| | |
|---|---|
| Title | ポリキューブの展開図に関する研究 |
| Author(s) | Xu, Dawei |
| Citation | |
| Issue Date | 2017-09 |
| Type | Thesis or Dissertation |
| Text version | ETD |
| URL | http://hdl.handle.net/10119/14825 |
| Rights | |
| Description | Supervisor:上原　隆平，情報科学研究科，博士 |

# Research on Developments of Polycubes

by

Dawei XU

**submitted to**
**Japan Advanced Institute of Science and Technology**
**in partial fulfillment of the requirements**
**for the degree of**
**Doctor of Philosophy**

*Supervisor:* Ryuhei Uehara

*School of Information Science*
*Japan Advanced Institute of Science and Technology*

Sept, 2017

**Doctoral Dissertation**

# Research on Developments of Polycubes

Dawei XU

*Supervisor:* Ryuhei Uehara

*School of Information Science*
*Japan Advanced Institute of Science and Technology*

Sept, 2017

# Abstract

In this thesis, we study the folding and unfolding problems of *polycubes*. A polycube is a kind of polyhedron that composed of identical cubes. It has an important role in the field of computational geometry.

The first problem is about the common developments that can fold to two or more incongruent polycubes, *boxes*. These developments are simple polyominoes that consist of unit squares of four connected so that the boxes and their corresponding developments have the same surface area. In searching for common developments that fold to two or more boxes, we start from a relatively simple task: finding two or more boxes that their surface areas are equal, but sizes are different.

The smallest surface area is 22 which admits to fold to two boxes of size $1\times1\times5$ and $1\times2\times3$. All common developments of these two boxes were enumerated by Matsui in 2011. The next smallest integer $n$ such that surface area $n$ can fold to two different boxes is 30. Previous study tried to list the common developments of two different boxes of size $1\times1\times7$ and $1\times3\times3$ of surface area 30, however, it failed due to the limitation of computational power. We fill this gap and research further.

By a new algorithm on a parallel computer, we first enumerate all common developments of two boxes of size $1\times1\times7$ and $1\times3\times3$. Starting from the partial development of surface area 1, we repeat adding one new unit square to all possible places of the partial development to generate new developments. In each loop, the new developments will get checked whether they can cover the boxes or not. The algorithm loops $n$ times, where $n = 30$.

Finally, we obtained all common developments of two boxes of size $1\times1\times7$ and $1\times3\times3$. There are 1080 common developments.

Surface area 30 is consistent to the surface area of a cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$. Therefore, next, we try to find a polyomino of surface area 30 that fold to these three different boxes. It is a great improvement from previously known one that the smallest surface area that folds to three different boxes is greater than 500. To achieve that, we propose a

new algorithm. It determines if a polyomino $P$ that can fold to two boxes of size $1\times1\times7$ and $1\times3\times3$ can also fold to a box of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$.

As a result, nine out of 1080 developments can fold to the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$. While eight developments have only one way of folding to the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$, the other development has two different ways of folding to it, for there is an angle of 26.6 degrees to the unit square's edge clockwise and counter-clockwise. That is, the last development can actually fold to 3 boxes in 4 different ways of folding: the box of size $1\times1\times7$, the box of size $1\times3\times3$, and 2 ways of the box of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$.

In these results, there are some common developments with a special property that they are central symmetric. We call them the *centrosymmetric* common developments.

The way to enumerate the centrosymmetric common developments is a bit different from the enumeration of normal common developments, which needs more computations. However, the size of data is relatively small, and it runs in loops of $n/2$ in total, where $n$ is its surface area.

As a result, in 2263 common developments of two boxes of $1\times1\times5$ and $1\times2\times3$, only 45 centrosymmetric common developments exist. We confirm the exact numbers of centrosymmetric common developments from surface area 22 to surface area 54. Especially, we show that there is no centrosymmetric common development of three different boxes of surface area 46. The surface area 46 is the smallest candidates of three boxes of integral size of $1 \times 1 \times 11$, $1 \times 2 \times 7$, and $1 \times 3 \times 5$. It is still open that if there is a common development of three boxes of these sizes, and we give a partial answer to this open problem.

The second part of this thesis is research on the "rep-cube", which is a new notion derived from the classic notion of "*rep-tile*" with the idea of the development. A *rep-tile* of order $k$ is a polygon that can be divided into $k$ replicas congruent to each another and similar to the original one. It was proposed by Solomon W. Golomb in 1962. In 2016, a new notion of rep-cube was proposed. A polyomino is a *rep-cube* of order $k$ if it is a development of a cube, and it can be divided into $k$ polyominoes such that each of them can fold to a cube. If each of these $k$ polyominoes has the same size, we call the original polyomino a *regular* rep-cube of order $k$.

A recent study shows that there exist regular rep-cubes of order $k$ for each $k =$

$2, 4, 5, 8, 9, 36, 50, 64$, and also $k = 36gk'^2$ for any positive integer $k'$ and an integer $g$ in $\{2, 4, 5, 8, 9, 36, 50, 64\}$. That is, there are infinitely many $k$ that allow regular rep-cube of order $k$. These results lead us to the following natural question: how many rep-cubes of order $k$ exist for some $k$?

As a consequence, we enumerate all regular rep-cubes of order 2 and 4 of surface area 12 and 24, respectively. For example, there are 33 rep-cubes of order 2 of surface area 12; that is, there are 33 dodecominoes that can fold to a cube of size $\sqrt{2} \times \sqrt{2} \times \sqrt{2}$ and each of them can be divided to two developments of the unit cube. Similarly, there are 7185 regular rep-cubes of order 4 of surface area 24.

**keywords:** Polyomino; folding algorithm; enumeration; polyhedron.

# Acknowledgement

I am grateful to my supervisor, Prof. Ryuhei Uehara, whose expertise, understanding, generous guidance and support made it possible for me to work on a topic that was of great interest to me. It was a pleasure working with him.

I am grateful to Assoc. Prof. Takashi Horiyama for giving his experience in enumerating algorithm and help me in off-campus research.

I would also like to thank Ms. Tomoko Taniguchi, her keen eyes and expertise in English help me remove spelling mistakes.

I would like to express my gratitude to the staff of RCACI (Research Center for Advanced Computing Infrastructure) for replying my question on parallel computers.

I would also like to thank my friends for accepting nothing less than excellence from me. Last but not the least, I would like to thank my parents, who have raised me up and provided me through moral and emotional support in my life.

Thank you!

# Contents

# List of Figures

# Chapter 1

# Introduction and Background

In 1525, the German painter and thinker Albrecht Dürer published his masterwork on geometry "On Teaching Measurement with a Compass and Straightedge", which opened the area of computational geometry with a lot of open problems [1]. That area consists of the folding and unfolding problems. Despite a long history, it has not been studied extensively in the mathematical literature until decades ago.

In general, we are interested in how polyhedra can be reconfigured to certain constraints depending on the type of object and the problem of interest. Typically, the process of *unfolding* approaches a more basic shape, whereas *folding* complicates the shape [2].

In the big picture of the area, there are three sections corresponding to the type of the object being folded: linkages, origami, and polyhedra. The problem we introduce belongs to the polyhedra area. We study the folding and unfolding problems of the *polycubes*, which are polyhedra that composed of identical unit cubes in face-to-face gluing manner.

The development is the unfolding obtained by slicing the surface of the solid, and it forms a single connected simple polygon without self-overlap [5]. In this thesis, as developments, we only consider orthogonal polygons that consist of unit squares in edge-to-edge gluing manner, which are called *polyominoes* [16].

Recently, a research presents a new paper-building design methodology using the research of common development of plural boxes. Paper, as an innovative material, has many advantages such as durability, lightweight, environment-friendly, and aesthetic property. In particular, in resource-limited societies, these advantages are extremely impor-

tant. The designer can simplify the paper building or part of the paper building as plural orthogonal polyhedra, which are actually boxes. Then the designer can choose a proper common development that can fold to these boxes to build the house.

## 1.1 Common Developments of Boxes

In the first half of this thesis, we study common developments that can fold to two or more incongruent orthogonal convex polyhedra, namely, *boxes*. This folding problem is very natural however quite counterintuitive; for a given polyomino that consists of unit squares, the problem asks are there two or more ways to fold it to different simple convex orthogonal polyhedra (Figure 1.1).

The research starts from a problem proposed by Lubiw and O'Rourke in 1996 [3], which is about polygons that can fold to a (convex) polyhedron. In 2007, Demaine and O'Rourke published a scholarly book, which is about geometric folding algorithms [5].

In general, we can state the development/folding problem as follows:

---

1 **Input** A polygon $P$ and a polyhedron $Q$.

2 **Output** Determine whether $P$ can fold to $Q$ 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 [11, 12]. Except that, we have quite a few results from the mathematical viewpoint. Hence, we can tackle this problem from the viewpoints of computational geometry and algorithms.

For example, Demaine and O'Rourke gave an $O(n^3)$ time algorithm for a convex polyhedron $Q$ in [5]. Precisely, the algorithm determines if $P$ can fold to *some convex* polyhedron $Q$, however, it does not give the concrete *shape* of $Q$. The algorithm computes the matching of edges for gluing in $P$ to fold to a convex polyhedron $Q$, and it also determines the vertices of $Q$ by checking the curvature of the vertices. However, it does not give the crease pattern of $P$ for folding, which forms the set of edges on $Q$. We remark that the shape is uniquely determined from its convexity of $Q$ by the Alexandrov's Theorem (see [4]). Moreover, such a shape can be determined by the pseudopolynomial

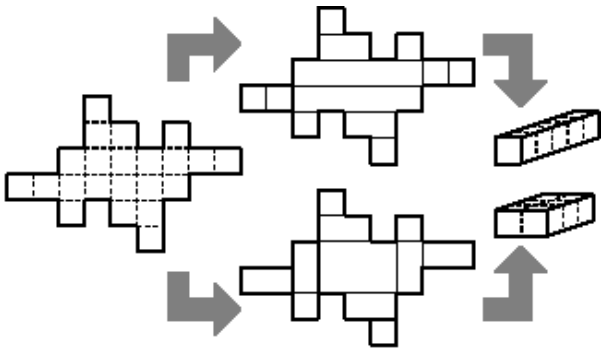


Figure 1.1: A polygon folding to two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$ in [9].

Figure 1.2: Cubigami.

time algorithm for Alexandrov's Theorem proposed in [4]. However, the algorithm in [4] runs in $O(n^{456.5})$ time (with some geometric parameters), and hence, it is not practical so far. Thus, we still have to decide the crease lines to make the shape by experiments. In other words, it is quite difficult to determine the *shape* $Q$ obtained from a polygon $P$ by folding even if we can find its edge matching of gluing. Recently, some restricted cases were discussed in [13]; they give efficient algorithms for the cases that $P$ is a petal polygon, and $Q$ is a pyramid or its variants.

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. We call this kind of polygon as polyomino, which was proposed by Solomon W. Golomb in 1954.

Such polygons have many applications including toys and puzzles. For example, the puzzle "cubigami" (Figure 1.2) 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. We can find some related results in the books on geometric folding algorithms by Demaine and O'Rourke [5, 1].

In searching for common developments that fold to two or more boxes, we start from a relatively simple task: finding two or more boxes whose surface areas are equal, but sizes are different.

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 may admit to fold to two or more

| $2(ab+bc+ca)$ | $a \times b \times c$ |
|---|---|
| 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 | $1 \times 1 \times 15$, $1 \times 3 \times 7$, $2 \times 3 \times 5$ |

Table 1.1: A part of possible size $a \times b \times c$ of boxes and its common surface area $2(ab + bc + ca)$.

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 1.1. From the table, we can say that the smallest surface area is 22 to have a common development of two boxes, and then their sizes are $1 \times 1 \times 5$ and $1 \times 2 \times 3$ [7]. In fact, Abel et al. have confirmed that there exist 2263 common developments of two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$ by an exhaustive search [6]. On the other hand, the smallest surface area that may admit to fold to three boxes is 46, which may fold to three boxes of size $1 \times 1 \times 11$, $1 \times 2 \times 7$, and $1 \times 3 \times 5$. However, the number of polyominoes of area 46 seems to be too huge to search. This number is strongly related to the enumeration and counting of polyominoes [16]. 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 of [8] (see the OEIS (https://oeis.org/A000105) for the references). Therefore, 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 $1 \times 3 \times 5$.

We follow the previous research and extend it from the case of the surface area 22 in Table 1.1. The next area after 22 in the table is 30, which admits to fold to two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$. When Abel et al. confirmed the area 22 in 2011, it took around 10 hours. Thus, we cannot use the straightforward way in [6] for the area 30.

Also, if we use the same way as one for area 22 shown in [6], it will consume too much memory. Therefore, we use a hybrid search of the breadth first search and the depth first

search. We first enumerate all common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$. There are 1080 common developments in total.

In [14], they proposed that there was a polygon that folded to two boxes of size $1 \times 2 \times 4$ and $\sqrt{2} \times \sqrt{2} \times 3\sqrt{2}$ (Figure 1.3). Its crease lines of the second one are not folded along the edges of the unit squares. In this context, we can observe that the area 30 may admit to fold to another box of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ by folding along the diagonal lines of rectangles of size $1 \times 2$. 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 these 1080 common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$.

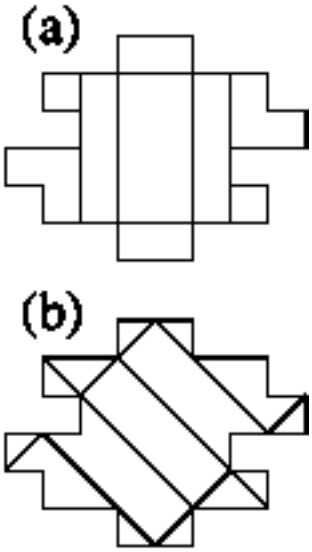



Figure 1.3: The common development shown in [14]. (a) It folds to a box of size $1 \times 2 \times 4$, and (b) it also folds to a box of size $\sqrt{2} \times \sqrt{2} \times 3\sqrt{2}$.

We remark that this is a special case of the development/folding problem mentioned above. In our case, $P$ is one of the 1080 polygons that consists of 30 unit squares, and $Q$ is the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$. We can use a pseudopolynomial time algorithm for Alexandrov's Theorem proposed in [4], however, it runs in $O(n^{456.5})$ time, which is not practical [13]. Alexandrov's Theorem states that every metric with the global topology and local geometry required of a convex polyhedron is in fact the intrinsic metric of some convex polyhedra. Thus, if $P$ is a development of a convex polyhedron, then the shape is uniquely determined [22].

For a given orthogonal polygon and the size of a box, to check whether the orthogonal

polygon can fold to the box, Horiyama and Mizunashi recently developed an efficient algorithm [21]. That algorithm runs in $O((n+m)\log n)$ time, where $n$ is the number of vertices in $P$, and $m$ is the maximum number of line segments that appear on a crease line. We remark that the parameter $m$ is hidden and can be huge comparing to $n$. In our case, $P$ is a polyomino, and this hidden parameter is linear to the number of unit squares in $P$.

Based on these results, we propose an efficient algorithm specialized in our case that checks whether a polyomino $P$ of area 30 can fold to 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 $1\times1\times7$ and $1\times3\times3$ can also fold to the third box of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ and obtain an affirmative answer. We find that nine of 1080 common developments of two boxes of size $1\times1\times7$ and $1\times3\times3$ can fold to the third box of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ (Figure 3.16). Moreover, one of the nine common developments of three boxes has another way of folding. Precisely, the last one (Figure 3.16(9)) admits to fold to 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 Figure 3.17.

We summarize the main results:

**Theorem 1.** *(1) There are 1080 polyominoes of area 30 that admit to fold (along the edges of unit squares) to two boxes of size* $1\times1\times7$ *and* $1\times3\times3$*. (2) Among the above 1080, nine polyominoes can fold to the third box of size* $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ *if we admit to fold along diagonal lines. (3) Among these nine polyominoes, one can fold to the third box in two different ways.*

In these results, there are several common developments that have a special property that they are central symmetric. We call it the *centrosymmetric* common development.

The way to enumerate the centrosymmetric common developments is a bit different to the enumeration of normal common developments, which needs more computation. However, the size of data is relatively small. Moreover, let $n$ be the surface area of common developments, then the algorithm runs in $n/2$ loops.

Based on these observations, we enumerated all centrosymmetric common developments of surface area $n$ for each of $n = 22, 30, 34, 38, 46, 54$. As results, we confirmed that there is no centrosymmetric common development of three boxes of surface area $n$ up to 54. These results give a partial answer to an open problem that asks if there is a common

development of three boxes of surface area 46 (and 54).

## 1.2 Research of Rep-cubes

As mentioned above, the definition of *polyomino* is a "simply connected" set of unit squares. This definition was introduced by Solomon W. Golomb in 1954. Since then, a set of polyominoes has been playing an important role in recreational mathematic society (see, e.g., [16], [18]). In Figure 82 in [18], it was shown that the set of 12 pentominoes can fold to a cube of size $\sqrt{10} \times \sqrt{10} \times \sqrt{10}$ by gluing them properly.

Eight years after Golomb proposed the notion of polyomino, he proposed another notion of "*rep-tile*" in 1962 [19]. A *rep-tile* of order $k$ is a polygon that can be divided into $k$ replicas congruent to one another and similar to the original (see [19, Chap 19]). Some examples of rep-tile are shown in Figure 1.4.

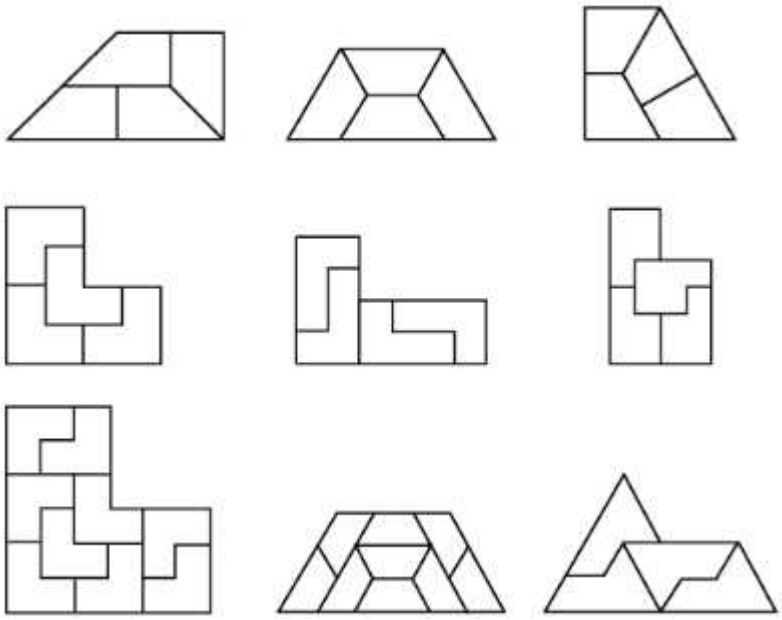


Figure 1.4: Rep-tiles.

Inspired by the notion of rep-tile and polyomino, a question was proposed: Is there any polyomino that can fold to a cube and can be divided into $k$ pieces such that each piece is a development of a cube for some $k$?

To answer this question, a new notion of rep-cube was proposed in 2016 [17]. A *rep-cube* of order $k$ is a polyomino that is a development of a cube, and it can be divided

into $k$ polyominoes such that each of them can be folded to a cube. If each of these $k$ polyominoes has the same size, we call the original polyomino a *regular* rep-cube of order $k$. It becomes another branch of the geometric folding problem.

For each of rep-cubes, crease lines for the developments may be different. Although each polyomino consists of unit squares, the crease lines may not be orthogonal to the edges of unit squares when it folds to a cube. For example, a regular rep-cube of order 2 folds to a cube by folding along the diagonals of unit squares, see Figure 1.5.

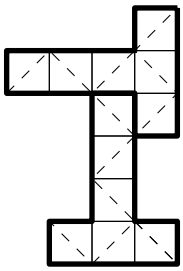


Figure 1.5: A regular rep-cube of order 2; each $T$ shape can fold to a cube, and this shape itself can fold to a cube of size $\sqrt{2} \times \sqrt{2} \times \sqrt{2}$ by folding along the dotted lines.

In [17], they show that there exist regular rep-cubes of order $k$ for each $k = 2, 4, 5, 8, 9, 36, 50, 64$, and also $k = 36gk'^2$ for any positive integer $k'$ and an integer $g$ in $\{2, 4, 5, 8, 9, 36, 50, 64\}$. That is, there are infinitely many $k$ that allows regular rep-cubes of order $k$.

In this thesis, we enumerate all regular rep-cubes of order $k$ for small $k$. We mention that the following problem is not so easy to solve efficiently: for a given polygon $P$, determine if $P$ can fold to a cube or not. We can use a similar technique above that checks the positional relationships of each unit square on the development in our case.

As a result, we enumerate all regular rep-cubes of order 2 and 4 of certain surface areas. For example, there are 33 rep-cubes of order 2 of surface area 12; that is, there are 33 dodecominoes that can fold to a cube of size $\sqrt{2} \times \sqrt{2} \times \sqrt{2}$, and each of them can be divided into two developments of a unit cube. Similarly, there are 7185 rep-cubes of order 4 of surface area 24.

We summarize the main results of this part:

**Theorem 2.** *(1) There are 33 regular rep-cubes of order 2 of surface area 12 that can fold to a cube of size* $\sqrt{2} \times \sqrt{2} \times \sqrt{2}$. *(2) There are 7185 regular rep-cubes of order 4 of surface area 24 that can fold to a cube of size* $2 \times 2 \times 2$.

## 1.3 Contents of Thesis

This thesis is organized as follows. In Introduction, we give a brief introduction about the polyominoes and polycubes, the history of the problems, and the application. The big picture of our original problem is about the folding and unfolding problems of polygons and polyhedra.

In Preliminaries, we give definitions to the terms we use in this thesis, and we describe some common techniques we use in the following chapters.

In the third chapter, we describe the approach of searching for the common developments of three boxes and the results of experiments.

As one special kind of development, the centrosymmetric common developments appear to be hard to find. The way for enumerating the centrosymmetric common developments is different from the normal ones, which needs more computation.

In Chapter 4, we describe the approach of enumerating all regular rep-cubes of order 2 and 4 and the results of experiments. We also propose a new notion of *uniform* rep-cubes that is a special kind of regular rep-cubes consist of pieces of the same shape.

In the last chapter, the conclusion of this thesis is given. We describe the main contribution. Some discussions are conducted, and open challenges are given. These challenges are worth solving, which can contribute to the computational geometry domain. The searching for a common development of three boxes still does not come to an end. Besides, the problem of finding regular rep-cubes of a larger order can be the future subject.

# Chapter 2

# Preliminaries

## 2.1 Developments

In [5], we can find the definition of the *development* of a polyhedron as the *net*. In this thesis, we use the "development" instead of "net" since the word "net" has several meanings. Briefly, the development is the unfolding obtained by cutting the surface of the solid, and it forms a single connected simple polygon without self-overlap. The *common development* of two (or more) solids is the development that can fold to the solids. In this thesis, as developments, we only consider orthogonal polygons that consist of unit squares, which are called *polyominoes*. A polyomino is a simple polygon that consists of unit squares of four connected [16]. Two unit squares of the polyomino are *adjacent* if they share an edge interior to the polygon [20].

Polyominoes obtained from a development by removing some unit squares are called *partial developments* of it. We call a convex orthogonal polyhedron a *box*.

## 2.2 Theorems of Developments

The followings are known theorems of developments of convex polyhedra, which are useful in our research.

**Theorem 3.** *[9] Let P be a polygon that can fold to a box B. If P has a cut inside without hole, we glue it and obtain P' with no cut inside. Then P' can also fold to B (Figure 2.1).*

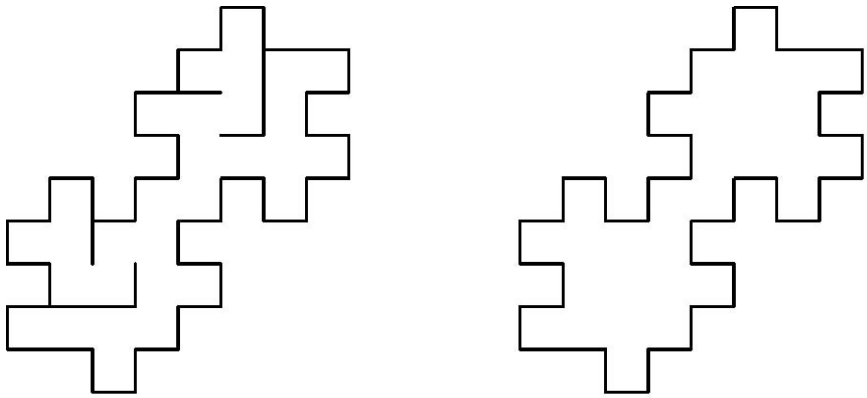


Figure 2.1: Development with a cut (left), development with no cut (right).

**Theorem 4.** *[9] Let B be an orthogonal box and P a polygon that can fold to B. Then, P is not necessarily simple (Figure 2.2).*

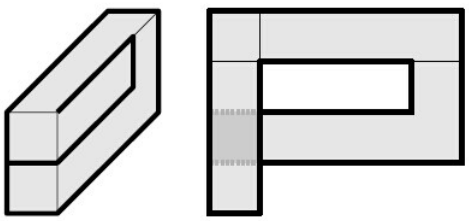


Figure 2.2: Development with an overlap.

**Theorem 5.** *[9] Let B be an orthogonal box and P a polygon that can fold to B. By Theorem 3, we assume that P contains no unnecessary cuts. We lay out P on the plane, and let P' be a silhouette of P. Then P is simple if and only if P' does not contain a hole.*

In this thesis, we only consider a simple polyomino without hole as a development of a polycube. In fact, as shown in Theorem 4, there are some polyomino $P$ which can be obtained from a development of a box such that its silhouette $P'$ is not simple, that is, $P'$ has a hole. In this case, to fold a box from $P'$, we have to cut the line in $P$ that joins the boundary of $P$ and the hole inside. This is another (further) topic, and we do not consider this case.

| $2(ab+bc+ca)$ | $a \times b \times c$ |
|---|---|
| 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 | $1 \times 2 \times 14$, $1 \times 4 \times 8$, $2 \times 2 \times 10$, $2 \times 4 \times 6$ |
| 90 | $1 \times 1 \times 22$, $2 \times 5 \times 5$, $3 \times 3 \times 6$ |
| 94 | $1 \times 1 \times 23$, $1 \times 2 \times 15$, $1 \times 3 \times 11$, $1 \times 5 \times 7$, $3 \times 4 \times 5$ |
| 112 | $1 \times 2 \times 18$, $2 \times 2 \times 13$, $2 \times 3 \times 10$, $2 \times 4 \times 8$, $4 \times 4 \times 5$ |
| 118 | $1 \times 1 \times 29$, $1 \times 2 \times 19$, $1 \times 3 \times 14$, $1 \times 4 \times 11$, $1 \times 5 \times 9$, $2 \times 5 \times 7$ |
| 136 | $1 \times 2 \times 22$, $2 \times 2 \times 16$, $2 \times 4 \times 10$, $2 \times 6 \times 7$, $3 \times 4 \times 8$ |
| 150 | $1 \times 1 \times 37$, $1 \times 3 \times 18$, $3 \times 3 \times 11$, $3 \times 4 \times 9$, $5 \times 5 \times 5$ |

Table 2.1: A part of possible size $a \times b \times c$ of boxes and its common surface area $2(ab + bc + ca)$.

## 2.3 Boxes with the Same Surface Area

In searching for common developments that fold to two or more boxes, we have a set: $P(S) = \{(a,\ b,\ c) \mid (ab + bc + ac) \times 2 = S\}$ ($S$, $a$, $b$, $c$ are positive integers, $0<a\leq b\leq c$). When $|P(S)|\geq 2$, there may exist a common development of $|P(S)|$ boxes of corresponding size. The list of boxes with the same surface area is shown in Table 2.1.

The smallest surface area that different boxes can appear in Table 2.1 is 22 which admits to fold to two boxes of size $1\times1\times5$ and $1\times2\times3$. All common developments of these two boxes have been already known. The next smallest integer $n$ such that surface area $n$ can fold to two different boxes is 30.

## 2.4 The Existing Algorithm

In searching for common developments of two different boxes of size 1×1×7 and 1×3×3 of surface area 30, we applied the algorithm that connects a unit square at a time from scratch. It starts from polyomino of one unit square. By edge-to-edge gluing unit squares, a large number of partial common developments are generated. The algorithm stops when the surface area of polyominoes is equal to the size of the surface area of given boxes.

## 2.5 Adjacency Lists of Boxes

To check whether a polyomino can cover a box without overlap or not, we use the adjacency list of the box.

We use an array to represent the position relationship of unit squares on the box. Every box has its unique position relationship of unit squares. Since Alexandrov's Theorem implies that, if $P$ is a development of a box, then the shape is uniquely determined. For example, Figure 2.3 and Table 2.2 are shown as the unfolding and adjacency list of the box of size $1 \times 1 \times 1$. Every unit square has four unique adjacency neighbors in four numbered directions. Thus, we can tell that a development can fold to the box or not.

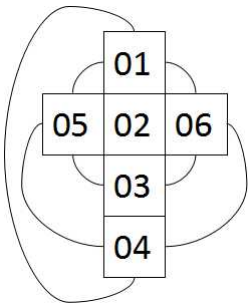



Figure 2.3: An unfolding of the box of size $1 \times 1 \times 1$.

## 2.6 Removal of Redundant Developments

During the processing, many redundant developments are generated. These redundant developments include the rotation and the reflection of the "canonical" development.

Table 2.2: The adjacency list of the box of size $1 \times 1 \times 1$.

| Square ID | UP | RIGHT | DOWN | LEFT |
|---|---|---|---|---|
| 01 | 04 | 06 | 02 | 05 |
| 02 | 01 | 06 | 03 | 05 |
| 03 | 02 | 06 | 04 | 05 |
| 04 | 03 | 06 | 01 | 05 |
| 05 | 01 | 02 | 03 | 04 |
| 06 | 01 | 04 | 03 | 02 |

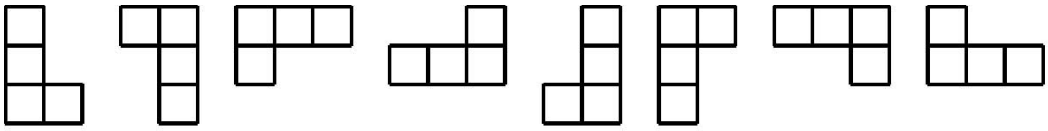


Figure 2.4: Polyominoes of the same shape.

The algorithm should not process and store the redundant developments to improve the efficiency. To reduce the amount of data, all polyominoes involved in this algorithm are made free polyominoes. *Free* polyominoes are polyominoes that are considered unchanged by rigid transformation (translation, rotation, reflection or glide reflection).

In general, each polyomino can be represented in *eight* different ways (Figure 2.4), we sort these shapes and pick the first one as the canonical one to store.

More precisely, the algorithm lists all rotations and reflections of a development in matrices, then sorts these matrices by a certain ordering and picks the matrix on top to the storing process.

The steps are shown as below.

**Step 1**: The algorithm puts the development in a matrix of size $X \times Y$, such that $Y \geq X$.

**Step 2**: In general, each polyomino can be represented in *eight* different arrays, we sort these arrays and pick the first one to store. Then the algorithm transforms the matrix to an array, we will describe the detail later. The algorithm sorts those eight arrays and

picks the first one to the storing process.

Top right array of Figure 2.5 is the first one after the sorting, so it is picked by the algorithm.

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 1 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 1 | 0 | 0 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 0 | 1 | 1 |

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

Figure 2.5: Matrix representation of polyominoes of the same shape.

By this way, all rotation and the reflection of the original development are represented as the same binary array. They are regarded as the same one polyomino in storing process.

## 2.7 Serialization and Storing

The polyomino $P_i$ is represented in the matrix in processing, and we compress the matrix into a binary array in storing to shrink the size.

Storing the matrix of the development straightforwardly consumes too much memory. It is necessary to compress the matrix into a smaller size. Every row of the matrix of 0s and 1s can transform to a binary number. In this way, a matrix is transformed into a binary array. The height and width values of the matrix are put in the top of the array for transform back to the matrix (Figure 2.6).

When we need to process this compressed development, the algorithm does the reverse operation: First, it reads the height and width values to create an initialized matrix. Then it reads the array, transforms the data from a binary number to a line of 0s and 1s line by line. At last, we get the original partial development represented by a matrix.

The compressed developments are stored in a hash table to manage (Figure 2.7).

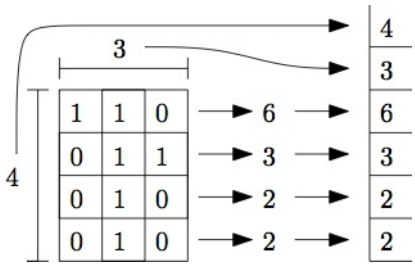



Figure 2.6: Compression of the development.

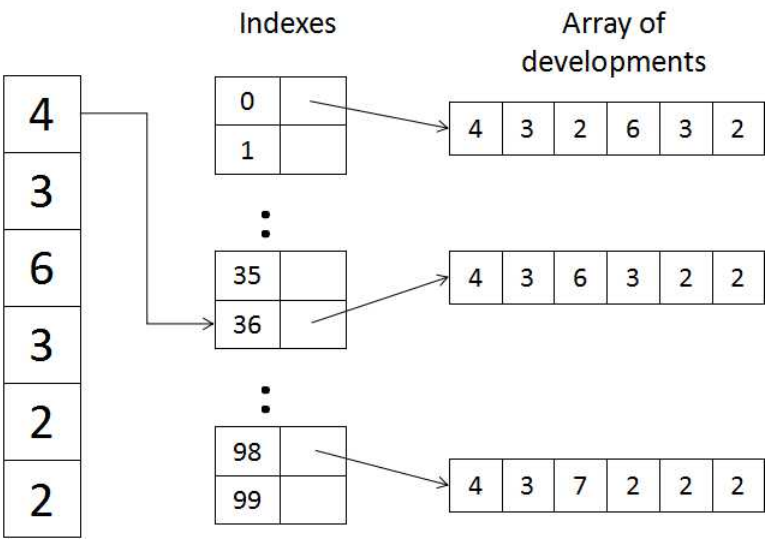



Figure 2.7: Storage of the development.

# Chapter 3

# Common Developments of Three Boxes

In this chapter, we give the details of our algorithm for finding all common developments of three boxes and further research of finding for the centrosymmetric common developments.

As the first step, we find all common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$. Then, we check each polyomino in the results of the first step if it can fold to a cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$.

In the results of enumerating all common developments of three boxes of size $1 \times 1 \times 7$, $1 \times 3 \times 3$ and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$, there are some centrosymmetric common developments. We find all centrosymmetric common developments of surface area from 22 to 54.

## 3.1 Enumerating Common Developments

### 3.1.1 Enumerating all Common Developments of Two Boxes of Size $1 \times 1 \times 7$ and $1 \times 3 \times 3$

Here, we describe the 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 [6] for finding two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$.

Let $s$ be the surface area of two given boxes, and $S_i$ be the set of all common partial developments of area $i$ of these two boxes. Then, $S_1$ consists of a unit square, and each $S_i$ with $i > 1$ is a subset of polyominoes of size $i$ that can be computed from $S_{i-1}$ by the

breadth first search for each $i = 2, 3, \ldots, s$.

The algorithm will check whether $P_i$ can cover the boxes without overlap or not. We name this Procedure CheckCover, and the details will be described in the next section. As mentioned in Preliminaries, we use the adjacency list of the boxes in checking. We let $B_i$ be the adjacency list of the $i$th given box.

Our final goal is to obtain the set $S_s$ that contains all common developments of surface area $s$ from the set $S_1$.

More precisely, it repeats the following process for each $i = 2, 3, \ldots$:

**Algorithm 1:** Outline of the exhaustive search algorithm.

**Input** : The surface area of two boxes, $s$;
the adjacency list of two boxes, $B_1$, $B_2$;

**Output**: All common developments of two boxes in $S_s$;

```
Initialize S_1 by a set of unit square;
for i = 2 to s do
    foreach partial development P_{i-1} in S_{i-1} do
        attach a square q_1 to P_{i-1} at each possible adjacency square on the
        boundary of P_{i-1} to obtain a new polyomino P_i;
        if CheckCover(P_i, B_1, s)==1 and CheckCover(P_i, B_2, s)==1 and P_i has
        no hole then
            store P_i into S_i; // P_i is a partial development of the boxes
return S_s;
```

As shown in Algorithm 1, the algorithm works in a loop as follows, it picks up a polyomino $P_{i-1}$ in $S_{i-1}$, then attaches a square $q_1$ by edge-to-edge gluing to $P_{i-1}$ at each possible adjacency empty square on the boundary of $P_{i-1}$ as shown in Figure 3.1. By this step, we generate a new polyomino $P_i$. The polyomino $P_i$ still needs to be examined whether it can fold to a part of the boxes if size $1 \times 1 \times 7$ and $1 \times 3 \times 3$ or not. This loop terminates at $i = s$ when the polyomino $P_s$ can fold to the given boxes. For example, consider the case $s = 30$. When $i = 1$, $S_1$ contains one unit square. When $i$ comes to 2, the algorithm selects one polyomino $P_1$ from $S_1$, since $i - 1 = 1$, then it attaches a square $q_1$ by edge-to-edge gluing to $P_1$ clockwise, and it calls Procedure CheckCover to check the

component can fold to part of the boxes or not. Then, the algorithm stores it into $S_2$ if the generated one is a partial development and no hole in it. The loop of $i = 2$ ends after it tried all $P_1$. When $i$ comes to 3, the algorithm selects a polyomino $P_2$ from $S_2$ one by one, attaches $q_1$ to $P_2$, calls Procedure CheckCover to check the component, checks for holes, and stores it as a partial development to $S_3$, and it repeats the loop. The loop of $i = 3$ ends after the algorithm tried all elements of $S_2$, then $i$ comes to 4, 5, ..., 30. The algorithm simply repeats the loop of gluing, checking and storing steps. At last, the $S_{30}$ contains all common developments that can fold to two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$.

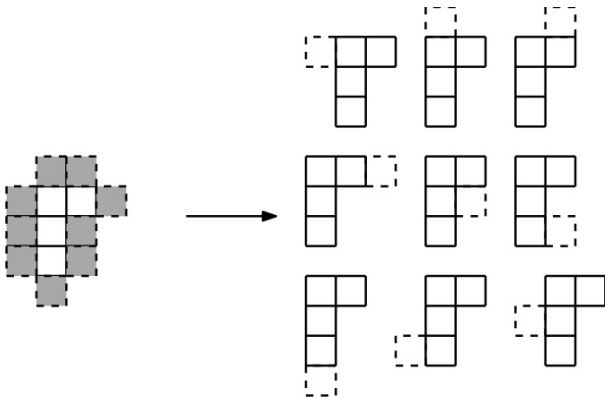
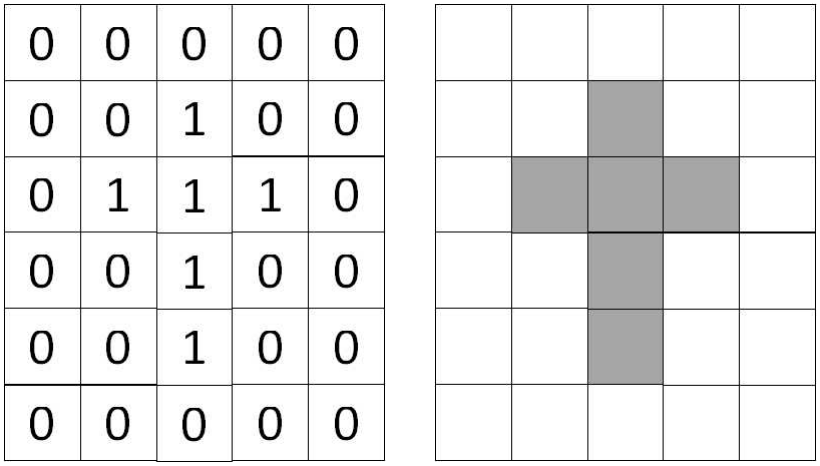


Figure 3.1: Possible results of adding one square.

As mentioned in Preliminaries, to reduce the amount of data, all polyominoes involved in the algorithm are free polyominoes. By this way, all rotations and reflections of the canonical development are represented as the same binary array. To compress data, all partial developments are serialized to the binary arrays and stored in a hash table in the storing process. This method consumes less memory.

The polyomino $P_i$ is represented in matrix of 0 and 1 in processing (Figure 3.2), and we compress the matrix into a binary array in storing to shrink the size. We maintain the sets $S_{i-1}$ and $S_i$ in two hash tables of size $O(\max_i\{i|S_i| + (i-1)|S_{i-1}|\})$. At the start of each loop, we access $S_{i-1}$ from one hash table and store $S_i$ into another hash table, and make the hash table for $S_{i-1}$ empty in the end of each loop. With the increasing of $i$, the size of the hash tables can be incredibly huge that may cause memory overflow on a normal personal computer (e.g. $i = 17$ when the surface area is 30).




(a) (b)

Figure 3.2: Matrix and corresponding development.

### 3.1.2 Improving Performance by Parallel Computing

When the surface area $i$ is not too large, we can get the $S_i$ from $S_{i-1}$ by the breadth-first search of adding a square to the $P_{i-1}$.

This simple idea works up to 22 for two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$ in [6] since the maximum number of $|S_i \cup S_{i-1}|$ takes $1.01 \times 10^7$ when $i = 18$. It ran in 10 hours in 2011 and 5 hours in 2014 in our experiments on a usual PC. However, for the surface area 30, it does not work even on a parallel computer (CRAY XC30) due to memory overflow when $i = 22$.

Thus, we divide the computation to two phases. In the first phase, we compute $S_i$ for each $i = 2, \ldots, 16$. As a result, we have $S_{16}$ that consists of 7486799 common partial developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$.

In the second phase, we divide $S_{16}$ into 75 disjoint subsets $S_{16}^j$ with $1 \leq j \leq 75$. For each $S_{16}^j$, we divide it into many small groups of common partial developments. For example, if each small group contains 25 common partial developments. Then, a $S_{16}^j$ contains 4000 small groups when $1 \leq j \leq 74$ ($S_{16}^{75}$ contains rest of them). In the parallel computer, we use 250 processes (each process on a core, in total it takes 250 cores) to process these small groups. Each process independently calculates a small group by the BFS algorithm at a time. On average, it takes about 40 minutes for a process to calculate a small group of 25 common partial developments from $i = 17$ to $i = 30$. In the final step, we merge $S_{30}^j$ with $1 \leq j \leq 75$, remove duplicates, and obtain $S_{30}$.

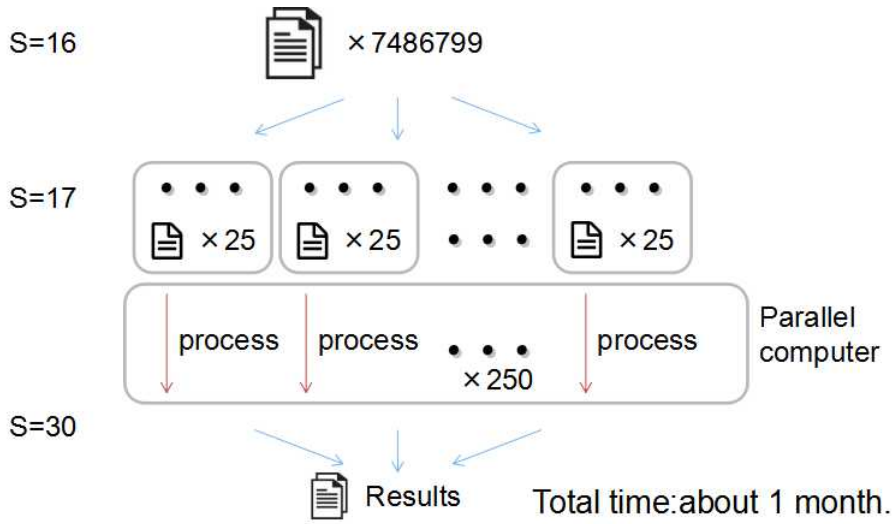



Figure 3.3: The detail of parallel computing.

The program runs, in total, in about 1 month on the parallel computer (CRAY XC30), and we obtain 1080 common developments in $S_{30}$ of two boxes of size $1\times1\times7$ and $1\times3\times3$. It is worthwhile to note that the total calculation time depends on the amount of requests to the parallel computer. If there is another request with a higher priority, we may have to wait for some time. We give the detail of the results in Section 3.2.1. If we conduct the same calculation on a normal notebook computer with a single process, it may takes a much longer time.

### 3.1.3 Enumerating all Common Developments of Three Boxes of Size $1 \times 1 \times 7$, $1 \times 3 \times 3$, and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$

We check each of the common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$, with the Procedure CheckCover. If it can fold to the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$, then it can fold to three boxes of size $1 \times 1 \times 7$, $1 \times 3 \times 3$, and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$.

Let $S_{30}$ be the set of all common developments of two boxes of size $1 \times 1 \times 7$ and $1 \times 3 \times 3$, and $B_3$ be the adjacency list of the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$. As shown in Algorithm 2, the algorithm checks all common developments $P_{30}$ in $S_{30}$. If $P_{30}$ can fold to the third box, the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$, then algorithm stores $P_{30}$ into $S'_{30}$. At last, all elements in $S_{30}$ are checked and the algorithm outputs $S'_{30}$, which is the set of all common developments of three boxes of size $1 \times 1 \times 7$, $1 \times 3 \times 3$, and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$.

**Algorithm 2:** Outline of the algorithm for checking the third box.

**Input** : The common developments of two boxes, $S_{30}$, the adjacency list of the cube, $B_3$;

**Output**: All common developments of three boxes $S'_{30}$;

```
foreach common development P_30 in S_30 do
    if CheckCover(P_30, B_3, 30)==1 then
        store P_i into S'_30; // P_30 is a common development of three boxes
return S'_30;
```

The program runs less than 1 minute on a laptop computer, and we obtain 9 common developments in $S'_{30}$ of three boxes of size $1 \times 1 \times 7$, $1 \times 3 \times 3$, and $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$. We give the detail of the result in Section 3.2.2.

### 3.1.4 Checking the Box

In this section, we give the details of Procedure CheckCover.

The previous study mainly focuses on the "normal" orthogonal polyhedra. That is, the crease lines are orthogonal to the edges of unit squares.

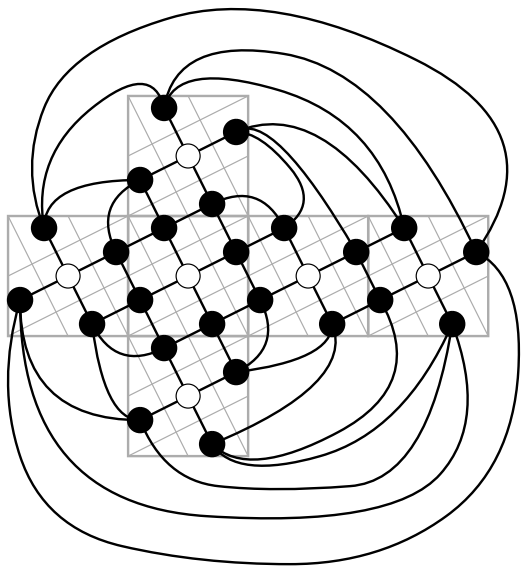


Figure 3.4: The graph induced by the cube.

However, the development of the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ cannot fold along the edges

of unit squares, which means the previous method of checking the development can fit the box or not is no more available. We have two algorithms for the approach of checking the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$. The first algorithm checks for each $P$ the vertices of the cube on the development boundary. The angle between the folding line and an edge of a unit square is 26.6 degrees. Besides, the vertex of the cube is made of three squares, which means every vertex unfolded on the development should be surrounded by 3 squares. Therefore, we need to check whether there is a development that fits such requirements: all vertices of the cube are on the boundary of development, constituting a grid of size $\sqrt{5}$ of 26.6 degrees to the unit square's edge, the number of the squares around each vertex should be exactly 3.

The second algorithm checks the positional relationships of each unit square on the development. Consider any polyhedron, e.g., a box, if it can be unfolded to a development that is made of unit squares, we can get an adjacency graph of unit squares on the box as shown in Figure 3.4. At this point, for any development, if its unit squares have the same positional relationships as the adjacency list, it can fold to the box. Besides, the first algorithm does not consider the dislocation of the center square, which can make a hole and an overlap on the cube (Figure 3.5). Therefore, we apply the second algorithm to handle it.

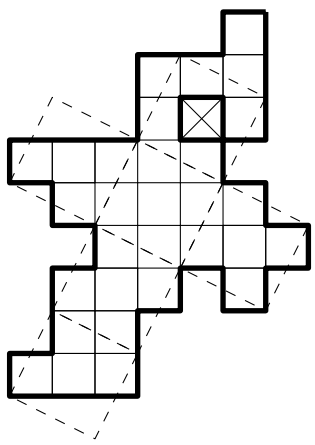


Figure 3.5: A hole and an overlap occur when we fold along the dotted lines to make a cube while the number of the squares around each vertex is 3.

Procedure CheckCover can check if a polyomino $P_i$ can fold to (a part of) the box or not. For example, the box is the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ (Figure 3.6), the adjacency list

**Procedure** CheckCover($P_i$, $B$, $s$)

**Input** : Polyomino $P_i$ in $S_i$;
adjacency list $B$ of the given box;
integer of the surface area of the given boxes $s$;

**Output**: Whether $P_i$ can fold to the box $B$ or not;

```
for i = 1 to s do
    mark the leftmost of the topmost square with number i as the start point;
    foreach marked square in P_i do
        mark its unmarked adjacent squares as the adjacency matrix of the box
        if every square of P_i is marked by a unique number then
            return 1; // P_i can fold to the cube
        else if any square of P_i is marked by 2 or more numbers then
            break; // P_i has an overlap to fold to the cube
        else if 2 or more squares of P_i are marked by the same number then
            break; // P_i has an overlap to fold to the cube
```

of the cube is shown as Table 3.1. The detail of the algorithm consists of 4 steps shown as below.

**Step 1:** With the adjacency list of the box, the algorithm marks one unit square on the partial development as a *start point* (basically, the leftmost of the topmost square for it is always the first to be visited in traversal).

**Step 2:** According to the Table 3.1, the algorithm marks the "start point" with a number.

By default, the procedure runs in $s$ loops to try all possible numbers of the start point. It works for ordinary boxes like $1 \times 1 \times 7$ and $1 \times 3 \times 3$. However, the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ can take less trials with some special tricks (Section 3.1.4).

**Step 3:** Assume the partial development can fold to the part of the target box as the start point is numbered. Then starting from the marked start point, the algorithm marks the neighbor-squares of the start point with the corresponding neighbor numbers on the adjacency list (Figure 3.7). If there is no neighbor of the start point in a certain direction, then skip it.

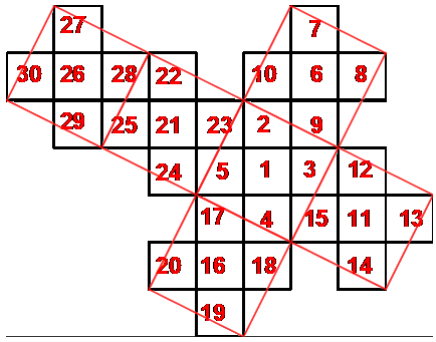



Figure 3.6: An unfolding of the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$.

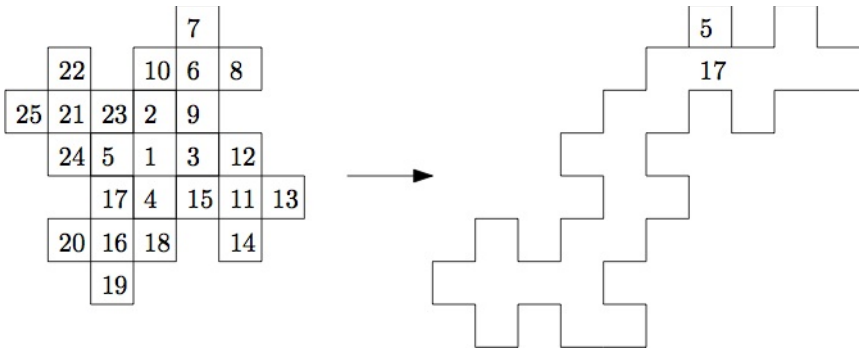



Figure 3.7: Neighbor-squares of start point marked in step 3.

Table 3.1: The adjacency list of the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$.

| Square ID | UP | RIGHT | DOWN | LEFT |
|---|---|---|---|---|
| 01 | 02 | 03 | 04 | 05 |
| 02 | 10 | 09 | 01 | 23 |
| 03 | 09 | 12 | 15 | 01 |
| 04 | 01 | 15 | 18 | 17 |
| 05 | 23 | 01 | 17 | 24 |
| 06 | 07 | 08 | 09 | 10 |
| 07 | 30 | 29 | 06 | 22 |
| 08 | 29 | 13 | 12 | 06 |
| 09 | 06 | 12 | 03 | 02 |
| 10 | 22 | 06 | 02 | 23 |
| 11 | 12 | 13 | 14 | 15 |
| 12 | 09 | 08 | 11 | 03 |
| 13 | 08 | 29 | 28 | 11 |
| 14 | 11 | 28 | 19 | 18 |
| 15 | 03 | 11 | 18 | 04 |
| 16 | 17 | 18 | 19 | 20 |
| 17 | 05 | 04 | 16 | 24 |
| 18 | 04 | 15 | 14 | 16 |
| 19 | 16 | 14 | 28 | 27 |
| 20 | 24 | 16 | 27 | 25 |
| 21 | 22 | 23 | 24 | 25 |
| 22 | 07 | 10 | 21 | 30 |
| 23 | 10 | 02 | 05 | 21 |
| 24 | 21 | 05 | 17 | 20 |
| 25 | 30 | 21 | 20 | 27 |
| 26 | 27 | 28 | 29 | 30 |
| 27 | 20 | 19 | 26 | 25 |
| 28 | 19 | 14 | 13 | 26 |
| 29 | 26 | 13 | 08 | 07 |
| 30 | 25 | 26 | 07 | 22 |

Repeat this step, the algorithm extends the marking to further neighbor-squares of the marked ones (Figure 3.8) until every square is uniquely marked.

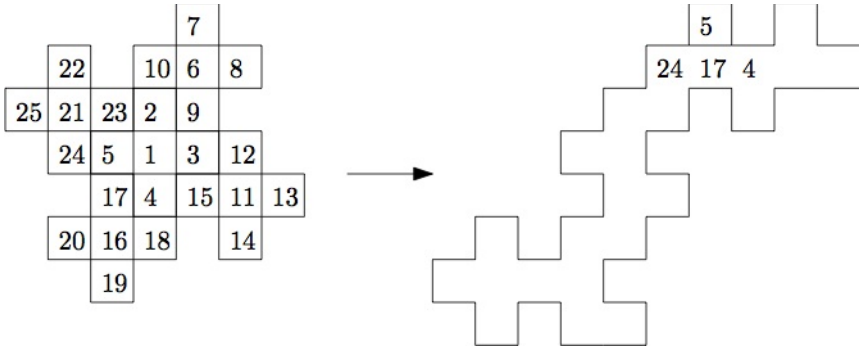



Figure 3.8: Further neighbor-squares of start point marked in step 3.

**Step 4:** If any two squares are marked with the same number, or one square is marked by two different numbers by two or more its neighbors, then it means an overlap occurs and $P$ should be discarded. After the overlap checking, we have two conditions to examine $P$; (1) all squares in connected $P$ are marked with their unique numbers, which means $P$ can wrap the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$ with consistency, and (2) $P$ has the same area of the cube. If $P$ only satisfies the first condition, then $P$ is a partial development. Otherwise, if $P$ satisfies these two conditions, it is a common development (Figure 3.9).

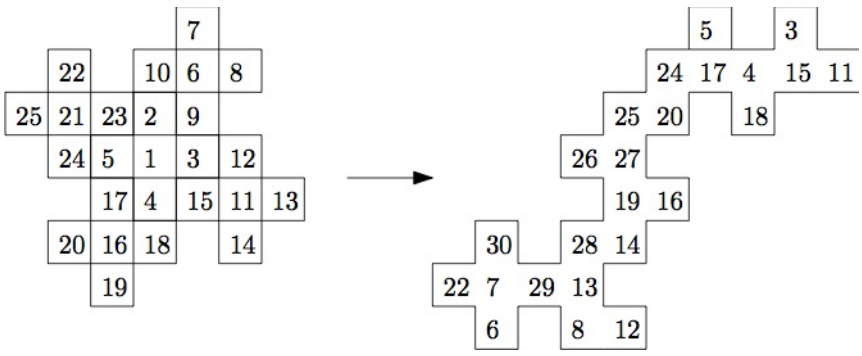



Figure 3.9: Every square is marked in step 4.

After checking all 5 possible start points on $P$, the algorithm repeats checking on the reflection of $P$ since there are two ways of folding to the cube of size $\sqrt{5} \times \sqrt{5} \times \sqrt{5}$. These two folding ways mirror each other as shown in Figure 3.10. It is necessary to check the reflection of $P$.

### 3.1.5 Some Tricks for the Cube of Size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$

The cube $Q$ of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ is made up of six same square faces. Thus, 30 unit squares of $P$ are partitioned into six same groups, which is a group of a surface. Each of these groups contains 2 small groups. One consists of only one unit square, which is the center square of one face of $Q$. The other small group consists of 4 unit squares, located on the corner. In Figure 3.6, squares 1 to 5 are on the same surface, and the square 1 is the center square.

Once we fix the way of folding of a unit square, we can determine the square grid defined by the folding line, which completely gives the folding lines to fold $Q$. For a fixed unit square $s$ in $P$, carefully counting (Figure 3.10), there are 5 choices of the situation of $s$ (another 5 choices for the reflection of $P$). To make the algorithm runs faster, start point can be numbered as 1~5.

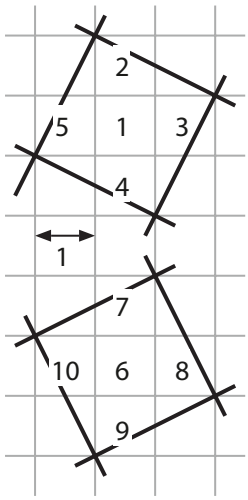



Figure 3.10: There are 10 choices for letting a unit square $s$ with respect to the square grid of size $\sqrt{5}$.

## 3.2 Results of Enumerating Common Developments of Boxes

### 3.2.1 Common Developments of Two Boxes of Size 1×1×7 and 1×3×3 of Surface Area 30

The program is compiled in C language to find the common developments of two boxes of size 1×1×7 and 1×3×3 of surface area 30 (Figure 3.11). Since a normal PC cannot withstand large amounts of data generated, the running environment is the parallel computer Cray XC30.

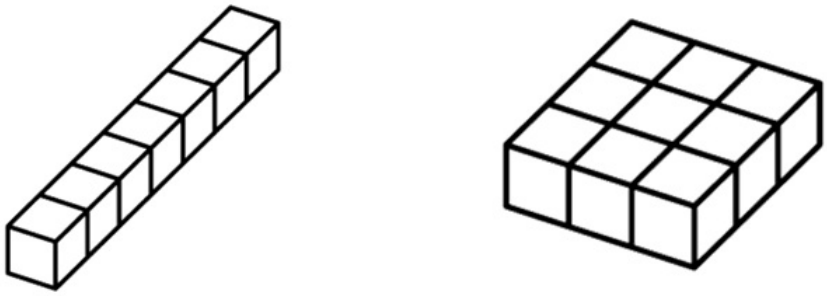


Figure 3.11: Boxes of size 1×1×7 and 1×3×3 of surface area 30.

Experimental results are shown in Table 3.2 and Table 3.3, and the charts in Figure 3.12 and Figure 3.14 for more intuitive expressions. The charts show the number of partial developments changes with the increasing of the surface area.

Processing first 16 steps takes 3 hours, generating 7.5 millions of the partial common developments as intermediate results. Processing every single development from step 17 to step 30 takes 40 minutes. Processing all 7.5 millions of developments by the parallel computer takes about 1 month.

The number of developments of surface area $s$=1 to $s$=16 is the same as the previous study. After the step 17, the process starts from each single partial development of the surface area 17. We can find the number of partial developments starts increasing rapidly from $s$=9, reaches its maximum at $s$=24, then decreases rapidly.

The final result is 1080 that is the number of common developments which can fold to

two boxes of size 1×1×7 and 1×3×3 of surface area 30. Some examples of the common developments are shown in Figure 3.13.

Table 3.2: The number of partial developments of surface area from 1 to 16.

| Set of partial developments | Number of developments |
|---:|---:|
| $S_1$ | 1 |
| $S_2$ | 1 |
| $S_3$ | 2 |
| $S_4$ | 5 |
| $S_5$ | 12 |
| $S_6$ | 35 |
| $S_7$ | 108 |
| $S_8$ | 368 |
| $S_9$ | 1283 |
| $S_{10}$ | 4601 |
| $S_{11}$ | 16405 |
| $S_{12}$ | 57879 |
| $S_{13}$ | 200209 |
| $S_{14}$ | 682639 |
| $S_{15}$ | 2288392 |
| $S_{16}$ | 7486799 |

### 3.2.2 Common Developments of Three Boxes of Size 1×1×7, 1×3×3, and $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ of Surface Area 30

The program is written in C language to find the common developments of three boxes of size 1×1×7, 1×3×3, and $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ of surface area 30 (Figure 3.15). The running environment is ASUS A43S laptop.

Our new algorithm based on the adjacency list checked whether each of 1080 common developments of two boxes of size 1×1×7 and 1×3×3 of surface area 30 can fold to the

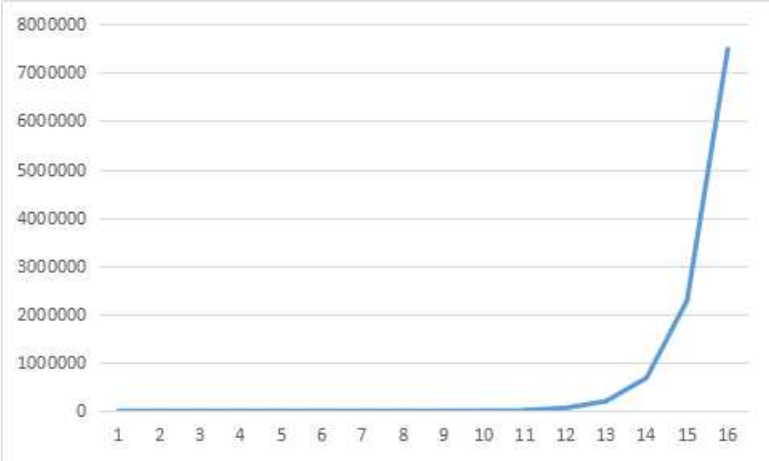



Figure 3.12: The number of partial developments of surface area from 1 to 16.

Table 3.3: The number of partial developments of surface area from 17 to 30 (from a randomly picked development of surface area 16).

| Set of partial developments | Number of developments |
|---:|---:|
| $S_{17}$ | 22 |
| $S_{18}$ | 230 |
| $S_{19}$ | 1430 |
| $S_{20}$ | 5834 |
| $S_{21}$ | 16589 |
| $S_{22}$ | 33997 |
| $S_{23}$ | 50497 |
| $S_{24}$ | 53420 |
| $S_{25}$ | 38989 |
| $S_{26}$ | 18605 |
| $S_{27}$ | 5249 |
| $S_{28}$ | 714 |
| $S_{29}$ | 28 |
| $S_{30}$ | 1 |

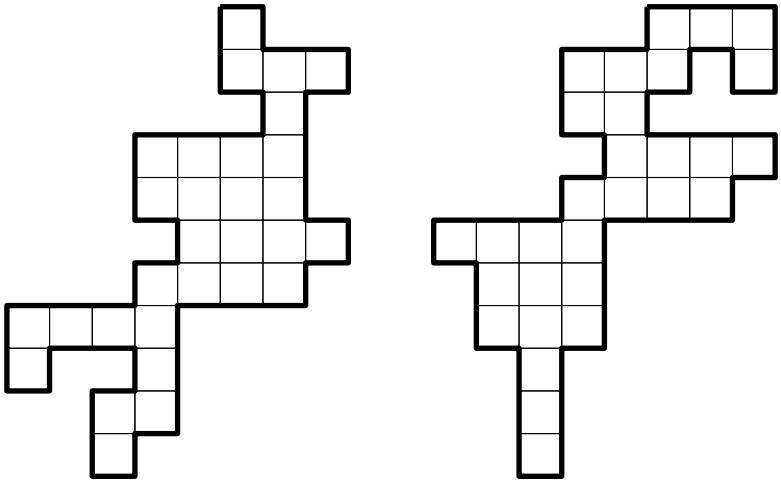


Figure 3.13: Some common developments of two boxes of size 1×1×7 and 1×3×3.

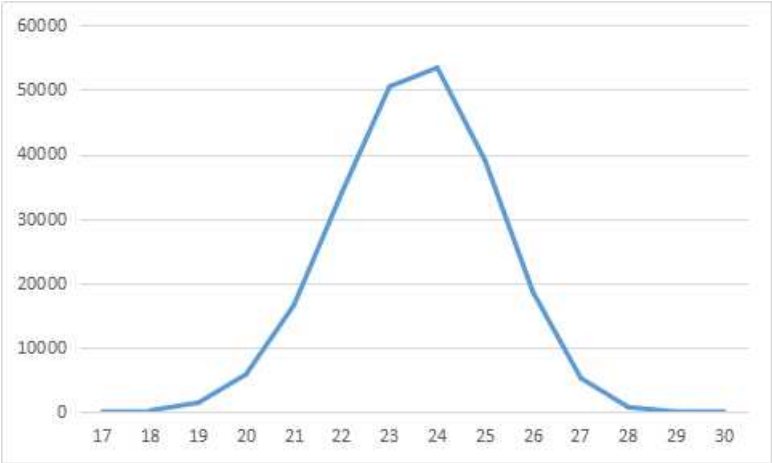



Figure 3.14: The number of partial developments of surface area from 17 to 30 (from a randomly picked development of surface area 16).

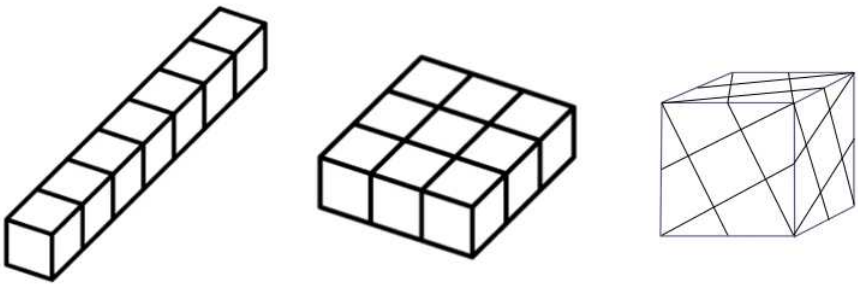


Figure 3.15: Boxes of size 1×1×7, 1×3×3, and $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ of surface area 30.

cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$. As a result, nine out of 1080 developments can fold to the cube (Figure 3.16). While eight developments have only one way of folding to the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$, the other development has two different ways of folding to the cube of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$, for there is an angle of 26.6 degrees to the unit square's edge clockwise and anti-clockwise. That is, the last development can actually fold to 3 boxes of size $1\times1\times7$, $1\times3\times3$, and $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ in two different ways as shown in Figure 3.17.

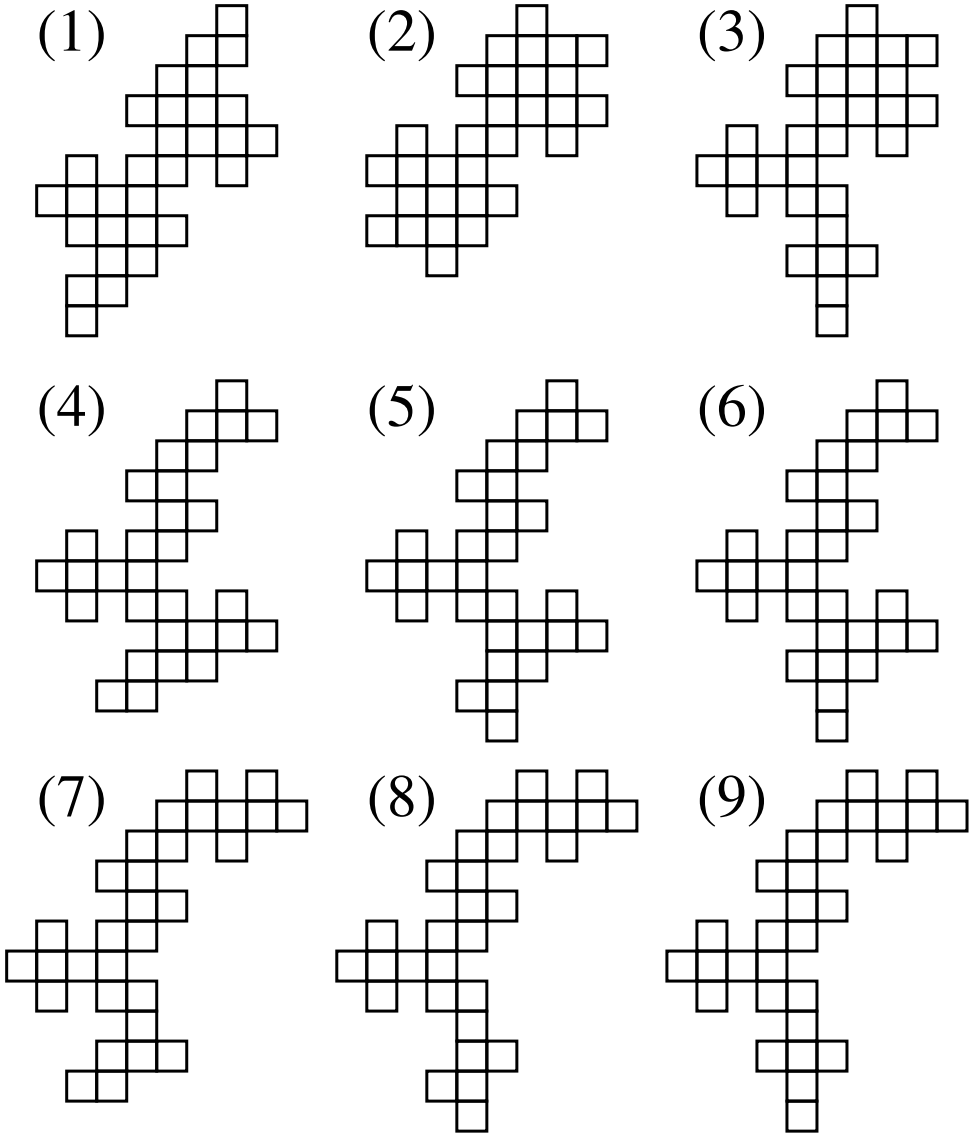



Figure 3.16: Developments having three or four ways of folding to three different boxes of surface area 30.

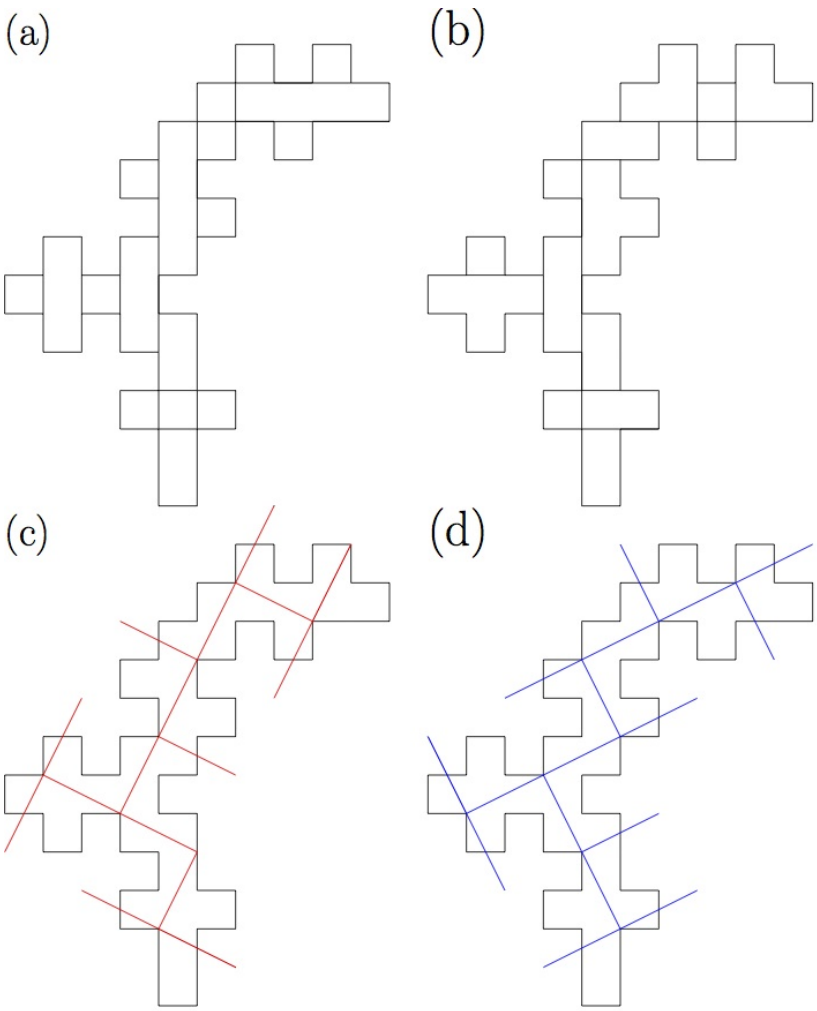



Figure 3.17: The development having four ways of folding: (a)$1\times1\times7$ (b) $1\times3\times3$ (c) $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$ (d) $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$.

## 3.3 Enumerating Centrosymmetric Common Developments

We first note the surface area of a box of size $a \times b \times c$ should be an even number since it is $2(ab + bc + ac)$. Thus we do not need to consider centrosymmetric polyominoes of odd surface area in this section.

Assume $P_i$ is a centrosymmetric polyomino of surface area $i$, where $i$ is an even integer. For any square $q_1$ on $P_i$, there is a corresponding square $q_2$ that is central symmetric to $q_1$. Removing both $q_1$ and $q_2$ from $P_i$, we get a new centrosymmetric polyomino $P_{i-2}$ ($P_{i-2}$ should be connected and has no hole inside). Repeat this step then we can get $P_{i-4}$, $P_{i-6}$, ... until the smallest centrosymmetric polyomino that has no more $q_1$ and $q_2$ that can be removed. Thus, we can generate all centrosymmetric partial developments of surface area $i$ from $i - 2$.

As shown in Figure 3.18, there are two possible locations for the center of symmetry on a unit square. Since the surface area of a box is an even integer, the center of symmetry should be on the edge, not inside the unit square. By a simple case analysis, the smallest surface area that admits both kinds of centrosymmetric polyominoes is 4 (Figure 3.19).

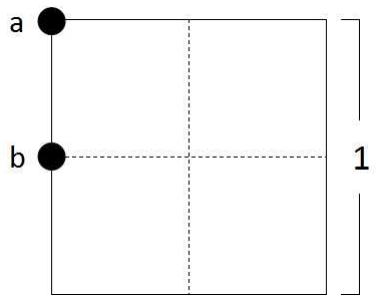
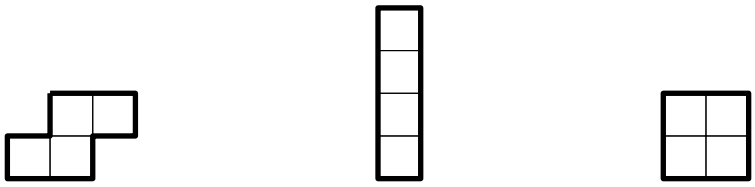



Figure 3.18: Possible locations for the center of symmetry on a unit square.

Let $S_i$ be the set of all common partial developments of area $i$ of two boxes. Let $B_i$ be the adjacency list of the $i$th given box. The differences from the previous exhaust algorithm are, this algorithm starts from $S_4$, and it runs in a shorter loop. First, $S_4$ consists of three tetrominoes, which are all centrosymmetric common partial developments (Figure 3.19). Each $S_i$ with $i > 4$ is a subset of polyominoes of size $i$ that can be computed from $S_{i-2}$ by the breadth first search for each even number $i = 6, 8, \ldots$.

Our final goal is to obtain the set $S_n$ that contains all centrosymmetric common developments of surface area $n$ from the set $S_4$.



(a) (b) (c)

Figure 3.19: Centrosymmetric common partial developments of $S_4$.

---

**Algorithm 3:** Outline of the exhaustive search algorithm for centrosymmetric common developments.

---

**Input** : The surface area of two boxes, $s$;
the adjacency list of two boxes, $B_1$, $B_2$;

**Output**: All centrosymmetric common developments in $S_s$;

```
Initialize S_4 by three polyominoes in Figure 3.19;
for i = 6, 8 ... s do
    foreach partial development P_{i-2} in S_{i-2} do
        attach a square q_1 to P_{i-2} at each possible adjacency square on the
        boundary of P_{i-2} to obtain a new polyomino P'_i;
        attach another square q_2 to P'_i in central symmetrical the position to q_1 to
        obtain a new centrosymmetric polyomino P_i;
        if CheckCover(P_i, B_1, s)==1 and CheckCover(P_i, B_2, s)==1 and P_i has
        no hole then
            store P_i into S_i; // P_i is a partial development of the box
return S_s;
```

---

This Algorithm 3 works in a loop as follows, it picks up a polyomino $P_{i-2}$ in $S_{i-2}$, then attaches a square $q_1$ by edge-to-edge gluing to $P_{i-2}$ at each possible adjacency empty square on the boundary of $P_{i-2}$. On the corresponding empty square that is central symmetric to $q_1$, it attaches another square $q_2$ to make a new centrosymmetric polyomino $P_i$ as shown in Figure 3.20. The polyomino $P_i$ still needs to be examined whether it

can fold to a part of the given boxes or not. This loop terminates at $i = s$ when the polyomino $P_s$ can fold to the given boxes. For example, consider the case $s = 22$. First, when $i = 4$, $S_i$ contains three centrosymmetric common developments. When $i$ starts from 6, the algorithm selects one polyomino $P_4$ from $S_4$, since $i - 2 = 4$, then it attaches a square $q_1$ by edge-to-edge gluing to $P_4$ clockwise, and attaches another $q_2$ on the central symmetrical position. Then it calls Procedure CheckCover (it is the same as the previous section) to check if the component can fold to a part of the given boxes or not. After that, the algorithm stores it into $S_6$ if the generated one is a partial development. Repeat these steps. The loop of $i = 6$ ends after it tried all $P_4$. Since the algorithm attaches two squares in one loop, then $i$ will be increased by 2. When $i$ comes to 8, the algorithm selects a polyomino $P_6$ from $S_6$ one by one, attaches $q_1$ and $q_2$ to $P_6$, calls Procedure CheckCover to check the component and stores the partial development to $S_8$, and repeats the loop. The loop of $i = 8$ ends after the algorithm tried all elements of $S_6$, then $i$ comes to 10, 12, 14, ..., $s$. The algorithm simply repeats the loop of attaching, checking and storing steps. At last, when $s = 22$, the $S_{22}$ contains all centrosymmetric common developments that can fold to two boxes of $1 \times 1 \times 5$ and $1 \times 2 \times 3$.

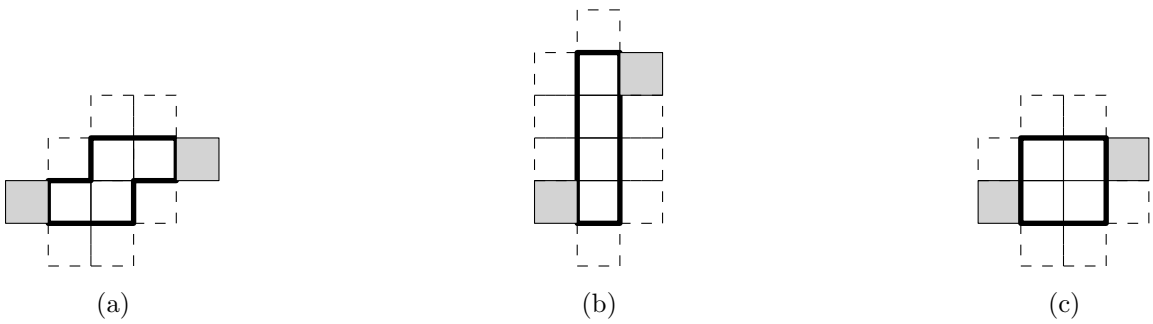



Figure 3.20: Attach $q_1$ and $q_2$ to make a new centrosymmetric polyomino $P_i$.

## 3.4 Results of Enumerating Centrosymmetric Common Developments

As a result, among 2263 common developments of two boxes of size $1 \times 1 \times 5$ and $1 \times 2 \times 3$, only 45 centrosymmetric common developments exist. The exact numbers of centrosymmetric common developments have been confirmed from surface area 22 to surface area

Table 3.4: The number of centrosymmetric common developments for each surface area.

| Surface area $S$ | Number of developments |
|---:|---:|
| 22 ($1 \times 1 \times 5$ and $1 \times 2 \times 3$) | 45 |
| 30 ($1 \times 1 \times 7$ and $1 \times 3 \times 3$) | 3 |
| 34 ($1 \times 1 \times 8$ and $1 \times 2 \times 5$) | 28 |
| 38 ($1 \times 1 \times 9$ and $1 \times 3 \times 4$) | 148 |
| 46 ($1 \times 1 \times 11$ and $1 \times 2 \times 7$) | 20 |
| 46 ($1 \times 2 \times 7$ and $1 \times 3 \times 5$) | 1 |
| 46 ($1 \times 1 \times 11$ and $1 \times 3 \times 5$) | 3 |
| 54 ($1 \times 1 \times 13$ and $1 \times 3 \times 6$) | 46 |
| 54 ($1 \times 1 \times 13$ and $3 \times 3 \times 3$) | 97 |
| 54 ($1 \times 3 \times 6$ and $3 \times 3 \times 3$) | $\geq 1920$ |

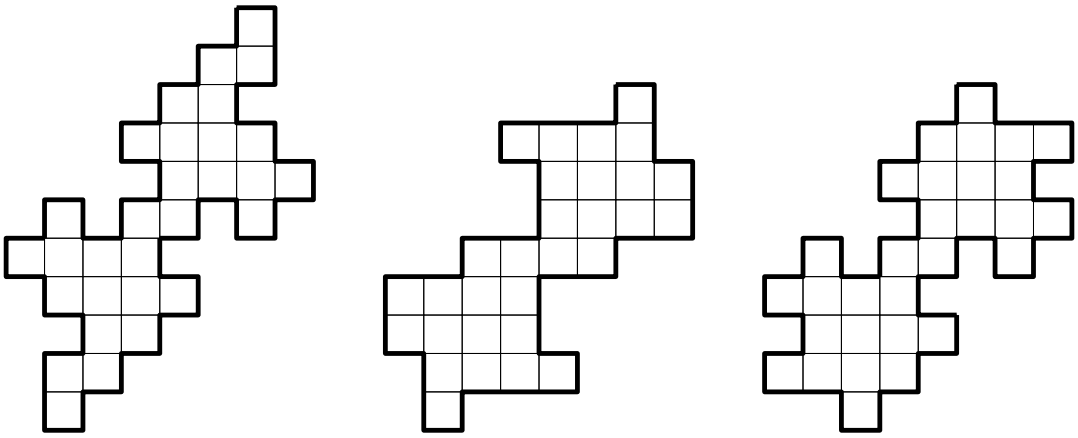
54, as shown in Table 3.4.

We here give two remarks. First, for the surface area 46, we have 24 centrosymmetric common developments in total, however, there are no centrosymmetric common developments for three boxes of size $1 \times 1 \times 11$, $1 \times 2 \times 7$, and $1 \times 3 \times 5$. Second, for the surface area 54, it is remarkable that while there are 46 and 98 centrosymmetric common developments of two boxes of $1 \times 1 \times 13$ and $1 \times 3 \times 6$, and of $1 \times 1 \times 13$ and $3 \times 3 \times 3$, respectively, there are no centrosymmetric common developments of three boxes of size $1 \times 1 \times 13$, $1 \times 3 \times 6$, and $3 \times 3 \times 3$.

Some centrosymmetric common developments are shown in Figure 3.21 and Figure 3.22.

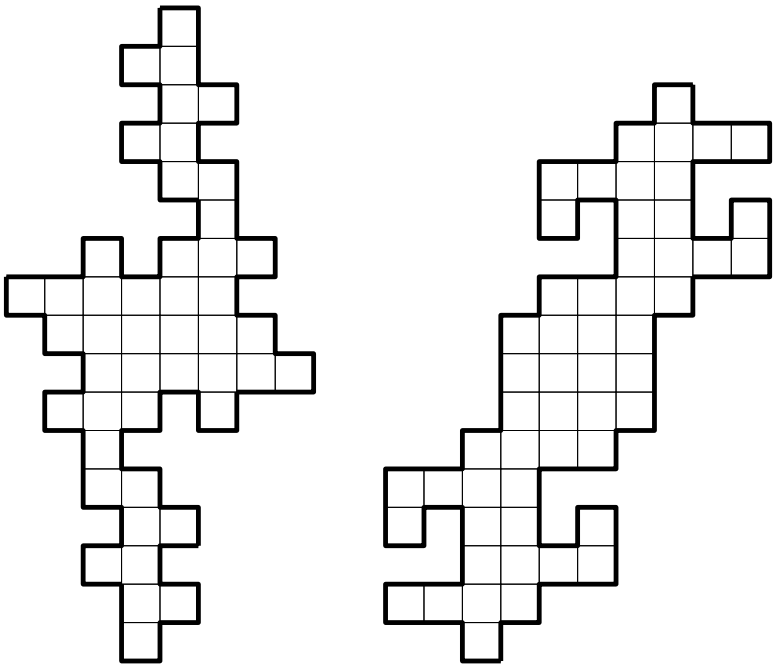
From the results, we summarize them to the following theorem:

**Theorem 6.** *The centrosymmetric common developments can only fold to two boxes, and none of them can fold to three boxes for surface area up to 54.*



(a) (b) (c)

Figure 3.21: All centrosymmetric common developments of surface area 30 that can fold to two boxes of $1 \times 1 \times 7$ and $1 \times 3 \times 3$.



(a) $1 \times 2 \times 7$ and $1 \times 3 \times 5$ (b) $1 \times 1 \times 13$ and $1 \times 3 \times 6$

Figure 3.22: Centrosymmetric common developments of other surface areas.

# Chapter 4

# Rep-cubes

In this chapter, we describe the exhaustive search algorithm for generating all regular rep-cubes of order $k$ (for $k = 2$ and $k = 4$), which consist of minimal unit squares.

## 4.1 Enumerating Rep-cubes

**Algorithm 4:** Outline of the exhaustive search algorithm.

**Input** : The integer of the order for the rep-cube, $k$;

**Output**: All rep-cubes in $S_k$;

```
Initialize S_1 by 11 developments of the unit cube;
for i = 2 to k do
    foreach partial development P_{i-1} in S_{i-1} do
        foreach development P_1 in S_1 do
            attach P_1 to P_{i-1} at each possible adjacency square on the boundary of
            P_{i-1} to obtain a new polyomino P_i;
            if CheckCoverSingle(P_i)==1 then
                store P_i into S_i; // P_i is a partial development of the box of size
                                    // √k × √k × √k
return S_k;
```

Algorithm 4 gives the outline of this algorithm. It works as follows: Let $S_i$ be the set of all polyominoes of size $6 \times i$ such that (1) it is composed by $i$ developments of unit

cube, (2) it can cover a part of a cube of size $\sqrt{k} \times \sqrt{k} \times \sqrt{k}$. That is, $S_1$ is the set of all developments of one unit cube, which consists of 11 polyominoes, and each set $S_i$ with $i > 1$ is a subset of polyominoes of area $6 \times i$ that can be computed from $S_{i-1}$ by the breadth first search for each $i = 2, 3, \ldots$. Let $P_i$ be any polyomino in $S_i$, e.g., $P_1$ is one of the 11 polyominoes in $S_1$ (Figure 4.1).

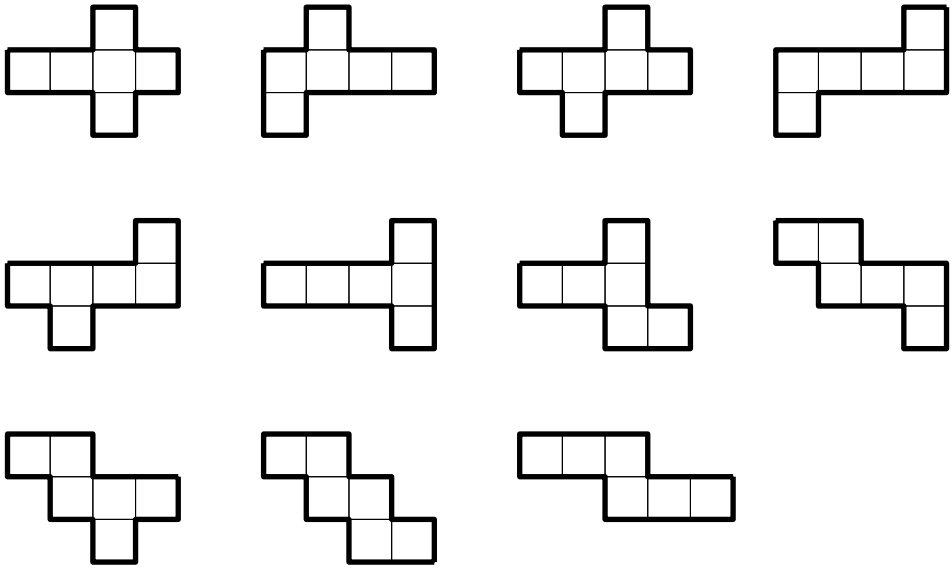


Figure 4.1: The set $S_1$ of all developments of one unit cube.

As in Section 3, in the term of search of development, each element in $S_i$ is called a *partial development* of the cube of size $\sqrt{k} \times \sqrt{k} \times \sqrt{k}$.

In Procedure CheckCoverSingle, the algorithm checks if $P_i$ can cover the cube of size $\sqrt{k} \times \sqrt{k} \times \sqrt{k}$ without overlap. The details will be described later.

Our final goal is to obtain the set $S_k$ that contains all regular rep-cubes of order $k$ from the set $S_1$.

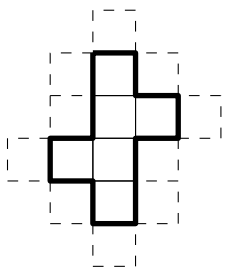


Figure 4.2: All possible adjacency empty squares on the boundary of a development of a $1 \times 1 \times 1$ cube.

The algorithm works in a loop as follows, it picks up a polyomino $P_{i-1}$ in $S_{i-1}$ and

a polyomino $P_1$ in $S_1$, then attach $P_1$ by edge-to-edge gluing to $P_{i-1}$ at each possible adjacency empty square on the boundary of $P_{i-1}$ as shown in Figure 4.2. We note that we have to consider not only the overlap, but also the reflection of $P_1$ if $P_1$ is not congruent to its reflection. By this step, we generate a new polyomino $P_i$, which is a component of $i$ developments of the unit cube. The polyomino $P_i$ still needs to be examined whether it can fold to a part of the cube of size $\sqrt{k} \times \sqrt{k} \times \sqrt{k}$ or not. This loop terminates at $i = k$ when the polyomino $P_k$ can fold to a complete cube. For example, consider the case $k = 4$. We already know that when $i = 1$, $S_1$ contains 11 developments of the unit cube. When $i$ comes to 2, the algorithm selects one polyomino $P_1$ from $S_1$ one by one, since $i - 1 = 1$, and picks another $P_1'$ in $S_1$ by the piece, note that $P_1$ and $P_1'$ can be the same one. Then it attaches $P_1'$ by edge-to-edge gluing to $P_1$ clockwise, and it calls Procedure CheckCoverSingle to check the component can fold to a part of the cube or not. Then, the algorithm stores it into $S_2$ if the generated one is a partial development. Repeat these steps with respect to the rotation and reflection of $P_1'$. The loop of $i = 2$ ends after it tried all combinations of $P_1'$ and $P_1$. When $i$ comes to 3, the algorithm selects a polyomino $P_2$ from $S_2$ one by one and picks $P_1'$ in $S_1$, attaches $P_1'$ to $P_2$, calls Procedure CheckCoverSingle to check the component and stores the partial development to $S_3$, and it repeats the loop for the rotation and reflection of $P_1'$. The loop of $i = 3$ ends after the algorithm tried all combinations of $P_1'$ and $P_2$, then $i$ comes to 4. In the loop of $i = 4$, it picks a polyomino $P_3$ from $S_3$ and $P_1'$ in $S_1$, and repeats the gluing, checking and storing steps. At last, the $S_4$ contains all rep-cubes of order 4.

As mentioned in Introduction, the folding lines of the cube of size $\sqrt{k} \times \sqrt{k} \times \sqrt{k}$ are not necessarily along the edges of unit squares. Rep-cubes of order 2, or 4 have different folding ways, which means we need a universal method to check whether the partial development can fold to the box or not.

As mentioned in Preliminaries, we proposed an algorithm that can check the positional relationships of each unit square on the development. Consider any polyhedron, if it can be unfolded to a development that is made of unit squares, we can get an adjacency list of unit squares on the box. Different developments of the same polyhedron keep a part of the same adjacency list. That is, we can tell whether a polyomino can fold to a cube like $\sqrt{k} \times \sqrt{k} \times \sqrt{k}$ by checking the positional relationship of the unit squares in

Procedure CheckCoverSingle.

We pick the first development $P$ in Figure 4.4 as an example (Figure 4.3). We first mark a unit square with the number 1 as the start point (here we pick the leftmost of the topmost square). Then, we mark all of its neighbor-squares a number according to the adjacency list of the cube of size $\sqrt{2} \times \sqrt{2} \times \sqrt{2}$ as Table 4.1 in all four directions. In this step, if a marked unit square is marked again with a different number, then it means an overlap occurs and $P$ should be discarded. Expand this step to its farther neighbors until every square of $P$ is marked with a number.

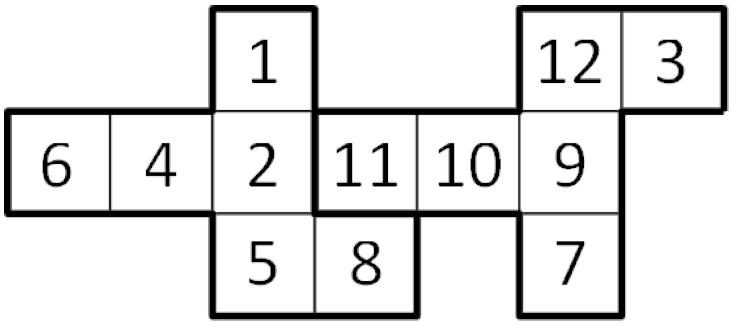



Figure 4.3: Every square of $P$ is marked with a unique number according to the adjacency list.

After this step, if every square in connected $P$ is marked with its unique number, $P$ can wrap the cube of size $\sqrt{k} \times \sqrt{k} \times \sqrt{k}$ with consistency. On the other hand, if (1) one square is marked with different numbers by its neighbors or (2) two or more squares are marked with the same number, then an overlap occurs in this folding way of $P$. We check all possible start points and directions for each $P$.

The polyomino $P_i$ is represented in the matrix in processing, and we compress the matrix into a binary array in storing to shrink the size. We maintain the sets $S_{i-1}$ and $S_i$ in two hash tables of size $O(\max_i\{i|S_i| + (i-1)|S_{i-1}|\})$. At the start of each loop, we access $S_{i-1}$ from one hash table and store $S_i$ into another hash table, and make the hash table for $S_{i-1}$ empty in the end of each loop. With the increasing of $i$, the size of the hash table can be incredibly huge that may cause memory overflow on a normal personal computer (e.g. $i = 3$ when $k = 5$).

As mentioned in Preliminaries, to reduce the amount of data, all polyominoes involved in the algorithm are free polyominoes. By this way, all rotation and the reflection of the original development are represented as the same binary array. In the storing process, all

Table 4.1: Adjacency list of cube of size $\sqrt{2} \times \sqrt{2} \times \sqrt{2}$.

| Square number | Up | Right | Down | Left |
|---|---|---|---|---|
| 1 | 12 | 11 | 2 | 3 |
| 2 | 1 | 11 | 5 | 4 |
| 3 | 12 | 1 | 4 | 6 |
| 4 | 3 | 2 | 5 | 6 |
| 5 | 4 | 2 | 8 | 7 |
| 6 | 3 | 4 | 7 | 9 |
| 7 | 6 | 5 | 8 | 9 |
| 8 | 7 | 5 | 11 | 10 |
| 9 | 6 | 7 | 10 | 12 |
| 10 | 9 | 8 | 11 | 12 |
| 11 | 10 | 8 | 2 | 1 |
| 12 | 9 | 10 | 1 | 3 |

partial developments are serialized to the binary arrays and stored in a hash table. This method consumes less memory.

## 4.2 Results of Enumerating Rep-cubes of Order $k$ for $k = 2, 4$

As a result of finding the rep-cube of order 2, by putting two developments of a cube aside, there are 2424 polyominoes of area 12. Among these 2424 dodecominoes, there are 33 regular rep-cubes of order 2 that can fold to a cube of size $\sqrt{2} \times \sqrt{2} \times \sqrt{2}$ and each of them can be divided into two developments of the unit cube (Figure 4.4 and Figure 4.5).

For the case of finding the regular rep-cube of order 4, we also implement this algorithm. As a result, we got the number of partial development of $i$ pieces as Table 4.2, which means there are 7185 rep-cubes of order 4. Some examples of these rep-cubes are shown in Figure 4.6.

**Procedure** CheckCoverSingle($P_i$)

**Input** : Polyomino $P_i$ in $S_i$;

**Output**: Whether $P_i$ can wrap to cube of size $\sqrt{k} \times \sqrt{k} \times \sqrt{k}$ or not;

```
foreach square in P_i do
    mark the square 1 as the start point
    foreach marked square in P_i do
        mark its unmarked adjacent squares as the adjacency matrix of the cube of
        size √k × √k × √k;
        check its marked adjacent squares has the same number as the adjacency
        matrix;
        if every square of P_i is marked by a unique number then
            return 1; // P_i can fold to the cube
        else if any square of P_i is marked by 2 or more numbers then
            break; // P_i has an overlap to fold to the cube
        else if 2 or more squares of P_i are marked by the same number then
            break; // P_i has an overlap to fold to the cube
return 0;
```

As shown in Figure 4.4 and Figure 4.5, we can observe that 17 rep-cubes out of 33 consist of two developments of the same shape. We may call them *uniform* rep-cubes. It is worth mentioning that 4 uniform rep-cubes marked in Figure 4.4 are also central symmetrical shapes and there are no such ones in non-uniform rep-cubes. In addition, we also find uniform rep-cubes of order 4, which are 158 in total. Ninety-eight of these uniform rep-cubes are made of pieces of the shape shown in Figure 4.7 (b).

In Figure 1 of [17], they gave three uniform rep-cubes of order 2 (Figure 1), 4, and 9. On the other hand, in [17], they also show a regular rep-cube of order 50 that contains all kinds of 11 developments of a unit cube. It may be worth focusing on these special cases for a larger $k$.

In the analysis of the results, we found two different patterns that make the same rep-cube. As shown in Figure 4.8, the difference in the composition does not affect the way of folding. Finding this kind of rep-cube can be an interesting topic for future research.

Table 4.2: The number of partial developments of regular rep-cubes of order 4.

| Set of partial developments | Number of developments |
|---:|---:|
| $S_1$ | 11 |
| $S_2$ | 2345 |
| $S_3$ | 114852 |
| $S_4$ | 7185 |

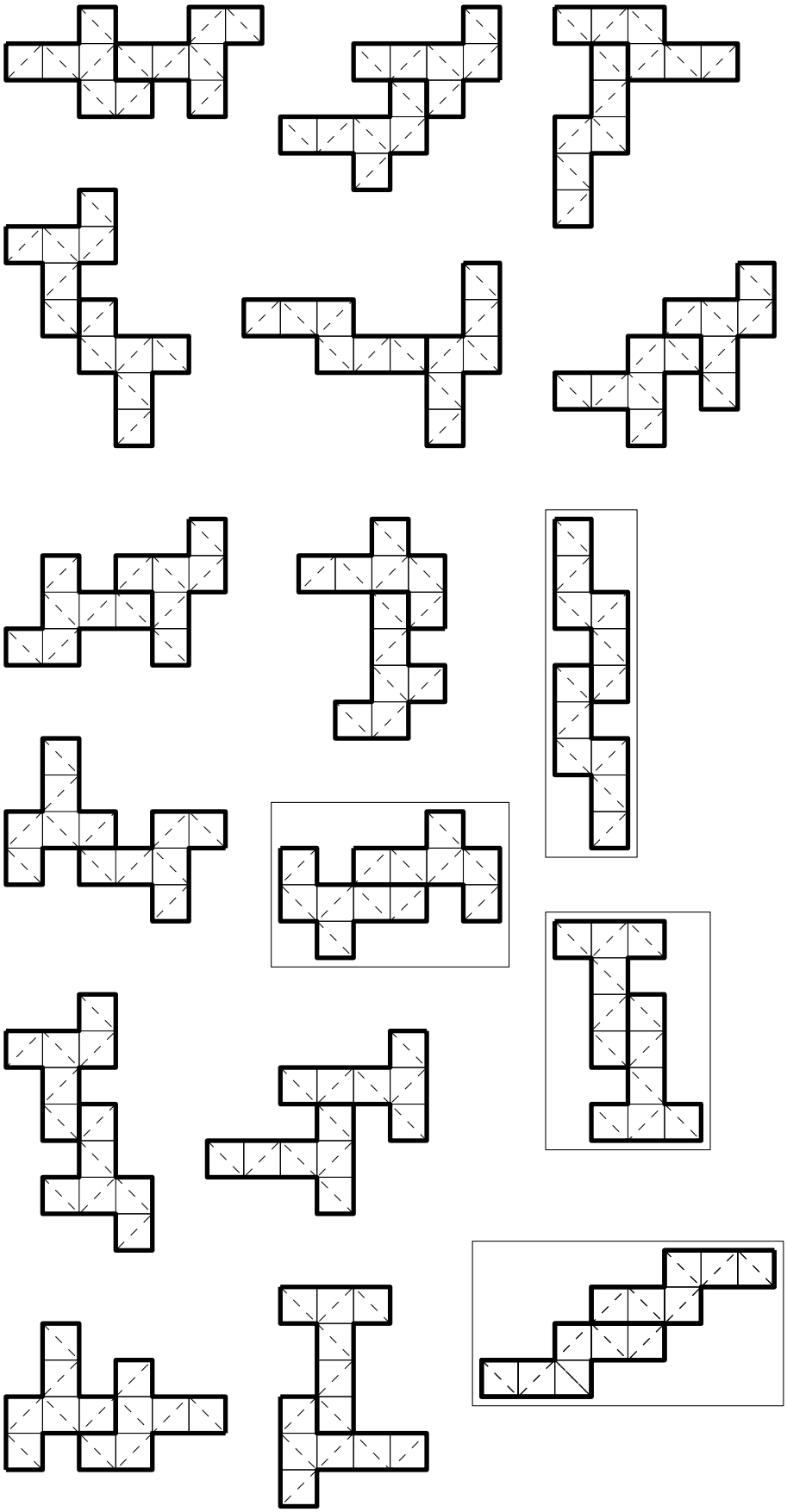


Figure 4.4: All 17 *uniform* rep-cubes of order 2, marked rep-cubes are central symmetric.

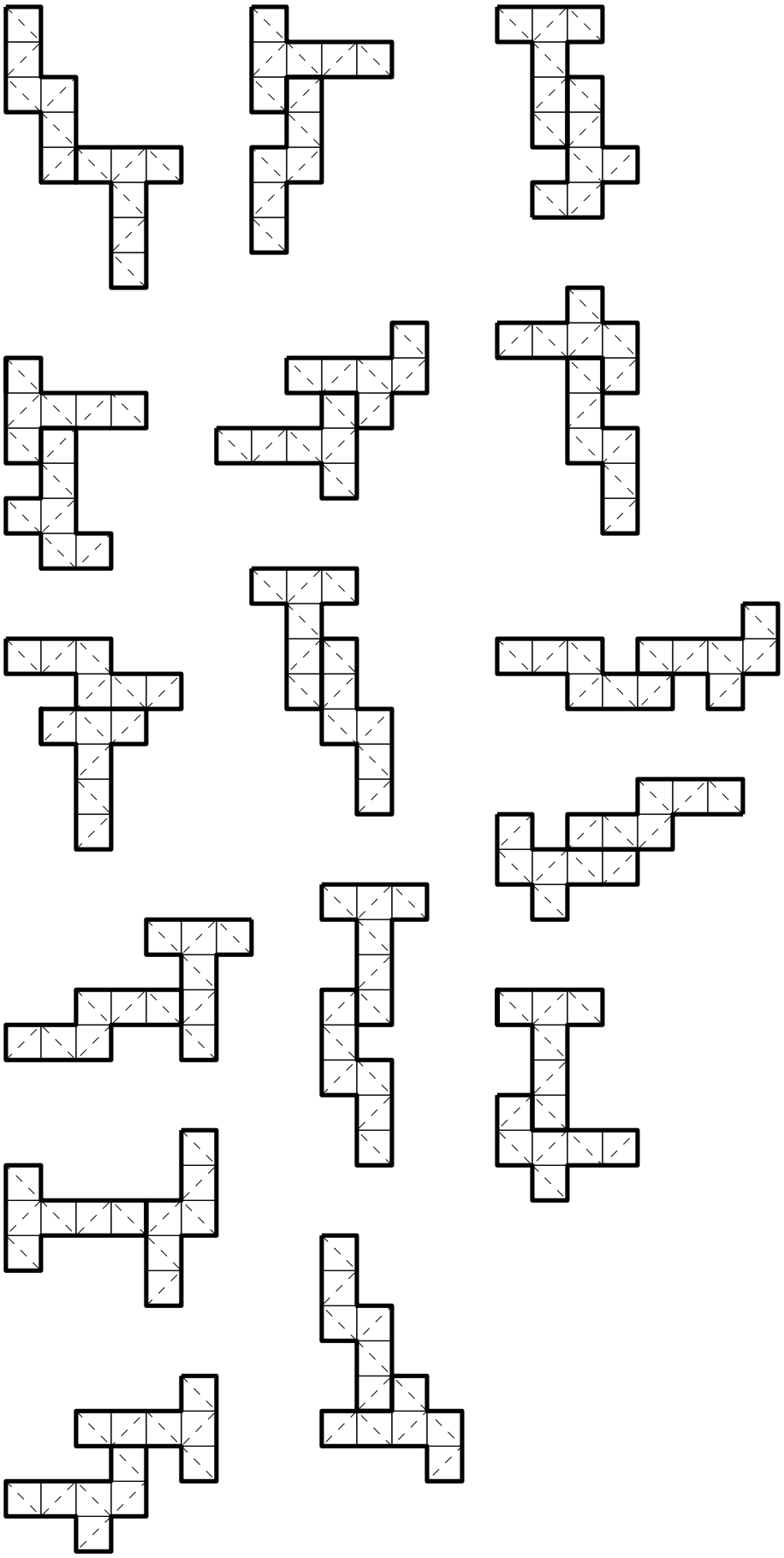
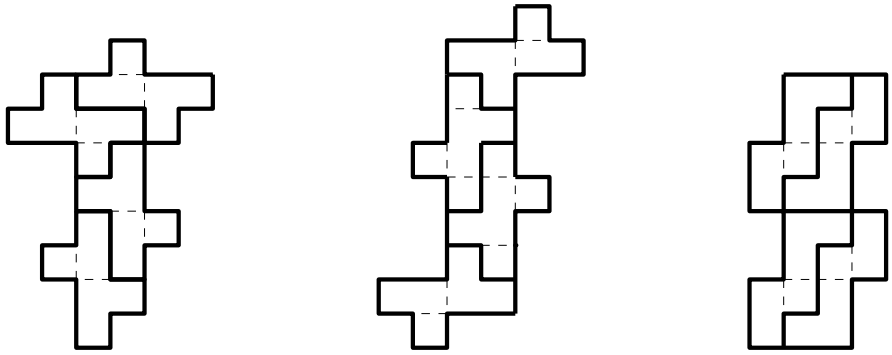
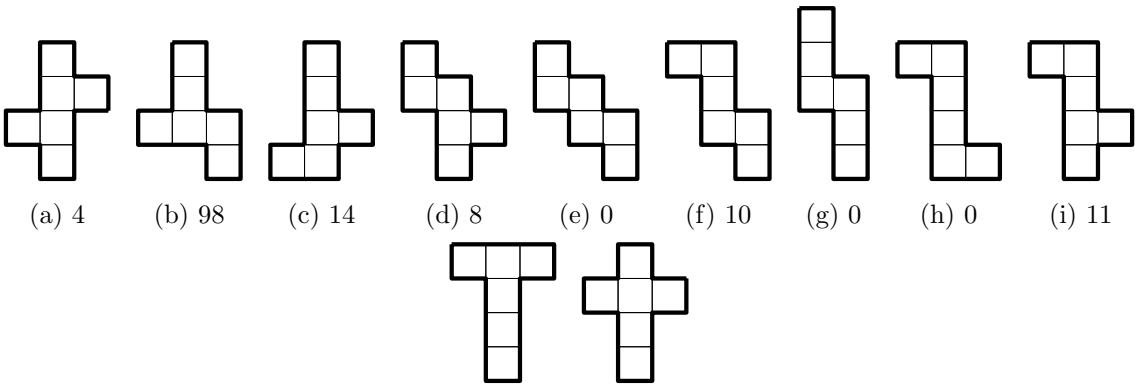


Figure 4.5: All regular rep-cubes of order 2 that are not *uniform*.



(a) (b) (c)

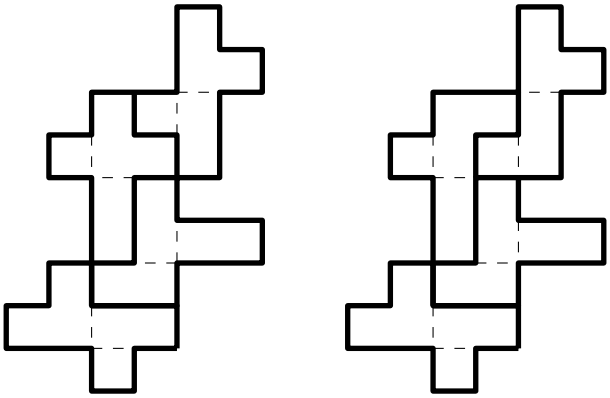
Figure 4.6: Some *uniform* rep-cubes of order 4.



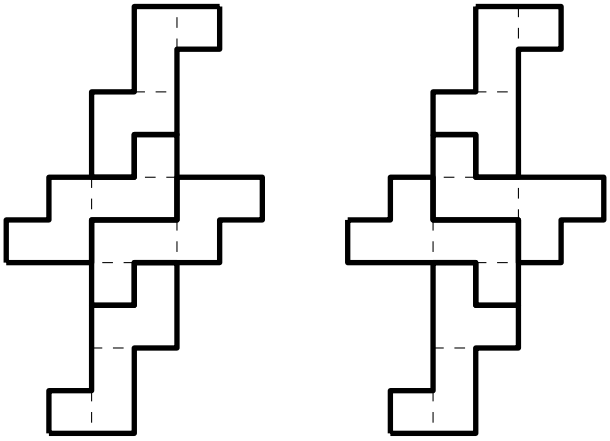
(a) 4 (b) 98 (c) 14 (d) 8 (e) 0 (f) 10 (g) 0 (h) 0 (i) 11

(j) 9 (k) 4

Figure 4.7: List of the numbers of uniform rep-cubes of order 4 made by each of 11 shapes.



(a)



(b)

Figure 4.8: Two different patterns can make the same rep-cube of order 4.

# Chapter 5

# Concluding Remarks

## 5.1 Contribution and Conclusion

The main contribution of our research is that we give a new algorithm for checking whether a polyomino can fold to a box of given size.

As the applications of this new algorithm, we partially solved the problem of "finding common developments of 3 boxes," "checking if there are centrosymmetric common developments of 3 boxes," and "enumeration of all rep-cubes for small $k$."

In the previous study, Shirakawa and Uehara showed infinite families of polyominoes that fold to three different boxes [15]. However, the smallest polyomino contained 532 unit squares. In this thesis, we show that there exist polyominoes of 30 unit squares that fold to three boxes if we allow us to fold along slanted lines.

As a follow-up to the study, we have also found the number of centrosymmetric common developments of surface area from 22 to 54. However, all of these common developments can only fold to two boxes. This discovery may help our future research.

Serval examples of rep-cubes are shown in [17], but there is no systematic way to find them. Previous study also proposed that regular rep-cubes of order $k$ exist for each $k = 2, 4, 5, 8, 9, 36, 50, 64$, and also $k = 36gk'^2$ for any positive integer $k'$ and an integer $g$ in $\{2, 4, 5, 8, 9, 36, 50, 64\}$. In this thesis, we provide a collection of regular rep-cubes of order 2 and order 4, and introduce a new notion of "uniform rep-cube".

## 5.2 Future Work

There are much future work in this area. For example, does there exist a polyomino that folds to four or more boxes? Is there any upper bound of the number of boxes which can be folded from one polyomino? In Theorem 2 of [23], the previous study indicates that there is no upper bound of the number of boxes such that they share the same surface area. However, it is still hard to imagine that one polyomino can fold to $n$ different boxes, for large $n$, e.g., 10000.

If we strictly fold along the edges of unit squares, the smallest possible surface area that may fold to three different boxes is 46 [15], which may produce three boxes of size $1 \times 1 \times 11$, $1 \times 2 \times 7$, and $1 \times 3 \times 5$. To solve this problem, we give a partial answer. When we restrict to centrosymmetric ones, there is no polyomino that can fold to these three boxes. It is still open for general polyominoes of surface area 46.

In the field of the rep-cubes, we wonder there is a characterization of an integer $k$ that whether the regular rep-cube of the order $k$ exists or not.

General development/folding problems also remain open, and these are what we want to know in the further study. In [10], Shirakawa et al. found a common development of a unit cube and an almost regular tetrahedron, however, a common development of two platonic solids are still open.

# Bibliography


[1] Joseph O'Rourke, "How to Fold It: The Mathematics of Linkages, Origami, and Polyhedra," Cambridge University Press, 2011.

[2] Erik D. Demaine and Joseph O'Rourke, "A survey of folding and unfolding in computational geometry," Combinatorial and computational geometry, Vol. 52, pp. 167–211, 2005.

[3] Anna Lubiw, and Joseph O'Rourke, "When can a polygon fold to a polytope," AMS Conference, 1996.

[4] Daniel Kane, Gregory N. Price, and Erik D. Demaine, "A pseudopolynomial algorithm for Alexandrov's Theorem," Workshop on Algorithms and Data Structures, Springer Berlin Heidelberg, 2009.

[5] Erik D. Demaine, and Joseph O'Rourke, "Geometric Folding Algorithms: Linkages, Origami, Polyhedra," Cambridge: Cambridge university press, 2007.

[6] Zachary Abel and Erik D. Demaine and Martin L. Demaine and Hiroaki Matsui, Günter Rote, and Ryuhei Uehara, "Common development of several different orthogonal boxes," Canadian Conference on Computational Geometry (CCCG), pp. 77–82, 2011.

[7] 松井寛彰, "複数の凸多面体を折ることができる展開図に関する研究," master thesis, 2013.

[8] Toshihiro Shirakawa and Ryuhei Uehara, "Common Developments of Three Different Orthogonal Boxes," International Journal of Computational Geometry and Applications, Vol. 23, No.1, pp. 65–71, 2013.

[9] Jun Mitani, and Ryuhei Uehara, "Polygons folding to plural incongruent orthogonal boxes," Canadian Conference on Computational Geometry (CCCG), pp. 39–42, 2008.

[10] Toshihiro Shirakawa and Takashi Horiyama and Ryuhei Uehara, "Construct of Common Development of Regular Tetrahedron and Cube," European Workshop on Computational Geometry (EuroCG), pp. 47–50, 2011.

[11] Jin Akiyama, "Tile-Makers and Semi-Tile-Makers," The Mathematical Association of America, Vol. 114, pp. 602–609, 2007.

[12] Jin Akiyama and Chie Nara, "Developments of Polyhedra Using Oblique Coordinates," Journal of the Indonesian Mathematical Society, Vol. 13, pp. 99–114, 2007.

[13] Zachary Abel and Erik D. Demaine and Martin L. Demaine and Hiro Ito and Jack Snoeyink and Ryuhei Uehara, "Bumpy Pyramid Folding," Canadian Conference on Computational Geometry (CCCG), pp. 258–266, 2014.

[14] Therese Biedl, Timothy Chan, Erik D. Demaine, Martin L. Demaine, Anna Lubiw, Ian Munro, and Jeffrey Shallit, "Notes from the University of Waterloo Algorithmic Problem Session," Sept. 8, 1999.

[15] Toshihiro Shirakawa and Ryuhei Uehara, "Common Developments of Three Incongruent Orthogonal Boxes," International Journal of Computational Geometry and Applications, vol. 23, pp. 65–71, 2013.

[16] Solomon W. Golomb, "Polyominoes," Princeton University Press, 1994.

[17] Zachary Abel, Brad Ballinger, Erik D. Demaine, Martin L. Demaine, Jeff Erickson, Adam Hesterberg, Hiro Ito, Irina Kostitsyana, Jayson Lynch and Ryuhei Uehara, "Unfolding and Dissection of Multiple Cubes, Tetrahedra, and Doubly Covered Squares," Journal of Information Processing, accepted, 2017.

[18] Martin Gardner, "Hexaflexagons, Probability Paradoxes, and the Tower of Hanoi," Cambridge, 2008.

[19] Martin Gardner, "Knots and Borromean Rings, Rep-Tiles, and Eight Queens," Cambridge, 2014.

[20] Greg Aloupis, Prosenjit K. Bose, Sébastien Collette, Erik D. Demaine, Martin L. Demaine, Karim Douïeb, Vida Dujmović, John Iacono, Stefan Langerman, Pat Morin, "Common unfoldings of polyominoes and polycubes," Computational Geometry, Graphs and Applications, Springer Berlin Heidelberg, pp. 44–54, 2011.

[21] Takashi Horiyama and Koichi Mizunashi, "Folding Orthogonal Polygons into Rectangular Boxes," Korea-Japan Joint Workshop on Algorithms and Computation, 2016.

[22] Alexander Bobenko, and Ivan Izmestiev, "Alexandrov 's theorem, weighted Delaunay triangulations, and mixed volumes," Annales de l'institut Fourier, Vol. 58, No. 2, 2008.

[23] Dawei Xu, Toshihiro Shirakawa, Takashi Horiyama and Ryuhei Uehara, "Common Developments of Three Incongruent Boxes of Area 30," International Conference on Theory and Applications of Models of Computation, pp. 236–247, 2015.

# Publications

# Bibliography

## International Journal

[1] Dawei Xu, Takashi Horiyama, Toshihiro Shirakawa, Ryuhei Uehara, "Common Developments of Three Incongruent Boxes of Area 30," Computational Geometry; Theory and Applications, vol. 64, pp. 1–12, 2017.

## International Conferences

[2] Dawei Xu, Toshihiro Shirakawa, Takashi Horiyama and Ryuhei Uehara, "Common Developments of Three Incongruent Boxes of Area 30," International Conference on Theory and Applications of Models of Computation, pp. 236–247, May 18–20 2015.

[3] Dawei Xu, Takashi Horiyama, Ryuhei Uehara, "Rep-cubes: Unfolding and Dissection of Cubes," 29th Canadian Conference on Computational Geometry (CCCG), July 26–28, 2017, accepted.

[4] Dawei Xu, Toshihiro Shirakawa, Ryuhei Uehara, "Common Unfolding of Three Different Boxes of Surface Area 30," 17th Japan Conference on Discrete and Computational Geometry and Graphs, Sept. 15–16, 2014.

[5] Dawei Xu, Toshihiro Shirakawa, Ryuhei Uehara, "Common Unfolding of Three Different Boxes," The 8th Annual Meeting of Asian Association for Algorithms and Computation (AAAC), May 9–10, 2015.

[6] Dawei Xu and Ryuhei Uehara, "Enumeration of all developments," The International Conference for High Performance Computing, Networking, Storage and Analysis (SC 15), poster talk, Nov. 15–20, 2015.