$, the paper segment in $[0, d_0]$ or $[L - d_n, L]$ may cover some other crease lines, and we cannot make any simple fold anymore. When they cover no crease line, the line $f_i$ becomes the edge of the paper strip and it plays the same role then. This issue will be discussed in Section 4. We first assume on the distances to avoid this issue. Let $\min = \min\{2d_0, d_1, d_2, \ldots, d_{n-1}, 2d_n\}$ and $\max = \max\{2d_0, d_1, d_2, \ldots, d_{n-1}, 2d_n\}$. When we have $1 \leq \max/\min \leq 2$, we can use the simple pleat folding algorithm as follows. When we fold the paper strip by a simple folding along the leftmost crease line $f_i$ at the $i$th step, the left paper segment from $f_i$ is of length at most $\max/\min \cdot d_{\min}$, and the $(i+1)$st paper segment is of length at least $\min \cdot d_{\min \max} \geq \max/\min \cdot d_{\min}$. Therefore, the leftmost paper segment does not cover the crease line $f_{i+1}$. We call the distances are almost the same in this situation. Now we are ready to state our problem:

**Input:** A paper strip $P$ with parallel $n$ lines $\ell_0, \ldots, \ell_n$, where $d_i$ is the distance between $\ell_i$ and $\ell_{i+1}$ for each $d = 0, \ldots, n$ ($\ell_0$ and $\ell_{n+1}$ denote the left and right edges of $P$).

**Operation:** All-layer simple folding.

**Goal:** Finding a shortest sequence of simple foldings that overlaps all lines $\ell_i$ $(1 \leq i \leq n)$ on a line, and no other paper segment is on the line.

We first consider the case that the distances are almost the same. That is, we only make a simple folding along some crease line $f_i$. We assume that the (folded) paper strip $P$ is on the interval $[i, j]$ for some $i < j$. We...
When our algorithm makes a simple fold $P$ at $f_k$ with $i < f_k < j$, it flips the left part of $P$ at the crease $f_k$ if $f_k \leq (i + j)/2$, and it flips the right part of $P$ at $f_k$ if $f_k > (i + j)/2$. Then, we can observe that $P$ always shrinks without changing its position. Precisely, we have the following observation:

**Observation 1** Assume $P$ is placed on the interval $[i, j]$ for some $i < j$. After making a simple folding, $P$ is placed on the interval $[i', j']$ such that either (1) $i < i' \leq (i + j)/2$ and $j' = j$ or (2) $i' = i$ and $(i + j)/2 < j' < j$. Moreover, the sequence of line segments (or distances) in $[i', j']$ is not changed by the simple folding.

Two examples are shown in Figure 2. We here note that in this simple folding, the direction (mountain or valley) of simple folding does not matter. Therefore, we do not consider the direction of each folding hereafter. Intuitively, we start from a paper strip $P$ placed on $[0, L]$, the interval shrinks after each simple folding, and eventually, we obtain the paper strip placed on $[f_j, f_{j+1}]$ for some $j$. Then we can make one complete straight cut of all given line segments (without cutting any other part). Our goal is finding the shortest sequence of $f_k$s for it.

Let $(d_0, d_1, \ldots, d_n)$ be the input of the problem, and $(f_1, \ldots, f_{n-1})$ be the middle points. At the point $f_i$ with $f_i \leq L/2$, we can fold $P$ at $f_i$ only if every pair of corresponding line segments overlaps. More precisely, we say that $P$ is simple foldable at $f_i$ if and only if $d_{i+j} = d_{i-j}$ for every $j = 1, 2, \ldots, i-1$ and $d_{2i} \geq d_0$ (since $d_0$ is the length of the leftmost paper segment). Such a simple folding at $f_i$ is said to be valid. We can define it for a point $f_i$ with $f_i > L/2$ in the same way. We can also define a valid simple folding for a folded $P$ in the same manner. When a folded state of $P$ is obtained by a sequence of valid simple foldings, the folded state is also said to be valid. Let $f_i$ be a valid point of $P$ with $f_i \leq L/2$. Then it is easy to see that the sequence $(d_1, \ldots, d_i, \ldots, d_{2i-1})$ is a palindrome of odd length, and $d_0 \leq d_{2i}$.

In this paper, palindromes of odd lengths play an important role. For a given string $S = (s_0, s_1, \ldots, s_n)$, a maximal palindrome centered at $s_i$ is defined by a maximal palindrome of odd length at center $s_i$ in $S$. We will use the following result by Manacher:

**Theorem 1 ([8])** For a given string $S = (s_0, s_1, \ldots, s_n)$ of length $n$, let $p_i$ be the length of the maximal palindrome centered at $s_i$. Then the sequence $(p_0, p_1, \ldots, p_n)$ can be computed in $O(n)$ time and $O(n)$ space.

---

We here note that, by Observation 1 and Theorem 1, we can solve the simple fold and cut problem for a given $P$ in $O(n^4)$ time and $O(n^2)$ space if the distances are almost the same:

**Proposition 2** If the distances are almost the same, there is an algorithm for solving the simple fold and cut problem for $P$ in $O(n^4)$ time and $O(n^2)$ space.

**Proof.** For the input of the problem, let $P$ be the set of intervals $[f_i, f_j]$ for each $0 \leq f_i < f_j \leq L$. By Observation 1, each valid folded state $P$ can be represented by an interval $[f_i, f_j]$ for some $0 \leq f_i < f_j \leq L$. Now let $G = (P, E)$ be a directed graph defined as follows. For each pair of $P_1, P_2 \in P$, $(P_1, P_2) \in E$ if the folded state represented by $P_2$ can be achieved from the folded state represented by $P_1$ by a simple folding. Then the solution of the problem is given by a shortest path on $G$ from the interval $[0, L]$ to an interval $[f_j, f_{j+1}]$ for some $j$. Since $|P| = O(n^4)$ and each vertex $P_i$ is of degree $O(n)$, the construction of $G$ takes $O(n^5)$ time with $O(n^2)$ space with precomputation of the sequence of lengths of maximal palindromes by Theorem 1. Then the shortest path problem can be solved in $O(n^4)$ time with $O(n^2)$ space by the breadth first search. 

---

3 Almost the Same Case

We will improve the running time in Proposition 2 from $O(n^4)$ time to $O(n^2)$ time. That is, the main theorem in this section is as follows:

**Theorem 3** There is an algorithm for solving the simple fold and cut problem for $P$ when the distances are almost the same in $O(n^2)$ time and $O(n^2)$ space.

Here we show a simple but crucial lemma.

**Lemma 4** Let $(d_0, d_1, \ldots, d_n)$ be the input of the problem, and $(f_1, \ldots, f_{n-1})$ be the middle points. Let $P'$ be a folded state of $P$ placed on $[f_i, f_j]$. We assume that two simple foldings at $f_k$ and $f_{k'}$ are both valid for some $k$ and $k'$ with $i < k < k' < (i + j)/2$. Then for any valid simple folding sequence $F = (f_{k_1}, \ldots, f_{k_r})$, we have another valid simple folding sequence $F' = (f_{k'_1}, \ldots, f_{k'_r})$ that is as short as $F$.

**Proof.** In $F$, after first valid simple folding at $f_k$, we obtain the folded state $P''$ in the interval $[f_k, f_j]$. On the other hand, after $f_{k'}$, the folded state $P'''$ is in the interval $[f_{k'}, f_j]$ with $f_k < f_{k'}$. Then, by Observation 1, $P'''$ is the subsequence of $P''$. Precisely, the latter part $[f_{k'}, f_j]$ in $P'''$ is the same as $P''$. Therefore, we first replace the first $f_k$ in $F$ by $f_{k'}$ and remove all valid simple foldings $f_{k'}$ with $f_k < f_{k'} < f_{k'}$, then we obtain a valid simple folding sequence $F'$. Then $F'$ is as short as $F$, which completes the proof.
We can obtain the same lemma for the right part of \( P \). Therefore, when we consider the shortest sequence of valid simple foldings for any folded state \( P' \) of \( P \) placed on \([f_i, f_j]\), it is sufficient to consider two valid simple foldings at \( f_i \) and \( f_r \), where \( f_i \) is the maximum valid simple foldable point with \( f_j \leq (f_i + f_j)/2 \) and \( f_r \) is the minimum valid simple foldable point with \((f_i + f_j)/2 < f_r\).

### 3.1 Data Structures for Our Algorithm

Let \((d_0, d_1, \ldots, d_n)\) be the input of the problem, and \((f_1, \ldots, f_{n-1})\) be the middle points on \( P \). Our algorithm is based on dynamic programming. We will use the following tables.

**Palindromes:** To check valid simple folding, we will use the sequence \((p_0, p_1, \ldots, p_{n-1})\), where \( p_i \) is the length of the maximal palindromic centered at \( f_i \) for each \( i = 0, \ldots, n - 1 \). By Theorem 1, the sequence can be computed in \( O(n) \) time and \( O(n) \) space.

**LLINE, RLINE:** For positive integers \( i \) and \( l \), \( \text{LLINE}[i][l] \) indicates whether the paper segment \([f_i, f_{i+l}]\) can be folded to left along the line \( f_{i+l} \) or not. Intuitively, when \( \text{LLINE}[i][l] = 1 \), the paper segment \([f_i, f_{i+l+2}]\) gives us a palindromic with respect to the distances. More precisely, we define \( \text{LLINE}[i][l] = 1 \) if \( d_l = d_{l+2}, d_{l+1} = d_{l+2l-1}, \ldots, d_{l+l-1} = d_{l+l+1} \). Otherwise, we define \( \text{LLINE}[i][l] = 0 \). We note that if \( L < i + 2l \), we define \( \text{LLINE}[i][l] = 0 \). For the sake of simplicity, we also define \( \text{LLINE}[i][0] = 1 \) for any \( i \).

In the same manner, we also define \( \text{RLINE}[i][l] \). Precisely, we define \( \text{RLINE}[i][l] = 1 \) if \( d_l = d_{l-2}, d_{l-1} = d_{l-2l+1}, \ldots, d_{l-l-1} = d_{l-1} \).

For the initial state of \( P \) where \( P \) is placed on an interval \([0, L]\), by Lemma 4, we can observe that the shortest valid simple folding sequence starts from either the crease line \( f_i \) or \( f_j \) such that \( 0 < f_m \) be the central crease line, that is, \( n = \lceil n/2 \rceil \), (1) \( f_1 \) is the maximum index in \( 0 \leq l \leq m \) such that \( \text{LLINE}[0][l-i] = 1 \), and (2) \( f_2 \) is the minimum index in \( m < r \leq n \) such that \( \text{RLINE}[n][r-j] = 1 \). This observation holds for general folded state of \( P \) as follows.

**Lemma 5** Let \( P' \) be the folded state of \( P \) which is obtained by some simple foldings in the manner stated in Observation 1. That is, we can assume \( P' \) is placed on an interval \([f_i, f_j]\) for some crease lines \( f_i \) and \( f_j \) with \( i < j \). Let \( f_m \) be the central crease line given by \( m = \lfloor (i + j)/2 \rfloor \). Then the shortest valid simple folding sequence starts from either the crease line \( f_i \) or \( f_j \) such that (1) \( f_i(j) \) is the maximum index in \( i \leq l \leq m \) such that \( \text{LLINE}[i][l-i] = 1 \), and (2) \( f_j(i) \) is the minimum index in \( m < r \leq j \) such that \( \text{RLINE}[j][j-r] = 1 \).

**Proof.** By Lemma 4 and the definitions of \( \text{LLINE, RLINE} \), it follows.

Let \( \text{sf}[i][j] \) be the minimum number of simple foldings of the folded state \( P' \) placed on an interval \([f_i, f_j]\) of the paper strip \( P \). Then by Lemma 5, we have the following observation.

**Observation 2** The minimum number of simple foldings is given by \( \text{sf}[0][n - 1] \) which satisfies the following recursive definitions.

\[
\begin{align*}
\text{sf}[i][j] &= \infty \text{ if } i > j, \\
\text{sf}[i][j] &= 0 \text{ if } j = i, \\
\text{sf}[i][j] &= \min \left\{ \text{sf}[f(i, j) + 1][j], \text{sf}[i][f_r(i, j) - 1] \right\} + 1 \text{ otherwise},
\end{align*}
\]

where \( f(i, j) \) and \( f_r(i, j) \) are defined as shown in Lemma 5.

Hereafter, we will show the implementation of the computation of Observation 2 which runs in \( O(n^2) \) time and \( O(n^2) \) space.

### 3.2 Algorithm Description and Computational Complexity

In this section, we prove Theorem 3 by showing the details of implementation of our main algorithm. It computes the minimum number of simple foldings by computing \( \text{sf}[0][n - 1] \) in Observation 2. The algorithm consists of initialization of auxiliary arrays and computation of \( \text{sf}[i][j] \).

**3.2.1 Initialization step**

For a given input \((d_0, d_1, \ldots, d_n)\), the algorithm first computes the middle points \((f_1, \ldots, f_{n-1})\). Next it computes the sequence \((p_0, p_1, \ldots, p_{n-1})\), where \( p_i \) is the length of the maximal palindromic centered at \( f_i \) for each \( i = 0, \ldots, n - 1 \). By Theorem 1, the sequence can be computed in \( O(n) \) time and \( O(n) \) space. The algorithm next computes \( \text{LLINE}[i][l] \) for each \( i \) and \( l \). (We prepare a two-dimensional array to store the table \( \text{LLINE}[i][l] \)). By the definition of \( \text{LLINE}[i][l] \), \( \text{LLINE}[i][l] = 1 \) if and only if \( p_{i+l} > 2l \). Therefore, it can be computed in \( O(n^2) \) time with \( O(n^2) \) space.

In order to compute the maximum and minimum indices in Lemma 5, we use two auxiliary tables (or arrays) \( \text{Fl}(i, j) \) and \( \text{Fr}(i, j) \). For \( m = \lfloor (i + j)/2 \rfloor \), each element \( \text{Fl}(i, j) \) gives the maximum index \( l \) in \([i, m]\) such that \( \text{LLINE}[i][l-i] = 1 \), and \( \text{Fr}(i, j) \) gives the minimum index \( r \) in \([m + 1, j]\) such that \( \text{RLINE}[j][j-r] = 1 \). For a fixed \( i \), it is not difficult to see that \( \text{Fl}(i, j) \) is changed from \( \text{Fl}(i, j - 1) \) only when the corresponding \( \text{LLINE}[i][m-i] = 1 \), otherwise, \( \text{Fl}(i, j) = \text{Fl}(i, j - 1) \).
1). More precisely, we have (0) $F_L(i, 0) = i$ (since $LLine[i][0] = 1$), (1) $F_L(i, j) = m$ if $LLine[i][m - i] = 1$, and (2) $F_L(i, j) = F_L(i, j - 1)$ if $LLine[i][m - i] = 0$. Therefore, the table $F_L(i, j)$ can be computed in $O(n^2)$ time and $O(n^2)$ space. For the table $Fr(i, j)$, we can use a symmetric argument, and it can be computed in the same manner.

### 3.2.2 Computation of $sf[i][j]$

Now we turn to the computation of the minimum number of simple foldings. It can be done by computing the array $sf[i][j]$. Using the auxiliary arrays in the previous section, we can describe this computation as follows: $sf[i][i] = sf[i][i + 1] = 0$ for each $i$, and $sf[i][j] = \min\{sf[F_L(i, j) + 1][j], sf[i][Fr(i, j) - 1]\} + 1$ for each $j = i, \ldots, n - 1$.

This equation is a recursive function with respect to the interval $[i, j]$. That is, $sf[i][j]$ is defined by $sf[i'][j']$ and $sf[i'][j']$ such that $i < i'$ and $j' < j$. Therefore, we can use the standard dynamic programming technique with respect to the length of the interval. That is, the algorithm first initializes all elements of $sf[i][i] = sf[i][i + 1] = 0$ and next computes all elements of $sf[i][i + 2]$, then computes all elements of $sf[i][i + 3]$, and so on. This can be done in $O(n^2)$ time and $O(n^2)$ space, which completes the proof of Theorem 3.

### 4 General Case

In the general case, we have to take care of the case that the leftmost paper segment or the rightmost paper segment covers some lines $\ell_i$ to be cut after folding.

To avoid this, the leftmost or rightmost paper segment has to be simple folded locally to shrink its length. Some simple examples are given in Figure 3. We here note that any intermediate paper segment, say the $i$th paper segment between $\ell_i$ and $\ell_{i+1}$, cannot be shrunk by any all-layer simple fold: If we fold along some crease line except $f_k$, either $\ell_i$ or $\ell_{i+1}$ should overlap with some paper inside of this segment. Then we cannot separate this cut line by any all-layer simple fold, and hence we will fail to perform one straight cut for this line. That is, if we want to shrink the $i$th paper segment, we have to fold along $f_k$ first, and make it the leftmost or rightmost paper segment of length $d_i/2$ in the folded state. Then we can observe the following:

**Observation 3** For any given numbers $d$ and $D$ with $d < D$, the minimum number of simple foldings to make a paper segment of length $D$ to one of length at most $d$ is $\lceil \log_2 D - \log_2 d \rceil$.

**Proof.** It is easy to see that to reduce the length of a paper segment of length $D$ to at most $d$, the optimal way is to repeat in simple folding in half $k$ times such that $k$ is the minimum integer with $D/2^k < d$. Solving this, we obtain the claim.

In the case that distances are almost the same, one simple folding can always be done at any $f_i$ if it is valid. In general case, as observed above, it is not necessarily true anymore. We suppose that the folded state $P'$ is on an interval $[i, j]$ as shown in Observation 1, and we want to make a simple folding along $f_k$ for some $k$ in $\langle i, j \rangle$. Then we have two basic operations that consist of two phases: we first shrink the leftmost paper strip and make a simple fold along $f_k$ when $k \leq (i + j)/2$, or we first shrink the rightmost paper strip and make a simple fold along $f_k$ when $k > (i + j)/2$. The extra simple foldings can be estimated by Observation 3. In order to describe this extra foldings, we introduce a function $\text{ef}(d_i, d_j)$ as follows. If $d_i$ is short enough, we need no extra foldings. We have three cases; we define $\text{ef}(d_i, d_j) = 0$ when (1) $0 < i < n$ and $d_i/2 \leq d_j$, (2) $i = 0$ and $j = n$, (3) $i = 0$, $j < n$, and $d_0 \leq d_j$, or (4) $i = n$, $j > 0$, and $d_n \leq d_j$. In the other cases, we need extra foldings as follows. When $0 < i < n$ and $d_i/2 > d_j$, we define $\text{ef}(d_i, d_j) = \lfloor \log_2(d_i/2) - \log_2 d_j \rfloor$. When $i = 0$ or $i = n$ (with $j \neq 0$ and $j \neq n$), we define $\text{ef}(d_i, d_j) = \lfloor \log_2 d_i - \log_2 d_j \rfloor$ when $d_i > d_j$.

Now we are ready to show the main theorem in this section:

![Figure 3: We have to roll up the leftmost paper segment of length 6; it should be folded in half (a) 3 times to make it of length $6/2^3 = 0.75 < 1$ when we fold it along the crease line $f_1$. (b) 2 times to make it of length $6/2^2 = 1.5 < 2$ when we fold it along the crease line $f_2$, and (c) 0 times when we fold it along the crease line $f_3$.](image)
Theorem 6 There is an algorithm for solving the simple fold and cut problem for $P$ in general distances in $O(n^3)$ time and $O(n^2)$ space.

Proof. The basic strategy is the same as the proof of Theorem 3. The key issue is that we cannot have the monotone property used in the proof of Lemma 4 anymore. When we have two valid foldings at $f_k$ and $f_{k'}$ with $i < k < k' < (i + j)/2$ on the folded state $P'$ on $[i, j]$ except the leftmost paper segment, $f_k$ can be better choice than $f_{k'}$ if $\text{ef}(d_i, d_{k-1})$ is much smaller than $\text{ef}(d_i, d_{k'-1})$. Thus we cannot use Lemma 5.

However, we use the following weaker claim. Let $P'$ be the folded state of $P$ which is obtained by some simple foldings, in the manner stated in Observation 1. That is, we can assume $P'$ is placed on an interval $[f_i, f_j]$ for some crease lines $f_i$ and $f_j$ with $i < j$. Let $f_m$ be the central crease line given by $m = \lceil (i + j)/2 \rceil$. Then, since no intermediate paper segment can be folded by an all-layer simple folding along a line except $f_i$s, the shortest valid simple folding sequence starts from either the crease line $f_i(i, j)$ or $f_j(i, j)$ such that (1) $f_i(i, j)$ is an index in $i \leq l \leq m$ such that $\text{LLine}[i][l - i] = 1$, and (2) $f_j(i, j)$ is an index in $m < r \leq j$ such that $\text{RLine}[j][j - r] = 1$.

Thus, we can use the same technique for finding the shortest sequence of simple foldings maintained by an array $\text{sf}[i][j]$ which is defined by $\text{sf}[i][i] = 0$ for each $i$, and

$$\text{sf}[i][j] = \min \left\{ \text{sf}[i'][j] + \text{ef}(d_i, d_{i'} - 1), \text{sf}[i][j'] + \text{ef}(d_j, d_{j'} - 1) \right\} + 1$$

for each $i' = i+1, \ldots, m$ and $j' = m+1, m+2, \ldots, j-1$. The correctness follows from the above discussion. This equation is still a recursive function with respect to the interval $[i, j]$. That is, $\text{sf}[i][j]$ is defined by $\text{sf}[i'][j]$ and $\text{sf}[i][j']$ such that $i < i'$ and $j' < j$. Therefore, we can use the standard dynamic programming technique with respect to the length of the interval. This can be done in $O(n^3)$ time and $O(n^2)$ space. \hfill \square

5 Concluding Remarks

In this paper, we investigated a one-dimensional variant of the simple fold and cut problem. We use the simplest model for simple folding. When we use the other models for simple folding discussed in [1], we will have a different problem. For example, the naïve algorithm stated in the introduction always works with $n - 1$ foldings for some simple folding models stronger than all-layer simple folding model. In this case, finding a shortest sequence of simple foldings seems to be a completely different problem, which is an interesting open problem.

On any simple folding model, the simple fold and cut problem for multiple simple polygons in two-dimensional space is still open. The authors conjecture that finding a shortest sequence of simple foldings for this general two-dimensional case is NP-complete.

Acknowledgements

The third author was supported in part by MEXT/JSPS KAKENHI Grant Number 17H06287 and 18H04091. This work was discussed at the 4th Workshop on Combinatorics and Enumeration Algorithms held on February 21–23, 2019 in Karuizawa, Japan supported by NII joint research fund. We thank the other participants of that workshop for providing a stimulating research environment.

References

Appendix

For an input \([1,1,2,2,2,1,2,2,2,1,2,1]\), the eventual values of our algorithm are given in Tables 1–6:

Table 1: Lengths of the maximal palindromes.

<table>
<thead>
<tr>
<th>(l)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: LLINE[i][l]

Table 3: RLINE[i][l]

Table 4: Ft(i, j)

Table 5: Fr(i, j)

Table 6: sf[i][j]