Packing Trominoes is NP-Complete, #P-Complete and ASP-Complete

Takashi Horiyama* Takehiro Ito† Keita Nakatsuka‡ Akira Suzuki† Ryuhei Uehara§

Abstract

We study the computational complexity of packing puzzles of identical polyominoes. Packing dominoes (i.e., 1×2 rectangles) into grid polygons can be solved in polynomial time by reducing to a bipartite matching problem. On the other hand, packing 2×2 squares is known to be NP-complete. In this paper, we fill the gap between dominoes and 2×2 squares, that is, we consider the packing puzzles of trominoes. Note that there exist only two shapes of trominoes: L-shape and I-shape. We show that their packing problems are both NP-complete. Our reductions are carefully designed so that we can also prove #P-completeness and ASP-completeness of the counting and the another-solution-problem variants, respectively.

1 Introduction

Since Golomb introduced the notion of polyominoes, many puzzles have been considered and solved for polyominoes [5]. Most puzzles are classified into two groups: the sliding-block puzzles and the packing puzzles (see Figure 1(a) and Figure 1(b), respectively). The computational complexity of solving sliding-block puzzles was a long standing open problem since Gardner introduced the problem in 1964 [4]. In 2005, it was settled by Hearn and Demaine [6]: generalized sliding-block puzzles are PSPACE-complete even if all blocks are of size 1×2. On the other hand, sliding-block puzzles are polynomial-time solvable if all blocks are of size 1×1 (see [7] for further details). Thus, in this sense, we have no gap for sliding-block puzzles.

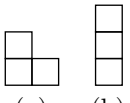
(a) (b)

Figure 1: (a) A sliding-block puzzle and (b) a packing puzzle.

In this paper, we consider packing puzzles, and we aim to fill this kind of gap. We consider the problem of so-called identical packing into the two dimensional plane: Given k identical polyominoes and a polygon P, determine whether the k polyominoes can be placed in P without overlap or not. We note that P may contain holes. Packing dominoes (i.e., 1×2 rectangles) into a polyomino P (i.e., a polygon made by connecting unit squares) can be solved in polynomial time by reducing to a bipartite matching problem. (See [8] for a bipartite matching algorithm.) On the other hand, packing 2×2 squares into a polyomino is known to be NP-complete [2, 3]. That is, there exist gaps between dominoes and 2×2 squares: Can you pack trominoes into a polyomino in polynomial time?

Here, a tromino is a (connected) polygon with three unit squares. As shown in Figure 2, trominoes have only two possible types of shapes: L-trominoes and I-trominoes. We consider the corresponding two problems, called 3L-PACKING and 3I-PACKING. In 3L-PACKING, an integer k and a polyomino P are given. P is given as a set of pairs of integers, corresponding to the positions (e.g., the centers) of unit squares in P. The 3L-PACKING problem is to determine whether k L-trominoes can be placed in P without overlap or not. 3I-PACKING is defined in the same manner, so as to pack I-trominoes.

We prove both problems are NP-complete by giving reductions from one-in-three 3SAT. The reductions are carefully designed so that each (valid) packing of trominoes has one-to-one correspondence with a (valid) solution of the original instance of one-in-three 3SAT. Therefore, our reductions also imply the results for the counting variant and the another-solution-problem variant of the tromino packing problems. Sim-



(a) (b)

Figure 2: (a) An L-tromino and (b) an I-tromino.

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

†Graduate School of Information Sciences, Tohoku University, {takehiro, a.suzuki}@ecei.tohoku.ac.jp

‡Faculty of Engineering, Saitama University, nakatsuka@al.ics.saitama-u.ac.jp

§School of Information Science, JAIST, uehara@jaist.ac.jp

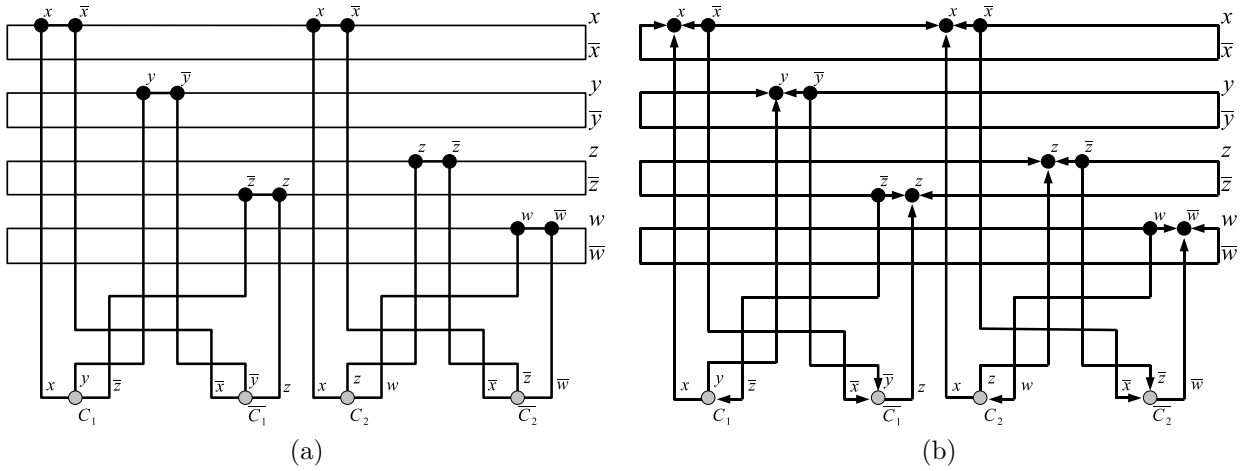


Figure 3: (a) Instance G_φ of 1-in-3 GO, and (b) its valid orientation.

ilar to other ASP-complete problems [10], the another-solution-problem variant is defined as follows: Given an instance (k, P) of 3L-PACKING or 3I-PACKING and its (valid) solution s, find a solution s' of (k, P) other than s. The counting variants and the another-solution-problem variants for the 3L-PACKING and 3I-PACKING problems are #P-complete and ASP-complete, respectively.

2 Reduction

To prove the NP-completeness of 3L-PACKING and 3I-PACKING, we introduce a graph orientation problem, called the one-in-three graph orientation problem (or 1-in-3 GO in short), and give reductions from one-in-three 3SAT to 3L-PACKING and 3I-PACKING via 1-in-3 GO.

2.1 Reduction to 1-in-3 Graph Orientation Problem

We first give a polynomial-time reduction from one-in-three 3SAT to 1-in-3 GO.

In one-in-three 3SAT, we are given a 3-CNF φ consisting of m clauses with n variables, where each clause C_j contains three literals (variables or their negations). One-in-three 3SAT is to determine whether there is a satisfying assignment to the variables so that each clause in φ has exactly one true literal. For example, given $\varphi = (x \vee y \vee \overline{z})(x \vee z \vee w)$, we have a satisfying assignment $(x, y, z, w) = (\text{False}, \text{False}, \text{False}, \text{True})$.

1-in-3 GO is defined as follows:

Definition 1 1-*in*-3 GO (*One-in-three Graph Orientation Problem*).

An undirected 3*-regular graph* $G = (V, E)$ *is given, where* V *can be partitioned into three* (*disjoint*) *node-subsets* V_ℓ*,* V_c *and* V_n *consisting of literal nodes, clause nodes, and negated-clause nodes, respectively. The objective is to determine whether we can assign a direction to each edge so that* (1) *every literal node in* V_ℓ *has in-degree* 0 *or* 3; (2) *every clause node in* V_c *has exactly one inbound edge, i.e., in-degree* 1; (3) *every negated-clause node in* V_n *has exactly one outbound edge, i.e., out-degree* 1.

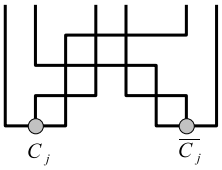


Figure 4: A clause node and its corresponding negated-clause node.

Figure 3(a) illustrates the undirected graph G_φ corresponding to an instance $\varphi = (x \vee y \vee \overline{z})(x \vee z \vee w)$. Note that a node in G_φ is depicted by a (black or gray) circle. The upper half of G_φ consists of n cycles, each of which corresponds to a variable in φ. Each cycle consists of some pairs of literal nodes, and the number of pairs equals to the number of occurrences of the variable in φ. If a pair of literal nodes is located in the upper half of its belonging cycle, the left (resp., right) node of the pair is labeled with a positive (resp., negative) literal. Otherwise, the left (resp., right) node is labeled with a negative (resp., positive) literal. The lower half of G_φ consists of m gadgets given in Figure 4, where the nodes labeled with C_j are clause nodes, and those labeled with $\overline{C_j}$ are negated-clause nodes. If clause C_j contains three literals ℓ_1, ℓ_2 and ℓ_3, the clause node labeled with C_j has exactly three edges connecting with literal nodes labeled with ℓ_1, ℓ_2 and ℓ_3. A negated-clause node labeled with $\overline{C_j}$ has exactly three edges connecting with literal nodes labeled with the negated literals $\overline{\ell_1}$, $\overline{\ell_2}$ and $\overline{\ell_3}$.

Therefore, G_φ contains $8m$ nodes and $12m$ edges in

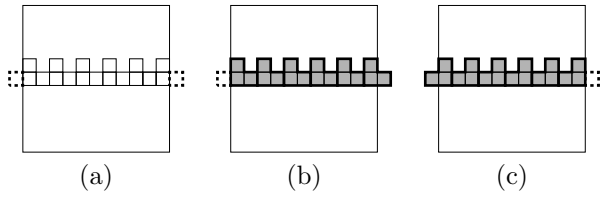


Figure 5: A line gadget and its two ways of packing.

total. We can draw G_φ within the grid of width $O(m)$ and height $O(n)$. Thus, we can obtain G_φ in $O(mn)$ time.

Figure 3(b) illustrates a valid orientation of the graph G_φ in Figure 3(a). Note that the two vertical edges emanating from a pair of literal nodes have different orientations, which implies that one is assigned True and another is assigned False. Also note that the literal nodes labeled with the same literal have the same vertical orientation, i.e., there is a consistency on the assignment to the variables. If a positive literal node has a downward (resp., upward) edge, its corresponding variable in φ is assigned True (resp., False). Restriction (2) on clause node labeled with C_j guarantees that exactly one literal in C_j is assigned True.

Thus, valid orientations to G_φ of 1-in-3 GO have one-to-one correspondence with satisfying assignments to φ of one-in-three 3SAT. We can easily check 1-in-3 GO is in NP, in #P and in ASP. Since one-in-three 3SAT is NP-complete [9], #P-complete [1] and ASP-complete [10], we have the following theorem.

Theorem 1 1-*in*-3 GO *is NP-complete, #P-complete and ASP-complete.*

2.2 Reduction to 3L-PACKING

Now, we give a polynomial-time reduction from 1-in-3 GO to 3L-PACKING. An intuitive correspondence between the two problems can be observed in Figure 5. We use the gadget in Figure 5(a) as a *line gadget*. We can place trominoes on both of the solid and dotted unit squares. If we pack as many L-trominoes as possible into this gadget, we have only two ways of packings as illustrated in Figures 5(b) and (c). We regard the line gadget in Figure 5(a) as an edge, and the ways of packings (b) and (c) as two corresponding orientations of the edge: Packing (b) corresponds to the orientation from left to right; in contrast, packing (c) corresponds to the orientation from right to left. The dotted unit square covered (resp., not covered) by a tromino represents that the orientation is outbound (resp., inbound).

If we need to bend an edge, we use a *corner gadget* in Figure 6(a). Similarly to the case of a line gadget, if we pack as many L-trominoes as possible into a corner gadget, we have only two ways of packings. The dotted unit square covered by a tromino represents that the orientation is outbound. By combining gadgets as in Figure 6(b), we can propagate the orientations of edges.

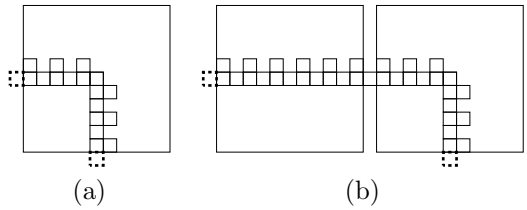


Figure 6: (a) A corner gadget, and (b) a combination of line and corner gadgets.

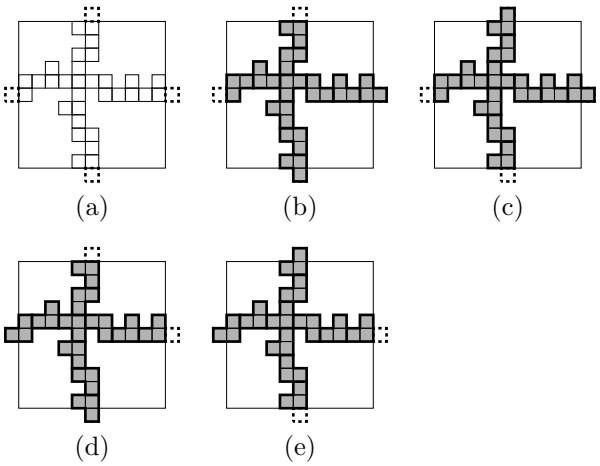


Figure 7: A cross gadget and its four ways of packings.

We respectively replace the crossing points of edges, pairs of literal nodes, clause nodes, and negated-clause nodes by cross, duplicator, clause, negated-clause gadgets, which are defined later. All but the duplicator gadget are of the same size. (More precisely, the normal size is 11×11.) The size of a duplicator gadget equals to that of a combination of two normal-size gadgets, since we replace two literal nodes by one duplicator gadget. As a result of the replacement, we can obtain a patchwork of the gadgets as a polyomino of 3L-PACKING.

A *cross gadget* is given in Figure 7(a). There are four ways, as illustrated in Figure 7(b)–(e), to cover the crossing unit square in Figure 7(a) by an L-tromino. Note that the left and right dotted unit squares always represent the same orientation; so do the upper and bottom dotted unit squares. In contrast, the orientations of vertical and horizontal directions are independent.

A pair of literal nodes is replaced by a *duplicator gadget* given in Figure 8(a). As mentioned before, the width of a duplicator gadget equals to that of a combination of two normal-size gadgets. Apart from the cross gadgets, if we pack as many L-trominoes as possible into a duplicator gadget, we have only two ways of packings. The two dotted unit squares in the left half of the gadget have the same orientation (inbound or outbound); so do the two dotted unit squares in the right half.

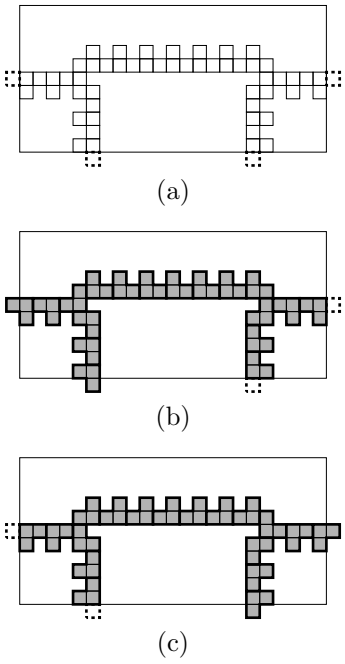


Figure 8: A duplicator gadget and its two ways of packings.

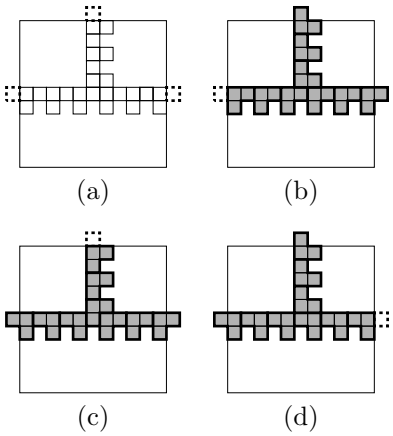


Figure 9: A clause gadget and its three ways of packings.

A clause node is replaced by a *clause gadget* given in Figure 9(a). We have three ways of packing. In every packing, there are exactly one inbound orientation and exactly two outbound orientations. We can regard this gadget as a three-forked road. The tromino covering the center of the three-forked road indicates which edge has inbound orientation. A negated-clause node is replaced by a *negated-clause gadget* given in Figure 10. Similar to the case of a clause gadget, we have three ways of packing. In every packing, however, there are exactly one outbound orientation and exactly two inbound orientations.

Since the drawing of graph G_φ is of size $O(mn)$, we use at most $O(mn)$ gadgets for replacing the elements in G_φ. All gadgets consist of constant number of unit squares, which implies a polynomial-time construction of P for the reduction.

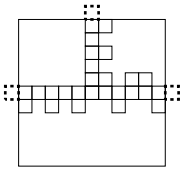


Figure 10: A negated-clause gadget.

We set the 'magic number' k of 3L-PACKING as $|P|/3$, where $|P|$ denotes the number of unit squares in P. This enforces that no unit squares in P remain uncovered. All of the above explained packings satisfy this constraint. Moreover, there is no other way of packing. Thus, valid packings to P of 3L-PACKING have one-to-one correspondence with valid orientations to G_φ of 1-in-3 GO. We can easily check 3L-PACKING is in NP, in #P and in ASP, since P is given as a set of the coordinates of all unit squares in P. By Theorem 1, we have the following theorem.

Theorem 2 3L-PACKING *is NP-complete, #P-complete and ASP-complete.*

2.3 Reduction to 3I-PACKING

By a similar argument as above, we give a polynomial-time reduction from 1-in-3 GO to 3I-PACKING. We use the gadgets in Figure 11(a)–(f) as line, corner, cross, duplicator, clause and negated-clause gadgets, respectively. The gadgets in Figure 11(a), (b) and (e) are straightforward for packing. We note that a crank in a line gadget (Figure 11(a)) guarantees exactly two ways of packings, which correspond to two orientations. These non-trivial gadgets work as follows.

(c) Cross gadget. In general, the cross gadgets are the most difficult part to design in these kinds of reductions, as mentioned in [7]. Such situation also occurs in 3I-PACKING as you can observe from Figure 11(c) and 12. We first observe that each of the left and right squares drawn in dotted line has two ways of packings since it is close to the corner as in the line gadget in Figure 11(a). The top and bottom dotted squares also have two ways since they are close to a three-forked road in the gadget. Next, the gadget (or the huge square) contains 127 squares. Thus, the gadget is covered by 43 trominoes, and two squares remain. The remaining squares can cover two out of the four dotted squares. Carefully tracing all possible packings, we can check that both of the top and bottom dotted squares cannot be covered at the same time, and both of the right and left dotted squares cannot be covered at the same time. Therefore, we only have four valid packings shown in Figure 12.

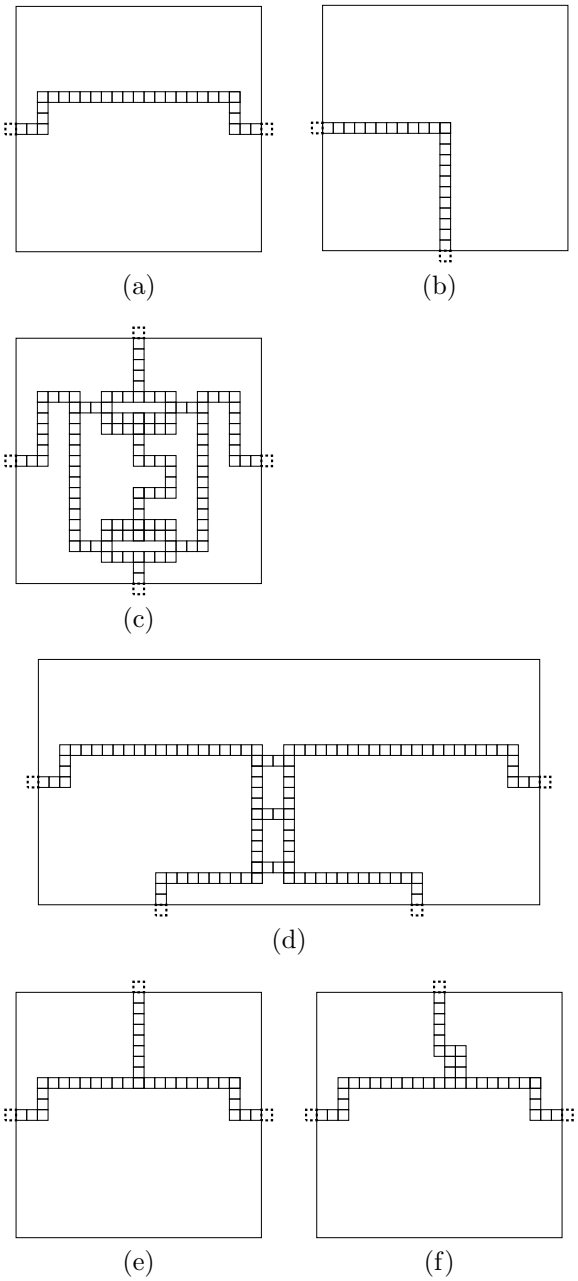


Figure 11: Gadgets for the reduction to 3I-PACKING.

(d) Duplicator gadget. At the center of the gadget in Figure 11(d), there are three bridges of length 4 joining the right and left parts. Each of the bridges can be packed in two possible ways by putting a tromino left or right. As in the cross gadget, since the number of the squares in the gadget is 106, we can place two squares at the positions of the dotted squares. use two squares at the dotted squares. The corners close to dotted squares force the configuration of trominoes. Contrary to the case of the cross gadget, these conditions allow us to pack I-trominoes in only two ways as shown in Figure 13.

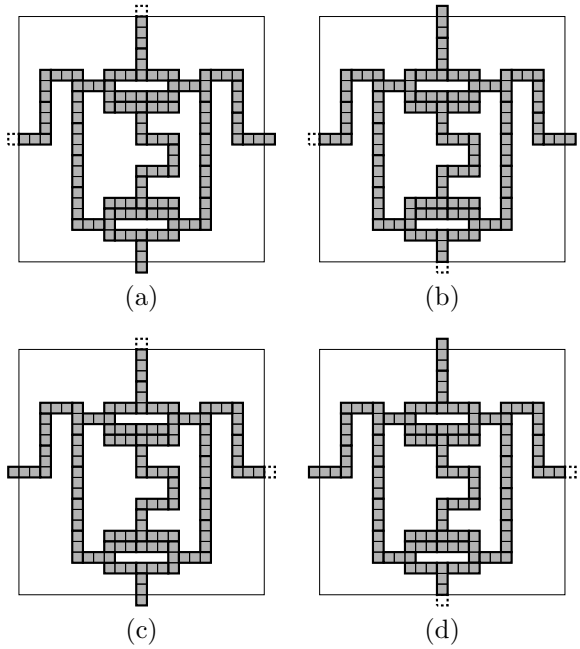


Figure 12: Four ways of packings for a cross gadget.

(f) Negated-clause gadget. In the negated-clause gadget given in Figure 11(f), the center rectangle of size 2×4 is critical. We can pack two trominoes in four ways here, but if they take different heights, we cannot pack trominoes in the horizontal segment between the left and right corners. Therefore, there are only two ways to pack the trominoes into the rectangle. This gives us three valid packings as illustrated in Figure 14.

The two ways of packings for a line gadget correspond to the orientations of the edge. A clause gadget has three ways of packings such that exactly one of the three dotted unit squares gives an inbound orientation. In contrast, a negated-clause gadget has three ways of packings such that exactly one of the three dotted unit squares gives an outbound orientation. Therefore, valid packings for P in 3I-PACKING have one-to-one correspondence with valid orientations of G_φ in 1-in-3 GO. We thus have the following theorem.

Theorem 3 3I-PACKING *is NP-complete, #P-complete and ASP-complete.*

3 Concluding Remarks

We have studied the computational complexity of packing problems with L-trominoes and I-trominoes. Our results fill the complexity gap between packing dominoes (i.e., 1×2 rectangles) and packing 2×2 squares.

We can extend our results to the following problems.

(1) Tromino-Packing (i.e., a mixture variant of packing trominoes): Given an integer k and a polyomino P,

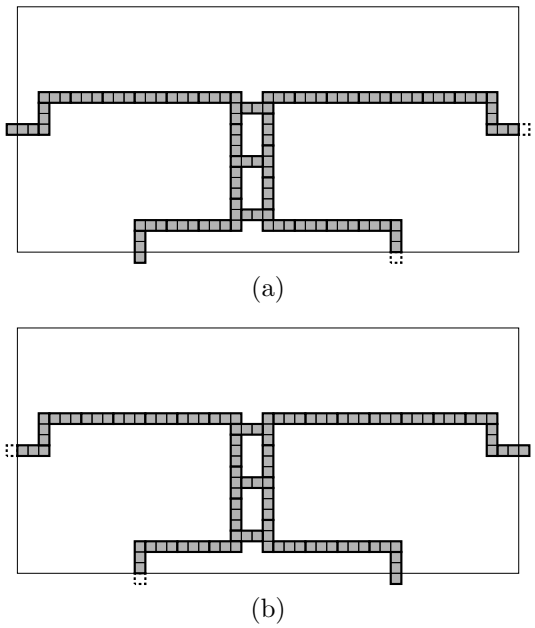


Figure 13: Two ways of packings for a duplicator gadget.

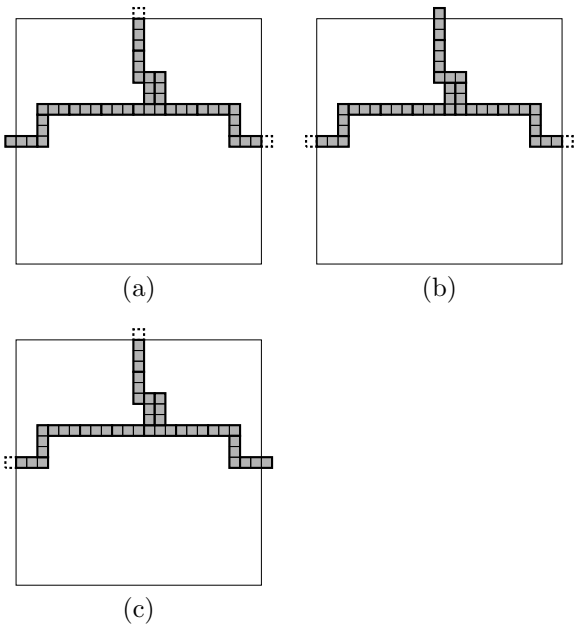


Figure 14: Three ways of packings for a negated-clause gadget.

determine whether k trominoes can be placed into P without overlap. Note that we can use both L-trominoes and I-trominoes in this variant. By constructing new gadgets, similar arguments establish that this variant is NP-complete, #P-complete and ASP-complete.

(2) 3L-Cover, 3I-Cover and Tromino-Cover: Given an integer k and a set S of points in the plane, determine whether k trominoes can cover all points in S. In 3L-COVER and 3I-COVER, we are only allowed to use identical L-trominoes and I-trominoes, respectively. In TROMINO-COVER, we are allowed to use both shapes. Trominoes are allowed to mutually overlap each other.

(3) 3L-Unique-Cover, 3I-Unique-Cover and Tromino-Unique-Cover: Given an integer k and a set S of points in the plane, determine whether k trominoes can uniquely cover all points in S, that is, no overlap of trominoes is allowed.

By converting the gadgets in this paper to the positions of points (and regarding the set of points as the set S), we can easily see the correspondence between the packing and the covering problems. Therefore, all these variants in (2) and (3) are NP-complete, #P-complete and ASP-complete.

References

[1] N. Creignou and M. Hermann, Complexity of generalized satisfiability counting problems. *Information and Computation*, 125, pp. 1–12, 1996.

[2] D. El-Khechen, M. Dulieu, J. Iacono and N. van Omme. Packing 2×2 unit squares into grid polygons is NP-complete. *Proc. 21st Canadian Conf. on Comput. Geom.*, pp. 33–36, 2009.

[3] R. J. Fowler, M. Paterson and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inf. Process. Lett.*, 12(3):133–137, 1981.

[4] M. Gardner. The hypnotic fascination of sliding-block puzzles. *Scientific American*, 210:122–130, 1964. (Also Chapter 7 of *Martin Gardner's Sixth Book of Mathematical Diversions*, University of Chicago Press, Chicago, 1984.)

[5] S. Golomb. *Polyominoes* (2nd edition). Princeton University Press, 1994.

[6] R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.

[7] R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation.* A K Peters Ltd., 2009.

[8] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Computing*, 2(4):225–231, 1973.

[9] T. J. Schaefer. The complexity of satisfiability problems. *Proc. 10th Ann. ACM Symp. on Theory of Computing*, pp. 216–226, 1978.

[10] T. Yato and T. Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(5):1052–1060, 2003.