Enumeration of Polyominoes for p4 Tiling

Takashi HORIYAMA* Masato SAMEJIMA*

Abstract

Polyominoes are the two dimensional shapes made by connecting n unit squares, joined along their edges. In this paper, we propose algorithms to enumerate polyominoes for p4 tiling, i.e., those covering the plane by only 90 degrees rotations around two rotation centers. The conventional methods are basically trial and error, i.e., they repeat generating polyominoes and checking whether the shapes have been already generated. Our approach is based on the reverse search, in which we design rules to generate the next. This technique has the following two characteristics: (1) No trial and error, which implies that we can reduce the computation time. (2) No need to store already enumerated polyominoes. Thus, we can also reduce the space complexity. We also implement the algorithm and enumerate all polyominoes for p4 tiling up to $n = 20$.

1 Introduction

Polyominoes are the two dimensional shapes made by connecting n equal-sized squares, joined along their edges. The first pentomino (i.e, the polyomino consisting with five squares) problem was proposed in 1907 by Dudeney, and the name polyomino was invented in 1953 by Golomb [5]. Since then, enumerating and counting the number of polyominoes has been of interests in combinatorial geometry, puzzle, and recreational mathematics (see, e.g., [8, 11]).

Another aspect of the interests is on polyomino tiling [6, 14]: Enumerate or count the number of ways to cover a given region by dominoes (or by specified polyominoes in general tiling), where neither overlapping nor hanging over the region is allowed. Enumeration and counting for covering the two dimensional plane are also considered. Recently, Fukuda et al. proposed a new tiling problem, in which polyominoes for p4 tiling are enumerated [3, 4]. More precisely, in the p4 tiling problem, the coordinates of the origin and the terminus are given. Our task is to enumerate polyominoes that cover the plane by only 90 degrees rotations around the origin and the terminus. The approaches in [3, 4] are basically trial and error, i.e., they repeat generating polyominoes and checking whether the shapes have been already generated.

*Graduate School of Science and Engineering, Saitama University, horiyama@al.ics.saitama-u.ac.jp

In this paper, we propose the use of the reverse search [1], which gives efficient algorithms in various combinatorial and geometric enumeration problem, see e.g., [2, 7, 9, 10, 13]. Under this search scheme, we first define a rooted tree whose nodes correspond to the polyominoes for p4 tiling. By designing rules for obtaining child nodes from any node, we can enumerate all nodes (i.e., all polyominoes) by traversing the tree. This technique has the following two characteristics: (1) No trial and error, since we have rules to generate the next. Thus, we can reduce the computation time. (2) No need to store already enumerated polyominoes. Thus, we can also reduce the space complexity.

The rest of this paper is organized as follows. The next section gives basic concepts on p4 tiling. Section 3 introduces the tree structures among polyominoes. In Section 4, we propose algorithms to enumerate all polyominoes for p4 tiling. Experimental results are also shown in the section. Section 5 gives concluding remarks.

2 Preliminaries

In the *p4 tiling problem*, the coordinates of the two rotation centers, the *origin* $(0, 0)$ and the *terminus* (x, y) are given, where x and y are integers. Since we can exchange the roles of the two rotation centers, without loss of generality, we can assume $y \geq x \geq 0$. A polyomino is a *p4 tiling* if it satisfies the following conditions: (1) It covers the plane by the p4 symmetry group [12], i.e., by the rotations of 90 degrees around the origin and the terminus. The cover (or called isohedral tiling) should not contain gaps nor overlaps. (2) It includes both of the origin and the terminus. (3) It is connected. Note that two squares are not connected if they meet in only one point. Our task is to enumerate all polyominoes for p4 tiling.

In [3, 4], the properties of a polyomino for p4 tiling are shown. (1) It consists of exactly $n = (x^2 + y^2)/2$ unit squares. Since x, y and n are integers, n can be 1, 2, 4, 5, 8, 9, 10, 13, 16, 17, 18, 20, ... Note that x and y have the same odd/even parity. (2) The p4 symmetry group induces n equivalence classes on the positions of the unit squares. (3) No two unit squares belong to the same equivalence class. Otherwise, we can move either of the two to another position by the repetition of the 90 degrees rotations, which implies an overlap for covering the plane.

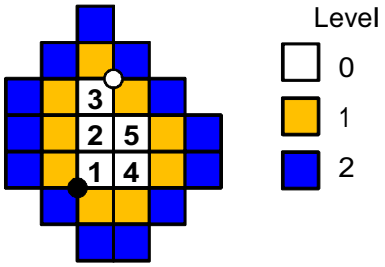

Figure 1: Levels for $n = 5$.

3 Family Trees

In this section, we define a tree structure $T_{(x,y)}$ among polyominoes for p4 tiling with the origin and the terminus (x, y). We first define neighboring polyominoes, and then describe a tree on the relation.

The *neighbors* of a polyomino P are the polyominoes by deleting a square from P and adding it to another position so that (1) the deleted and added positions belong to the same equivalence class, (2) the resulting polyominoes should include both of the origin and the terminus. (3) the resulting polyominoes should keep the connectivity. Note that these three have the same meaning with the three conditions of the polyominoes for p4 tiling defined in Section 2.

Let the root node of $T_{(x,y)}$, called *the root polyomino*, consist of the following two regions: (1) Rectangle of size xy whose down-left is the origin and up-right is (x, y). (2) Rectangle of size $(y-x)^2/2$ whose down-left is $(x, 0)$ and up-right is $((x + y)/2, y - x)$. We can see that the polyomino has size $xy+(y-x)^2/2 = (x^2+y^2)/2$. In case $x = 0$, the root polyomino consists of only rectangle (2).

The *parent* of a polyomino is one of its neighbors selected by the following three rules: (*Fundamental rule: child → parent*) Move a square so that its "level" becomes smaller. The *level* of a square is the distance from the root polyomino. The squares in the root polyomino have level 0. Those next to the root have level 1. Those next to the squares in level i have level $i + 1$.

(*Additional rule 1: child → parent*) If we can move two or more squares, we select the square of the smallest number. Figure 2 is an example of such situation. From the shape of the bottom, we can move both of the squares 4 and 5, and thus we have two choices to reach the shape of the top: (1) Move square 4 first and then square 5, and (2) move square 5 first and then square 4. According to the additional rule 1, we can uniquely obtain the parent by moving square 4.

(*Additional rule 2: child → parent*) If we can move a square to two or more possible positions of a smaller level, we select the position of the smallest level. Figure 3 is an example of such situation. The shape of the bottom-right in Figure 3 has the smallest movable

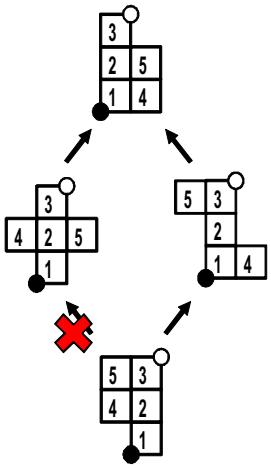

Figure 2: Additional rule 1 (child → parent).

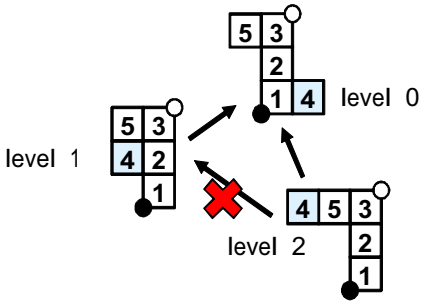

Figure 3: Additional rule 2 (child → parent).

square 4 of level 2. The square has two possible positions of level 0 and 1, where the three shapes are mutually neighbors to each other. According to the additional rule 2, we move square 4 to the position in level 0.

If two or more possible positions have the smallest level at the same time in the additional rule 2, we break the tie by their coordinates. For example, suppose we have two positions (x_1, y_1) in level ℓ_1 and (x_2, y_2) in level ℓ_2, where the coordinates give the down-left of the positions. Then, the position (x_1, y_1) is smaller than (x_2, y_2) if either of (1) $\ell_1 < \ell_2$, (2) $\ell_1 = \ell_2$ and $x_1 < x_2$, and (3) $\ell_1 = \ell_2$, $x_1 = x_2$ and $y_1 < y_2$ hold. We also regard this tie-breaking rule as a part of the level comparison in the additional rule 2.

For any polyomino P, by repeating the process of obtaining the parent, we can observe a sequence from P to the root polyomino. By merging such sequences for all polyominoes, we can obtain the *family tree* $T_{(x,y)}$ for p4 tiling, in which nodes correspond to the polyominoes for p4 tiling, and edges correspond to the pairs of polyominoes and their parents. Figure 4 illustrates the family tree $T_{(1,3)}$. The number of squares is $n = 5$.

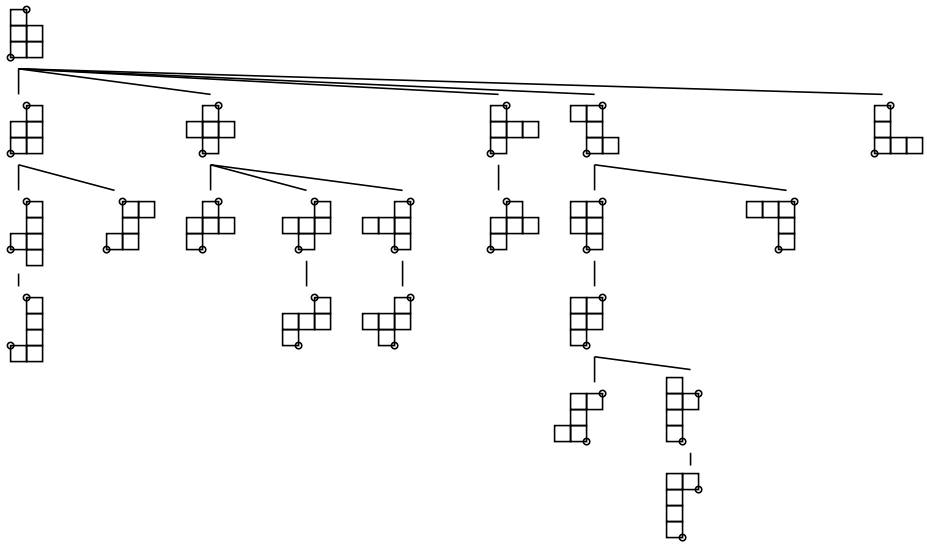

Figure 4: Family tree $T_{(1,3)}$.

4 Algorithms and Experimental Results

In this section, we propose an algorithm to enumerate polyominoes for p4 tiling by traversing the family tree $T_{(x,y)}$. The starting point is the root polyomino. If we have rules to obtain all children of any node in $T_{(x,y)}$, by repeatedly applying the rules, we can traverse $T_{(x,y)}$.

The rules from a parent to its children are obtained by reverting the rules from a child to its parent defined in Section 3. The reverse of the fundamental rule is as follows: (*Fundamental rule: parent → children*) Move a square so that its level becomes larger. To avoid generating the same polyomino twice, we use the following additional rules.

(*Additional rule 1: parent → children*) If we can move a square to the position of a smaller level, we prohibit its any move. This rule is the reverse of the additional rule 2 from a child to its parent. Figure 3 illustrates such situation. While square 4 in the left shape has a possible position in level 2 (the bottom-right shape), we prohibit the move since it can be in level 0 that implies the top-right shape is the parent of the bottom-right shape.

(*Additional rule 2: parent → children*) Move a square so that it has smaller number than the previously moved square, or it prevent the move of the previously moved square. In case the parent node is the root polyomino, we assume that we have the previously moved square that is larger than any of the squares. This rule is the reverse of the additional rule 1 from a child to its parent.

We implemented the algorithm and enumerated polyominoes for p4 tiling up to $n = 20$. Table 4 summarizes the number of polyominoes and their computation time. The second column indicates the the coordinates of the corresponding terminus. It takes only 10.74 seconds for 18-ominoes and 46.26 seconds even for 20-ominoes on Intel Xeon CPU 3.0GHz. Figure 5 is a partial list of enumerated 17-ominoes for p4 tiling.

Table 1: The numbers of polyominoes for p4 tiling and their computation time.

n	terminus	#polyominoes	Time (sec)
4	$(2,2)$	9	0.01
5	$(1,3)$	21	0.01
8	$(0,4)$	166	0.01
9	$(3,3)$	317	0.02
10	$(2,4)$	596	0.03
13	$(1,5)$	4,167	0.30
16	$(4,4)$	26,448	2.56
17	$(3,5)$	48,970	5.71
18	$(0,6)$	90,652	10.74
20	$(2,6)$	302,042	46.26

5 Concluding Remarks

We proposed an algorithm to enumerate polyominoes for p4 tiling. Our algorithm distinguishes two polyominoes that are identical with each other if we exchange the two rotation centers. To overcome this situation, we should introduce a lexicographic order on the polyominoes and generate only the lexicographically smallest (or the largest) polyominoes.

The techniques in this paper are easily applicable to other rotation symmetry groups p3 (120 degrees rotation) and p6 (60 degrees rotation). Since there are 17 symmetry groups for tiling the plane [12], we should address their enumerations.

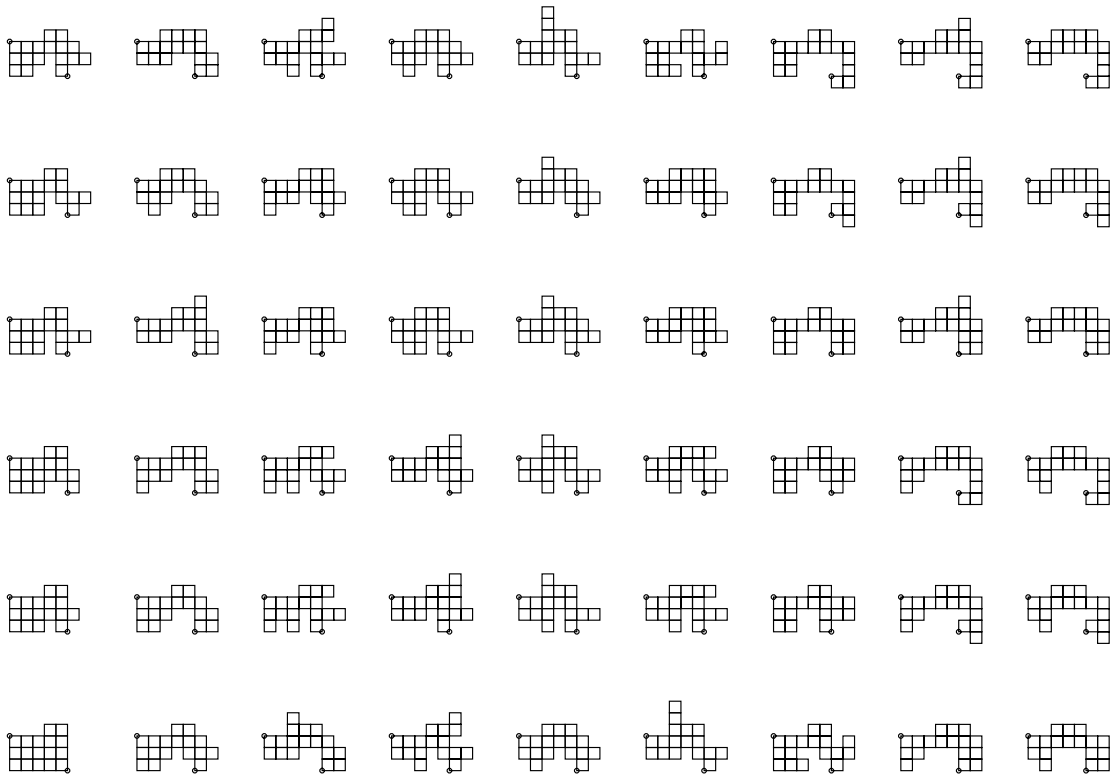

Figure 5: Partial list of 17-ominoes for p4 tiling.

References

[1] D. Avis and K. Fukuda, Reverse Search for Enumeration, *Discrete Appl. Math.*, 6, pp.21–46, 1996.

[2] D. Avis, K. Naoki, M. Ohsaki, I. Streinu, S. Tanigawa, Enumerating Constrained Non-crossing Minimally Rigid Frameworks, *Discrete and Computational Geometry*, vol.40, no.1, pp.31–46, 2007.

[3] H. Fukuda, N. Mutoh, G. Nakamura, and D. Schattschneider, A Method to Generate Polyominoes and Polyiamonds for Tilings with Rotational Symmetry, *Graphs and Combinatrics*, 23, pp.259–267, 2007.

[4] H. Fukuda, N. Mutoh, G. Nakamura, and D. Schattschneider, Enumeration of Polyominoes, Polyiamonds and Polyhexs for Isohedral Tilings with Rotational Symmetry, *Lecture Notes in Computer Science* 4535, pp.68–78, 2008.

[5] S. Golomb, *Polyominoes: puzzles, Patterns, Problems, and Packings*, Princeton University Press, Princeton, NJ, second edition, 1994.

[6] P. W. Kasteleyn, The Statistics of Dimers on a Lattice: I. The Number of Dimer Arrangements on a Quadratic Lattice, *Physica*, vol.27/12, pp.1209–1225, 1961.

[7] Y. Kikuchi, K. Yamanaka, and S. Nakano, A Simple Generation of Multi-dimensional Partitions, *IEICE Technical Reports*, COMP2008-49, pp.23–29, 2008.

[8] D. E. Knuth, `http://www-cs-faculty.stanford.edu/~knuth/programs.html`.

[9] Y. Matsui, R. Uehara, and T. Uno, Enumeration of Perfect Sequences of Chordal Graph. *Lecture Notes in Computer Science*, 5369, pp.859–870, 2008.

[10] S. Nakano, Enumerating Floorplans with n Rooms, *Lecture Notes in Computer Science*, 2223, pp.104–115, 2001.

[11] T. R. Parkin, L. J. Lander, and D. R. Parkin, Polyomino Enumeration Results, *SIAM Fall Meeting*, Santa Barbara, California, 1967.

[12] D. Schattschneider, The Plane Symmetry Groups Their Recognition and Notation, *American mathematical monthly*, 85, pp.439–450, 1978.

[13] K. Yamanaka, S. Kawano, Y. Kikuchi, and S. Nakano, Constant Time Generation of Integer Partitions, *IEICE Trans. Fundamentals*, vol.E90-A, no.5, pp.888–895, 2007.

[14] M. Yoshida, Jinkōki, 1627.