

Common Developments of Three Incongruent Boxes of Area 30

Dawei Xu¹, Takashi Horiyama²(✉), Toshihiro Shirakawa^{1,2},
and Ryuhei Uehara¹

¹ School of Information Science,
Japan Advanced Institute of Science and Technology, Nomi, Japan
{xudawei, uehara}@jaist.ac.jp

² Information Technology Center, Saitama University, Saitama, Japan
horiyama@al.ics.saitama-u.ac.jp

Abstract. We investigate common developments that can fold into plural incongruent orthogonal boxes. Recently, it was shown that there are infinitely many orthogonal polygons that folds into three boxes of different size. However, the smallest one that folds into three boxes consists of 532 unit squares. From the necessary condition, the smallest possible surface area that can fold into two boxes is 22, which admits to fold into two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$. On the other hand, the smallest possible surface area for three different boxes is 46, which may admit to fold into three boxes of size $1 \times 1 \times 11$, $1 \times 2 \times 7$, and $1 \times 3 \times 5$. For the area 22, it has been shown that there are 2,263 common developments of two boxes by exhaustive search. However, the area 46 is too huge for search. In this paper, we focus on the polygons of area 30, which is the second smallest area of two boxes that admits to fold into two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$. Moreover, when we admit to fold along diagonal lines of rectangles of size 1×2 , the area may admit to fold into a box of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$. That is, the area 30 is the smallest candidate area for folding three different boxes in this manner. We perform two algorithms. The first algorithm is based on ZDDs, zero-suppressed binary decision diagrams, and it computes in 10.2 days on a usual desktop computer. The second algorithm performs exhaustive search, however, straightforward implementation cannot be run even on a supercomputer since it causes memory overflow. Using a hybrid search of DFS and BFS, it completes its computation in 3 months on a supercomputer. As results, we obtain (1) 1,080 common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$, and (2) 9 common developments of three boxes of size $1 \times 1 \times 7$, $1 \times 3 \times 3$, and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$.

1 Introduction

Since Lubiw and O'Rourke posed the problem in 1996 [10], polygons that can fold into a (convex) polyhedron have been investigated in the area of computational geometry. In general, we can state the development/folding problem as follows:



Fig. 1. Cubigami.

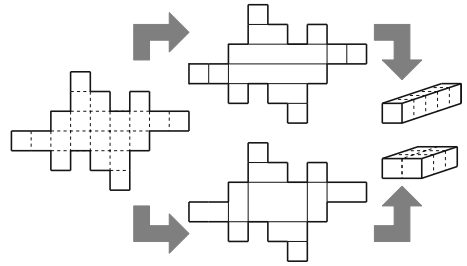


Fig. 2. A polygon folding into two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$ in [12].

Input : A polygon P and a polyhedra Q

Output: Determine whether P can fold into Q or not

When Q is a tetramonohedron (a tetrahedron with four congruent triangular faces), Akiyama and Nara gave a complete characterization of P by using the notion of tiling [2,3]. Except that, we have quite a few results from the mathematical viewpoint. Hence we can tackle this problem from the viewpoint of computational geometry and algorithms.

From the viewpoint of computation, one natural restriction is that considering the orthogonal polygons and polyhedra which consist of unit squares and unit cubes, respectively. Such polygons have wide applications including packaging and puzzles, and some related results can be found in the books on geometric folding algorithms by Demaine and O'Rourke [6,14]. However, this problem is counterintuitive. For example, the puzzle "cubigami" (Fig. 1) is a common development of all tetracubes except one (since the last one has surface area 16, while the others have surface area 18), which is developed by Miller and Knuth. One of the many interesting problems in this area asks whether there exists a polygon that folds into plural incongruent orthogonal boxes. This folding problem is very natural but still counterintuitive; for a given polygon that consists of unit squares, and the problem asks are there two or more ways to fold it into simple convex orthogonal polyhedra (Fig. 2). Biedl et al. first gave two polygons that fold into two incongruent orthogonal boxes [5] (see also Fig. 25.53 in the book by Demaine and O'Rourke [6]). Later, Mitani and Uehara constructed infinite families of orthogonal polygons that fold into two incongruent orthogonal boxes [12]. Recently, Shirakawa and Uehara extended the result to three boxes in a nontrivial way; that is, they showed infinite families of orthogonal polygons that fold into three incongruent orthogonal boxes [16]. However, the smallest polygon by their method contains 532 unit squares, and it is open if there exists much smaller polygon of several dozens of squares that folds into three (or more) different boxes.

Table 1. A part of possible size $a \times b \times c$ of boxes and its common surface area $2(ab + bc + ca)$.

$2(ab + bc + ca)$	$a \times b \times c$
22	$1 \times 1 \times 5, 1 \times 2 \times 3$
30	$1 \times 1 \times 7, 1 \times 3 \times 3$
34	$1 \times 1 \times 8, 1 \times 2 \times 5$
38	$1 \times 1 \times 9, 1 \times 3 \times 4$
46	$1 \times 1 \times 11, 1 \times 2 \times 7, 1 \times 3 \times 5$
54	$1 \times 1 \times 13, 1 \times 3 \times 6, 3 \times 3 \times 3$
58	$1 \times 1 \times 14, 1 \times 2 \times 9, 1 \times 4 \times 5$
62	$1 \times 1 \times 15, 1 \times 3 \times 7, 2 \times 3 \times 5$
64	$1 \times 2 \times 10, 2 \times 2 \times 7, 2 \times 4 \times 4$
70	$1 \times 1 \times 17, 1 \times 2 \times 11, 1 \times 3 \times 8, 1 \times 5 \times 5$
88	$1 \times 2 \times 14, 1 \times 4 \times 8, 2 \times 2 \times 10, 2 \times 4 \times 6$

It is easy to see that two boxes of size $a \times b \times c$ and $a' \times b' \times c'$ can have a common development only if they have the same surface area, i.e., when $2(ab + bc + ca) = 2(a'b' + b'c' + c'a')$ holds. We can compute small surface areas that admit to fold into two or more boxes by a simple exhaustive search. We show a part of the table for $1 \leq a \leq b \leq c \leq 50$ in Table 1. From the table, we can say that the smallest surface area is at least 22 to have a common development of two boxes, and their sizes are $1 \times 1 \times 5$ and $1 \times 2 \times 3$. In fact, Abel et al. have confirmed that there exist 2,263 common developments of two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$ [1]. On the other hand, the smallest surface area that may admit to fold into three boxes is 46, which may fold into three boxes of size $1 \times 1 \times 11$, $1 \times 2 \times 7$, and $1 \times 3 \times 5$. However, the number of polygons of area 46 seems to be too huge to search. This number is strongly related to the enumeration and counting of polyominoes, namely, orthogonal polygons that consist of unit squares [7]. The number of polyominoes of area n is well investigated in the puzzle society, but it is known up to $n = 45$, which is given by the third author (see the OEIS (<https://oeis.org/A000105>) for the references). Since their common area consists of 46 unit squares, it seems to be hard to enumerate all common developments of three boxes of size $1 \times 1 \times 11$, $1 \times 2 \times 7$, and $1 \times 3 \times 5$.

One natural step is the next one of the surface area 22 in Table 1. The next area of 22 in the table is 30, which admits to fold into two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$. When Abel et al. had confirmed the area 22 in 2011, it takes around 10 h. Thus we cannot use the straightforward way in [1] for the area 30. We first employ a nontrivial extension of the method based on a zero-suppressed binary decision diagram (ZDD) used in [4], which is so-called frontier-based search algorithm for enumeration [9]. Our first algorithm based on ZDD runs in around 10 days on an ordinary PC. To perform double-check, we also use supercomputer (CRAY XC30). We note that we cannot use the same way as one for area 22 shown in [1] since it takes too huge memory even on a supercomputer. Therefore, we use

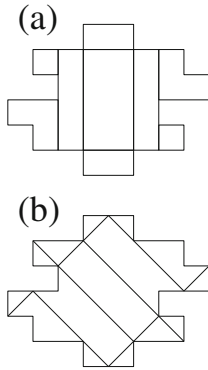


Fig. 3. The common development shown in [5]. (a) It folds into a box of size $1 \times 2 \times 4$ and (b) it also folds into a box of size $\sqrt{2} \times \sqrt{2} \times 3\sqrt{2}$.

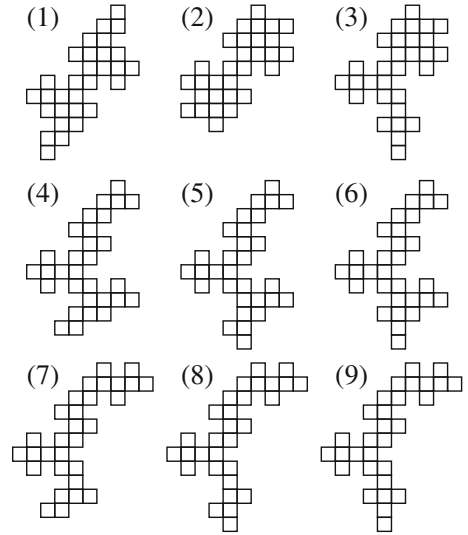


Fig. 4. Nine polygons that fold into three boxes of size $1 \times 1 \times 7$, $1 \times 3 \times 3$, and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$. The last one can fold into the third box in two different ways (Fig. 5).

a hybrid search of the breadth first search and the depth first search. Our first result is the number of common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$, which is 1,080.

Based on the obtained common developments, we next change the scheme. In [5], they also considered folding along 45° lines, and showed that there was a polygon that folded into two boxes of size $1 \times 2 \times 4$ and $\sqrt{2} \times \sqrt{2} \times 3\sqrt{2}$ (Fig. 3). In this context, we can observe that the area 30 may admit to fold into another box of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ by folding along the diagonal lines of rectangles of size 1×2 . This idea leads us to the problem that asks if there exist common developments of three boxes of size $1 \times 1 \times 7$, $1 \times 3 \times 3$, and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ among the common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$.

We remark that this is a special case of the development/folding problem above. In our case, P is one of the 1,080 polygons that consist of 30 unit squares, and Q is the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$. We note that we can use a pseudopolynomial time algorithm for Alexandrov’s Theorem proposed in [8], however, it runs in $O(n^{456.5})$ time, and it is not practical. Therefore, we develop the other efficient algorithm specialized in our case that checks if a polyomino P of area 30 can fold into a cube Q of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$. Using the algorithm, we check if these common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$ can also fold into the third box of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$, and give an affirmative answer. We find that nine of 1,080 common developments of two boxes can fold into the third box (Fig. 4). Moreover, one of the nine common developments of three

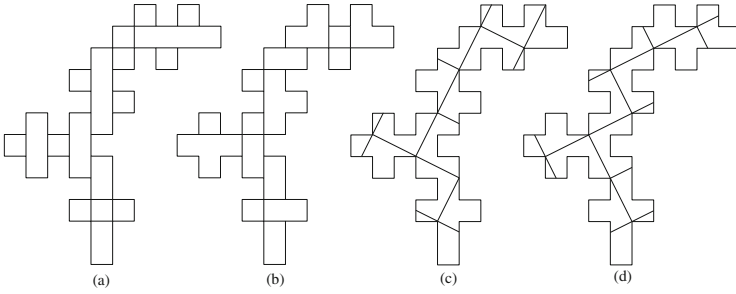


Fig. 5. The unique polygon folds into three boxes of size (a) $1 \times 1 \times 7$, (b) $1 \times 3 \times 3$, and (c)(d) $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ in four different ways.

boxes has another way of folding. Precisely, the last one (Fig. 4(9)) admits to fold into the third box of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ in two different ways. These four ways of folding are depicted in Fig. 5.

We summarize the main results in this paper:

Theorem 1. (1) *There are 1,080 polyominoes of area 30 that admit to fold (along the edges of unit squares) into two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$.* (2) *Among the above 1,080, nine polyominoes can fold into the third box of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ if we admit to fold along diagonal lines (Fig. 4).* (3) *Among these nine polyominoes, one can fold into the third box in two different ways (Fig. 5).*

2 Preliminaries

2.1 Problem Definitions

Demaine and O’Rourke [6, Chap. 21] give a formal definition of the *development* of a polyhedron as the *net*¹. Briefly, the development is the unfolding obtained by slicing the surface of the polyhedron, and it forms a single connected simple polygon without self-overlap. The *common development* of two (or more) polyhedra is the development that can fold into both (or all) of them. We only consider connected orthogonal polygons that consist of unit squares, which are called *polyominoes* [7], as developments. Polyominoes obtained from a development by removing some unit squares are called *partial developments* of it. We call a convex orthogonal polyhedron (folded from a polyomino) a *box*.

The cut edges of an edge development of a convex polyhedron form a spanning tree of the 1-skeleton (i.e., the graph formed by the vertices and the edges) of the polyhedron (See e.g., [6, Lemma 22.1.1]). Figure 6(a) and (b) are the 1-skeleton of a cube and its spanning tree, respectively. In our problem, given a box of size $a \times b \times c$, we divide the faces into unit squares, and cut the surface along edges of the unit squares. We call such a development a *unit square development*.

¹ Since the word “net” has several meaning, we use “development” instead of it to make clear.

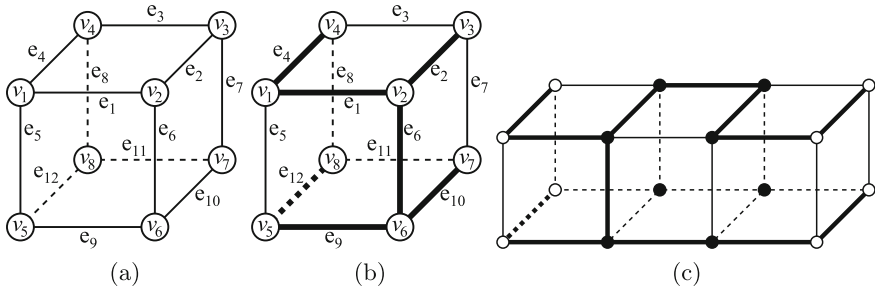


Fig. 6. 1-skeletons and spanning trees of a cube and a box of size $1 \times 1 \times 3$.

In Fig. 6(c), we regard the eight vertices (colored in white) as special, where the angle sum at each corner is 270° . We call them *corners*. The 1-skeleton of a box is given as $G = (V_c \cup V_o, E)$, where V_c and V_o denote the sets of eight corners and others, respectively, and E denote the set of edges of unit length. The cut edges of a unit square development form a tree spanning to the eight corners.

Now, we go back to the common development. It is easy to see that two boxes of size $a \times b \times c$ and size $a' \times b' \times c'$ have a common unit square development only if they have the same surface area, i.e., $2(ab + bc + ca) = 2(a'b' + b'c' + c'a')$. Such 3-tuples (a, b, c) can be computed by a simple enumeration for small areas (Table 1), but it seems that we have many corresponding 3-tuples for large area. In fact, this intuition can be proved as follows:

Theorem 2 [13]. *We say two 3-tuples (a, b, c) and (a', b', c') are distinct if and only if $a \neq a'$, $b \neq b'$, or $c \neq c'$. For any positive integer p , there are p distinct 3-tuples (a_i, b_i, c_i) for $i = 1, 2, \dots, p$ such that $a_i b_i + b_i c_i + c_i a_i = a_j b_j + b_j c_j + c_j a_j$ for any $1 \leq i, j \leq p$.*

Proof. For a given p , we let $a_i = 2^i - 1$, $b_i = 2^{2p-i} - 1$, $c_i = 1$ for $i = 1, 2, \dots, p$. Then we have $a_i b_i + b_i c_i + c_i a_i = (2^{2p} - 2^i - 2^{2p-i} + 1) + (2^{2p-i} - 1) + (2^i - 1) = 2^{2p} - 1$ for any i . It is easy to see that all 3-tuples (a_i, b_i, c_i) are distinct. Thus we have the theorem. □

By Theorem 2, we can consider any number of boxes that may share the common developments.

2.2 Enumeration by Zero-Suppressed Binary Decision Diagrams

A *zero-suppressed binary decision diagram (ZDD)* [11] is directed acyclic graph that represents a family of sets. As illustrated in Fig. 7, it has the unique source node², called *the root node*, and has two sink nodes 0 and 1, called *the 0-node* and *the 1-node*, respectively (which are together called the constant nodes). Each of the other nodes is labeled by one of the variables x_1, x_2, \dots, x_n , and has exactly two outgoing edges, called *0-edge* and *1-edge*, respectively. On every path from the root node to a constant node in a ZDD, each variable appears at most once in the same order. The size of a ZDD is the number of nodes in it.

² We distinguish *nodes* of a ZDD from *vertices* of a graph (or a 1-skeleton).

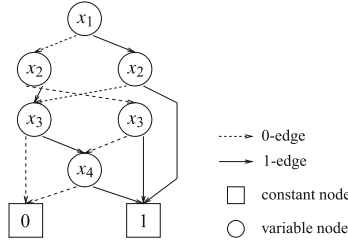


Fig. 7. A ZDD representing $\{\{1, 2\}, \{1, 3, 4\}, \{2, 3, 4\}, \{3\}, \{4\}\}$.

Every node v of a ZDD represents a family of sets \mathcal{F}_v , defined by the subgraph consisting of those edges and nodes reachable from v . If node v is the 1-node (respectively, 0-node), \mathcal{F}_v equals to $\{\{\}\}$ (respectively, $\{\}$). Otherwise, \mathcal{F}_v is defined as $\mathcal{F}_{0\text{-succ}(v)} \cup \{S \mid S = \{var(v)\} \cup S', S' \in \mathcal{F}_{1\text{-succ}(v)}\}$, where $0\text{-succ}(v)$ and $1\text{-succ}(v)$, respectively, denote the nodes pointed by the 0-edge and the 1-edge from node v , and $var(v)$ denotes the label of node v . The family \mathcal{F} of sets represented by a ZDD is the one represented by the root node. Figure 7 is a ZDD representing $\mathcal{F} = \{\{1, 2\}, \{1, 3, 4\}, \{2, 3, 4\}, \{3\}, \{4\}\}$. Each path from the root node to the 1-node, called *1-path*, corresponds to one of the sets in \mathcal{F} .

Now, we focus on the enumeration of developments by ZDDs. As denoted in Sect. 2.1, the cut edges of an edge development form a spanning tree of the 1-skeleton (e.g., edges $\{e_1, e_2, e_4, e_7, e_6, e_9, e_{10}\}$ in Fig. 6(b)). This conditions can be interpreted as follows:

Property 1. Given the 1-skeleton $G = (V, E)$ of a polyhedron, the cut edges of its edge development is the set of edges $E_d (\subseteq E)$ satisfying: (1) E_d has no cycle. (2) Subgraph of G induced by E_d has only one connected component. (3) Each vertex in V is adjacent to at least one edge in E_d .

Algorithm 1 [4] gives the frontier-based search [9] to construct a ZDD representing a family of spanning trees. It can be considered as one of DP-like algorithms. Each search node in the algorithm corresponds to a subgraphs of the given graph G . The search begins with node_{root} (i.e., the root node of the resulting ZDD) corresponding to $(V, \{\})$. In the search, we check whether we can adopt edge e_i or not, in the order of $i = 1, 2, \dots, m$, where m is the number of edges in G . In Line 4 of Algorithm 1, current search node is \hat{n} , and in case $x = 1$ (respectively, $x = 0$), we adopt (respectively, do not adopt) e_i . Search node n' corresponds to the resulting graph, and is pointed by the x -edge of \hat{n} in Line 13.

The key is to share nodes of the constructing ZDD (in Lines 9 and 10) by simple “knowledge” of subgraphs, and not to traverse the same subproblems more than once. Each search node \hat{n} in the algorithm has an array $\hat{n}.comp[]$ as an knowledge, where $\hat{n}.comp[v_j]$ indicates the ID of the connected component v_j belonging to. We can reduce the size of knowledge by maintaining the values of $\hat{n}.comp[]$ just for vertices incident to both a processed and an unprocessed edges. Such set of vertices are called the i -th *frontier* $F_i (\in V)$, which is formally

Algorithm 1. Construct ZDD

Input : Graph $G = (V, E)$ with n vertices and m edges
Output: ZDD representing a family of spanning trees in G

```

1  $N_1 := \{\text{node}_{\text{root}}\}$ .  $N_i := \{\}$  for  $i = 2, 3, \dots, m + 1$ 
2 for  $i := 1, 2, \dots, m$  do
3   foreach  $\hat{n} \in N_i$  do
4     foreach  $x \in \{0, 1\}$  do           // 0-edge and 1-edge
5        $n' := \text{CheckTerminal}(\hat{n}, i, x)$  // returns 0, 1, or nil
6       if  $n' = \text{nil}$  then           //  $n'$  is neither 0 nor 1
7         Copy  $\hat{n}$  to  $n'$ 
8         UpdateInfo( $n', i, x$ )
9         if there exists  $n'' \in N_{i+1}$  that is identical to  $n'$  then
10           $n' := n''$ 
11        else
12           $N_{i+1} := N_{i+1} \cup \{n'\}$ 
13      Create the  $x$ -edge of  $\hat{n}$  and make it point at  $n'$ 

```

defined as $F_i = (\cup_{j=1, \dots, i} e_j) \cap (\cup_{j=i+1, \dots, m} e_j)$, $F_0 = F_m = \{\}$. We check whether the subgraph corresponding to the search node \hat{n} consists a spanning tree in Procedure CheckTerminal. For more detail, see [9].

3 Algorithms for the First Two Boxes of Size $1 \times 1 \times 7$ and $1 \times 3 \times 3$

3.1 Algorithm Based on ZDDs

We first describe how to obtain all common unit cube developments of two incongruent boxes of sizes $1 \times 1 \times 7$ and $1 \times 3 \times 3$ by ZDDs. The strategy is simple: For each box, we enumerate sets of cut edges corresponding to unit cube developments, and convert them to the shapes of the developments, each of which is represented by a sequence of interior angles of a polyomino. Then, we obtain common developments that appear in both of the two boxes. The important thing is to enumerate the sets of cut edges efficiently. For obtaining unit cube developments, we generalize the algorithm given in Sect. 2.2. Once a ZDD is obtained, each of its 1-paths represents a set of cut edges. By traversing the ZDD, we can obtain 1-paths, and thus obtain the shapes of developments. The difference between the problem in Sect. 2.2 and ours can be seen in Fig. 6(b) and (c). In our problem, faces of our boxes are divided into unit squares, and we need to make a tree spanning to the eight corners, not spanning to all vertices. The cut edges of a unit square development of our box has the following property:

Property 2. Given the 1-skeleton $G = (V_c \cup V_o, E)$ of a box, the cut edges of its unit square development is the set of edges $E_d (\subseteq E)$ satisfying: (1) E_d has no cycle. (2) Subgraph of G induced by E_d has only one connected component of

size greater than 1. (3) Each vertex in V_c is adjacent to at least one edge in E_d . (4) No vertex in V_o is adjacent to exactly one edge in E_d .

Conditions (1) and (2) are essentially equivalent to those in Property 1. Condition (3) is to flatten the corners of the box into a plane. Conditions (2) and (3) guarantees that all vertices in V_c are connected. Condition (4) is to avoid a vertex in V_o adjacent to exactly one edge in E_d . (If there exists such an edge, we can eliminate it from E_d .) Conditions (2) and (4) guarantees that all vertices in V_o adjacent to two or more edges are connected to the vertices in V_c . Thus, we have a tree spanning the vertices in V_c .

To check the above conditions, we modify Procedures UpdateInfo and CheckTerminal. For counting the number of adopted edges adjacent to v_j and the size of connected component v_j belonging to, we prepare two arrays $\hat{n}.\text{deg}[\]$ and $\hat{n}.\text{size}[\]$. In Procedure 3, we initialize $\hat{n}.\text{deg}[v_j] := 0$ (i.e., the number of adopted edges in E_d adjacent to v_j is 0) and $\hat{n}.\text{size}[v_j] := 1$ (i.e., vertex v_j is a singleton) in Line 3. If edge $e_i = (v_{i_1}, v_{i_2})$ is adopted to E_d (i.e., $x = 1$), we update the degrees of v_{i_1} and v_{i_2} , and the size of their connected components in Lines 8, 9 and 12.

In Procedure 2, Condition (1) is checked in Lines 2–4. If vertex v_j leaves from the frontier, we have no chance to adopt its adjacent edges, which means the degree of v_j does not change. Thus, we check Conditions (3) and (4) in Lines 8 and 9, respectively. At the same time, we have no chance to grow the size of v_j 's connected components. Thus, we check whether we have two or more connected components in Lines from 14 to 16, and terminate the search if it holds. Otherwise, we have only one connected component, and hence we cannot adopt any edges in the remaining search. Thus, we check Conditions (3) and (4) in Lines from 17 to 22, and returns the result.

3.2 Algorithm Based on Exhaustive Search

Here we describe the exhaustive algorithm for generating all common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$. The basic idea is similar to one in [1]: Let L_i be the set of all common partial developments of area i of two boxes. Then L_1 consists of a unit square, and each L_i with $i > 1$ is a subset of polyominoes of size i that can be computed from L_{i-1} by the breadth first search. Each L_i is maintained by a huge hash table, which means that we use $O(\max_i \{i|L_i| + (i-1)|L_{i-1}|\})$ space for the computation of step i .

This simple idea works up to 22 for two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$ in [1] since the maximum number of $|L_i \cup L_{i-1}|$ takes 1.01×10^7 when $i = 18$. However, for the surface area 30, it does not work even on a supercomputer (CRAY XC30) due to memory overflow when $i = 22$.

Thus we divide the computation into two phases. In the first phase, we compute L_i for each $i = 2, \dots, 16$. As a result, we have L_{16} that consists of 7,486,799 common partial developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$. In the second phase, we partition L_{16} into 75 disjoint subsets L_{16}^j with $1 \leq j \leq 75$. For each L_{16}^j , we independently compute up to L_{30}^j in parallel by the BFS algorithm again. In the final step, we merge L_{30}^j with $1 \leq j \leq 75$, remove duplicates, and obtain L_{30} .

Procedure 2. CheckTerminalRevised(\hat{n}, i, x)

```

1 Let  $(v_{i_1}, v_{i_2})$  denote  $e_i \in E$ 
2 if  $x = 1$  then
3   if  $\hat{n}.comp[v_{i_1}] = \hat{n}.comp[v_{i_2}]$  then //  $v_{i_1}, v_{i_2}$  are in the same component
4     return 0 // we have a cycle by adding  $e_i$ 
5 Copy  $\hat{n}$  to  $n'$ 
6 UpdateInfo( $n', i, x$ )
7 foreach  $v_j \in \{v_{i_1}, v_{i_2}\}$  satisfying  $v_j \notin F_i$  do //  $v_j$  is leaving from the frontier
8   // Check the degree constraints for  $v_j$ 
9   if  $(v_j$  is in  $V_c$ ) and  $(\hat{n}.deg[v_j] = 0)$  then return 0
10  if  $(v_j$  is in  $V_o$ ) and  $(\hat{n}.deg[v_j] = 1)$  then return 0
11  if  $(\forall v_k \in F_i \hat{n}.comp[v_j] \neq \hat{n}.comp[v_k])$  then
12    //  $v_j$ 's connected component cannot connect to any other components
13    if  $(\hat{n}.size[v_j] > 1)$  then
14      if  $(\exists v_\ell \in F_i (\hat{n}.size[v_\ell] > 1))$  then
15        // we have two or more connected components of size  $> 1$ 
16        return 0
17      else // We cannot adopt any edges
18        foreach  $v_{j'} \in \cup_{i'=i+1, \dots, m} e_{i'}$  do
19          // Check the degree constraints for remaining vertices
20          if  $(v_{j'}$  is in  $V_c$ ) and  $(\hat{n}.deg[v_{j'}] = 0)$  then return 0
21          if  $(v_{j'}$  is in  $V_o$ ) and  $(\hat{n}.deg[v_{j'}] = 1)$  then return 0
22          return 1
23
24   $F_i := F_i \setminus \{v_j\}$ 
25 return nil

```

Procedure 3. UpdateInfoRevised(\hat{n}, i, x)

```

1 Let  $(v_{i_1}, v_{i_2})$  denote  $e_i \in E$ 
2 foreach  $v_j \in \{v_{i_1}, v_{i_2}\}$  such that  $v_j \notin F_{i-1}$  do //  $v_j$  is entering the frontier
3    $\hat{n}.comp[v_j] := j$  // The initial component ID is the index of  $v_j$ 
4    $\hat{n}.deg[v_j] := 0, \hat{n}.size[v_j] := 1$ 
5 if  $x = 1$  then // Merge two components of  $v_{i_1}, v_{i_2}$ 
6    $c_{min} := \min\{\hat{n}.comp[v_{i_1}], \hat{n}.comp[v_{i_2}]\}$ 
7    $c_{max} := \max\{\hat{n}.comp[v_{i_1}], \hat{n}.comp[v_{i_2}]\}$ 
8    $\hat{n}.deg[v_{i_1}] := \hat{n}.deg[v_{i_1}] + 1, \hat{n}.deg[v_{i_2}] := \hat{n}.deg[v_{i_2}] + 1$ 
9    $s = \hat{n}.size[v_{i_1}] + \hat{n}.size[v_{i_2}]$ 
10  foreach  $v_j \in F_i$  do
11    if  $\hat{n}.comp[v_j] = c_{max}$  then  $\hat{n}.comp[v_j] := c_{min}$ 
12    if  $(\hat{n}.comp[v_j] = c_{min})$  or  $(\hat{n}.comp[v_j] = c_{max})$  then  $\hat{n}.size[v_j] := s$ 
13 foreach  $v_j \in \{v_{i_1}, v_{i_2}\}$  such that  $v_j \notin F_i$  do //  $v_j$  is leaving the frontier
14   Forget  $\hat{n}.comp[v_j], \hat{n}.deg[v_j]$  and  $\hat{n}.size[v_{i_2}]$ 

```

4 Algorithm for the Third Box

Let L_{30} be the set of all common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$. We here note that if we can compute L_{30} efficiently, we can check in the same manner; that is, we generate all developments of the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ by cutting along the line of unit squares, and check if each one appears in L_{30} or not. Thus, in the first method based on ZDDs, we can use the same way again; we construct all developments of the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ based on the connection network on unit squares, and check if each one appears in L_{30} or not. In the second method based on the exhaustive search for two boxes, we check if each development in L_{30} can be folded into a cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$.

The program of the first method based on ZDDs runs on a usual desktop computer with Intel Xeon E5-2643 and 128 GB memory. It takes 0.10 and 71.53 s for obtaining the sets of cut edges of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$, respectively, and 7.7 days for converting the cut edges into the shapes of developments and for obtaining the common developments. For the third box of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$, It takes 354.64 s for obtaining cut edges, and 2.5 days for obtaining the common developments of the three boxes. It takes 10.2 days in total. The program of the second method runs, in total, in 3 months on the supercomputer (CRAY XC30), and we obtain 1,080 common developments in L_{30} of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$ ³ and 9 common developments of three boxes of size $1 \times 1 \times 7$, $1 \times 3 \times 3$ and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$.

5 Concluding Remarks

Recently, Shirakawa and Uehara showed infinite families of orthogonal polygons that fold into three incongruent orthogonal boxes [16]. However, the smallest polygon contains 532 unit squares. In this paper, we show that there exist orthogonal polygons of 30 unit squares that fold into three incongruent orthogonal boxes if we allow us to fold along slanted lines. In the original framework in [16], the smallest possible surface area that may fold into three different boxes is 46, which may produce three boxes of size $1 \times 1 \times 11$, $1 \times 2 \times 7$, and $1 \times 3 \times 5$. We conjecture that there exists an orthogonal polygon of 46 unit squares that admits to fold these three boxes. Some nontrivial properties in Figs. 4 and 5 may help to find it.

There are many future work in this area. For example, does there exist a polyomino that folds into four or more boxes? Is there some upper bound of the number of boxes which can be folded from one polyomino? We remind that Theorem 2 says that we have no upper bound by the constraint of the surface areas. But it is hard to imagine that one polyomino can fold into, say, 10,000 different boxes. General development/folding problems are also remained open. For example, Shirakawa et al. found a common development of a unit cube and an almost regular tetrahedron (with relative error $< 2.89200 \times 10^{-1796}$) [15], however, a common development of two Platonic solids are still open.

³ We note that the maximum number of partial developments is given when $j = 24$.

References

1. Abel, Z., Demaine, E., Demaine, M., Matsui, H., Rote, G., Uehara, R.: Common development of several different orthogonal boxes. In: 23rd Canadian Conference on Computational Geometry (CCCG 2011), pp. 77–82 (2011)
2. Akiyama, J.: Tile-makers and semi-tile-makers. *Math. Assoc. Amerika* **114**, 602–609 (2007)
3. Akiyama, J., Nara, C.: Developments of polyhedra using oblique coordinates. *J. Indonesia. Math. Soc.* **13**(1), 99–114 (2007)
4. Araki, Y., Horiyama, T., Uehara, R.: Common unfolding of regular Tetrahedron and Johnson-Zalgaller solid. In: Rahman, M.S., Tomita, E. (eds.) WALCOM 2015. LNCS, vol. 8973, pp. 294–305. Springer, Heidelberg (2015)
5. Biedl, T., Chan, T., Demaine, E., Demaine, M., Lubiw, A., Munro, J.I., Shallit, J.: Notes from the University of Waterloo Algorithmic Problem Session, 8 September 1999
6. Demaine, E.D., O’Rourke, J.: *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, Cambridge (2007)
7. Golomb, S.W.: *Polyominoes*. Princeton University Press, Princeton (1994)
8. Kane, D., Price, G.N., Demaine, E.D.: A pseudopolynomial algorithm for Alexandrov’s theorem. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 435–446. Springer, Heidelberg (2009)
9. Kawahara, J., Inoue, T., Iwashita, H., Minato, S.: Frontier-based search for enumerating all constrained subgraphs with compressed representation. Technical report TCS-TR-A-14-76, Division of Computer Science, Hokkaido University (2014)
10. Lubiw, A., O’Rourke, J.: When can a polygon fold to a polytope? Technical report 048. Department of Computer Science, Smith College (1996)
11. Minato, S.: Zero-suppressed bdds for set manipulation in combinatorial problems. In: 30th ACM/IEEE Design Automation Conference (DAC 1993), pp. 272–277 (1993)
12. Mitani, J., Uehara, R.: Polygons folding to plural incongruent orthogonal boxes. In: Canadian Conference on Computational Geometry (CCCG 2008), pp. 39–42 (2008)
13. Okumura, T.: Personal communication, August 2014
14. O’Rourke, J.: *How to Fold It: The Mathematics of Linkage, Origami and Polyhedra*. Cambridge University Press, Cambridge (2011)
15. Shirakawa, T., Horiyama, T., Uehara, R.: Construct of common development of regular tetrahedron and cube. In: 27th European Workshop on Computational Geometry (EuroCG 2011), pp. 47–50, 28–30 March 2011
16. Shirakawa, T., Uehara, R.: Common developments of three incongruent orthogonal boxes. *Int. J. Comput. Geom. Appl.* **23**(1), 65–71 (2013)