Computer-assisted Analysis of the Sparsest Packing and the Topswops Problem

Kento Kimura
Department of Computer Scinece, Gunma University.
Ph.D. Thesis, Supervised by Prof. Kazuyuki Amano

January 22, 2023

Acknowledgment

First, I would like to express my gratitude to my Ph.D. supervisor, Kazuyuki Amano. He was also my supervisor when I was in my bachelor's program and master's programs. I have discussed several problems with him for seven years, and he influences much of my thinking in this area. He has helped me learn knowledge in this area and the research skills for becoming a good researcher. Because of his solid support and encouragement, I could complete this Ph.D. thesis.

I am also deeply indebted to the co-referees of the thesis, Shin-ichi Nakano, Yoichi Seki, Ken-etsu Fujita, and Toru Araki. Their comments and suggestions for this thesis were constructive. I am fortunate to have many coauthors: Tetsuya Araki, Shin-ichi Nakano, and Kazuyuki Amano. I would like to thank all of them for our fruitful collaborations.

Finally, I would like to thank express to my family. My parents and grandmother were patient enough to support me for 28 years. I sometimes enjoyed talking and playing video games with my only older brother, and I was pleased with them.

Abstract

Many mathematical problems, such as those on the subject of mathematical puzzles, often are problems whose rule describing the problem is primitive and simple. Nevertheless, mathematical structures of solutions for such a problem can often be extremely challenging to solve. In this thesis, we investigated the sparsest packing and the Topswops problem by using computer-assisted and theoretical analysis.

The first is a card game called "Topswops." Given a deck of n cards numbered 1 to n, continue the following operation until the top card is 1: If the top card of the deck is k, then turn over a block of k cards at the top of the deck. Let $f(n)$ be the maximum number of steps of Topswops on n cards. Despite 50 years of research, the exact value of $f(n)$ has yet to be determined. In this thesis, by applying an algorithm developed by Knuth in a parallel fashion, we conclude that $f(18) = 191$ and $f(19) = 221$.

The second is a puzzle called "anti-slide." The anti-slide packing is a packing of identical pieces of some specified shape for a three-dimensional box in such a way that none of the pieces in the box can slide. Given a box of some specified size and identical pieces of some specified shape, we find a sparsest anti-slide packing. In this thesis, we analyzed the sparsest anti-slide packings for the three cases of T-tetrominoes, L-tricubes, and $2 \times 2 \times 1$ pieces. We consider the problem for the case of a two-dimensional square box using T-tetromino pieces. We show that, for a square box of side length n, the number of pieces in a sparsest packing is exactly $\lfloor 2n/3 \rfloor$ when $n \not\equiv 0 \pmod 3$, and is between $2n/3 - 1$ and $n - 1$ when $n \equiv 0 \pmod 3$. Next, we consider the problem for the case of a three-dimensional cubic box of side length n using L-tricubes. We find a new construction of an anti-slide packing of $n^2/2$ L-tricubes. Finally, we give the construction of

an anti-slide packing of $2 \times 2 \times 1$ pieces with volume density 0.48.

keyword — discrete mathematics, Topswops, enumeration, sparsest packing, anti-slide packing, integer programming

Contents

List of Figures

Chapter 1

Introduction

Many discrete mathematical problems are long-standing open, even though the rule describing the problem is primitive and simple. In this thesis, we focus on investigating the mathematical structure of a solution for two open problems.

Many problems described based on a mathematical rule can be formulated as an integer linear programming (IP) and a SAT naturally. The approach for the problem often is that we analyze the mathematical structure of the solutions for a model for the problem using solvers. Since recently mixed IP solvers and SAT solvers are implemented by the theory of algorithm and heuristic optimization, the power of the solvers has dramatically improved. The older solver could not output the solutions of instances for a problem that we would like to analyze, but the current solvers can output the solutions to the instances. The approach using the solver is effective and used in the research for discrete mathematics [1] and computational geometry [2].

Mathematical puzzles provided us with interesting problems in various fields in mathematics. In terms of complexity, many puzzles described in the literature have been shown to be NP-complete, which seems to be a common essence of puzzles [3, 4, 5, 6]. Major complexity results for puzzles are surveyed by Hearn and Demaine [7]. Also, the essence of a solution to the puzzle such as Hitori [8] is investigated.

However, there are some problems for which the approach using solvers cannot compete with the problems. For example, the problem of finding

the minimum number of wires of a depth-2 threshold circuit emulating the parity function can be formulated as an IP but the solver will be stuck. It is known that there exists a depth-2 threshold circuit computing parity with wires that is quadratic for the number of variables [9]. This is nearly close to the lower bound [10] but does not yet match it. Threshold circuits and threshold function are well-studied, such as threshold circuits for parity [11], the energy of threshold circuits for parity [12], and a hypercube cut by hyperplanes [13].

In this thesis, we investigated the sparsest packing which is similar to the (densest) packing problems and the problem of the card game which is similar to the operation like the sort.

1.1 Topswops

Consider a deck of face-up n cards numbered 1 to n arranged in random order, which can be viewed as a permutation on $\{ 1, 2, \ldots, n \}$. Given the deck of n cards, one player continues the following operation on the deck until the top card is 1: If the top card is k, then turn over a block of k cards at the top of the deck. This card game is called *Topswops*, originally invented by J.H. Conway in 1973. We call a deck before starting the game *initial*. For a natural number n, let $f(n)$ be the maximum number of steps until termination for Topswops on n cards.

For example, let $n = 5$ and $(3, 1, 4, 5, 2)$ be an initial deck. The game goes as

$$\begin{aligned}&(\underline{3, 1, 4}, 5, 2) \to (\underline{4, 1, 3, 5}, 2) \to (\underline{5, 3, 1, 4, 2}) \to (\underline{2, 4}, 1, 3, 5)\\ &\to (\underline{4, 2, 1, 3}, 5) \to (\underline{3, 1, 2}, 4, 5) \to (\underline{2, 1}, 3, 4, 5) \to (1, 2, 3, 4, 5),\end{aligned} \tag{1.1}$$

and terminates after seven steps.

The initial deck $(3, 1, 4, 5, 2)$ is finally transformed at the sorted position $(1, 2, 3, 4, 5)$. Note that the initial deck is not always transformed to the sorted position $(1, 2, \ldots, n)$ at the termination of the game. For instance, let $n = 6$ and $(4, 1, 6, 5, 2, 3)$ be an initial deck. The game progresses according to the rule of Topswops and terminates with the deck $(1, 4, 3, 2, 5, 6)$ after

ten steps. In this example, the deck $(1, 4, 3, 2, 5, 6)$ is not at the sorted position $(1, 2, 3, 4, 5, 6)$.

H.S. Wilf [14, pp. 81 – 82] proved that Topswops halts within a finite number of steps for any decks of n cards, which will be described in Section 2.1. We call an initial deck that needs $f(n)$ steps largest. For example, the initial deck $(3, 1, 4, 5, 2)$ is largest for $n = 5$, and the initial deck $(4, 1, 6, 5, 2, 3)$ is largest for $n = 6$.

As of 2021, the exact values of $f(n)$ for $n \leq 17$ in [15, A000375] were obtained by an exhaustive search with some pruning techniques. The exact values of the number of the largest decks for $n \leq 17$ in [15, A123398] also was known. In Al Zimmermann's Programming Contests on February 2011 [16], the players found several better decks for $n \geq 19$ by using local search.

The other version of the problem is to find the maximum number of steps until termination whenever the deck is finally transformed in ascending order. The problem has been solved for $n \leq 20$ [15, A000376 and A174498].

In related works, the problem of *sorting by reversals* is investigated well. Let $(a_1, a_2, \ldots, a_n)$ be a sequence of length n such that an element is a different number. For some sequence $a_1, a_2, \ldots, a_n$ of length n and some two indices i, j with $1 \leq i < j \leq n$, we transform the sequence

$$a_1, a_2, \ldots, a_{i-1}, \underline{a_i, a_{i+1}, \ldots, a_{j-1}, a_j}, a_{j+1}, \ldots, a_n$$

to

$$a_1, a_2, \ldots, a_{i-1}, \underline{a_j, a_{j-1} \ldots, a_{i+1}, a_i}, a_{j+1}, \ldots, a_n.$$

The problem is to find the minimum number of steps of the transformation for sorting on sequences of length n. G.A. Watterson et al. [17] and V. Bafna et al. [18] showed that the value of the minimum number of steps of the transformation is exactly $n - 1$. The version of the problem that is only allowed to do prefix reversals is also well-studied, called *Pancake Problem*.

In 1977, H. Dweighter et al. [19] commented that the steps of the transformation is at most $2n - 6$ and at least $n + 1$ for a sequence with length n. In 1979, W. H. Gates et al. [20] showed that the steps of the transformation is at most $(5n + 5)/3$. In 2009, B. Chitturi et al. [21] showed that the steps of the transformation is at most $18n/11$. Moreover,

M.H. Heydari et al. [22] showed that the steps of the transformation is at least $15n/14$ for some number n. L. Bulteau et al. [23] discussed the NP-completeness of the Pancake Problem. See the introduction of [24] for more background on the pancake problem.

In Chapter 2, we show the values of $f(18)$ and $f(19)$ via our computational experiments by applying an algorithm developed by Knuth in a parallel fashion.

1.2 Anti-slide

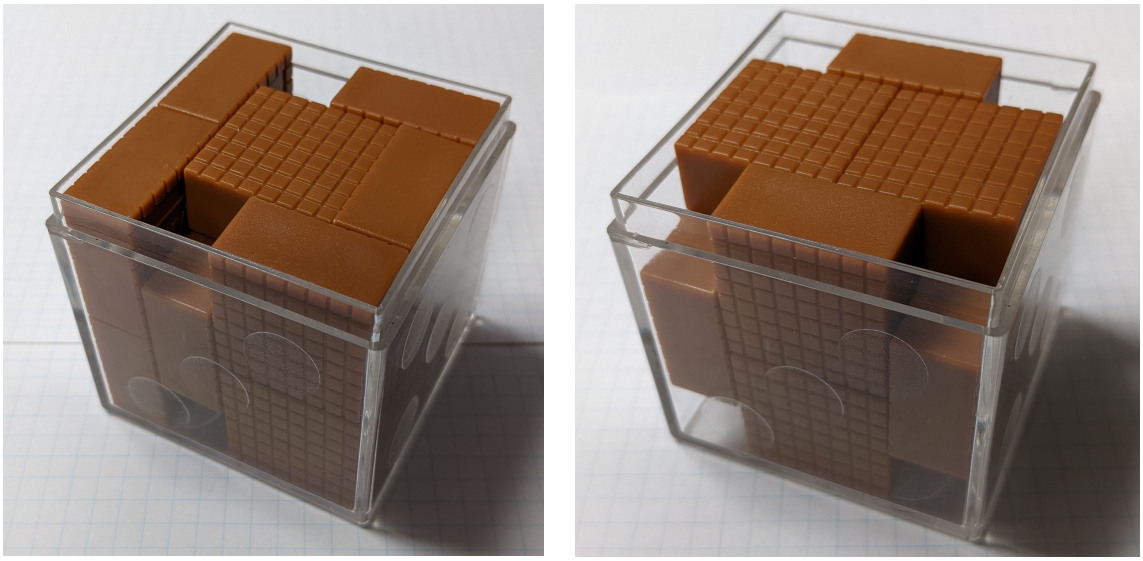
"*Anti-slide*" [25] is a puzzle that given a $4 \times 4 \times 4$ box and a number of $2 \times 2 \times 1$ pieces, consider how to pack $2 \times 2 \times 1$ pieces into a $4 \times 4 \times 4$ box in such a way that none of the pieces can move. This puzzle assumes that there is no friction between pieces or between pieces and the inner wall of a box, and each piece can fit inside the box. The objective is to find such a packing with the smallest number of pieces. The puzzle was originally invented by William Strijbos in 1994. For example, an anti-slide packing is shown in Fig. 1.1. Including Fig. 1.1 (b), there exist three anti-slide packings of 12 pieces. Since there exist no anti-slide packings with 11 or less pieces [25], optimal packings use 12 pieces.

Let us consider the generalized problem of this puzzle. Given a box of some specified size and identical pieces of some specified shape, the problem is to find a sparsest anti-slide packing for the box.

In previous our work [26], we investigated the asymptotic behavior of the minimum number of pieces in a stable packing for an $n \times n \times n$ box for various shapes of *polycubes* illustrated in Fig. 1.2 and observed that these could be categorized into three groups, which is described later in Section 4.1.

In Chapter 3, we will discuss the minimum number of pieces with anti-slide packings for a two-dimensional square box using T-tetrominoes. In Chapter 4, we focus on sparsest anti-slide packings for $2 \times 2 \times 1$ pieces and L-tricube pieces for a three-dimensional cubic box.

It is quite natural to consider that the dual of the sparsest packing is the densest packing. Indeed, packing problems have been well-studied in



(a) (b)

Fig. 1.1: An anti-slide packing for a $4 \times 4 \times 4$ box : (a) with 15 pieces, (b) with 12 pieces.

mathematics, discrete geometry, and computer science [27]. The problems ask for finding a densest packing of identical objects for a space where an object is not allowed to overlap other. Kepler conjecture is a version of the packing problem where an object is a sphere of the same radius, and a space is a three-dimensional Euclidean space. In 2014, Hales et al. [28] proved the Kepler conjecture using generic proof assistants. See, e.g., the introduction of [28] for a long history of the research.

1.3 Related papers

The result in this thesis is based on the following one published paper.

- Kento Kimura, Kazuyuki Amano, and Tetsuya Araki,
  On the Minimum Number of Pieces for Two-Dimensional Anti-Slide Using T-Tetrominoes,
  IEICE Transactions on Information and Systems, Volume E104.D,

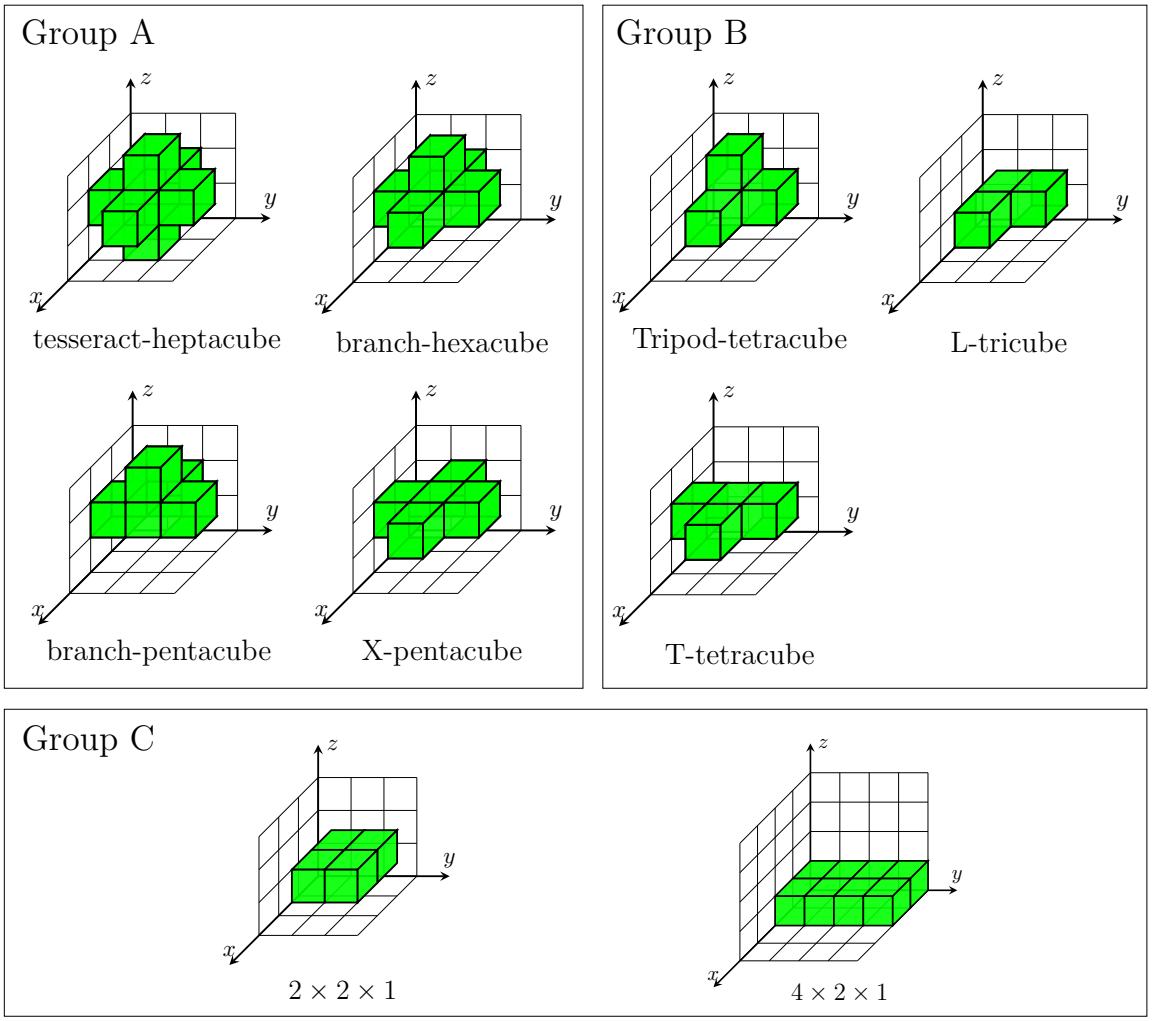


Fig. 1.2: Several shapes of polycubes analyzed in [26].

Issue 3, Pages 355 – 361 (March 2021).
DOI:10.1587/transinf.2020FCP0007

The rest results in this thesis is based on the following one preprint and one conference abstract.

- Kento Kimura, Atsuki Takahashi, Tetsuya Araki and Kazuyuki Amano, Maximum Number of Steps of Topswops on 18 and 19 Cards, arXiv:2103.08346 (Mar. 2021)

- Kento Kimura and Kazuyuki Amano,

Upper Bounds on the Minimum Number of Pieces for Anti-slide Packing,
The 24th Japan Conference on Discrete and Computational Geometry, Graphs, and Games, (Virtual, 2022.9.9 - 11).

The published paper is related to Chapter 3. The preprint is related to Chapter 2. The conference abstract is related to a part of Chapter 4.

1.4 Other papers by the author

We list the author's published paper that are not include in the list of the previous section.

- Kento Kimura, Kazuyuki Amano and Shin-ichi Nakano,
  Escape from the Room,
  Proceedings of the 28th International Computing and Combinatorics Conference (COCOON 2022), Lecture Notes in Computer Science (LNCS), Volume. 13595, Pages 232 – 241 (January 2023).

Chapter 2

Topswops

2.1 Introduction

In this chapter, we study the Topswops problem. Recall the card game called Topswops. A deck of n cards numbered 1 to n arranged in random order. Given the deck of n cards, one player continues the following operation on the deck until the top card is 1: If the top card is k, then turn over a block of k cards at the top of the deck. For a natural number n, let $f(n)$ be the maximum number of steps until termination for Topswops on n cards.

H.S. Wilf [14, pp. 81 – 82] showed the finiteness of Topswops. A card in a deck is said to be in *natural position* if the value of the card is the same as its position in the deck. For a deck of n cards, the Wilf number for the deck denotes $\sum_{i=1}^{n} \mathbf{1}[i\text{-th card is in natural position}] \cdot 2^{i-1}$ where $\mathbf{1}[\cdot]$ denotes 1 if the condition in the bracket is satisfied, and 0 otherwise. For example, let $(3, 2, 7, 8, 5, 1, 4, 6, 9)$ be a deck. If we read the deck down from top, then there are three cards 2, 5, and 9 in natural position; hence, the Wilf number of the deck is $2^{2-1} + 2^{5-1} + 2^{9-1} = 530$. H.S. Wilf showed the Wilf number for a deck increases until termination, as follows.

Claim 1. *H.S. Wilf [14, pp. 81 – 82] Let n be a natural number and A be a deck of n cards such that the top card is not 1. Then the Wilf number for a deck applying A the operation once is greater than the Wilf number for A.*

Since the Wilf number is upper bounded by $2^n - 1$, Claim 1 implies that the game terminates within $2^n - 1$ steps. The above discussion shows that

$f(n) \leq 2^n - 1$. The best-known upper bound on $f(n)$ is $F(n+1) - 1 = O(1.618^n)$ [29, 30], where $F(k)$ is the k-th Fibonacci number. See D. Berman [29] or Problems 108 of [30]. The best-known lower bound is $\Omega(n^2)$ [31]. There is an exponential gap between the upper and lower bounds on $f(n)$. This problem is long-standing open for 50 years.

Recall that we call an initial deck that needs $f(n)$ steps *largest*. For a natural number n, let $g(n)$ be the number of largest decks on n cards. As of 2021, exact values of $f(n)$ in OEIS [15, A000375] and $g(n)$ in OEIS [15, A123398] for $n \leq 17$ were obtained by using several brute-force searches, which is as follows:

$n =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$f(n) =$	0	1	2	4	7	10	16	22	30	38	51	65	80	101	113	139	159
$g(n) =$	1	1	2	2	1	5	2	1	1	1	1	1	1	4	6	1	2

In a related work, Komano et al. [32] discussed zero-knowledge proof protocol for Topswops.

In this chapter, we describe our effort for extending this list for $n = 18$ and 19. Namely, by applying an algorithm developed by Knuth [33] in a parallel fashion, we conclude that $f(18) = 191$ and $f(19) = 221$. We simultaneously find that $g(18) = 4$ and $g(19) = 1$.

The rest of this chapter is as follows. In Section 2.2, we give an explanation of Knuth's algorithm [30]. Then, in Section 2.3, we describe our computational experiments for determining $f(18)$ and $f(19)$. The code used in our experiments can be viewed on GitLab at `https://gitlab.com/kkimura/tswops`.

2.2 Knuth's algorithm

In this section, we explain an algorithm for finding a largest deck for Topswops used in our experiment, which was developed by Knuth [30, Solution of Problem 107] (see also [33] for the code itself). Three algorithms were described there, and we use the most efficient one, which is referred to as a "better" algorithm.

A card is in a card sleeve that is numbered the minus value $-i$, which means the i-th card in an initial deck from the top. The value written on the card sleeve should be visible on the back of the card. We consider that the game starts with an initial deck of all the face down cards in a card sleeve. We call a deck of all the face down cards *face-down*. If the top card is face down, then we turn it up and apply the prefix reversal on its value into the deck. The game can progress as the Topswops.

Let A be a face-down initial deck of n cards that can be viewed as an array with $(-1, -2, \ldots, -n)$. For example, let $(3, 1, 4, 5, 2)$ be an initial deck of face up cards. The game goes as

$$
\begin{aligned}
&(-1,-2,-3,-4,-5) = (\underline{3,-2,-3},-4,-5) && (\because \text{1st card is } 3.)\\
&\to (-3,-2,3,-4,-5) = (\underline{4,-2,3,-4},-5) && (\because \text{3rd card is } 4.)\\
&\to (-4,3,-2,4,-5) = (\underline{5,3,-2,4,-5}) && (\because \text{4th card is } 5.)\\
&\to (-5,4,-2,3,5) = (\underline{2,4},-2,3,5) && (\because \text{5th card is } 2.)\\
&\to (\underline{4,2,-2,3},5) \to (\underline{3,-2,2},4,5) \to (\underline{2,-2},3,4,5)\\
&\to (-2,2,3,4,5) = (1,2,3,4,5) && (\because \text{2nd card is } 1.)
\end{aligned}
$$

and terminates. The prefix reversals in this progress correspond to Eq. (1.1).

Let A' be a deck with the face down top card and at least two face down cards. Knuth introduced an algorithm that performs the prefix reversal for a deck A', the value k in the top of A', and steps c from the start of the game to A'.

Algorithm 1 The game progresses until the other face down card appears.

```
1: procedure TrySwops(A', k, c)
2:     c := c + 1
3:     A'_1 := k
4:     loop
5:         Turn over a block of A'_1 cards of A'.
6:         if A'_1 ≤ −1 then
7:             Go to line 9.
8:         c' := c' + 1
9:     return c'.
```

See Algorithm 1. Line 6 in Algorithm 1 shows that if the face down top of the deck appears, then the loop exits. The loop in Algorithm 1 will eventually halt to appear in the face down top of the deck. We repeatedly apply the algorithm into A' until the top of A' is face down and there is exactly one face down card in A'.

Next, we explain how to restore an initial deck. For a natural number n, let $[n]$ denote the set $\{ 1, 2, \ldots, n \}$. For an initial deck A, let $S(A)$ be a list $(d_1, d_2, \ldots, d_k)$ $(k \leq n)$ where d_i is the i-th card that is turned up at the top of the deck in the game starting from A. For example, $S((3, 1, 4, 5, 2)) = (3, 4, 5, 2, 1)$ (see Eq. (1.1)) and $S((3, 5, 4, 1, 2)) = (3, 4, 1)$. Notice that the length of $S(A)$ depends on A, but the last element of $S(A)$ is always 1.

An important property is that if A is largest, then the length of $S(A)$ must be n. This can be verified by seeing that if $S(A) = (d_1, \ldots, d_{k-1}, 1)$ for some $k < n$, then we can always create another deck A' such that the first k elements of $S(A')$ is $(d_1, \ldots, d_{k-1}, d')$ for $d' \in \{ 2, \ldots, n \} \setminus \{ d_1, \ldots, d_{k-1} \}$ and that the game for A' is strictly longer than the one for A.

Let P be the set of all lists $p = (p_1, p_2, \ldots, p_n)$ such that p is a permutation on $[n]$ and $p_n = 1$. Given a list $p \in P$, we can get an initial deck $S^{-1}(p)$ by the following algorithm. In Algorithm 1, the minus value $-i$ in A means that the i-th card in a deck is not specified yet.

Algorithm 2 Generate an initial deck with length n

```
1: procedure GenInitDeck(p)
2:     Let A be an array with (−1, −2, ..., −n).
3:     for i = 1, 2, ..., n do
4:         a_{−A_1} ≔ p_i
5:         A_1 ≔ p_i
6:         while A_1 > 1 do
7:             Turn over a block of A_1 cards of A.
8:     return (a_i)_{i∈[n]}
```

The above arguments suggest that we can determine $f(n)$ by examining all $(n-1)!$ lists in P together with Algorithm 2. Essentially, Knuth's algorithm enumerates these lists as well as corresponding decks in a depth-

first fashion. Moreover, the algorithm applies two pruning criteria to reduce the size of the search tree.

The behavior of the algorithm without the pruning is visualized as a tree structure, which is illustrated in Fig. 2.1 for $n = 4$. An initial deck denotes

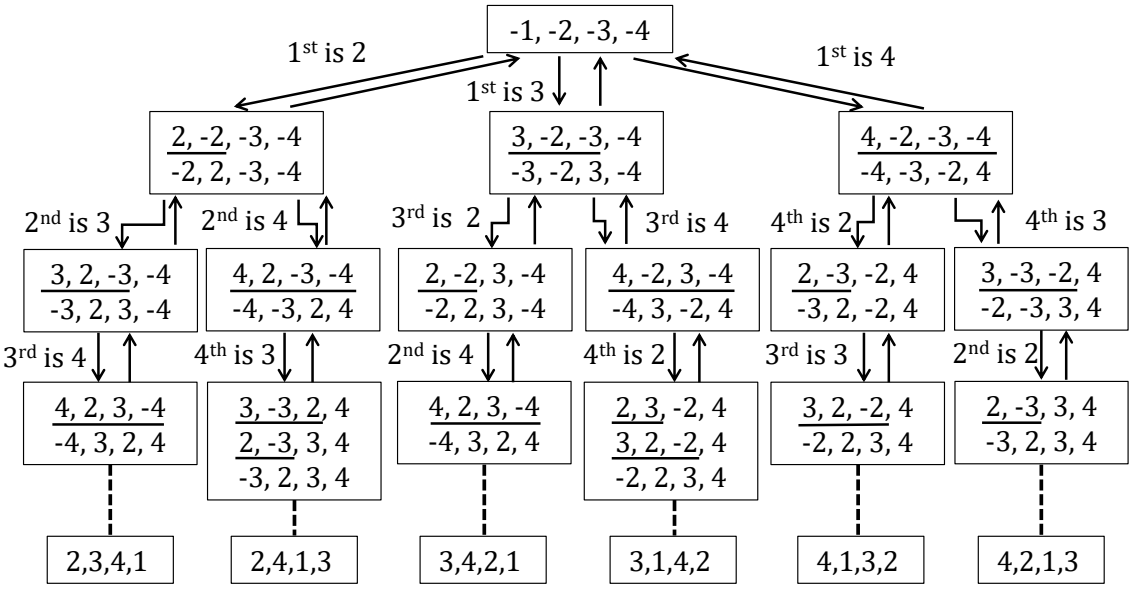


Fig. 2.1: Traverse the initial decks for $n = 4$ by the backtrack.

$-1, -2, -3, -4$, illustrated in Fig. 2.1. Each text labeled with a down arrow means that the card with the number is inserted into the i-th card sleeve. See the path from $(-1, -2, -3, -4)$ to $(4, 2, 1, 3)$. A list $p = (4, 3, 2, 1)$ that the path generates corresponds to $S((4, 2, 1, 3))$.

Let us see the two pruning criteria to reduce the size of the search tree. The first pruning is based on the fact that a largest deck must be a *derangement*, i.e., the k-th card from the top is not k for every $k \in [n]$. In order to explain the second pruning, we need some definitions. Let A_c be the deck obtained from A by executing c steps of the game. Let $T(A_c)$ denote the largest integer k such that the cards numbered $1, 2, \ldots, k$ are located at positions at $1, 2, \ldots, k$ (in an arbitrary order) in the deck A_c. It is obvious that if $f(T(A_c)) + c < f(n)$, then A is not largest. Although $f(n)$ is not known beforehand, we can use any lower bound $\ell(n)$ on $f(n)$ in the right hand side of inequalities for pruning. Note that the depth of the search

tree without pruning is $(n-1)$ and each node at depth k has $n-1-k$ children.

Let N be a set with $N = \phi$ in a power set of $\{ 2, 3, 4, \ldots, n \}$. Let A be an array of length n with integers. Let p be a partial permutation for a set in P. Note that p is viewed as a list in an algorithm. Let c be a positive integer. We implement the backtracking procedure of Knuth's algorithm into the recursive procedure.

Algorithm 3 The algorithm enumerates all possible initial decks.

```
1:  procedure NextNode(N, A, p, c)
2:      for all k ∈ N do
3:          if A_1 = −k then
4:              Continue;
5:          if k = T(A) then
6:              Let A′ be an array copied with A.
7:              Turn over a block of k cards of A′.
8:              if ℓ(T(A′)) + c + 1 < ℓ(n) then
9:                  Continue;
10:         else if ℓ(T(A)) + c + 1 < ℓ(n) then
11:             Continue;
12:         Let A′ be an array copied with A.
13:         Add k at the end of p.
14:         c′ ← TrySwops(A′, k, c)
15:         if |p| = n − 1 then
16:             if c′ ≥ ℓ(n) then
17:                 ℓ(n) := c′
18:                 Add 1 at the end of p.
19:                 Output the value of GenInitDeck(p).
20:                 Remove the last element of p.
21:         else
22:             NextNode(N \ { k }, A′, p, c′)
23:         Remove the last element of p.
```

Later, we will describe the behavior of the above procedure with the

behavior of the entrypoint. The entrypoint for the enumeration is as follows.

Algorithm 4 The procedure initializes and calls NextNode.

1: Let p be an empty list in which the element type is a natural number.
2: Let A be an array with $(-1, -2, \ldots, -n)$.
3: NEXTNODE($\{ 2, 3, 4, \ldots, n \}$, A, p, 0)

Line 3 in Algorithm 4 gives four parameters to NEXTNODE. N means a set of cards such that a card in A is face-down except for a card with 1. A means the deck with n card sleeves. p means a list p that records a card such that a card is turned up at the top of the deck. c means the current steps starting from the game.

We describe the behaviors of Knuth's algorithm using the case $n = 4$ illustrated in Fig. 2.1. Line 3 in Algorithm 4 shows NEXTNODE($\{ 2, 3, 4 \}$, $(-1, -2, -3, -4)$, $()$, 0) where $()$ denotes an empty list. Next, we see the NEXTNODE in Algorithm 3. **for all** $k \in N$ **do** corresponds to the texts of the arrow going down from text box $(-1, -2, -3, -4)$ in Fig. 2.1. In this example, we assign 2, 3, and 4 to k in order. We consider the case $k = 2$.

if $A_1 = -k$ **then** checks whether an initial deck of A is derangement. In order to check whether an initial deck for A is derangement, it is sufficient to verify whether a card with k is not in natural position. If a card with k is in natural position, then we go to Line 3 in Algorithm 3 and take the next element from N. In this case of $k = 2$, Lines 3 and 4 are ignored. Line 5 to 11 corresponds to the procedure expressing the second pruning. Note that Line 6 to 9 looks ahead to one step of A.

In Line 13, we add $k(= 2)$ at the end of p. Note that $|p|$ denotes the length of p. Since $p = (2)$, we have $|p| = 1$. This line corresponds to the transition from the above text box to the below text box via the down arrow labeled with k. Line 15 to 22 shows that if $|p| = n - 1$, then we output the larger initial deck, otherwise we call NEXTNODE. The current situation is that $N = \{ 2, 3, 4 \}$, $A' = (-2, 2, -3, -4)$, $p = (2)$, and $c = 1$. In this situation, since $|p| \neq 3$, we call NEXTNODE($N \setminus \{ 2 \}$, A', (2), 1). Line 23 removes the last element of p. After it goes through **for all** $k \in N$ **do**, it backtracks.

2.3 Experiments and Results

Since the search tree of Knuth's "better" algorithm is well-balanced, it is easy to be parallelized. First, we generate the search tree for the first few levels, which corresponds to the first few elements of the list p explained in the last section. Then, distribute the leaves of the tree to many threads and resume the generation in parallel by letting a given leaf as a root of a subtree.

For $n = 18$, we truncate the tree at level two and divide it into 240 subtrees. For $n = 19$, we truncate the tree at level three and divide it into $3,952$ subtrees. Each of these numbers is slightly smaller than the one in the original search tree, i.e., $272(= 17 \times 16)$ or $4,896(= 18 \times 17 \times 16)$, because of the pruning.

In our experiments, we use up to 172 threads in parallel spreading out over nine standard PCs. The computation takes about 7 hours for $n = 18$ (using 132 threads), and about 6 days for $n = 19$ (using 172 threads). This means that, if we run the code on a single thread, then the computation would take approximately 10^3 days for $n = 19$. The total numbers of traversed nodes are $43,235,268,208,065$ for $n = 18$ and $933,351,108,741,643$ for $n = 19$, respectively. The ratios to the number of nodes in the search tree without pruning, i.e., $\sum_{i=0}^{n-1} \prod_{j=1}^{i}(n-j)$, are 4.47% and 5.36%, respectively. The breakdown of the number of traversed nodes for $n = 19$ with respect to the levels of the tree is shown in Table 2.1.

By examining the result, we conclude that $f(18) = 191$ and $f(19) = 221$. The largest deck for $n = 18$ is unique. It is

(6 14 9 2 15 8 1 3 4 12 18 5 10 13 16 17 11 7),

which terminates at the sorted position (1 2 3 . . . 18). There are four largest decks for $n = 19$. These are

(9 4 19 17 10 1 11 15 12 8 5 2 18 13 16 7 3 14 6),
(12 15 11 1 10 17 19 2 5 8 9 4 18 13 16 7 3 14 6),
(12 1 18 11 3 14 2 6 8 16 5 4 15 10 13 17 19 7 9),
(12 1 18 11 2 3 14 6 8 16 5 4 15 10 13 17 19 7 9).

Interestingly, all these decks terminate at a same non-sorted position (1 10 9 8 7 6 5 4 3 2 11 12 13 14 15 16 17 18 19). The largest deck that terminates at the sorted position is known to take 207 steps (see A000376 of OEIS [15]), which is fourteen less than the value of $f(19)$.

Tab. 2.1: The number of traversed nodes for $n = 19$.

Level	# of traversed nodes	Level	# of traversed nodes
0	1	10	46335514956
1	17	11	304773283939
2	272	12	1716889839183
3	3952	13	8059154346527
4	52861	14	30428256670076
5	653126	15	89242470628183
6	7419100	16	200111553921243
7	77075852	17	326581145735086
8	726678384	18	276853558861087
9	6158057798	---	---
10	46335514956	Total	933351108741643

2.4 Concluding Remarks

In this chapter, we investigated Topswops problem and concluded that $f(18) = 191$ and $f(19) = 221$, $g(18) = 1$ and $g(19) = 4$ by applying an algorithm developed by Knuth [33] in a parallel fashion.

Using our method, we tried to obtain the exact value of $f(n)$ for $n \geq 20$. For $n = 20$, we truncate the tree at level four and divide it into exactly $68,310$ subtrees. The traversal for all nodes in the first subtree takes about $23,000$ seconds. Thus, by using one thread, the computation for $n = 20$ may take about $18,000$ days. If we can prepare many threads, we have the exact value of $f(20)$ by using our method, e.g., the computation takes about three months with 200 threads. Even if we use our method, it seems to be hard to obtain the values for $n \geq 21$.

There is an exponential gap between the upper and lower bounds on $f(n)$. As mentioned in Morales [31], we wonder whether there exists an initial deck with $\Omega(n^3)$ steps. To improve the upper and lower bounds would be an interesting future work.

Chapter 3

Anti-slide Packing for a Two-dimensional Box

3.1 Introduction

In this chapter, we study sparsest anti-slide packings for a two-dimensional square box. Recall that given a box of some specified size and identical pieces of some specified shape, the problem is to find a sparsest anti-slide packing for the box. We consider the *integral* and *orthogonal* version of the problem where a box is a two-dimensional square. We call an anti-slide packing *stable*. See the meaning of these words such as integral, orthogonal, and stable in Section 3.2.

Amano, Nakano, and Yamazaki [34] gave an integer linear programming (IP) formulation of the problem for finding a sparsest stable packing and obtained upper and lower bounds on the density of such a packing of $2 \times 2 \times 1$ pieces for a three-dimensional box. It is also natural to consider such packings for various shapes of the pieces. In our previous work [26], we analyzed stable packings for several shapes of *polycubes*. See the groups for several shapes of polycubes in Chapter 4.

In a related work, Takenaga, Xi, and Inada [35] considered a two-dimensional case using pentomino pieces. Namely, they enumerated anti-slide packings of pentomino pieces for a two-dimensional box of small size using ordered binary decision diagrams.

In this chapter, we focus on the problem of packing T-tetrominoes (Fig. 3.1) into a two-dimensional square box.

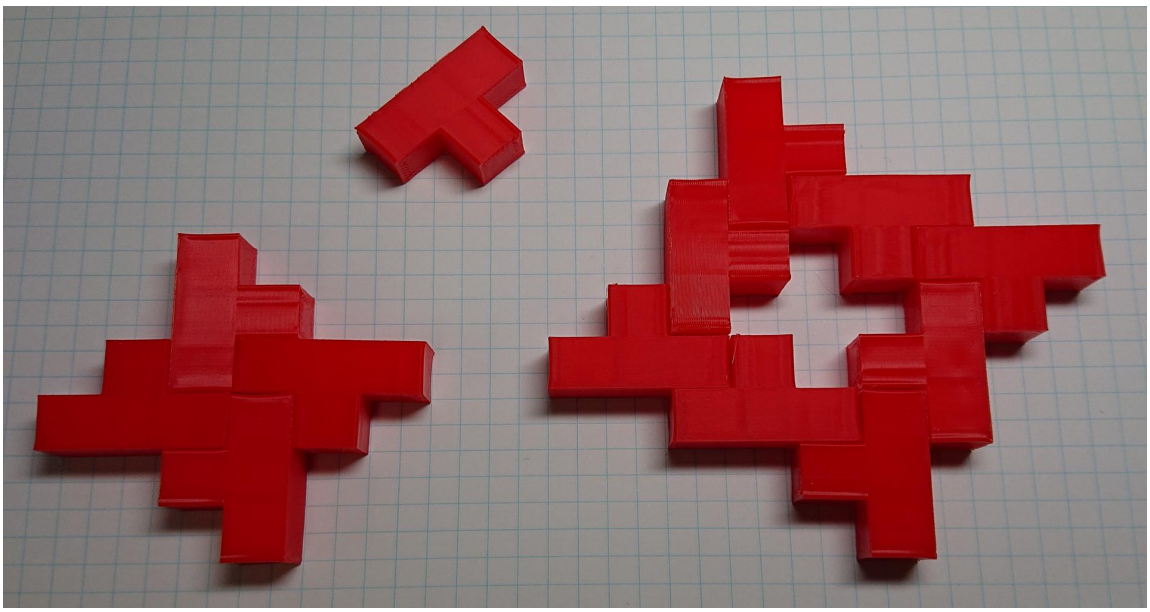


Fig. 3.1: A sparsest stable packing of T-tetrominoes: (left) for a 6×6 box. (right) for a 9×9 box.

The L-toromino is the simplest piece among the pieces. However, it seems that a sparsest packing for an $n \times n$ box is obtained by stacking $n-1$ L-torominoes diagonally, which is not so interesting. The T-tetromino is the second simplest. We will see in the following that the problem turns out to be highly non-trivial even for this simple shape.

First, we conducted computer searches based on the IP formulation developed in [34] to find sparsest stable packings of T-tetromino pieces for an $n \times n$ box for $n \leq 15$. Interestingly, the results suggest that the structure of a sparsest packing would be quite different depending on whether the side length of a box is divisible by three or not (see Section 3.2).

Let us see this more precisely. Let $F(n)$ be the minimum number of T-tetromino pieces of a stable packing for an $n \times n$ box. Our experimental results suggest that $F(n) \sim 2n/3$ for $n \not\equiv 0 \pmod 3$, and $F(n) \sim n$ for $n \equiv 0 \pmod 3$.

In this chapter, we determine the value of $F(n)$ exactly for the case of

$n \not\equiv 0 \pmod 3$ by proving matching upper and lower bounds of $F(n) = \lfloor 2n/3 \rfloor$. For the case of $n \equiv 0 \pmod 3$, we have succeeded only in proving the bound of $2n/3 - 1 \leq F(n) \leq n - 1$, although we believe that the upper bound is tight for $n \geq 9$. Note that when n is 6, the smallest number of pieces in a stable packing is 4 (Fig. 3.1).

The organization of this chapter is as follows. In Section 3.2, we define the terms and show our experimental results. In Section 3.3, we show the upper bounds on $F(n)$ by giving an explicit construction of stable packings of T-tetromino pieces for an $n \times n$ box. In Section 3.4, we show the lower bounds on $F(n)$. Finally, in Section 3.5, we discuss the reasons why $F(n)$ behaves differently depending on whether n is divisible by three or not.

3.2 Definitions and Experimental Results

For a positive integer α, $[\alpha]$ denotes a set $\{ 1, 2, \ldots, \alpha \}$.

We describe *integral* and *orthogonal*. Integral means that we assume all the coordinates of corner points of pieces are integers. By using the integral, we view the $n \times n$ box as a two-dimensional array of cells of unit size. Each cell is identified by $(i, j) \in [n] \times [n]$. Orthogonal means that we assume that each piece is axis-aligned and each piece slides to the orthogonal direction, i.e., parallel to x- or y-axes.

For an $n \times n$ box and a number of T-tetromino pieces, we say that a situation is a *packing* if each piece goes in the box and is restricted by the integral and the orthogonal conditions. We say that a packing is a *stable packing* if none of pieces in a packing can slide in any direction.

A T-tetromino is a polyomino shaped like the letter T consisting of four unit squares. We say that a piece is placed at (i, j) if the central square of the piece is adjusted to (i, j). Also, we say that a piece is placed with direction $d \in \mathcal{R}$, where $\mathcal{R} = \{ U, R, D, L \}$, each of which is shown in Fig. 3.2. For example, if a T-tetromino piece is placed at $(2, 2)$ with direction U, then the cells $(2, 2)$, $(3, 2)$, $(1, 2)$, and $(2, 3)$ are occupied by this piece.

By using the IP formulation in Amano et al. [34], the problem can be represented as an IP problem. We solved this problem by using an IP solver [36] for $n \leq 15$. The results are shown in Tab. 3.1, and examples of

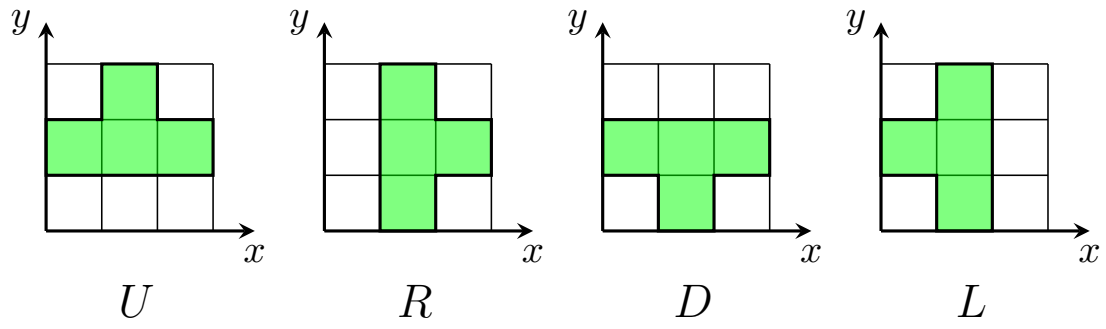


Fig. 3.2: The placement of a piece with direction U, R, D, and L.

packings that attain $F(n)$ for $n = 10, 11, \ldots, 15$ are shown in Fig. 3.3.

Tab. 3.1: Values of $F(n)$ for $n \leq 15$.

n	4	5	6	7	8	9	10	11	12	13	14	15
$F(n)$	2	3	4	4	5	8	6	7	11	8	9	14

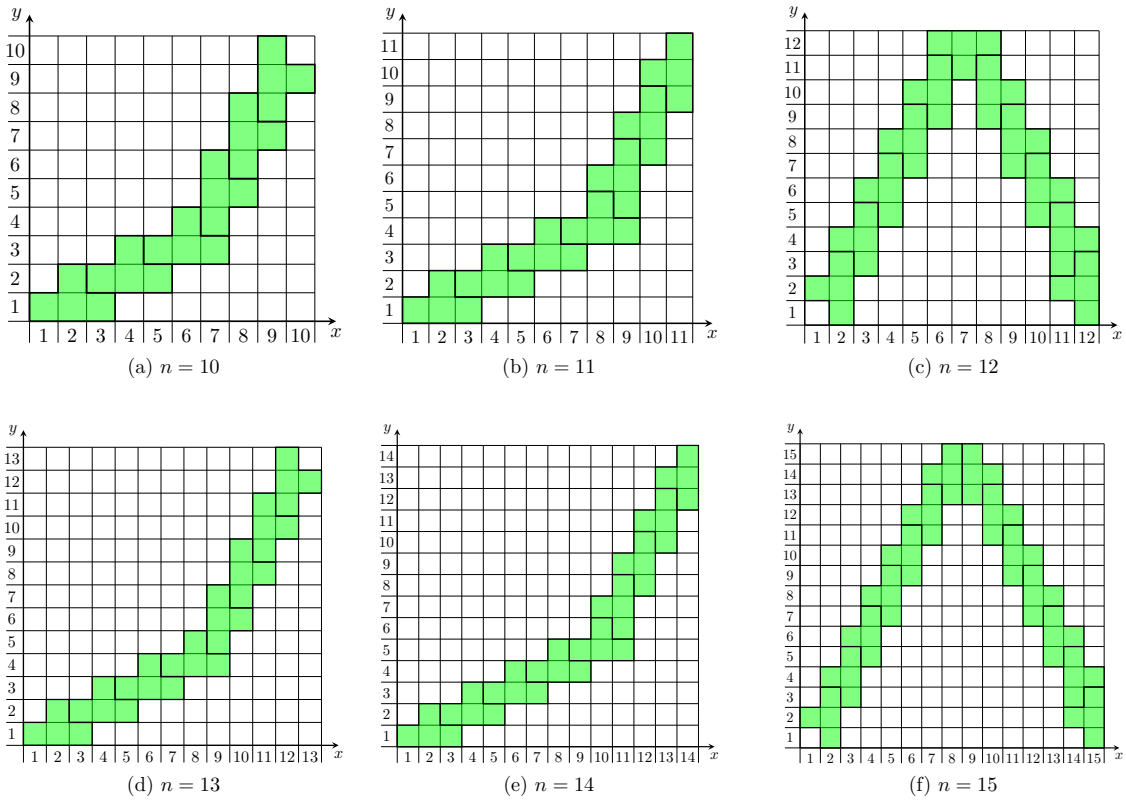
3.3 Upper Bound

From Fig. 3.3, we can observe that a sparsest packing for an $n \times n$ box looks like a diagonal line for $n \not\equiv 0 \pmod 3$ and a V-shape for $n \equiv 0 \pmod 3$. We believe that this holds also for higher values of n.

An easy calculation shows that the number of pieces used in these packings is $\sim \frac{2}{3}n$ for the former, and $\sim n$ for the latter. In this section, we show the upper bounds on $F(n)$ matching this by giving an explicit construction of stable packings.

Theorem 2. *For every $n \geq 4$ such that $n \equiv 1 \pmod 3$, we have*

$$F(n) \leq \frac{2(n-1)}{3},$$



(a) $n = 10$ (b) $n = 11$ (c) $n = 12$

(d) $n = 13$ (e) $n = 14$ (f) $n = 15$

Fig. 3.3: The sparsest stable packings for the $n \times n$ box for $10 \leq n \leq 15$.

and for every $n \geq 5$ *such that* $n \equiv 2 \pmod 3$, *we have*

$$F(n) \leq \frac{2n-1}{3}.$$

Proof. First we consider the case $n \equiv 1 \pmod 3$.

The proof is by induction. The base case, $F(4) = 2$, is obvious. Place two T-tetromino pieces as in the center part of Fig. 3.3(a). The induction step is also obvious by comparing Fig. 3.3(a) and (d). Precisely, we can obtain a specified stable packing for the $(n+3) \times (n+3)$ box by adding two pieces to a stable packing for the $n \times n$ box; add one piece with R-direction to the top right corner and the other piece with U-direction to the bottom left corner. This completes the proof for the case $n \equiv 1 \pmod 3$.

The proof for the case $n \equiv 2 \pmod 3$ is analogous, referring to Fig. 3.3(b) and (e). □

Theorem 3. *For every* $n \geq 6$ *such that* $n \equiv 0 \pmod 3$*, we have* $F(n) \leq n - 1$.

Proof. The proof is similar to the proof of Theorem 2.

We use the packing of Fig. 3.3(c) as the base for the case $n \equiv 0 \pmod 6$, and the packing of Fig. 3.3(f) as the base for the case $n \equiv 3 \pmod 6$.

Then, we can obtain a stable packing for the $(n + 6) \times (n + 6)$ box from a packing for the $n \times n$ box by adding six pieces: insert three pieces with L-direction in the *middle* of the left slope and insert three pieces with R-direction in the *middle* of the right slope. Similarly, we can obtain a stable packing for the 6×6 box consisting of five pieces from a packing for the 12×12 (Fig. 3.3(c)) box by removing six pieces: remove three pieces with L-direction on the left slope and three pieces with R-direction on the right slope. A packing for the 9×9 box consisting of eight pieces can be obtained from the packing for the 15×15 box (Fig. 3.3(f)) in a similar way. This suffices to prove the theorem. □

3.4 Lower Bound

In this section, we give the lower bound on $F(n)$, proving that the upper bound shown in Theorem 2 is exactly tight.

Theorem 4. *For every* $n \geq 4$*, we have* $F(n) \geq \frac{2}{3}n - 1$.

Since $F(n)$ takes only an integer, the upper bounds in Theorem 2 and the lower bound in Theorem 4 are shown to be identical.

Corollary 5. *For every* $n \geq 4$ *such that* $n \not\equiv 0 \pmod 3$*, we have* $F(n) = \left\lfloor \frac{2n}{3} \right\rfloor$. □

The rest of this section is devoted to the proof of Theorem 4. Roughly speaking, we will show that, given any stable packing, we can always pick a set of pieces of size $\geq 2n/3 - 1$ from the packing by applying an appropriate procedure.

3.4.1 Preparation

Consider a stable packing P_n for an $n \times n$ box. Let W_l denote the left vertical boundary of the box. Similarly, let W_r, W_d, and W_u denote the right vertical, the bottom horizontal, and the top horizontal boundary of the box. We introduce an algorithm that defines a "backbone" set $\mathcal{P}$ of pieces for a given packing P_n. See Algorithm 5. In what follows, we

Algorithm 5 Select a set of pieces from any stable packing

```
1:  procedure SELECT(P_n)
2:      Pick an arbitrary piece in P_n that touches W_l.
3:      Designate the picked piece as p_1.
4:      m := 1
5:      while p_m does not touch W_r do
6:          Pick an arbitrary piece that touches p_m from the right.
7:          m := m + 1
8:          Designate the picked piece as p_m.
9:      P_{l→r} := { p_1, ..., p_m }.
10:     Pick an arbitrary piece that touches W_d.
11:     Designate this picked piece as p'.
12:     if p' ∉ P_{l→r} then
13:         m := m + 1
14:         p_m := p'
15:     p'' := p'
16:     while p'' does not touch W_u do
17:         Pick an arbitrary piece that touches p'' from above.
18:         Designate this picked piece as p'.
19:         if p' ∉ P_{l→r} then
20:             m := m + 1
21:             p_m := p'
22:         p'' := p'
23:     P := { p_1, ..., p_m }
24:     output P and P_{l→r}.
```

verify that Algorithm 5 surely halts and outputs a set $\mathcal{P}$ for any stable

packing. Note that every piece in a stable packing must touch another piece or a boundary of the box in order to avoid sliding. For $m' = 1, 2, \ldots$, let $x(m')$ denote the maximum value of the x-coordinate of a cell occupied by $p_{m'}$ designated in line 8 (or in line 3 for $m' = 1$). Because every piece with the shape of T-tetromino is touched from right, $x(m')$ has the following properties: first, $x(m')$ is a monotonically non-decreasing function; second, if $x(m'+1) - x(m') = 0$, then $x(m'+2) - x(m'+1) \geq 1$. These two properties imply that $x(m) = n$ for some m with $m \leq 2n$. This guarantees that the first **while** loop in Algorithm 5 will eventually be terminated. $P_{l \to r}$ (in line 10) is a sequence of picked pieces in line between W_l and W_r. A similar argument also shows the termination of the second **while** loop in the algorithm. This suffices to show that Algorithm 5 halts for any stable packing. We should note that Algorithm 5 surely outputs a set $\mathcal{P}$ since every piece in a stable packing must touch another piece or a boundary of the box in order to avoid sliding.

An example of a stable packing and the corresponding output of Algorithm 5 is illustrated in Fig. 3.4. In this example, the algorithm outputs the set $\mathcal{P}$ of nine pieces labeled 1 to 9 and the set $P_{l \to r}$ is consisting of five pieces labeled 1 to 5 in Fig. 3.4.

The following definition is the key to our analysis.

Definition 6. *For a set of pieces $\mathcal{P}$ placed in an $n \times n$ box and for $i \in [n]$, we say that the i-th column (i-th row, respectively) is marked if a cell with the x-coordinate i (y-coordinate i, respectively) is occupied by some piece in $\mathcal{P}$.*

It is obvious that P_n with a stable packing marks all columns and rows. Suppose that $\mathcal{P} = \{p_1, \ldots, p_m\}$ is an output of Algorithm 5. It is clear that $\mathcal{P}$ marks all columns and rows. For each $k \leq m$, let $M(k)$ denote the total number of rows and columns that are marked by $\{ p_1, \ldots, p_k \}$.

For example, the piece labeled 1 in Fig. 3.4(b) marks the fifth, sixth, and seventh rows and the first and second columns, and thus $M(1) = 5$.

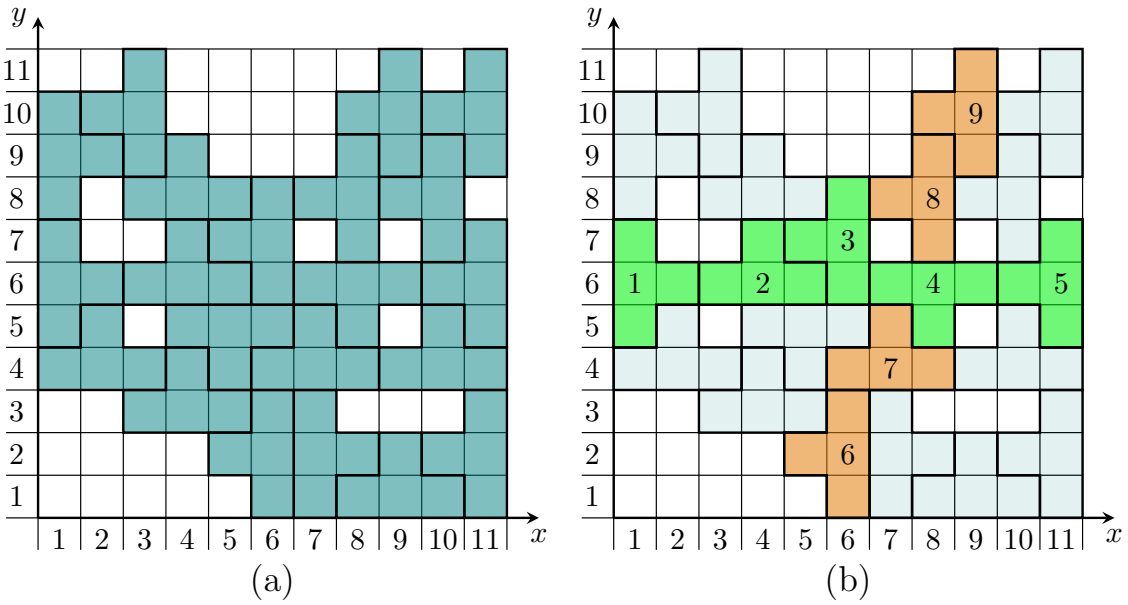


Fig. 3.4: (a): A stable packing for an 11×11 box. (b): The output of Algorithm 5 for the packing in (a) is the pieces labeled 1 to 9.

We define Q_4, Q_3, and $Q_{\leq 2}$ as follows:

$$\begin{aligned} Q_4 &:= \{ p_k \,|\, k \in [|\mathcal{P}|] \setminus \{1\}, M(k) - M(k-1) = 4 \}, \\ Q_3 &:= \{ p_k \,|\, k \in [|\mathcal{P}|] \setminus \{1\}, M(k) - M(k-1) = 3 \}, \\ Q_{\leq 2} &:= \{ p_k \,|\, k \in [|\mathcal{P}|] \setminus \{1\}, M(k) - M(k-1) \leq 2 \}. \end{aligned}$$

By the definition, it is obvious that $M(m) = 2n$ and that

$$\mathcal{P} = Q_4 \cup Q_3 \cup Q_{\leq 2} \cup \{ p_1 \}.$$

3.4.2 Analysis of $P_{l \to r}$ of $\mathcal{P}$

Given a set $P_{l \to r}$ of pieces and $k \in [|P_{l \to r}|]$, let $G_{\max,y}(k)$ and $G_{\min,y}(k)$ be the maximum and minimum values of the y-coordinate marked by the first k-th pieces of $P_{l \to r}$. Similarly, let $G_{\max,x}(k)$ be the maximum value of the x-coordinate marked by the first k-th pieces of $P_{l \to r}$.

For $k \in [|P_{l \to r}|]$, let $H_{\max,y}(k)$ and $H_{\min,y}(k)$ be the maximum and minimum values of the y-coordinate of a cell occupied by the k-th piece in

$P_{l\to r}$. Similarly, let $H_{\min,x}(k)$ be the minimum value of the x-coordinate of a cell occupied by the k-th piece in $P_{l\to r}$.

In addition, we define $X(k)$, $Y_{\max}(k)$ and $Y_{\min}(k)$ as follows:

$$\begin{aligned} X(k) &= G_{\max,x}(k) - G_{\max,x}(k-1), \\ Y_{\max}(k) &= G_{\max,y}(k) - G_{\max,y}(k-1), \\ Y_{\min}(k) &= G_{\min,y}(k-1) - G_{\min,y}(k), \end{aligned}$$

where $X(k)$, $Y_{\max}(k)$, and $Y_{\min}(k)$ are undefined when $k = 1$. According to the definitions of these functions and Algorithm 5, we have the following:

$$\begin{aligned} &0 \le X(k) \le 3, \\ &0 \le Y_{\max}(k) \le 2, \\ &0 \le Y_{\min}(k) \le 2, \\ &Y_{\max}(k) \cdot Y_{\min}(k) = 0. \end{aligned} \tag{3.1}$$

We call the cell $(G_{\max,x}(k), G_{\max,y}(k))$ the k-th *top corner*. Also, we call the cell $(G_{\max,x}(k), G_{\min,y}(k))$ the k-th *bottom corner*. We refer to a top corner or a bottom corner as a *corner*. We say that the k-th top corner (or similarly, k-th bottom corner) is *filled* if it is occupied by the k-th piece in $P_{l\to r}$.

We introduce two types of 0/1 variables:

$$\left\{ s_{\max}[k] \mid k \in [|P_{l\to r}|] \right\},$$

and

$$\left\{ s_{\min}[k] \mid k \in [|P_{l\to r}|] \right\}.$$

The variable $s_{\max}[k]$ takes the value 1 if and only if the k-th top corner is filled. Similarly, the variable $s_{\min}[k]$ takes the value 1 if and only if the k-th bottom corner is filled.

The following lemma says that every piece in Q_4 must be picked in the first while loop in Algorithm 5.

Lemma 7. $Q_4 \subset P_{l\to r}$.

Proof. The proof is immediate by seeing that every piece picked in the second while loop (in lines 18–26) in Algorithm 5 can mark only rows since all columns have been marked during the first while loop (in lines 5–9). □

The following lemma says that $p_k \in Q_4$ fills neither the k-th top nor bottom corner. In addition, $p_k \in Q_4$ touches p_{k-1} that fills the $(k-1)$-th top or bottom corner.

Lemma 8. *For every integer k such that $2 \leq k \leq |P_{l\to r}|$, if $p_k \in Q_4$, then $s_{\max}[k] = 0$, $s_{\min}[k] = 0$, and either $s_{\max}[k-1] = 1$ or $s_{\min}[k-1] = 1$.*

Proof. Suppose that $p_k \in Q_4$. This means that

$$X(k) + Y_{\max}(k) + Y_{\min}(k) = 4.$$

By Eq. (3.1), there are four possibilities for the values of $X(k), Y_{\max}(k)$, and $Y_{\min}(k)$.

$$\text{Case I: } \begin{cases} X(k) = 3 \\ Y_{\max}(k) = 1 \\ Y_{\min}(k) = 0 \end{cases} \quad \text{Case II: } \begin{cases} X(k) = 3 \\ Y_{\max}(k) = 0 \\ Y_{\min}(k) = 1 \end{cases}$$

$$\text{Case III: } \begin{cases} X(k) = 2 \\ Y_{\max}(k) = 2 \\ Y_{\min}(k) = 0 \end{cases} \quad \text{Case IV: } \begin{cases} X(k) = 2 \\ Y_{\max}(k) = 0 \\ Y_{\min}(k) = 2 \end{cases}$$

Case I. The situation of this case is illustrated in Fig. 3.5. Since $X(k) = 3$ and $Y_{\max}(k) = 1$, the piece p_k is with U-direction and the piece p_{k-1} fills the $(k-1)$-th top corner. This implies that $s_{\max}[k] = 0$ and $s_{\max}[k-1] = 1$.

It remains to show that $s_{\min}[k] = 0$. To this purpose, it is sufficient to show that $H_{\min,y}(k) - G_{\min,y}(k) \geq 1$. In this case, we have

$$H_{\min,y}(k) = G_{\max,y}(k-1).$$

This can be verified as follows.

$$\begin{aligned} &H_{\min,y}(k) - G_{\min,y}(k) \\ &\geq G_{\max,y}(k-1) - G_{\min,y}(k) \\ &= G_{\max,y}(k-1) - G_{\min,y}(k-1) \geq 1. \end{aligned}$$

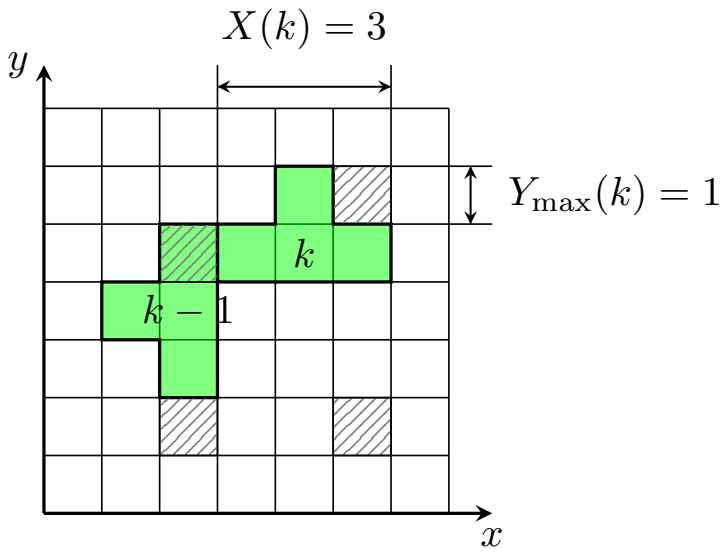


Fig. 3.5: A piece with $p_k \in Q_4$, $X(k) = 3$, and $Y_{\max}(k) = 1$ does not fill the k-th corner shown in the striped pattern and touches p_{k-1} that fills the $(k-1)$-th top corner. This corresponds to Case I in the proof of Lemma 8.

Case II. This case is "upside down" to Case I, and can be verified analogously.
Cases III and IV. In these cases, we can easily verify that the piece p_k is with R-direction. The rest of the proof is similar to the proof for Case I. □

The next lemma says that if $p_k \in Q_3$ and the $(k-1)$-th top and bottom corners are not filled, then the k-th top and bottom corners are not filled as well.

Lemma 9. *For every integer k such that $2 \leq k \leq |P_{l\to r}|$, if $p_k \in Q_3$, $s_{\max}[k-1] = 0$, and $s_{\min}[k-1] = 0$, then $s_{\max}[k] = 0$ and $s_{\min}[k] = 0$.*

Proof. Suppose that $p_k \in Q_3$, which means that

$$X(k) + Y_{\max}(k) + Y_{\min}(k) = 3.$$

Suppose also that $s_{\max}[k-1] = 0$ and $s_{\min}[k-1] = 0$. Then, by Eq. (3.1),

the values of $X(k)$, $Y_{\max}(k)$, and $Y_{\min}(k)$ are one of the following five cases.

$$\text{Case I: } \begin{cases} X(k) = 3 \\ Y_{\max}(k) = 0 \\ Y_{\min}(k) = 0 \end{cases}$$

$$\text{Case II: } \begin{cases} X(k) = 2 \\ Y_{\max}(k) = 1 \\ Y_{\min}(k) = 0 \end{cases} \quad \text{Case III: } \begin{cases} X(k) = 2 \\ Y_{\max}(k) = 0 \\ Y_{\min}(k) = 1 \end{cases}$$

$$\text{Case IV: } \begin{cases} X(k) = 1 \\ Y_{\max}(k) = 2 \\ Y_{\min}(k) = 0 \end{cases} \quad \text{Case V: } \begin{cases} X(k) = 1 \\ Y_{\max}(k) = 0 \\ Y_{\min}(k) = 2 \end{cases}$$

Case I. Since $X(k) = 3$, the direction of the piece p_k is U or D. If p_k is U-direction, then $s_{\max}[k] = 0$ is obviously true. Then, we have

$$H_{\min,y}(k) - G_{\min,y}(k) \geq 1.$$

This implies that $s_{\min}[k] = 0$.

If p_k is D-direction, then this is "upside down" to the situation where p_k is U-direction, and hence can be shown analogously. These two cases imply that p_k fills neither the k-th top corner nor bottom corner, i.e., $s_{\max}[k] = 0$ and $s_{\min}[k] = 0$ as desired.

Case II. The situation of Case II is illustrated in Fig. 3.6. According to the assignment of Case II, the direction of the piece p_k is U or R. If p_k is R-direction, then $s_{\max}[k] = 0$ and $s_{\min}[k] = 0$ are obviously true.

We now suppose that p_k is U-direction. This case guarantees that $s_{\max}[k] = 0$. It remains to show that $s_{\min}[k] = 0$. This is the same proof that $H_{\min,y}(k) - G_{\min,y}(k) \geq 1$ of Case I of Lemma 8, which completes the proof.

Case III. This case is "upside down" to Case II, and hence can be shown analogously.

Case IV. According to the assignment of Case IV, the direction of the piece p_k is R. Then $s_{\max}[k] = 0$ and $s_{\min}[k] = 0$ are obviously true.

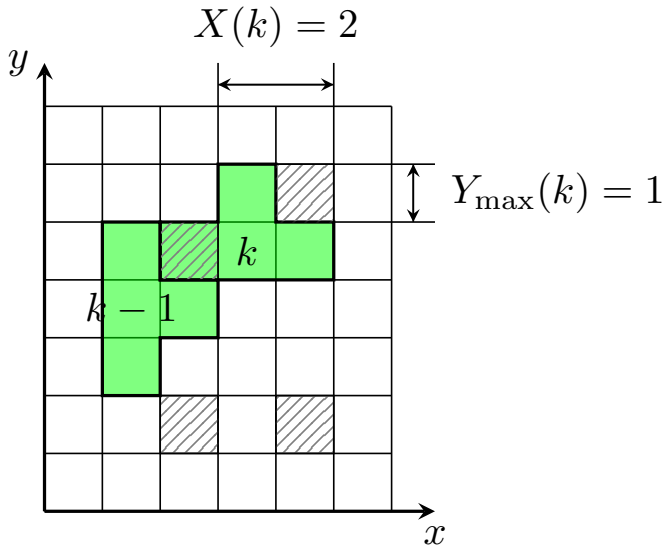


Fig. 3.6: A piece $p_k \in Q_3$ such that $X(k) = 2$ and $Y_{\max}(k) = 1$ does not fill the k-th corners shown in the striped pattern and touches p_{k-1} that does not fill the $(k-1)$-th top corner. This corresponds to Case II in the proof of Lemma 9.

Case V. This case is "upside down" to Case IV, and hence can be shown analogously.

This completes the proof of the lemma. □

Here, we write the indices of elements in Q_4 as $\ell_1, \ldots, \ell_{|Q_4|}$ in ascending order. The key lemma states that if p_{ℓ_i} and $p_{\ell_{i+1}}$ exist, there is at least one piece $p_k \in Q_{\leq 2}$ for filling a $(\ell_{i+1} - 1)$-th corner.

Lemma 10. *For every integer i such that $1 \leq i \leq |Q_4| - 1$, there is an integer k such that $\ell_i < k < \ell_{i+1}$ and $p_k \in Q_{\leq 2}$.*

Proof. The proof is by contradiction. Suppose that there exists an integer i such that $1 \leq i \leq |Q_4| - 1$, and $p_k \in Q_3$ holds for every k such that $\ell_i < k < \ell_{i+1}$.

Lemma 8 immediately yields $s_{\max}[\ell_i] = 0$ and $s_{\min}[\ell_i] = 0$. By applying Lemma 9 successively for $k = k_i + 1, k_i + 2, \ldots, \ell_{i+1} - 1$, we conclude that $s_{\max}[k_{i+1} - 1] = 0$ and $s_{\min}[k_{i+1} - 1] = 0$. However, Lemma 8 also yields

$s_{\max}[\ell_{i+1}-1]=1$ or $s_{\min}[\ell_{i+1}-1]=1$ since $p_{\ell_{i+1}} \in Q_4$, which gives the desired contradiction. □

3.4.3 Proof of the Main Theorem

Lemma 10 immediately yields the following:

Lemma 11. $|Q_4| - |Q_{\leq 2}| \leq 1$. □

Intuitively, Lemma 11 says that each piece in an output $\mathcal{P}$ of Algorithm 5 marks at most (about) three rows or columns on average. By combining this with the fact that the total number of marked rows or columns is $2n$, we obtain a desired lower bound of (roughly) $2n/3$.

Let us see the formal proof of the main theorem.

Proof of Theorem 4. Fix an integer $n \geq 4$ and a stable packing P_n for an $n \times n$ box. We obtain $\mathcal{P}$ by applying Algorithm 5 for P_n, and then define Q_4, Q_3, and $Q_{\leq 2}$.

By the definition, we have $|\mathcal{P}| = |Q_4| + |Q_3| + |Q_{\leq 2}| + 1$. We can bound $M(|\mathcal{P}|)$ from above by

$$\begin{aligned} M(|\mathcal{P}|) &\leq 4|Q_4| + 3|Q_3| + 2|Q_{\leq 2}| + 5 \cdot 1 \\ &= 3(|Q_4| + |Q_3| + |Q_{\leq 2}| + 1) \\ &\qquad + (|Q_4| - |Q_{\leq 2}|) + 2 \\ &= 3|\mathcal{P}| + (2 + |Q_4| - |Q_{\leq 2}|). \end{aligned}$$

By recalling that $M(|\mathcal{P}|) = 2n$, this is equivalent to

$$\frac{2}{3}n - \frac{2 + |Q_4| - |Q_{\leq 2}|}{3} \leq |\mathcal{P}|.$$

By applying Lemma 11, we have

$$\frac{2}{3}n - 1 \leq |\mathcal{P}| \leq F(n),$$

which completes the proof of the theorem. □

3.5 Concluding Remarks

In this chapter, we investigated the anti-slide packings for a square box of side length n using T-tetrominoes and proved that the number of pieces in a sparsest packing solution $F(n) = \lfloor 2n/3 \rfloor$ when $n \not\equiv 0 \pmod 3$ and $2n/3-1 \leq F(n) \leq n-1$ when $n \equiv 0 \pmod 3$. There is still a substantial gap between the upper and lower bounds for the case that the side length is divisible by 3. We believe that the upper bound is in fact tight, although we have not succeeded in proving this.

One may wonder whether a diagonal packing would be possible even for the case $n \equiv 0 \pmod 3$. However, a simple observation of the packings shown in Fig. 3.3 reveals that this is not the case. We can observe that, for every $k = 1, 2, \ldots$, the sum of the largest x-coordinate and the largest y-coordinate of a cell occupied by the k-th piece, counting from the bottom left corner, in a diagonal packing is not divisible by 3. We conjecture that an extra $n/3$ pieces are needed to prevent all of these pieces from sliding. To show this formally would be an interesting future work.

Chapter 4

Anti-slide Packing for a Three-dimensional Box

4.1 Introduction

In this chapter, we study sparsest anti-slide packings for a three-dimensional cubic box.

There are several possible definitions [35, 37] of the anti-slide problem. We consider the *integral*, *orthogonal*, and *snagged* version of the problem: First, integral means that each piece is aligned so that every corner of the piece matches the coordinate of integers. Second, orthogonal means that each piece slides toward an orthogonal direction, i.e., parallel to x, y, or z-axes. Third, snagged means that each piece will not slide toward a direction if the piece touches the other piece or the inner wall of a finite box on the straight line of the direction.

We say that a piece is *fixed* if the piece will not slide toward any orthogonal directions. We say that a packing of pieces is *stable* if every piece in the packing is fixed. In our model, an anti-slide packing is called a stable packing. The *density* of a packing is defined to be the ratio of the volume occupied by pieces.

In 1994 and 1996, Knuth [38] made a program to find all stable packings for a small box implementing the data structure *dancing links* [39]. In 2007, Knuth [40] also gave a construction of a stable packing of $2 \times 2 \times 1$ pieces

whose volume density is $11/16 = 0.6875$ when the size of a box goes to infinity. Amano et al. [34] used an IP approach to show that the minimum density is between 0.288 and $2/3 \sim 0.666\dot{}$.

In our previous work [26], we investigated the asymptotic behavior of the minimum number of pieces in a stable packing for an $n \times n \times n$ box for various shapes of *polycubes* illustrated in Fig. 4.1 and observed that these could be categorized into three groups. Namely, each polycube in Group A

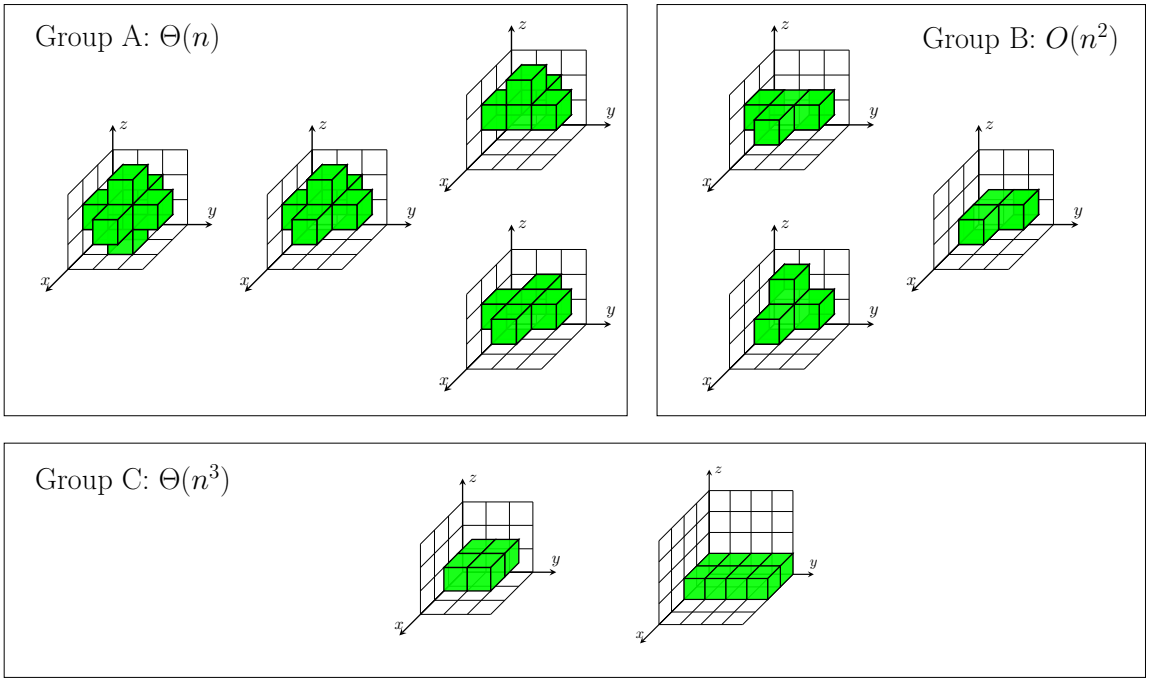


Fig. 4.1: Several shapes of pieces analyzed.

and B has a packing with $\Theta(n)$ and $O(n^2)$ pieces, respectively. Note that a non-zero density lower bound shown for a piece in Group C implies that it needs $\Theta(n^3)$ pieces to pack without sliding. We *believe* that a piece in Group B needs $\Omega(n^2)$ pieces to pack.

In this chapter, we focus on the sparsest anti-slide packing for $2 \times 2 \times 1$ pieces and L-tricube pieces. Firstly, we give a new construction of a stable packing of L-tricubes. We will discuss this in Section 4.2. Secondly, we show that there is a stable packing of $2 \times 2 \times 1$ pieces when a box is torus. We will describe our construction in Section 4.3. Finally, we show that

there is a stable packing of $2 \times 2 \times 1$ pieces without the assumption of the torus. We will describe our construction in Section 4.4.

In related works, an anti-slide packing is well-studied in the enumeration and computational complexity. Takenaga et al. [35] enumerated anti-slide packings of pentomino pieces for a small two-dimensional box using the ordered binary decision diagram. Minamisawa et al. [37] discussed the NP-completeness of the anti-slide puzzle. Note that the model in Minamisawa et al. assumes real-world physical phenomena.

Notations. For positive integers n and m with $n < m$, $[n]$ denotes the set $\{ 1, 2, \ldots, n \}$ and $[n, m]$ denotes $\{ n, n+1, \ldots, m \}$. By using the integral, we view the $l \times m \times n$ box as a three-dimensional array of cells of unit size. Each cell is identified by $(i, j, k) \in [l] \times [m] \times [n]$. In this chapter, we assume without loss of generality that $l \geq m \geq n$.

We say that a $2 \times 2 \times 1$ piece is placed at (i, j, k) if the cell in the piece closest to the origin in the array is aligned to (i, j, k). Also, we say a $2 \times 2 \times 1$ piece is placed with direction $d \in [3]$, each of which is is shown in Fig. 4.2. For example, if we place a $2 \times 2 \times 1$ in a $4 \times 4 \times 4$ box at

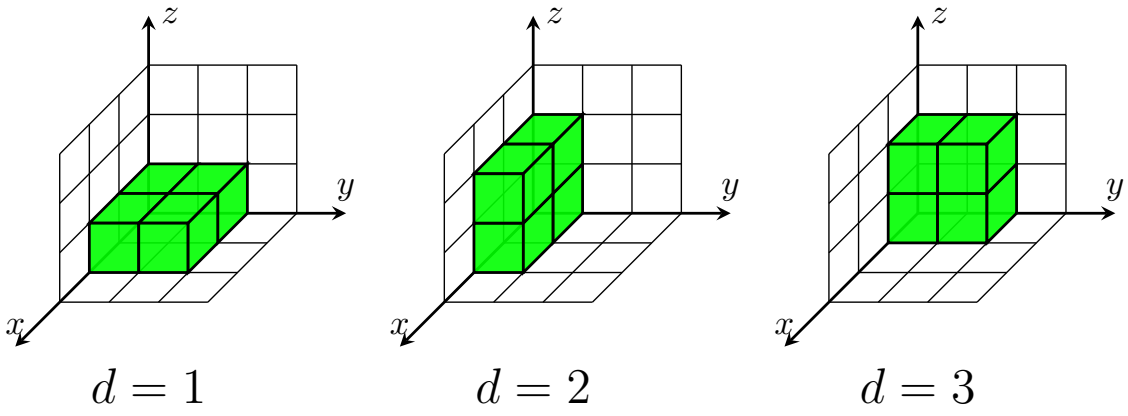


Fig. 4.2: Placements of a piece with direction $d = 1$, 2, and 3.

$(1, 2, 1)$ with the direction $d = 1$ then the cells $(1, 2, 1)$, $(2, 2, 1)$, $(1, 3, 1)$, and $(2, 3, 1)$ are occupied by the piece. The placement and occupancy of an L-tricube is defined in the same way. Note that the number of directions of an L-tricube is 12.

Let us see the IP formulation for the anti-slide packing problem developed by Amano et al. [34]. We can give the IP model for the polycubes in Fig. 4.1. Here, we will describe the IP model for the case of $2 \times 2 \times 1$ piece.

We introduce two types of 0-1 variables

$$\{\, p[i,j,k,d] \mid (i,j,k) \in [l] \times [m] \times [n] \,\}$$

and

$$\{\, o[i,j,k] \mid (i,j,k) \in [l] \times [m] \times [n] \,\}.$$

The variable $p[i,j,k,d]$ takes value 1 if there is a piece placed at (i,j,k) with direction d, and 0 if there is no such piece. The variable $o[i,j,k]$ takes value 1 if the cell (i,j,k) is occupied by some placed piece, and 0 otherwise.

We will give a set of linear constraints so that the solution of the model corresponds to a stable packing. There exists at least one piece placed in the box, which is expressed as

$$\sum_{(i,j,k),d} p[i,j,k,d] \geq 1. \tag{4.1}$$

where the summation ranges over all $(i,j,k) \in [l] \times [m] \times [n]$ and all directions $d \in \{\, 1,2,3 \,\}$.

An occupancy of a cell is expressed as

$$o[i,j,k] = \sum_{(i',j',k'),d} p[i',j',k',d] \tag{4.2}$$

where the summation ranges over all (i',j',k') and d such that the cell (i,j,k) is occupied if a piece is placed at (i',j',k') with direction d.

A piece occupies the cells along the shape of the piece. Thus, a placement of a piece is written as

$$p[i,j,k,d] \leq o[i',j',k'] \tag{4.3}$$

for each cell (i',j',k') occupied by the piece.

Finally, we will express the constraint that every piece of a stable packing is fixed. Consider that a piece is placed at (i,j,k) with direction d. If the

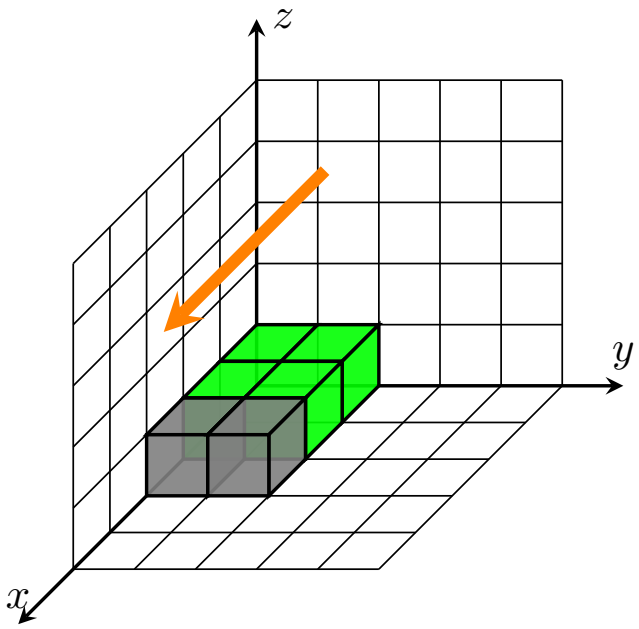


Fig. 4.3: Either one of the cells, colored gray cubes, is occupied if a placed piece will not slide toward the positive x-axis.

piece will not slide toward the positive x-axis, then either one of the cells is occupied. This situation is illustrated in Fig. 4.3. This situation is expressed as

$$p[i,j,k,d] \leq \sum_{(i',j',k')} o[i',j',k']. \tag{4.4}$$

where (i',j',k') is a cell needed for the piece to be fixed. Similarly, we add the constraint for the situation for positive y, z, negative x, y, and z -axes.

We easily understand that a feasible solution satisfying all the conditions described in Eqs. (4.1) to (4.4) is corresponding to a stable packing. The objective function is to minimize the number of pieces in the box. This is expressed by

$$\sum_{(i,j,k),d} p[i,j,k,d] \tag{4.5}$$

where the summation ranges over all $(i,j,k) \in [l] \times [m] \times [n]$ and all directions $d \in \{ 1,2,3 \}$.

4.2 Minimum Number of a Stable Packing for L-tricubes

In this section, we focus on the L-tricube, which would be the easiest shape to get a quadratic lower bound. We analyze stable packings of L-tricubes in terms of projection. We also obtain a new construction of a stable packing of L-tricubes.

Let $L(n)$ be the minimum number of L-tricube pieces of a stable packing for an $n \times n \times n$ box. We conducted computer searches based on the IP formulation in Section 4.1 to find a better upper bound on $L(n)$. We add some heuristically found constraints to the IP model, and obtain the values for $n \leq 11$. The results are shown in Tab. 4.1, and the example of the packing for $n = 8$ is shown in Fig. 4.4. Our experimental results suggest

Tab. 4.1: Values of the upper bound on $L(n)$ for $n \leq 11$.

n	4	5	6	7	8	9	10	11
$L(n) \leq$	9	15	23	31	40	50	61	72

that $L(n)$ tends to $n^2/2$ when n goes to infinity.

Next, we consider the lower bound on $L(n)$. For a stable packing of L-tricubes, we say that a pixel on the xy-plane (resp., xz-plane and yz-plane) is *marked* if the orthogonal projection of the set of pieces in the packing onto the plane includes this pixel. Given a stable packing, the *shadow size* of the packing is defined as the maximum number of marked pixels in a plane among xy, xz and yz -planes. Let $T(n)$ denote the minimum of a shadow size over all anti-slide packings of L-tricubes for an $n \times n \times n$ box. If one can show a quadratic lower bound on $T(n)$, then it would immediately imply $L(n) = \Omega(n^2)$.

For example, given a stable packing illustrated in Fig. 4.5 (a) for a $7 \times 7 \times 7$ box, the shadow size of the packing is 49 in Fig. 4.5 (b) because each of the number of marked pixels colored by gray in a plane among xy, xz and yz -planes is 18, 49, and 49.

We performed a computer experiment using an IP solver to find a stable packing having a small value of $T(n)$. The results are shown in Tab. 4.2.

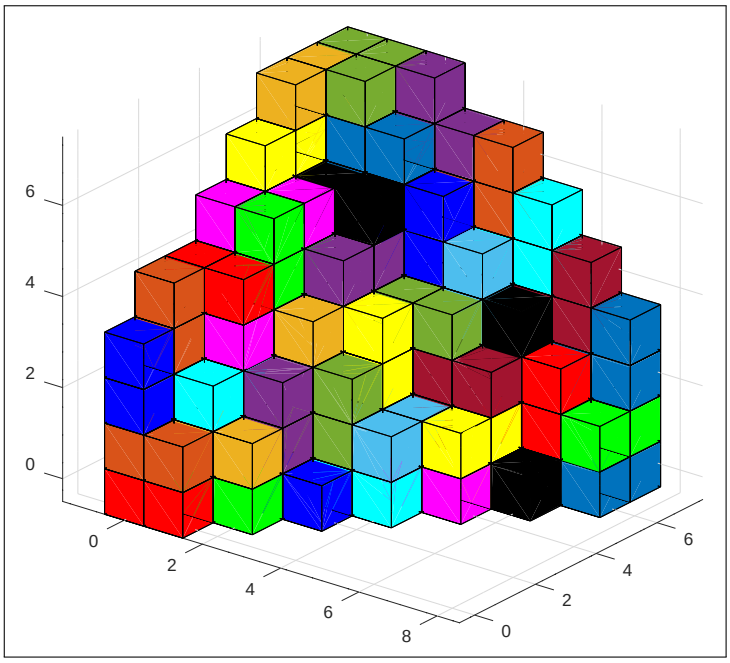


Fig. 4.4: A packing of L-tricube pieces for an $8 \times 8 \times 8$ box.

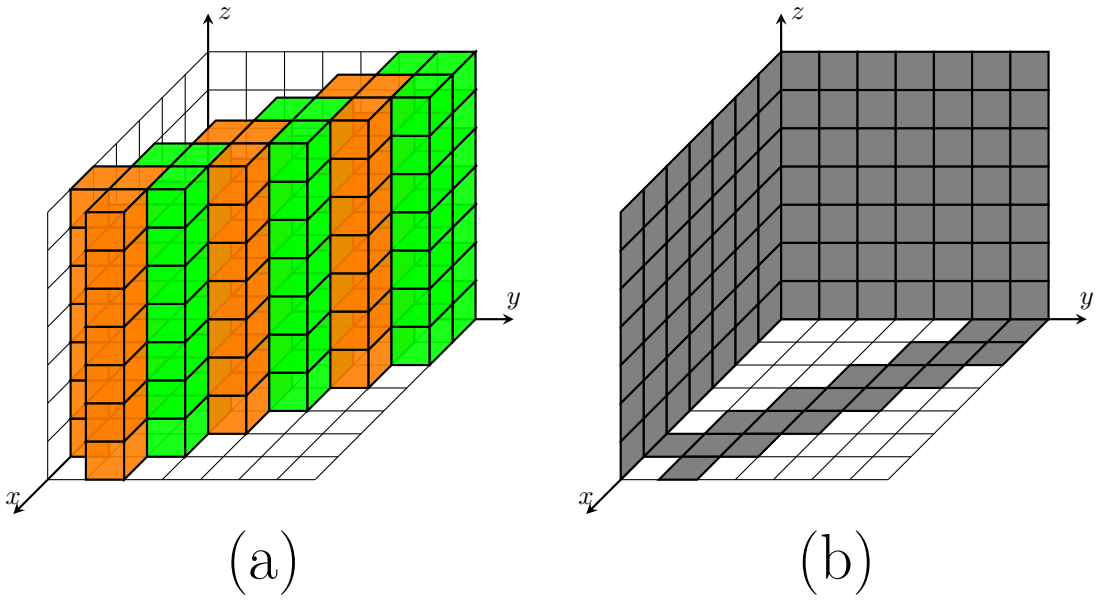


Fig. 4.5: A shadow size for the stable packing for a $7 \times 7 \times 7$ box. A marked pixel is colored gray.

The example of the packing for $n = 6$ is shown in Fig. 4.6 (a). Fig. 4.6 (b)

Tab. 4.2: Values of the upper bound on $T(n)$ for $n \leq 8$.

n	4	5	6	7	8
$T(n) \leq$	13	19	26	35	45

shows the projection on the xz-plane given by this packing. The projections

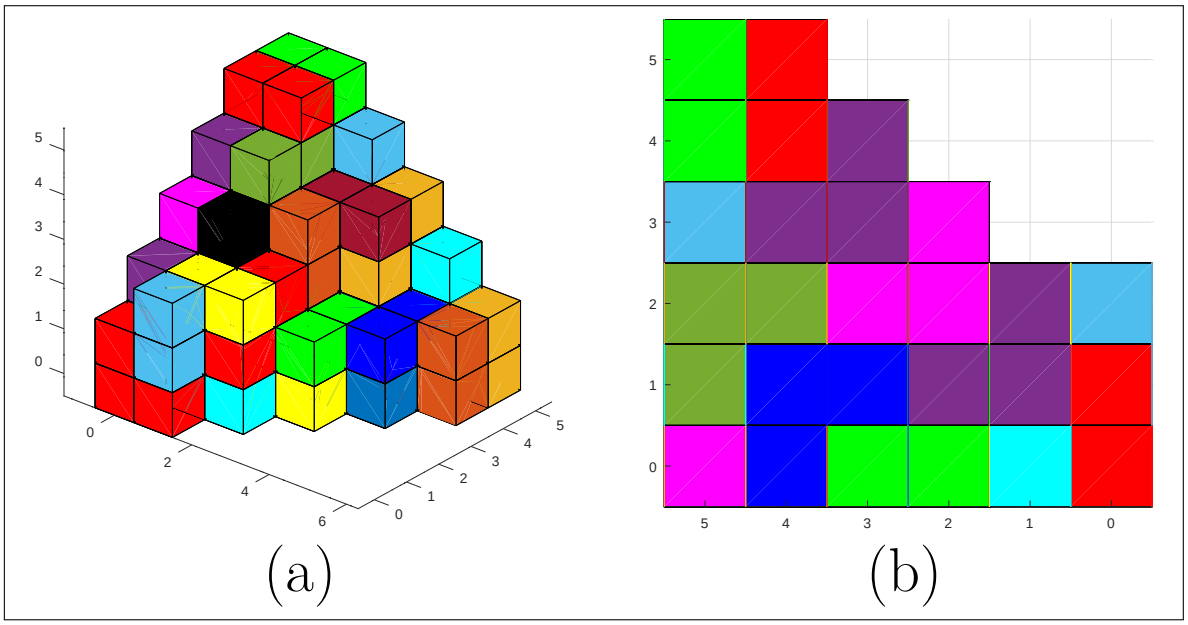


Fig. 4.6: A packing of L-tricube pieces for a $6 \times 6 \times 6$ box.

on the xy-plane and the yz-plane are similar. The result suggests that $T(n)$ would be quadratic in n. We believe that the asymptotic behavior of $L(n)$ is quadratic in n:

Conjecture 12. $L(n) = \Omega(n^2)$. *More ambitiously,* $L(n) \sim n^2/2$.

4.3 Density for $2 \times 2 \times 1$ Pieces for Torus

In this section, we show that there is a stable packing of density $12/25 = 0.48$ when a box is *torus*. We say that a three-dimensional array is a torus with

size $l \times m \times n$ if $(i, j, k) \in \mathbb{N}^3$ is equivalent to

$$((i-1) \bmod l + 1, (j-1) \bmod m + 1, (k-1) \bmod n + 1)$$

for every $(i, j, k) \in \mathbb{N}^3$. We say that a packing for a torus is *quasi-stable* if none of the pieces in the box can slide and all pieces of the packing move together as one block.

Let $R(n)$ denote the minimum ratio of the volume occupied by pieces in a quasi-stable packing of $2 \times 2 \times 1$ pieces for the three-dimensional torus of side length n. We conducted a computer search for a sparse packing of a torus of a size with $l \leq 8$. We use an IP solver based on the IP formulation developed in Amano et al.[34].

The sparsest packing obtained so far is for the $5 \times 5 \times 5$ torus and it uses 15 pieces in Fig. 4.7. The volume density is $4 \cdot 15/5^3 = 12/25$. Since we can obtain a quasi-stable packing for a larger torus by repeating its placement in the direction of x, y, and z-axes, we have the following bound on $R(n)$.

Theorem 13. $R(5k) \leq 12/25$ *for every* $k \geq 1$.

4.4 Minimum Density for $2 \times 2 \times 1$ Pieces

In the previous section, we gave a packing of density 0.48 that uses $2 \times 2 \times 1$ pieces for a torus. In this section, we extend this result to show that the same density can be achieved for a finite box.

Let $F_3(n)$ denote the minimum number of $2 \times 2 \times 1$ pieces of a stable packing for an $n \times n \times n$ box. The *density* of a packing is defined to be the ratio of the volume occupied by pieces. Let $\overline{V}$ denote the upper limit of the minimum density of a stable packing consisting of $2 \times 2 \times 1$ pieces where the size of a box approaches infinity, i.e.,

$$\overline{V} \coloneqq \limsup_{n \to \infty} \frac{4 \cdot F_3(n)}{n^3}. \tag{4.6}$$

As mentioned in [34], we believe that the limit $\lim_{n \to \infty} \frac{4 \cdot F_3(n)}{n^3}$ also exists.

The aim of this section is to show the following upper bound on $\overline{V}$:

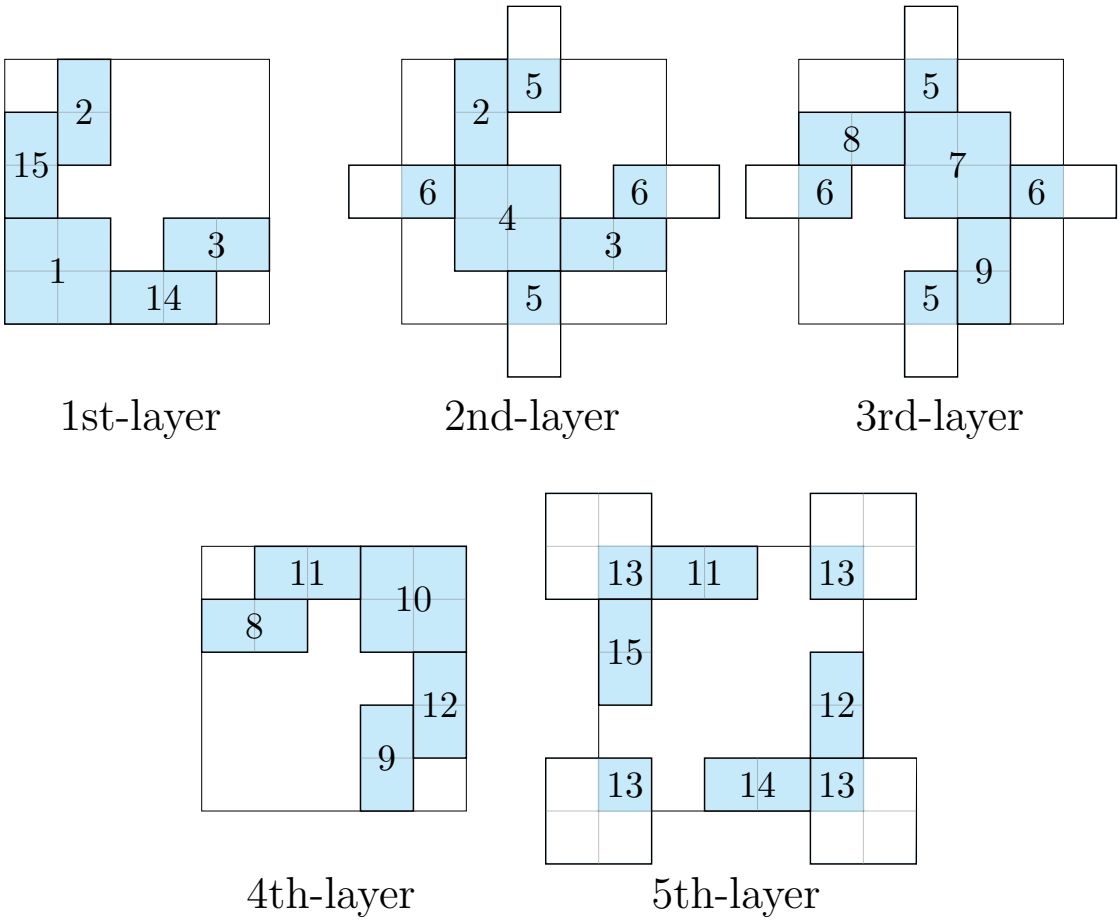


Fig. 4.7: A sparsest quasi-stable packing of $2 \times 2 \times 1$ pieces for a $5 \times 5 \times 5$ torus. Each number represents a piece.

Theorem 14. $\overline{V} \leq 0.48$.

Since we have a placement of density 0.48 for a $5 \times 5 \times 5$ torus, we can obtain an arbitrary large pattern having the same density by repeating the $5 \times 5 \times 5$ pattern along the x-, y- and z-axis. A problem is that the total pattern does not fit a box since the boundary of the pattern is not flat. In order to resolve this, we should give the way of padding a small number of pieces so that a packing becomes flat and stable. Actually, this is exactly what we will discuss in the following.

In what follows, we show the upper bound on $F_3(n)$ by giving an explicit construction of stable packings.

Lemma 15. *For every positive integer* $n \geq 36$, $F_3(n) \leq S(\lfloor (n-6)/5 \rfloor) + (n^3 - (n-8)^3)/4$ *where* $S(k) = 15k^3 + \frac{(5k+6)^3 - (5k)^3}{4}$.

Once we have the lemma, Theorem 14 is obvious since it will imply

$$\frac{4 \cdot F_3(n)}{n^3} = \frac{4 \cdot (15(n/5)^3 + O(n^2))}{n^3} = \frac{12}{25} + o(1) = 0.48 + o(1).$$

The rest of this section is devoted to the proof of Lemma 15.

A cell in a finite box is called *outer* if it touches the boundary of the box. For two cells $c = (i, j, k)$ and $c' = (i, j, k)$ in a finite box, we say that c and c' are *neighbors* if $|i - i'| + |j - j'| + |k - k'| = 1$ holds.

Definition 16. *A stable packing for a box is called stretchable, if, for every two neighboring outer cells in the box, at least one of two cells is occupied by some piece in the packing.*

This property is useful for expanding a stable packing. Later on, we will show that every stretchable packing for an $l \times m \times n$ box can be expanded to a packing for an $(l+2) \times (m+2) \times (n+2)$ box with the same property by padding an appropriate number of pieces.

Here, we introduce the notion of *building block* that represents a pattern of the placement of pieces.

Definition 17. *Given a placement of pieces in an* $l \times m \times n$ *box and a sub-array* $\mathcal{A} = [i, i'] \times [j, j'] \times [k, k'] \subset [l] \times [m] \times [n]$, *an* $(i' - i + 1) \times (j' - j + 1) \times (k' - k + 1)$ *box with the placement of pieces in* $\mathcal{A}$ *is called a building block. Note that a building block may contain a "part" of a piece, e.g.,* $1 \times 1 \times 1$ *or* $2 \times 1 \times 1$ *part of a* $2 \times 2 \times 1$ *piece.*

Given a building block, a positive integer k and some axis, we can form a combined building block by arranging k building blocks along with the designated axis. For two building blocks B and B', we say that B is equal to B', denoted by $B = B'$, if the placement of B is same as the placement of B'.

When $\mathcal{A} = [l] \times [m] \times [k, k']$ such that $[k, k'] \subset [n]$, a building block on $\mathcal{A}$ is called xy*-board*. Similarly, when $\mathcal{A} = [l] \times [j, j'] \times [n]$ ($\mathcal{A} = [i, i'] \times [m] \times [n]$, respectively) a building block on $\mathcal{A}$ is called xz*-board* (yz*-board*, respectively).

Definition 18. *Let P be a stable packing for an $l \times m \times n$ box. We say that P is self-expanded for the z-axis if P satisfies the following: There exists an interval $[k_1, k_2]$ such that three xy-boards B_1, B_2 and B_3 given by the interval $[k_1, k_2]$, $[k_2 + 1, 2k_2 - k_1 + 2]$ and $[2k_2 - k_1 + 3, 3k_2 - 2k_1 + 4]$ are all equal. Given such a packing, we can obtain a stable packing for an $l \times m \times (n + k_2 - k_1 + 1)$ box by inserting the placement of B_2 (of width $k_2 - k_1 + 1$) into $z = k_2 + 1$ in the box. Self-expanded packing for the x- and y-axis is defined in the same way.*

We insert the placement of B_2 into $z = k_2 + 1$ for an $l \times m \times n$ box and the obtained stable packing for an $l \times m \times (n + k_2 - k_1 + 1)$ box is illustrated in Fig. 4.8.

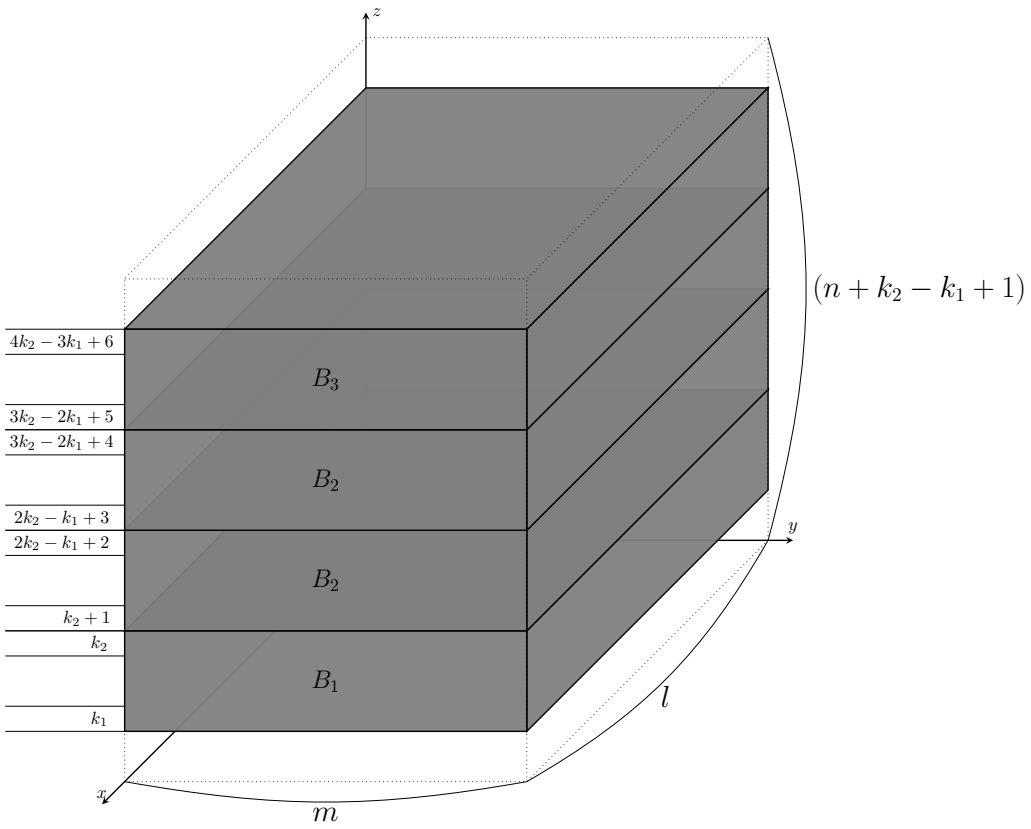


Fig. 4.8: The stable packing for an $l \times m \times (n + k_2 - k_1 + 1)$ box.

When a stable packing is self-expanded for all three axes, the packing is simply called *expandable*.

In what follows, a stretchable and expandable stable packing is called a *adjustable* stable packing. In order to show Lemma 15, we first show the following theorem that resolves the cases $n \equiv 1 \pmod 5$.

Theorem 19. *For every positive integer $k \geq 5$, there exists an adjustable stable packing for a $(5k+6) \times (5k+6) \times (5k+6)$ box with at most $15k^3 + ((5k+6)^3 - (5k)^3)/4$ pieces.*

The proof is by construction. We briefly explain our idea for obtaining the construction of a packing. First, we find an adjustable stable packing for a small-size box. Next, we insert boards into the middle layer of the packing to enlarge its size.

Proof of Theorem 19. We prove the theorem by induction on k. For the sake of the proof, we further impose a condition to a packing that the packing is self-expanded for x-, y- and z-axis with respect to the interval $[9, 13]$. We refer to this condition as Condition (*). The meaning of the numbers 9 and 13 will be clarified later.

The base case, that is $k = 5$, is established by giving an explicit adjustable packing for a $31 \times 31 \times 31$ box. We found this packing by computer search. The packing is consisting of $5,027$ pieces, which is smaller than $S(5) = 5416.5$. Due to the space constraint, we omit the description of the packing itself. Instead, we will explain how we found this packing after the proof of the theorem.

We now proceed to the induction step. We assume for the induction hypothesis that we have an adjustable stable packing for a $(5k+6) \times (5k+6) \times (5k+6)$ box that uses at most $15k^3 + ((5k+6)^3 - (5k)^3)/4$ pieces and satisfies Condition (*). We will expand this packing to an adjustable packing for the box of side length $5(k+1)+6$.

Let Q be an adjustable stable packing for a $(5k+6) \times (5k+6) \times (5k+6)$ box. By the induction hypothesis, Q is self-expanded for the z-axis with respect to the interval $[9, 13]$. Thus, by inserting the placement of the xy-board B given by the interval $[14, 18]$ into $z = 14$ in the box, we can obtain a stable packing for a $(5k+6) \times (5k+6) \times (5(k+1)+6)$ box. We designate

the packing as Q_1. The number of inserted pieces is shown to be at most

$$15k^2 + \frac{5 \cdot (5k+6)^2 - 5 \cdot (5k)^2}{4}. \tag{4.7}$$

Note that Q_1 is self-expanded for the z-axis. Since the placement of B is a board in Q, Q_1 is stretchable. In addition, it is easy to verify that Q_1 is also self-expanded for x- and y-axis with respect to the interval $[9, 13]$. Thus, Q_1 is expandable, hence is adjustable.

We repeat this procedure two more times, i.e., for the x- and y-axis. We insert yz-board B' given by the interval $[14, 18]$ into Q_1 to obtain a stable packing for a $(5(k+1)+6) \times (5k+6) \times (5(k+1)+6)$ box. We designate the packing as Q_2. Note that Q_2 is adjustable. The number of inserted pieces is at most

$$15(k+1)k + \frac{5 \cdot (5(k+1)+6)(5k+6) - 5 \cdot (5k(k+1))}{4}. \tag{4.8}$$

Finally, we insert xz-board B' given by the interval $[14, 18]$ into Q_2 to obtain a stable packing for a $(5(k+1)+6) \times (5(k+1)+6) \times (5(k+1)+6)$ box. We designate the packing as Q_3. Note that Q_3 is adjustable. The number of inserted pieces is at most

$$15(k+1)^2 + \frac{5 \cdot (5(k+1)+6)^2 - 5 \cdot (5(k+1))^2}{4}. \tag{4.9}$$

The total number of pieces in the packing for a $(5(k+1)+6) \times (5(k+1)+6) \times (5(k+1)+6)$ box is upper bounded by

$$15(k+1)^3 + \frac{(5(k+1)+6)^3 - (5(k+1)^3)}{4} \tag{4.10}$$

as desired. □

Construction of the base packing for $31 \times 31 \times 31$ box: Below we explain how we found a $31 \times 31 \times 31$ packing that used in the base case in the proof of Theorem 19.

Since there exists the stable packing for $5 \times 5 \times 5$ torus illustrated in Fig. 4.7, we have the building block of size $5 \times 5 \times 5$ with the same placement of pieces to the packing and designate this building block as $\mathcal{B}_0$.

We create a $15 \times 15 \times 15$ building block consisting of the placement instantiated from a $5 \times 5 \times 5$ building block as follows: First, we arrange three placements instantiated from $\mathcal{B}_0$ along with x-axis, and then we designate the placement of this combined building block as $\mathcal{B}_1$. Note that $\mathcal{B}_1$ is a $15 \times 5 \times 5$ box. Second, we arrange three $\mathcal{B}_1$s along with y-axis and then we designate the arrangement of the combined building block as $\mathcal{B}_2$. Note that $\mathcal{B}_2$ is a $15 \times 15 \times 5$ box. Finally, we arrange three $\mathcal{B}_2$s along with z-axis and then we designate the arrangement of the building block as $\mathcal{B}_3$. Note that $\mathcal{B}_3$ is a $15 \times 15 \times 15$ box.

Next, by using building block $\mathcal{B}_3$, we will create 27 building blocks. We package a $21 \times 21 \times 21$ building block with the three space around $\mathcal{B}_3$ inside the the building block. We designate the $21 \times 21 \times 21$ building block as $\mathcal{B}$. For each x, y, and z axis, we divide the $21 \times 21 \times 21$ building block into 8, 5 and 8 lengths along with the axis, and then obtain the 27 building blocks, and let P be the set of the building blocks: There are eight $8 \times 8 \times 8$ building blocks; There are twelve $8 \times 8 \times 5$ building blocks; There are six $8 \times 5 \times 5$ building blocks; There is one $5 \times 5 \times 5$ building block.

Finally, we will obtain the stable packing for a $31 \times 31 \times 31$ box: We designate a yz-board with $[8, 12]$ in $\mathcal{B}$ as D. We insert two building blocks of D into $x = 8$ in $\mathcal{B}$ and designate the obtained $31 \times 21 \times 21$ building block as $\mathcal{B}'$. Next, we designate a xz-board with $[8, 12]$ in $\mathcal{B}'$ as D. We push in two building blocks of D into $y = 8$ in $\mathcal{B}'$ and designate the obtained $31 \times 31 \times 21$ building block as $\mathcal{B}''$. Lastly, we designate a xy-board with $[8, 12]$ in $\mathcal{B}''$ as D. We push in two building blocks of D into $z = 8$ in $\mathcal{B}''$ and designate the obtained $31 \times 31 \times 31$ building block as $\mathcal{B}'''$.

Thus, we add the constraints that the placement instantiated from $\mathcal{B}'''$ and the stable packing is stretchable to the IP formulation developed in [34]. By using an IP solver, we conducted a computer search for an adjustable stable packing of the $31 \times 31 \times 31$ box consisting of the building blocks over P. The obtained adjustable stable packing for the $31 \times 31 \times 31$ box uses $5,027$ pieces. Thus, the base case holds.

By Theorem 19, we completed the cases $n \equiv 1 \pmod 5$. Below we

show that the same density bound can be achieved for every n such that $n \not\equiv 1 \pmod 5$.

When n is even, it is easy to see that an adjustable packing for an $(n+2) \times (n+2) \times (n+2)$ box can be obtained from an adjustable packing for an $n \times n \times n$ box by putting $((n-2)^3 - (n^2))/4 = O(n^2)$ pieces in a suitable way around the boundary of the box. When n is odd, the following lemma guarantees that such an extension is also possible.

Lemma 20. *For every odd integer $o \geq 5$, there exists a stretchable stable packing for a $2 \times o \times o$ box and also for a $2 \times (o+2) \times o$ box.*

Proof. Our proof is by induction on odd integer o.

The base case is $o = 5$. The packings for the box of size $2 \times 5 \times 5$ and $2 \times 7 \times 5$ are illustrated in Fig. 4.9 is obvious.

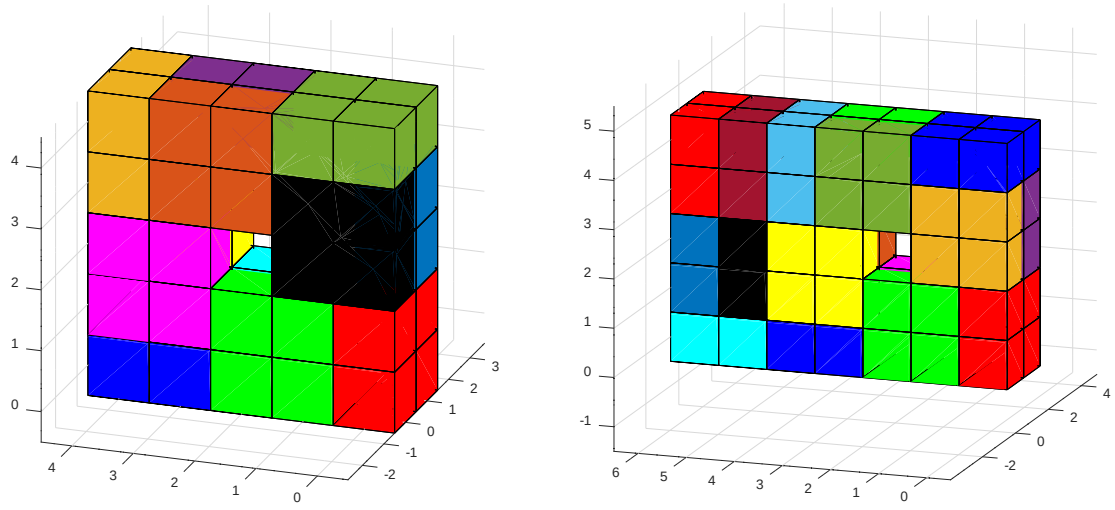


Fig. 4.9: The stretchable stable packings for the box of size $2 \times 5 \times 5$ and $2 \times 7 \times 5$.

For $o \geq 7$, a stretchable packing for a $2 \times o \times o$ box is obtained by combining the packing for a $2 \times o \times 2$ box and the packing for a $2 \times o \times (o-2)$ box. Similarly, a stretchable packing for a $2 \times (o+2) \times o$ box can be obtained by combining the packing for a $2 \times o \times o$ box and the packing for a $2 \times 2 \times o$ box. See Figure Fig. 4.10.

□

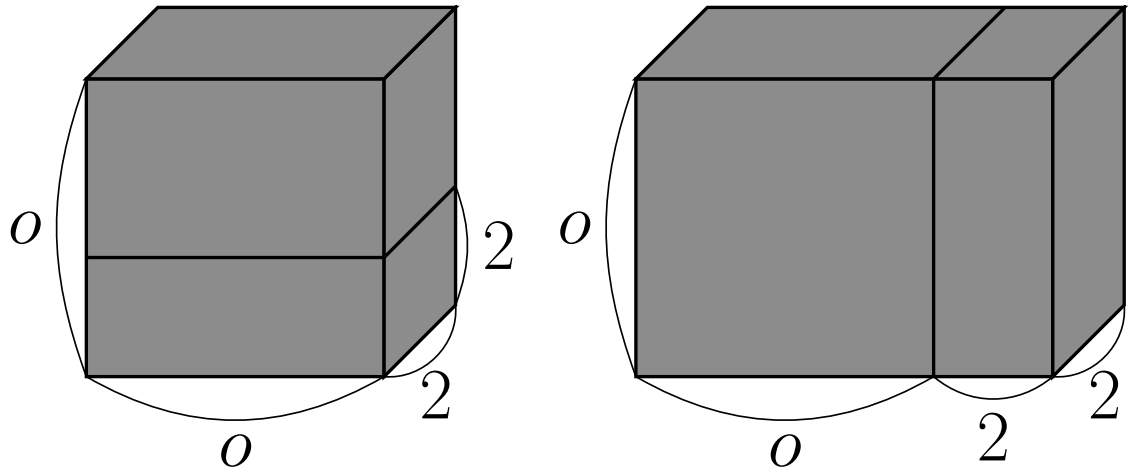


Fig. 4.10: The stretchable stable packings for the box of size $2 \times o \times o$ and $2 \times (o+2) \times o$.

By padding the placement of pieces given in the lemma repeatedly, we can expand an adjustable packing for a box of side length n to an adjustable packing for a box of side length $n+2$, for an odd n.

Proof of Lemma 15. We give a stable packing for every $n \geq 36$ satisfying the statement of the lemma.

By using Lemma 20 if necessary, we can extend an adjustable packing for a box of side length m to the one of side length $m+2$ by adding at most $((m+2)^3 - m^3)/4$ pieces.

By Theorem 19, we have an adjustable stable packing for a $(5k+6) \times (5k+6) \times (5k+6)$ box that uses at most $S(k)$ pieces for every $k \geq 5$. It is easy to see that, for every $n \geq 36$, we can obtain a packing for a box of side length n by applying the above extension at most four times to a packing given in Theorem 19 (for a suitable choice of k). A simple calculation on the number of inserted pieces completes the proof of the lemma. □

4.5 Concluding Remarks

In this chapter, for each shape of piece, we investigated the sparsest anti-slide packings for a three-dimensional box of side length n. For the case

of $2 \times 2 \times 1$ pieces, we showed that there is a stable packing of density $12/25 = 0.48$. Currently, we do not whether this value can be improved or not.

Chapter 5

Conclusion

In this thesis, we investigated the sparsest packing which is similar to the (densest) packing problems and the problem of the card game which is similar to the operation like the sort.

Topswops Problem is long-standing open for 50 years. We extended the sequences for $f(n)$ and $g(n)$. Our results sets a new record for the first time in 12 years [15, COMMENTS in A000375]. We concluded that $f(18) = 191$ and $f(19) = 221$, $g(18) = 1$ and $g(19) = 4$, by applying an algorithm developed by Knuth [33] in a parallel fashion.

As mentioned in Section 2.4, using our method, the traversal for all nodes for $n = 20$ may take about $18,000$ days. If we can prepare many threads, we have the exact value of $f(20)$ by using our method. For example, when we use 200 threads, the computation for $n = 20$ takes only about three months. Since the computation for any $n \geq 21$ takes much longer than for $n = 20$, we would not attempt to obtain the exact values of $f(n)$ for $n \geq 21$ by using our method.

There is an exponential gap between the upper and lower bounds on $f(n)$ for a given positive integer n. Morales [31] showed that there is an initial deck with $\Omega(n^2)$ steps. They [29, 30] showed that a largest deck requires $O(1.618^n)$ steps. To improve the upper and lower bounds would be an interesting future work.

In Anti-slide Problem, we studied the problem where a box is two-dimensional or three-dimensional.

First, we discussed Anti-slide Problem for the two-dimensional square

box using T-tetrominoes in Chapter 3. We determined the value of $F(n)$ exactly for the case of $n \not\equiv 0 \pmod 3$ by proving matching upper and lower bounds of $F(n) = \lfloor 2n/3 \rfloor$. Moreover, we showed $F(n) \leq n - 1$ for the case of $n \equiv 0 \pmod 3$, but cannot prove $F(n) = n - 1$ for this case yet.

Second, we discussed Anti-slide Problem for the three-dimensional cubic box in Chapter 4. In our previous work [26], we investigated the asymptotic behavior of the minimum number of pieces of a stable packing for an $n \times n \times n$ box for various shapes of polycubes. The result obtained in Chapter 3 indicated that $L(n) \leq n(n-1)$ because a stable packing for an $n \times n \times n$ box is obtained by stacking $n - 1$ L-tricubes diagonally on an $n \times n$ box. We observed that these could be categorized into three groups, which is illustrated in Fig. 5.1. Each polycube in Group A, B, and C has

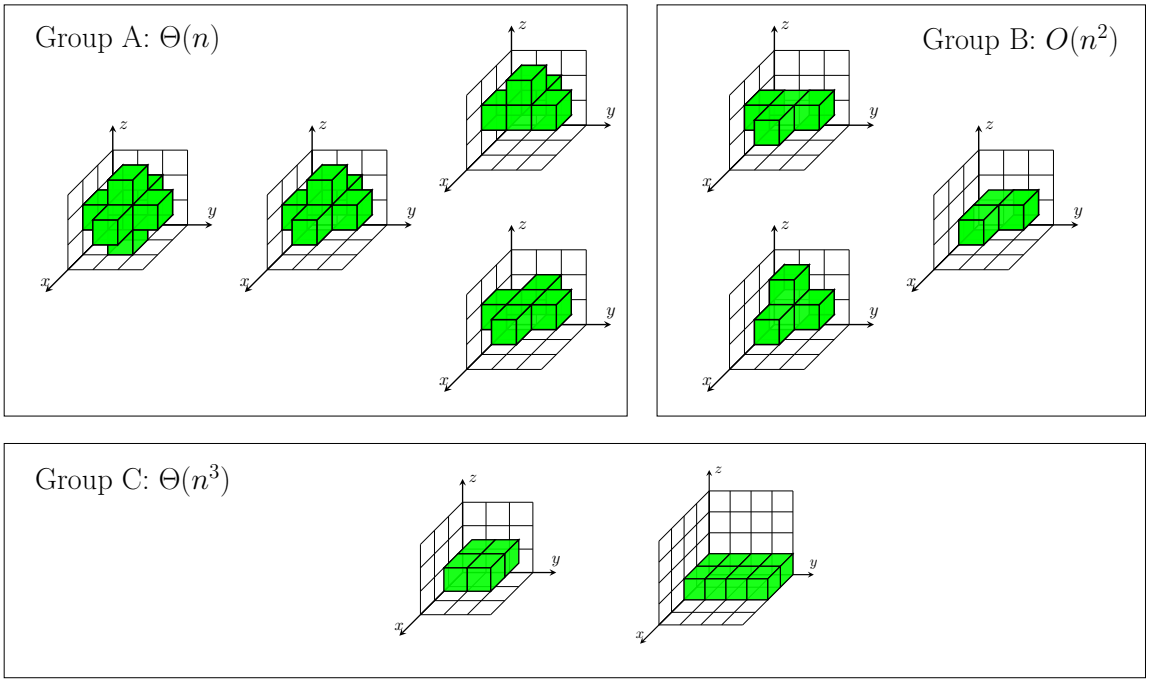


Fig. 5.1: (Recall) Several shapes of pieces analyzed.

a stable packing with $\Theta(n)$, $O(n^2)$, and $\Theta(n^3)$ pieces, respectively. We believe that a polycube in Group B needs $\Omega(n^2)$ pieces to pack.

We focused on a sparsest stable packing for L-tricube pieces and $2 \times 2 \times 1$ pieces. We gave a new construction of a stable packing of L-tricubes. We conjectured that $L(n) \simeq n^2/2$ when n goes to infinity. Next, we investigated

sparsest stable packings of $2 \times 2 \times 1$ pieces. We showed a stable packing of density $12/25 = 0.48$ when a box is torus. We succeeded to remove the assumption that a box is torus without increasing the density, hence we concluded that $\overline{V} \leq 0.48$.

We list below some of the other problems and the topics in this thesis.

- Can we design a more efficient algorithm to find largest decks on n cards, which is considering the multithread?
- Does $f(n) = \Theta(n^3)$?
- Does $L(n) = \Omega(n^2)$?
- What is the exact value of $\overline{V}$?
- Consider a polycube different from the polycubes in Fig. 5.1. Which groups is the polycube categorized in?
- Consider that we fix any one group and any one polycube in the group in Fig. 5.1. What is the relationship between the arrangement of a stable sparsest packing of pieces and the asymptotic behavior of the minimum number of the pieces?
- Consider two types of pieces for a stable packing, e.g., tesseract-hypercubes and L-tricubes. What is the value of the minimum total number of two pieces?

Bibliography

[1] K. Amano, M. Yoshida, Depth Two (n-2)-Majority Circuits for n-Majority, IEICE Trans. on Fund. of Elect., Comm. & Comp. Sci., Vol. E101.A, Issue 9, pp. 1543–1545 (2018).

[2] R. Tadaki and K. Amano, Search for developments of a box having multiple ways of folding by SAT solver, arXiv:2005.02645 (May 2020).

[3] A. Uejima and K. Oe, The Computational Complexity of Creek Puzzles on Several Grids, J. Inf. Process., Vol. 28, pp. 911–918 (2020).

[4] E. D. Demaine, Y. Okamoto, R. Uehara, and Y. Uno, Computational Complexity and an Integer Programming Model of Shakashaka, IEICE Trans. Fund. Vol. E97.A, pp.1213–1219 (2014).

[5] Z. Abel, E. D. Demaine, M. L. Demaine, S. Eisenstat, J. Lynch, and T. B. Schardl, Finding a Hamiltonian Path in a Cube with Specified Turns Is Hard, J. Inf. Process., Vol. 21, No. 3, pp. 368–377 (2013).

[6] K. Haraguchi and H. Ono, Blocksum Is NP-complete, IEICE Trans. Inf. & Syst., Vol. E96.D, No. 3, pp. 481–488 (2013).

[7] R. A. Hern and E. D. Demaine, Games, Puzzles and Computation, A.K. Peters Ltd. (2009).

[8] A. Suzuki, M. Kiyomi, Y. Otachi, K. Uchizawa, and T. Uno, Hitori Numbers, J. Inf. Process., Vol. 25, pp. 695–707 (2017).

[9] K. Y. Siu, V. Roychowdhury, and T. Kailath, Discrete Neural Computation: A Theoretical Foundation, Prentice Hall, 1995.

[10] R. Paturi and M.E. Saks, Approximating Threshold Circuits by Rational Functions, Inf. & Comp., Vol. 112, Issue 2, pp. 257–272 (1994).

[11] R. Paturi and M.E. Saks, On threshold circuits for parity, Proc. 31st Annual Symp. on Found. of Comp. Sci., vol. 1 pp. 397–404 (1991).

[12] K. Uchizawa, Size, Depth and Energy of Threshold Circuits Computing Parity Function, Proc. 31st Inter. Symp. on Algo. and Comp. (ISAAC 2020), LIPIcs, vol. 181, pp. 54:1–54:13 (2020).

[13] G. Yehuda and A. Yehudayoff, Slicing the hypercube is not easy, arXiv:2102.05536 (2021).

[14] M. Gardner, Time Travel and Other Mathematical Bewilderments, W.H. Freeman & Co., 1987.

[15] OEIS Foundation Inc., The On-Line Encyclopedia of Integer Sequences, available from <http://oeis.org>(retrieved December 2022).

[16] Al Zimmermann's Programming Contests, available from <http://azspcs.com/Contest/Cards>(retrieved December 2022).

[17] G.A. Watterson, W.J. Ewens, T.E. Hall and A. Morgan, The chromosome inversion problem, Journal of Theoretical Biology, Vol. 99, Issue 1, pp. 1–7 (1982).

[18] V. Bafna and P. A. Pevzner, Genome Rearrangements and Sorting by Reversals, SIAM Journal on Computing, Vol. 25, No.2, pp. 272–289 (1996)

[19] H. Dweighter, M. R. Garey, D. S. Johnson, and S. Lin, E2569, Journal of The American Mathematical Monthly, Vol. 84, No. 4, pp. 296 (1977)

[20] W. H. Gates and C. H. Papadimitriou, Bounds for sorting by prefix reversal, Discrete Mathematics, Vol. 27, Issue 1, pp. 47–57 (1979).

[21] B. Chitturi, W. Fahle, Z. Meng, L. Morales, C.O. Shields, I.H. Sudborough and W. Voit, An (18/11)n upper bound for sorting by prefix reversals, Theore. Comp. Sci., Vol. 410, Issue 36, pp. 3372–3389 (2009).

[22] M.H. Heydari, and I.H. Sudborough, On the Diameter of the Pancake Network, Journal of Algorithms, Vol. 25, Issue 1, pp. 67–94 (1997).

[23] L. Bulteau, G. Fertin, and I. Rusu, Pancake Flipping is hard, J. Comp. & Sys. Sci., Vol. 81, Issue 8, pp. 1556–1574 (2015).

[24] T. Araki, T. Horiyama, S. Nakano, Y. Okamoto, Y. Otachi, R. Uehara, T. Uno and K. Yamanaka, Sorting by Five Prefix Reversals, SIG Technical Reports, Vol. 179, No. 1, pp. 1–7 (2020).

[25] Anti-slide, available from <http://www.johnrausch.com/PuzzleWorld/puz/anti_slide.htm>(retrieved December 2022).

[26] K. Kimura, K. Amano, and T. Araki. The Analysis of Anti-slide (in Japanese), IEICE Technical Report, Vol. 119, No. 340, COMP2019–43, pp. 101–107 (2019).

[27] P. Brass, W.O.J. Moser, and J. Pach, Research Problems in Discrete Geometry, Springer, (2006).

[28] T. Hales, M. Adams, G. Bauer, T. Dang, J. Harrison, L. Hoang, C. Kaliszyk, V. Magron, S. Mclaughlin, T. Nguyen, Q. Nguyen, T. Nipkow, S. Obua, J. Pleso, J. Rute, A. Solovyev, T. Ta, N. Tran, T. Trieu, J. Urban, K. Vu and R. Zumkleler A Formal Proof of the Kepler Conjecture, Forum of Mathematics, Pi, vol. 5, pp. e2 (2017).

[29] D. Berman, M.S. Klamkin, D.E. Knuth, and J.H. Conway, Problem 76–17, SIAM Review, Vol. 19, No. 4, pp. 739–741 (1977).

[30] D.E. Knuth, The Art of Computer Programming Volume 4 Fascicle 2, Addison-Wesley Prof., pp. 119 (2005).

[31] L. Morales, H. Sudborough, A quadratic lower bound for Topswops, Theore. Comp. Sci., Vol. 411, No. 44, pp. 3965–3970 (2010).

[32] Y. Komano and T. Mizuki, Physical Zero-Knowledge Proof Protocol for Topswops, Proc. Inf. Sec. Prac. and Exp. 2022 (ISPEC 2022), Lecture Notes in Computer Science (LNCS), Vol. 13620, pp. 537–553 (2022).

[33] D.E. Knuth, topswops-fwd.w (the source code of a "better" algorithm), available from <https://www-cs-faculty.stanford.

edu/~knuth/programs/topswops-fwd.w>(retrieved December 2022).

[34] K. Amano, S. Nakano, and K. Yamazaki. Anti-Slide, J. Inf. Process., Vol. 23, No. 3, pp. 252–257 (2015).

[35] Y. Takenaga, Xi Yang, and A. Inada. Anti-Slide Placements of Pentominoes, Proc. the 22nd Japan Conf. on Disc. and Comp. Geom., Graphs, and Games (JCDCG$\hat{3}$), pp. 121–122 (2019).

[36] Gurobi Optimizer, available from <https://www.gurobi.com/>(retrieved December 2022).

[37] K. Minamisawa, R. Uehara, and M. Hara, Mathematical Characterizations and Computational Complexity of Anti-Slide Puzzles, Proc. the 15th International Conf. and Work. on Algo. and Comp. (WALCOM 2021), LNCS Vol. 12635, pp. 321–332 (2021).

[38] "antislide3.w.gz". available from <https://www-cs-faculty.stanford.edu/~knuth/programs/antislide3.w.gz>(retrieved December 2022).

[39] D.E. Knuth, The Art of Computer Programming Volume 4 Fascicle 5, Addison-Wesley Prof. (2019).

[40] S. Grabarchuk, Age of Puzzles. available from <http://www.ageofpuzzles.com/Masters/DonaldEKnuth/DonaldEKnuth.pdf>(retrieved December 2022).