Theoretical Computer Science 983 (2024) 114292

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Computational complexity of jumping block puzzles ☆,☆☆

Masaaki Kanzaki [a], Yota Otachi [b], Giovanni Viglietta [c], Ryuhei Uehara [d,*]

[a] *Third System Department, SI Division, JMAC SOFT Co., Ltd. Johnai, Nagaoka, Niigata 940-0061, Japan*
[b] *Department of Mathematical Informatics, Graduate School of Informatics, Nagoya University, Furocho, Chikusa-ku, Nagoya, Aichi 464-8601, Japan*
[c] *Department of Computer Science and Engineering, University of Aizu, Aizu Wakamatsu, Fukushima 965-8580, Japan*
[d] *School of Information Science, Japan Advanced Institute of Science and Technology, Asahidai, Nomi, Ishikawa 923-1292, Japan*

ARTICLE INFO

Communicated by A. Fink

Keywords:
Combinatorial reconfiguration
Computational complexity
Jumping block puzzle
Sliding block puzzle
Token jumping

ABSTRACT

In the context of computational complexity of puzzles, the sliding block puzzles play an important role. Depending on the rules and set of pieces, the sliding block puzzles can be polynomial-time solvable, NP-complete, or PSPACE-complete. On the other hand, a relatively new notion of jumping block puzzles has been proposed in the puzzle community. This is a counterpart to the token jumping model of the combinatorial reconfiguration problems in the context of block puzzles. We investigate some variants of jumping block puzzles, which are based on actual puzzles, and a natural model from the viewpoint of combinatorial reconfiguration, and determine their computational complexities. More precisely, we investigate two generalizations of two actual puzzles which are called Flip Over puzzles and Flying Block puzzles and one natural model of jumping block puzzles from the viewpoint of combinatorial reconfiguration. We prove that they are PSPACE-complete in general. We also prove the NP-completeness of these puzzles in some restricted cases, and we give polynomial-time algorithms for some restricted cases.

1. Introduction

Recently, the *reconfiguration problems* attracted the attention of theoretical computer science. These problems arise when we need to find a step-by-step transformation between two feasible solutions of a problem such that all intermediate results are also feasible and each step abides by a fixed reconfiguration rule, that is, an adjacency relation defined on feasible solutions of the original problem. Reconfiguration problems have been studied extensively for several well-known problems, including INDEPENDENT SET [13,16], SATISFIABILITY [9,18], SET COVER, CLIQUE, MATCHING [13], VERTEX COLORING [2–4], and SHORTEST PATH [15].

In the reconfiguration problems that consist in transforming a vertex subset (e.g., an independent set), there are three natural reconfiguration rules called "token sliding," "token jumping," and "token addition and removal" [16]. In these rules, a vertex subset is represented by the set of *tokens* placed on each of the vertices in the set. (We cannot put two or more tokens on a vertex.) In the token sliding model, we can slide a token on a vertex to another along an edge joining these vertices. On the other hand, a token can jump to any other vertex in the token jumping model. (In the token addition and removal model, we can remove and add some tokens in a specific range. That is, the total number of tokens can be changed in a given range.)

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

☆☆ A preliminary version appeared in the proceedings of the 27th International Computing and Combinatorics Conference (COCOON 2021), Lecture Notes in Computer Science, Vol. 13025, pp. 655–667, 2021.

* Corresponding author.
E-mail addresses: kanzaki11114@gmail.com (M. Kanzaki), otachi@nagoya-u.jp (Y. Otachi), viglietta@gmail.com (G. Viglietta), uehara@jaist.ac.jp (R. Uehara).

https://doi.org/10.1016/j.tcs.2023.114292
Received 9 February 2022; Received in revised form 22 September 2023; Accepted 31 October 2023

Available online 10 November 2023

Fig. 1. A typical sliding block puzzle and the 15-puzzle.

Fig. 2. An example of the Flip Over puzzle. Each piece has its front side and back side, and the goal is to make them back-side up.

In the puzzle community, several puzzles have been invented which can be seen as realizations of some reconfiguration problems. Among them, the family of *sliding block puzzles* (Fig. 1) has been playing an important role bridging recreational mathematics and theoretical computer science. A classic puzzle is called the Dad puzzle; it consists of rectangle pieces in a rectangle frame, and the goal is to slide a specific piece (e.g., the largest one) to the goal position. The Dad puzzle was invented in 1909 and the computational complexity of this puzzle was open since Martin Gardner mentioned it in 1960s [6]. After almost four decades, Hearn and Demaine proved that the puzzle is PSPACE-complete in general (a comprehensive survey can be found in [10]). When all pieces are unit squares, we obtain another famous puzzle called the *15-puzzle*; in this puzzle, we slide each unit square using an empty area and arrange the pieces in order. From the viewpoint of combinatorial reconfiguration, this puzzle has remarkable properties in the general form of the (n^2-1)-puzzle. For given initial and final arrangements of pieces, the decision problem asks if we can transform from the initial arrangement to the final arrangement. Then the decision problem can be solved in linear time by checking their parities. For a yes-instance, while a feasible solution can be found in $O(n^3)$ time, finding a shortest solution is NP-complete (see [20,5]).

To see how the computational complexity of a reconfiguration problem depends on its reconfiguration rule, it is natural to consider the "jumping block" variant as a counterpart of sliding block puzzles. In fact, some realizations of *jumping block puzzles* have been invented by the puzzle community. As far as the authors know, "Flying Block" was the first jumping block puzzle, which was designed by Dries de Clercq and popularized at the International Puzzle Party by Dirk Weber in 2008.[1] The Flying Block consists of four polyominoes (see Section 2 or [8] for the notion of polyominoes) within a rectangular frame. The goal is similar to the Dad puzzle: moving a specific piece to the goal position. In one move, we first pick up a piece, rotate it if necessary, and then put it back into the frame. A key feature is that each piece has a small tab to pick it up. Because of this feature, flipping a piece before putting it back into the frame is not allowed.

Later, Fujio Adachi invented another jumping block puzzle which is called *Flip Over* (or Turn Over). It was invented in 2016 and popularized at the International Puzzle Party by Naoyuki Iwase in 2019.[2] This puzzle consists of four polyominoes in an orthogonal frame, which is not a rectangle. The operation is a bit different from Flying Block; you can *flip* the piece in addition to the rotation if you like. Each piece has its front side and back side (distinguished by their colors). The goal of this puzzle is to make all pieces back-side up. A simple example is given in Fig. 2.

The model of Flying Block is natural as puzzle because the goal is the same as the Dad puzzle. On the other hand, the model of Flip Over forces us to move all front-sided pieces at least once, and hence the number of front-sided pieces is a trivial lower bound of the number of moves of the puzzle.

As far as the authors know, the jumping block puzzles have never been investigated in the context of the reconfiguration problems. In this paper, we first investigate the computational complexities of the jumping block puzzles under the models given by actual puzzles. Namely, we consider the computational complexities of the jumping block puzzles under the models of Flip Over and Flying Block.

From the viewpoint of reconfiguration problems, a natural variant of the jumping block puzzles is as follows. We are given a set of polyominoes, an orthogonal frame, and the initial and the goal arrangements of the polyominoes in the frame. Then the problem asks whether we can move the initial arrangement to the goal arrangement by a sequence of polyomino jumps. We also investigate the computational complexity of this puzzle.

Due to a technical reason, we will show our results on the flip over puzzles in Section 3, the flying block puzzles in Section 4, and the reconfiguration problem in Section 5. For these variants of jumping block puzzles, the results in this paper can be summarized in Tables 1, 2, and 3. (In these tables, "#moves" means the number of moves of each piece, "frame" means the shape of the frame (and

[1] You can find "Flying Block", "Flying Block II", and "Flying Block III" at http://www.robspuzzlepage.com/sliding.htm (accessed in December 2021).
[2] Some commercial products can be found at http://www.puzzlein.com/ (in Japanese; accessed in December 2021).

Table 1
Results on the flip over puzzles.

#moves	frame	symmetric pieces	asymmetric pieces	#vacs	complexity	reference
Any	rect of constant height	rects of size $1 \times k$	4-omino	1	PSPACE-complete	Theorem 2
Any	rect of height 2	rects of height 1	polyomino of height 2	any	NP-hard	Theorem 3
≤ 1	rect	rects of size $1 \times k$	4-omino	any	NP-complete	Theorem 4
≤ 1	rect	rects of size $1 \times k$	4-ominoes	1	NP-complete	Theorem 5
≤ 1	rect of height 2	rects of height 1 and polyomino of height 2	polyomino	any	NP-complete	Theorem 6

Table 2
Results on the flying block puzzles. (We need no asymmetric piece in this case.)

#moves	frame	symmetric pieces	#vacs	complexity	reference
Any	rect of constant height	rects of size $1 \times k$	1	PSPACE-complete	Theorem 7
Any	rect of height 2	rects of height 1 and polyomino of height 2	any	NP-hard	Theorem 8
≤ 1	rect of height 2	rects of height 1 and polyomino of height 2	any	NP-complete	Theorem 8
≤ 1	rect	rects of size $1 \times k$	any	NP-complete	Theorem 9
≤ 1	rect of height 1	rects of height 1	any	weakly NP-complete	Theorem 10

Table 3
Results on the reconfiguration problem on the jumping block puzzles.

#moves	frame	symmetric pieces	asymmetric pieces	#vacs	complexity	reference
Any	rect of constant height	rects of size $1 \times k$	0	1	PSPACE-complete	Theorem 15
Any	rect of height 2	rects of height 1	polyomino	any	NP-hard	Theorem 16
Any	rect of height 1	any	any	any	poly-time	Observation 12
Any or ≤ 1	any	rects of size 1×2	0	any	poly-time	Observation 13 & Theorem 14

"rect" means a simple rectangle without holes), k is an integer with $1 \leq k \leq 3$, "(asymmetric/symmetric) pieces" means the pieces of the puzzle, "#vacs" means the number of vacant squares.) One considerable NP-complete problem for the reconfiguration problem not in Table 3 is the following (Theorem 17): For a given instance of the reconfiguration problem on the jumping block puzzle on a rectangular frame of height 1, it is NP-complete to decide whether there is a jumping sequence of length at most t from the initial feasible packing to the goal feasible packing for a given integer t.

2. Preliminaries

A *polyomino* is a polygon formed by joining one or more unit squares edge to edge. A polyomino is also called a *k-omino* if it consists of k unit squares. When $k = 1, 2, 3, 4$, we sometimes call it *monomino*, *domino*, *tromino*, and *tetromino*, respectively. Some polyominoes have their own names; e.g., we have two different trominoes, which are called *I-tromino* formed by three consecutive unit squares, and *L-tromino* formed by L-shape by three unit squares. We also use *L-tetromino*, which is an L-shape tetromino, as a minimum asymmetric polyomino with respect to the line symmetry. An instance of the *jumping block puzzle* consists of a set $\mathcal{P}$ of polyominoes $P_1, P_2, \ldots, P_n$ and a polyomino F. Each polyomino $P_i \in \mathcal{P}$ is called a *piece*, and F is called a *frame*. Each piece P_i has its *front side* and *back side* (in the figures in this paper, a bright color and a dark color indicate front side and back side, respectively).

A *feasible packing* of $\mathcal{P}$ to F is an arrangement of all pieces of $\mathcal{P}$ into F such that each piece is placed in F, no two pieces overlap (except at edges and vertices), each vertex of each piece of $\mathcal{P}$ is placed on a grid point in F, and each edge of each piece of $\mathcal{P}$ is parallel to some edge of F.

For a feasible packing of $\mathcal{P}$ to F, a *flip* of P_i is an operation that consists of the following steps: (1) pick up P_i from F, (2) translate, rotate, and flip P_i if necessary, and (3) put P_i back into F so that the resulting arrangement is a feasible packing. On the other hand, a *fly* of P_i is an operation that consists of the following steps: (1) pick up P_i from F, (2) translate and rotate P_i if necessary, and (3) put P_i back into F so that the resulting arrangement is a feasible packing.

Then the *flip over puzzle* problem is formalized as follows:

FLIP OVER PUZZLE
Input: A set $\mathcal{P}$ of polyominoes in an orthogonal frame F.
Operation: Pick up a piece, translate, rotate, and flip it if desired, and put it back into F.
Goal: Make every piece back-side up in F.

Precisely, for a given feasible packing X of $\mathcal{P}$ to F, the *flip over puzzle* asks whether X can be reconfigured by a sequence of flips to a feasible packing in which all pieces are back-side up in F. We first observe that each piece can be flipped in place if it is line-symmetric. That is, when all polyominoes are line-symmetric, it is a yes-instance and all pieces can be flipped in a trivial way. In contrast, we will show that the existence of one asymmetric piece changes the computational complexity of this puzzle.

On the other hand, the *flying block puzzle* problem is formalized as follows:

FLYING BLOCK PUZZLE
Input: A set $\mathcal{P}$ of polyominoes in an orthogonal frame F, a specific piece P in $\mathcal{P}$, and a goal position of P in F.
Operation: Pick up a piece, translate and rotate it if desired, and put it back into F.
Goal: Move P to the goal position.

In contrast to the flip over puzzle, in the flying block puzzle, the input consists of three elements: a feasible packing of $\mathcal{P}$ to F, a specific piece P, and the goal position of P in F. That is, the *flying block puzzle* asks if we can move the specific piece P to the goal position starting from the feasible packing by a sequence of flies. We also note that we cannot flip pieces in the flying block puzzle (since each piece has a tab to pick in real puzzles).

In the context of reconfiguration problems, we can introduce the *C2C*[3] *jumping block puzzle* as follows:

C2C JUMPING BLOCK PUZZLE
Input: A set $\mathcal{P}$ of polyominoes, an orthogonal frame F, an initial feasible packing $\mathcal{S}$ in F, and a goal feasible packing $\mathcal{T}$ in F.
Operation: Pick up a piece, translate, rotate, and flip it if desired, and put it back into F.
Goal: Determine if we can arrange the pieces from $\mathcal{S}$ to $\mathcal{T}$ by a sequence of operations.

In C2C jumping block puzzle, we consider the case that each piece has no label. That is, congruent pieces are not distinguished in $\mathcal{S}$ and $\mathcal{T}$. Therefore, we have to solve a matching problem between $\mathcal{S}$ and $\mathcal{T}$; see Fig. 17 for example. We can consider a natural labeled (or colored) variant; see Concluding remarks in Section 6 for further details.

In order to determine the computational complexity of the problems defined above, we use the notion of the *constraint logic*, which was introduced by Hearn and Demaine [10,11]. A *constraint graph* $G = (V, E, w)$ is an edge-weighted undirected 3-regular graph such that (1) each edge $e \in E$ has weight 1 or 2, (we sometimes describe the weight 1 by *red* and 2 by *blue*) and (2) each vertex is either an *AND vertex* or an *OR vertex* such that (2a) an AND vertex is incident to two red edges and one blue edge, and (2b) an OR vertex is incident to three blue edges. A *configuration* of a constraint graph is an orientation of the edges in the graph. A configuration is *legal* if the total weight of the edges pointing to each vertex is at least 2. The *NCL* problem on the constraint logic is defined as follows.

NCL
Input: A constraint graph $G = (V, E, w)$, a legal configuration C_0 for G, and an edge $e_t \in E$.
Question: Is there a sequence of legal configurations $(C_0, C_1, \ldots)$ such that (1) C_i is obtained by reversing the direction of a single edge in C_{i-1} with $i \geq 1$, and (2) the last configuration is obtained by reversing the direction of e_t?

The *Bounded NCL* problem is a variant of the NCL problem with the additional restriction that every edge can be reversed at most once. For these two problems, the following theorem is known:

Theorem 1 *([10]). (1) NCL is PSPACE-complete even on planar graphs and (2) Bounded NCL is NP-complete even on planar graphs.*

3. Complexity of flip over puzzles

In this section, we focus on the flip over puzzles. We first show that this puzzle is PSPACE-complete even on quite restricted cases. Next, we show that it is NP-hard even if the frame F is a rectangle of height 2. Lastly, we show that it is NP-complete if the number of flips of each piece is $O(1)$.

3.1. PSPACE-completeness in general

The first result for the flip over puzzle is the following theorem:

Theorem 2. *The flip over puzzle is PSPACE-complete even with all the following conditions*[4]*:*

(1) *the frame is a rectangle of constant height without holes,*
(2) *every line-symmetric piece is a rectangle of size* $1 \times k$ *with* $1 \leq k \leq 3$*,*
(3) *there is only one asymmetric piece of size* 4*, and*
(4) *there is only one vacant unit square.*

We note that all polyominoes of size at most 3 (i.e., monomino, domino, I-tromino, and L-tromino) are line-symmetric. Therefore, Theorem 2 is tight in the sense that it contains only one minimum asymmetric piece.

[3] C2C stands for Configuration-To-Configuration.

[4] Hereafter, in each theorem, the condition (0) specifies the condition on the number of movements for each piece, the condition (1) indicates the size of the frame, the condition (2) is the condition of most line-symmetric pieces, the condition (3) is the condition of asymmetric pieces, and the condition (4) is the condition of vacant spaces.

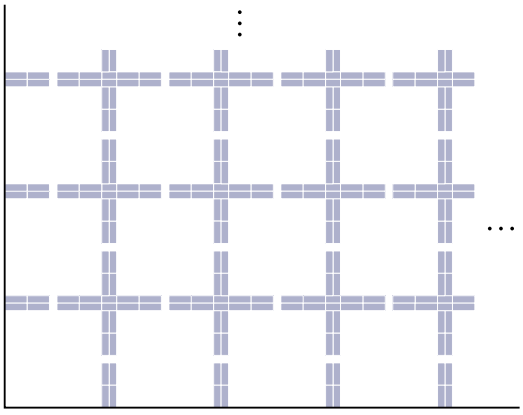

Fig. 3. The cell border, which is a framework of embedding gadgets.

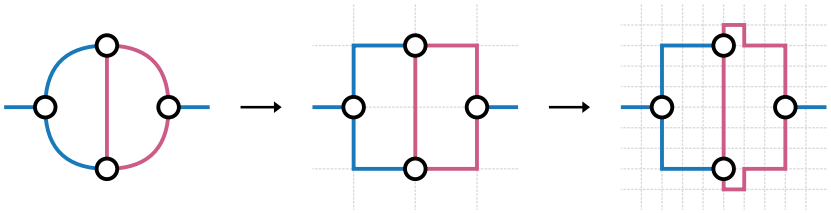

Fig. 4. A grid drawing of the graph in Step (1). Two red edges incident to an AND vertex are collinear. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Proof. We give a polynomial-time reduction from the NCL problem to the flip over puzzle. Let $G = (V, E, w)$, e_t, and C_0 be the input of the NCL problem. That is, G is an edge-weighted planar graph, C_0 is a legal configuration of G and e_t is an edge in E. It is known that NCL is still PSPACE-complete even if the input NCL graph is planar and has constant bandwidth and it is given with a rectilinear embedding[5] of constant height (see [22, Thm. 11] for the details). Thus we assume that the rectlinear embedding is of constant height without loss of generality.

The framework of the reduction is similar to the reduction of the NCL problem to the sliding block puzzle shown in [10, Sec. 9.3]: The frame F is a big rectangle, and it is filled with a regular grid of gate gadgets, within a "cell border" construction. The internal construction of the gates is such that none of the cell-border blocks may move, thus providing overall integrity to the configuration. In our reduction, the cell border has width 2 (Fig. 3), and each gadget in a cell will be designed of size 13×13. We first prepare a sufficiently large cell border for embedding G. Then we construct an instance of the flip over puzzle in three steps.

Step (1) We first compute a rectilinear embedding of G, which can be done in $O(n^2)$ time [17]. To simplify the construction, we bend some edges (by refining the grid) to make two red edges incident to an AND vertex collinear (Fig. 4). We then define the frame F so that each *cell* of size 13×13 contains one of an AND vertex, an OR vertex, a unit straight segment of an edge (*wire*), a unit turning segment of an edge (*corner*), and an empty cell.

Step (2) The vertices, unit segments of edges, and empty cells are replaced by the gadgets shown in Fig. 5. We note that these gadgets are designed in a square of size 13×13 and green trominoes are shared by two adjacent gadgets. As we will see later, we have only one vacant square in the resulting arrangements. Thus each of the tetrominos in our gadgets has only at most two options of its location. Under this assumption, we can confirm that each of AND and OR gadgets have five and seven feasible arrangements by jumping polyomino as shown in Fig. 6. For a given constraint graph G with its legal configuration C_0 for G, we choose one of these options according to C_0. Since it is a legal configuration for G, all green trominoes are aligned with consistency. We embed these gadgets into the frame F to represent the graph, with margins of gap 2. If a vacant space of size 1×2 formed by adjacent empty cells remains at some place between two gadgets, we fill it by a domino. We here note that, in our gadgets, the orientation of an edge is represented by green trominoes in an opposite way. As shown in Figs. 6 and 9, when an edge is incoming to the corresponding vertex, then the corresponding green tromino is "out" of the gadget, and it is "in" the gadget when the edge is outgoing from the vertex.

[5] A *rectilinear embedding* of a graph G is an embedding of G to a square grid such that (1) each vertex is placed on a grid point, (2) each edge joins two grid points by a sequence of straight orthogonal line segments of integer length with turning points on grid points, (3) no pair of vertices and turning points shares the same grid point. See [17] for the further details.

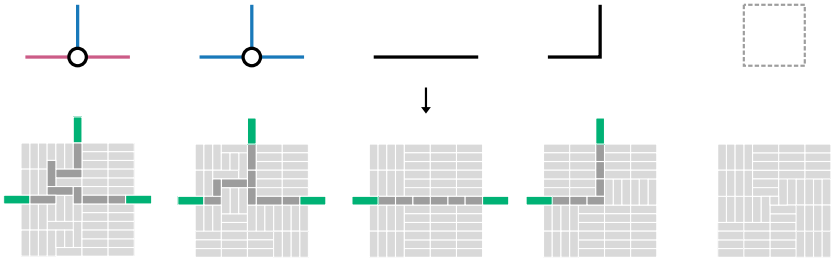

Fig. 5. Gadgets of size 13×13 for an AND vertex, an OR vertex, a wire, a corner, and an empty cell.

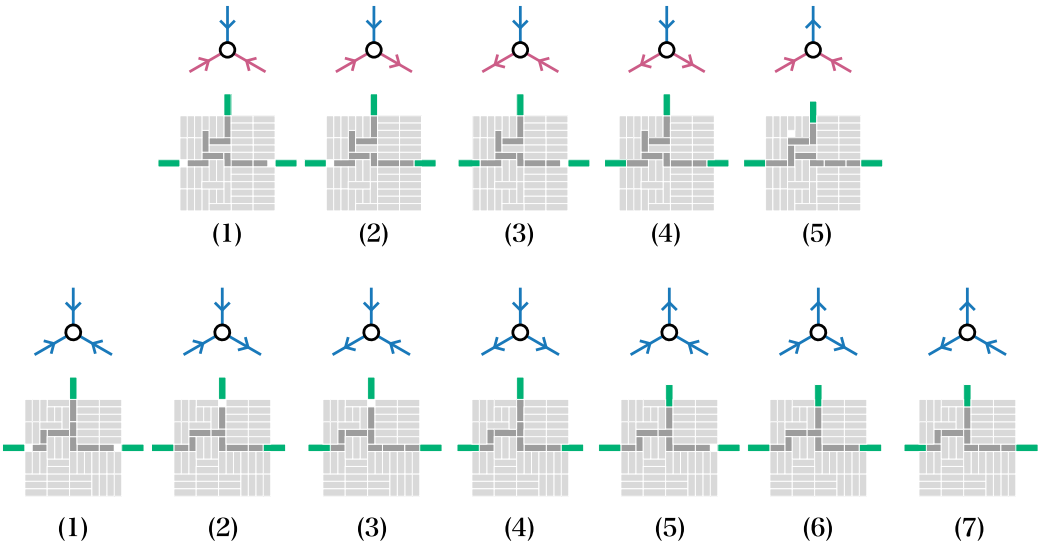

Fig. 6. Possible arrangements of gadgets.

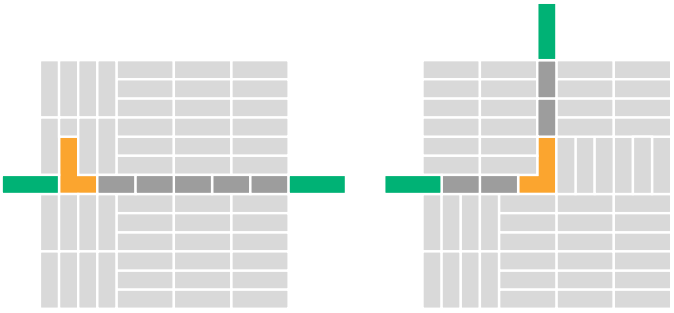

Fig. 7. Two edge gadgets for e_t for embedding the L-tetromino.

Step (3) Pick up any unit segment of the edge e_t, and embed the left gadget in Fig. 7 if it is a straight line, and the right gadget in Fig. 7 if it is a turn. We note that each of them contains the unique asymmetric L-tetromino and this L-tetromino can be flipped when the leftmost I-tromino moves to the left. Finally we put monominoes (polyominoes of unit size) to fill all vacant unit squares except one. (This one vacant unit square can be arbitrary.) This is the end of our reduction. An example of the construction is given in Fig. 8.

It is clear that this reduction can be done in polynomial time, we can construct an instance of the flip over puzzle with consistency for a given constraint graph G with its legal configuration C_0 for G, the instance constructed from G satisfies the conditions (1) to (4) in Theorem 2 since each gadget has a constant size, and the flip over puzzle belongs to PSPACE. We now show that G is a yes instance of the NCL problem if and only if the instance of the flip over puzzle constructed from G is a yes instance.

The basic idea is the same as one used in [10, Sec. 9.3]; when an edge e changes the direction and points to a vertex v, the corresponding *I-tromino* (of size 1×3 and in green in the figure) moves out one unit. As shown in Fig. 9, we designed two gadgets so that (1) when both of the I-trominoes corresponding to the red edges moved out from an AND gadget, the I-tromino corresponding

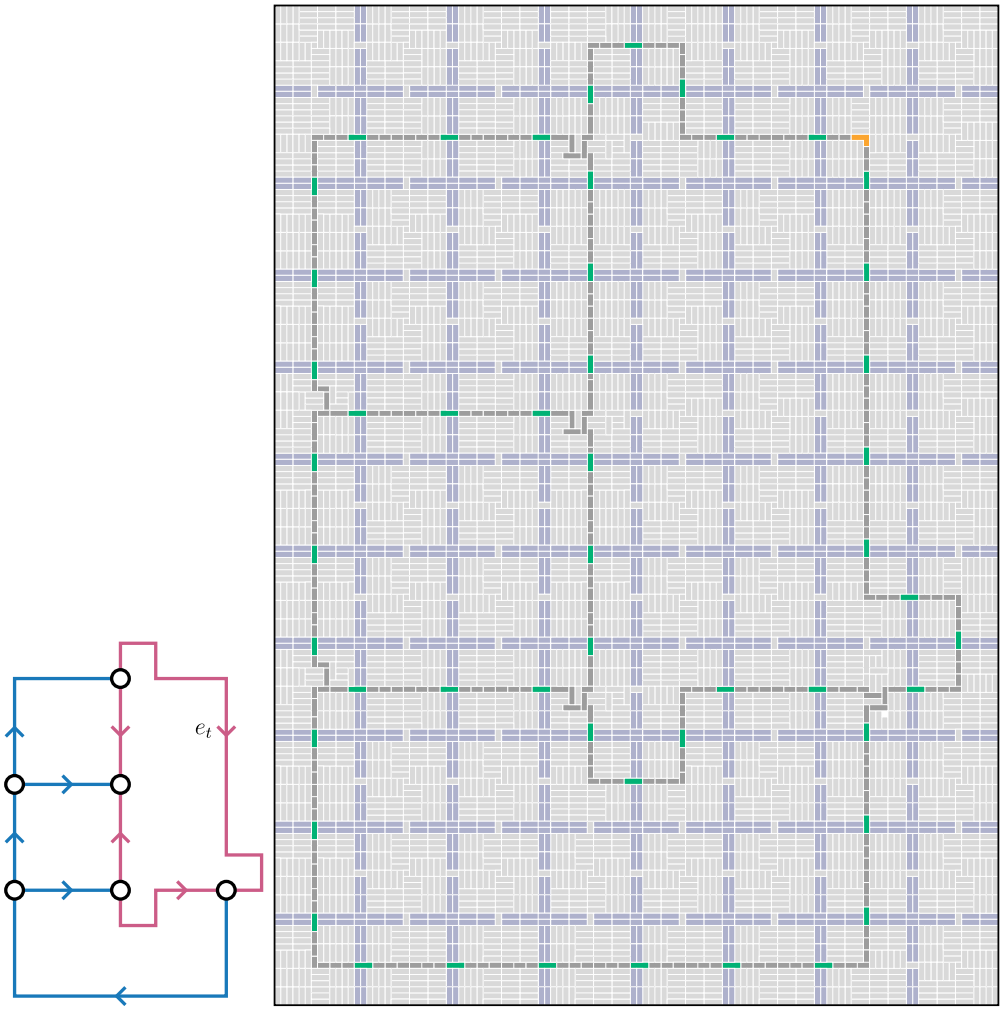

Fig. 8. Completed reduction after Step (3). The unique L-shape is placed at the top-right corner, and two of three vacant spaces in two AND gadgets are filled by a monomino, and the other is left as the unique vacant space.

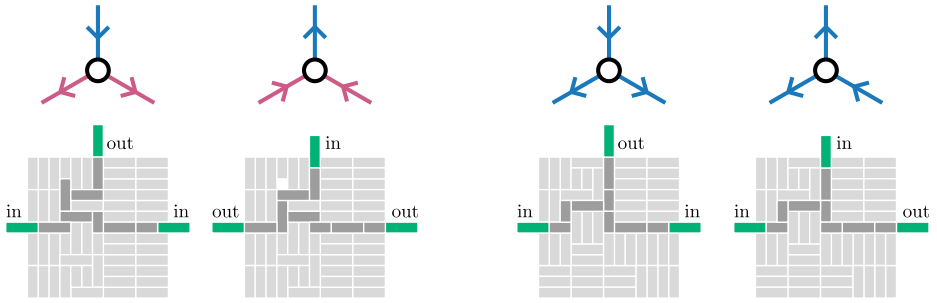

Fig. 9. Movements of gadgets.

to the blue edge of the AND gadget can move in (as shown in Fig. 6(5)), and (2) when at least one of the I-trominoes corresponding to the blue edges moved out from an OR gadget, the I-tromino corresponding to the blue edge of the OR gadget can move in (as shown in Fig. 6(2)-(7)).

In [10, Sec. 5.2.3], they introduce a notion of *protected OR*, and show that the NCL problem is still PSPACE-complete even if all OR gadgets are protected OR. In our context, it implies that we can assume that we only have the patterns in Fig. 6(4)(6)(7) for given C_0, and patterns in Fig. 6(2)(3)(5) never happen in our reconfiguration. We also observe that we cannot have Fig. 6(1) in our reconfiguration since it violates the condition of a protected OR and hence C_0 does not contain this configuration. Therefore, in a

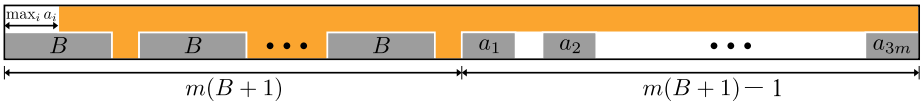

Fig. 10. Reduction for NP-hardness.

feasible reconfiguration of the instance of the NCL problem, all OR gadgets in the constructed instance of the flip over puzzle contain no unit square.

We then observe that we have only one vacant unit square in the instance of the flip over puzzle constructed here. However, by jumping monominoes, we can move the vacant unit square to anywhere we need as long as it is occupied by a monomino.

Based on the arguments above, an AND gadget in the flip over puzzle can simulate an AND vertex in the NCL graph and an OR gadget in the flip over puzzle can simulate a protected OR vertex in the NCL graph. The correctness for a wire gadget, a corner gadget, and an empty cell is trivial. Thus, using the same arguments in [10, Sec. 9.3], we can show that the flip over puzzle simulates the movements of the NCL problem and vice versa. Therefore, if we can eventually reverse e_t in the instance of the NCL problem, we can make a vacant unit square next to the unique L-tetromino in the instance of the flip over puzzle, and then we can make it back-side up using the vacant unit square next to the L-tetromino. Thus we have the theorem. □

3.2. NP-hardness on a frame of height 2

In Theorem 2, we proved that the problem is PSPACE-complete even if the frame is a rectangle of constant height. On the other hand, if the frame is a rectangle of size $1 \times m$ (or height 1), the problem is trivial: each piece should be a rectangle of height 1 and hence it can be flipped to back side in-place in one step. Interestingly, the problem is intractable when the frame has height 2.

Theorem 3. *The flip over puzzle is NP-hard even with all the following conditions:*

(1) *the frame is a rectangle of height* 2*,*
(2) *every line-symmetric piece is a rectangle of height* 1*, and*
(3) *there is only one asymmetric piece.*

Proof. We reduce the following 3-PARTITION problem to our problem.

3-PARTITION
Input: Positive integers $a_1, a_2, a_3, \dots, a_{3m}$ such that $\sum_{i=1}^{3m} a_i = mB$ for some positive integer B and $B/4 < a_i < B/2$ for $1 \le i \le 3m$.
Output: Determine whether we can partition $\{1, 2, \dots, 3m\}$ into m subsets $A_1, A_2, \dots, A_m$ so that $\sum_{i \in A_j} a_i = B$ for $1 \le j \le m$.

It is well known that the 3-PARTITION problem is strongly NP-complete [7]. Without loss of generality, we assume that $m < B/2$ (otherwise, we multiply each a_i by $2m$). For a given instance $\langle a_1, a_2, \dots, a_{3m} \rangle$ of 3-PARTITION, we construct an instance of the jumping block puzzle as follows. It consists of a frame F and a set of pieces $\mathcal{P} = \{P_1, P_2, \dots, P_{4m+1}\}$. The frame F is a $2 \times (2m(B+1)-1)$ rectangle. For $1 \le i \le 3m$, the piece P_i is a $1 \times a_i$ rectangle. The pieces $P_{3m+1}, P_{3m+2}, \dots P_{4m}$ are $1 \times B$ rectangles. The unique asymmetric piece P_{4m+1} is drawn in Fig. 10. It almost covers the whole frame F except (1) one rectangle of size $1 \times \max_i a_i$, (2) m rectangles of size $1 \times B$ in the left side, and (3) one rectangle of size $1 \times (m(B+1)-1)$ in the right side. The initial feasible packing of $\mathcal{P}$ to F is also described in Fig. 10. All pieces are placed front-side up. The asymmetric piece P_{4m+1} is placed in F as shown in Fig. 10, and m rectangles of size $1 \times B$ are occupied by m pieces $P_{3m+1}, P_{3m+2}, \dots, P_{4m}$ of size $1 \times B$. The other $3m$ polyominoes $P_1, P_2, \dots, P_{3m}$ are put in the rectangle of size $1 \times (m(B+1)-1)$ in the right side of P_{4m+1} in arbitrary order. We assume that the piece P_{3m+i} occupies the i-th vacant space of size $1 \times B$ from the center of P_{4m+1}.

Now, we show that all pieces in $\mathcal{P}$ can be flipped back-side up if and only if the original instance of 3-PARTITION is a yes instance. Since all pieces in $\mathcal{P}$ are rectangles except P_{4m+1}, it is sufficient to ask whether P_{4m+1} can be flipped.

We first assume that all pieces in $\mathcal{P}$ can be flipped. It is easy to see that P_{4m+1} can be flipped only in the horizontal direction. We first observe that no piece of size $1 \times B$ can be moved to any other place before flipping P_{4m+1} as follows: First, the vacant space at the top-left corner is too small for it. Since $B/4 < a_i < B/2$ and $B/4 < a_i \le \max_i a_i < B/2$, we can move at most one piece of size $1 \times a_i$ to the top-left corner. Even if we move the longest piece of size $1 \times \max_i a_i$ to the top-left corner, by the assumption $m < B/2$, we can make a vacant space of width at most $\max_i a_i + (m-1) < B/2 + B/2 = B$ in the right-hand side, which is not enough to put a piece of size $1 \times B$. Now we consider the situation just before the asymmetric piece P_{4m+1} is flipped. The top-left vacant space of size $1 \times \max_i a_i$ flips to the top-right corner, and the corresponding top-right part of P_{4m+1} flips to the top-left corner. Therefore, there is no piece in this vacant space when we flip P_{4m+1}. In the bottom-left half in P_{4m+1}, we have m vacant spaces of size $1 \times B$, and each space is filled by a piece P_i of size $1 \times B$ for $3m+1 \le i \le 4m$. When the piece P_{4m+1} is flipped, all these m vacant spaces are flipped to the right half of the frame. On the other hand, the total length of the remaining pieces $P_1, P_2, \dots, P_{3m}$ is mB. They should be in the right half of the frame, and P_{4m+1} cannot overlap with any of them when it is flipped. Therefore, these pieces $P_1, P_2, \dots, P_{3m}$ should fit into m vacant spaces of size $1 \times B$ in P_{4m+1} when these vacant spaces come from the left side to the right side by the flip of P_{4m+1}. In order to do that, $P_1, P_2, \dots, P_{3m}$ should be partitioned into m groups so that each group exactly fills a vacant space of size $1 \times B$ in total. This partition clearly gives us a solution of the instance of the 3-PARTITION problem.

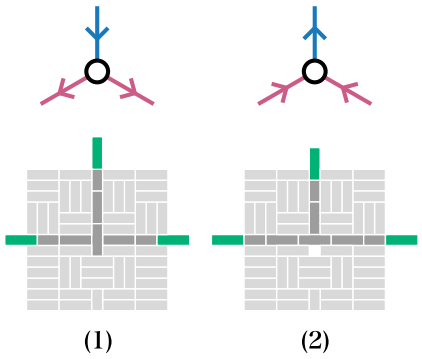

Fig. 11. AND gadget for Theorem 4.

We next assume that the instance of the 3-PARTITION problem is a yes instance. Then $\{1,\dots,3m\}$ has a partition into m 3-subsets $A_1, A_2, \dots, A_m$, where each A_i consists of i_1, i_2, i_3 with $a_{i_1} + a_{i_2} + a_{i_3} = B$. Using the left-top vacant space as a working space, we can sort the pieces $P_1, \dots, P_m$ into *any order* in the right-bottom space. (E.g., we can simulate the bubble sort.) Thus we sort $P_1, \dots, P_m$ in such a way that for $1 \le i \le m$, the positions $3i-2, 3i-1, 3i$ are occupied by the pieces $P_{i_1}, P_{i_2}, P_{i_3}$. Finally, we adjust the positions of the pieces to have a vacant space of unit size after every third piece. After this process, since we use a solution of 3-PARTITION, $P_1, \dots, P_{3m}$ form m rectangles of size $1 \times B$. Therefore we can flip P_{4m+1}. This completes the proof of Theorem 3. □

3.3. NP-completeness with constant flips

In the proof of Theorem 3, we reduced the 3-PARTITION problem to the flip over puzzle. We saw that the instances constructed there had polynomial-length yes-witnesses, however, we do not know whether the flip over puzzle on a frame of height 2 belongs to NP or not. In this section, we focus on the flip over puzzle with *constant flips*. In this model, we require that the number of flips for each piece does not exceed a fixed constant c. Then, if an instance of the puzzle has a solution, the solution can be represented by a sequence of moves, and the number of moves is bounded above by cn, where n is the number of pieces. Therefore, each yes-instance has a witness of polynomial length, which implies that puzzles with this restriction belong to NP. In this section, we show that the flip over puzzle is still NP-complete even if each piece can be flipped at most once with some additional restrictions. In the following, we only show the NP-hardness in the proofs since the problem clearly belongs to NP.

Theorem 4. *The flip over puzzle with constant flips is NP-complete even if it satisfies all the following conditions:*

(0) *each piece can be moved at most once,*
(1) *the frame is a rectangle without holes,*
(2) *every line-symmetric piece is a rectangle of size* $1 \times k$ *with* $1 \le k \le 3$*, and*
(3) *there is only one asymmetric piece of size* 4.

Proof. The claim can be proved by following the same strategy of the proof of Theorem 2 in Section 3.1. That is, we give a polynomial-time reduction from the bounded NCL problem. The construction from the NCL graph can be done in almost the same way as shown in Section 3.1 except Step (3) and the AND gadget. This time, we do not fill each of the vacant unit squares by monominoes in Step (3). The monominoes are the only pieces that may move twice or more if every edge can be reversed at most once. Thus the resultant puzzle satisfies the condition (0). On the other hand, if we remove all monominoes from the gadgets, the AND gadget in Fig. 5 can produce an empty domino, and hence it does not work as desired. To avoid it, we replace the AND gadget in Fig. 5 by one given in Fig. 11. We note that the AND gadget in Fig. 11 does not work in the proof of Theorem 2 since this gadget requires at least two vacant unit squares. On the other hand, when we can flip each piece at most once, it is easy to see that we can reconfigure the gadget from Fig. 11(1) to Fig. 11(2) by a sequence of flips that satisfies the condition (0). We then can observe that an instance of the bounded NCL problem is a yes instance if and only if the constructed instance of the flip over puzzle has a solution with the conditions (0), (1), (2), and (3). □

Theorem 5. *The flip over puzzle with constant flips is NP-complete even if it satisfies all the following conditions:*

(0) *each piece can be moved at most once,*
(1) *the frame is a rectangle without holes,*
(2) *every line-symmetric piece is a rectangle of size* $1 \times k$ *with* $1 \le k \le 3$*,*
(3) *all asymmetric pieces are of size* 4*, and*
(4) *there is only one vacant unit square.*

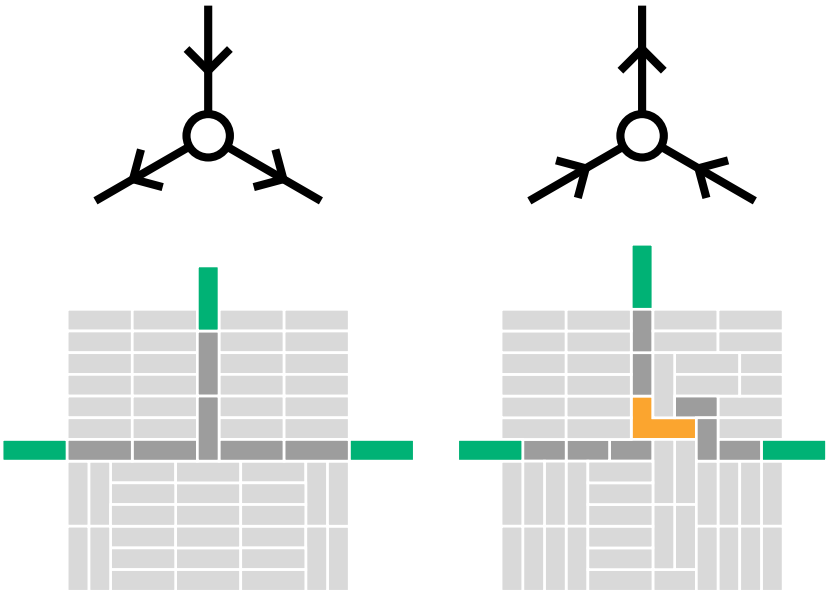

Fig. 12. The two gadgets for replacing the two types of vertices.

Proof. In order to prove NP-hardness, we give a polynomial-time reduction from the *Hamiltonian cycle problem* on 3-regular planar digraphs where each vertex has indegree 1 or 2, which is a classic well-known NP-complete problem [19]. We use an idea similar to the one in [21]: For a given 3-regular planar digraph G, we replace the two types of vertices by the two types of gadgets shown in Fig. 12, respectively, and join them by the same wire gadget in Section 3.1. In [21], a specific peg visits every vertex exactly once along the Hamilton cycle. Our construction is done by making one (and unique) vacant unit square in a wire gadget instead of the specific peg. As shown in Corollary 19 in Appendix A, we note that every edge joining an outdegree-1 vertex to an indegree-1 vertex should belong to every Hamiltonian cycle. Such an edge is called a *permanent arc* in [19]. Furthermore, the Hamiltonian cycle problem on 3-regular planar digraphs is still NP-complete even if (1) the numbers of outdegree-1 vertices and indegree-1 vertices are the same, and (2) each outdegree-1 vertex should be joined to an indegree-1 vertex. That is, we can assume that any instance of this restricted Hamiltonian cycle problem has a permanent edge, and we can find it in polynomial time. Therefore, the vacant unit square can be put in any permanent edge. The gadget for this vacant unit square on a permanent edge is given in the left in Fig. 13(b). We have only one choice at this start point as shown in the right in Fig. 13(b). That is, this gadget forces us the direction of the edge. An example of the construction is given in Fig. 13(c).

The unique vacant unit square moves like the peg in [21]: when it visits an indegree-1 vertex as shown in the left side of Fig. 12, the central vertical I-trominoes in the figure move the vacant unit square to center, and it will be brought to left or right. On the other hand, when the vacant unit square visits an indegree-2 vertex from one of the two incoming directions, then the L-tetromino can move it to the outgoing direction of the vertex. Fig. 13(d) illustrates the configuration of the puzzle after visiting the vertices $3, 2, 6, 5, 1, 4, 8, 7$ in this order. It is easy to observe that all asymmetric pieces are flipped, and the remaining pieces are all rectangles, which can be flipped without any vacant space.

Therefore, when G has a Hamiltonian cycle, the instance of the flip over puzzle has a solution with the condition (0).

Now we suppose that the instance of the flip over puzzle has a solution. By Theorem 18, we can assume that the instance consists of k gadgets in the left and right in Fig. 12, respectively. Since the instance has only one vacant square, it is easy to see that the wire and turn gadgets are one-way. On the other hand, the L-tetromino in the right gadget in Fig. 12 can be flipped when the signal gets into the gadget and goes out from the gadget. Therefore, each of these gadgets can be passed once. By Theorem 18, when the signal goes out from the gadget, the next gadget should be the left gadget in Fig. 12. Therefore, the vacant square should visit these two gadgets in Fig. 12 alternately. Since it is a solution, the vacant square visits every left gadget exactly once, and it should visit the right gadget exactly once. This implies that the original graph G has a Hamiltonian cycle. This completes the proof. □

Theorem 6. *The flip over puzzle with constant flips is NP-complete even with all the following conditions:*

(0) *each piece can be moved at most once,*
(1) *the frame is a rectangle of height* 2*,*
(2) *all but two pieces are rectangles of height* 1*, and*
(3) *there is only one asymmetric piece.*

Proof. The basic idea is similar to the proof of Theorem 3. We reduce the 3-PARTITION problem to our problem as shown in Fig. 14. For given positive integers $a_1, a_2, a_3, \ldots, a_{3m}$, F is a rectangle of size $2 \times (3mB + m + 1)$. Each P_i is a rectangle of size $1 \times a_i$ for $1 \le i \le 3m$.

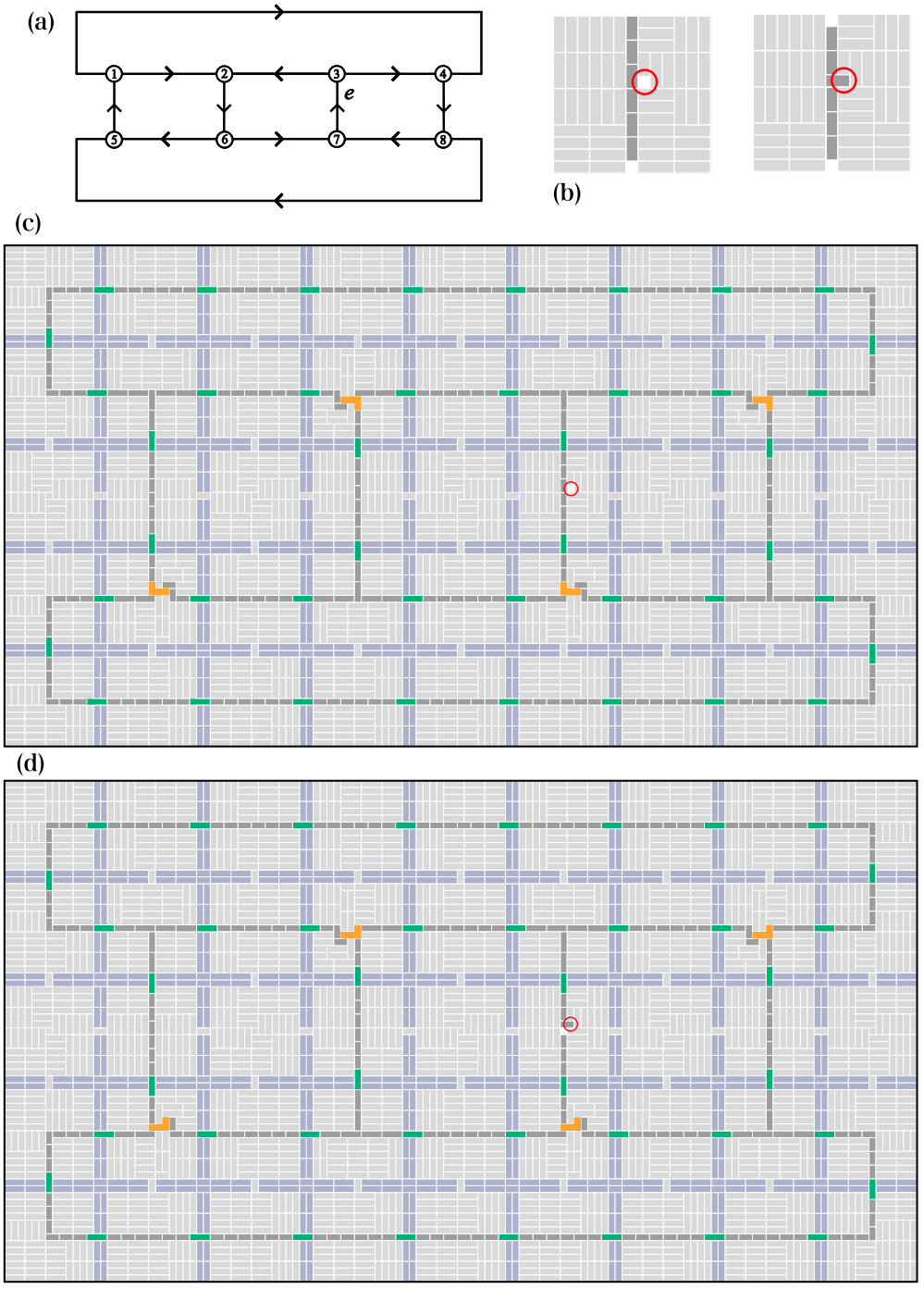

Fig. 13. An example of construction. (a) Given directed 3-regular planar graph G, (b) gadget for the start point, (c) constructed puzzle, and (d) the configuration after visiting the vertices $3, 2, 6, 5, 1, 4, 8, 7$ in this order.

As shown in Fig. 14, the large piece P_{3m+1} is obtained from a rectangle of size $2 \times (3mB + m - 1)$ by removing m rectangles of size $1 \times B$ in center and two rectangles of size $1 \times (mB - 1)$ on the left and right sides to make m vacant rectangles and two vacant L-shapes on the both sides in F. The last piece P_{3m+2} is a long L-shape polyomino as shown in Fig. 14. Since each block can be flipped at most once, the only way to flip all the pieces is (0) flipping P_{3m+1}, (1) flipping P_i for each i with $1 \le i \le 3m$ into m vacant rectangles of $1 \times B$ in center of P_{3m+1}, and (2) flipping P_{3m+2} from the left side in F to the right side in F. Therefore, the instance of the 3-PARTITION problem is a yes instance if and only if the constructed instance of the flip over puzzle with the conditions has a solution. □

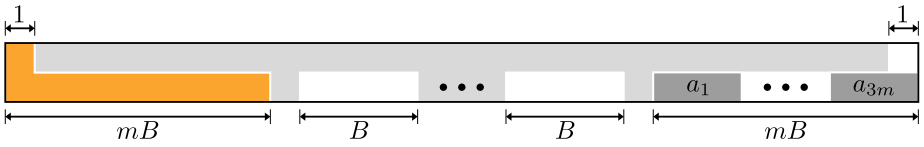

Fig. 14. Gadgets for the flip over puzzle of height 2.

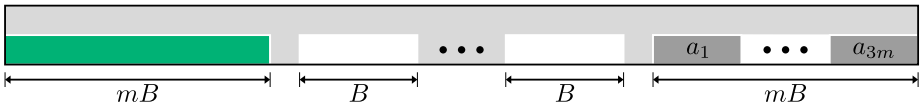

Fig. 15. Reduction to the flying block puzzle.

4. Complexity of flying block puzzles

In this section, we turn to the flying block puzzles. In flying block puzzles, we can translate and rotate pieces but cannot flip them. In the flip over puzzle in Section 3, almost all pieces are rectangles except a few asymmetric pieces. Since a rectangle does not change by a flip, we can inherit most of the results for flip over puzzles in Section 3 to ones for flying block puzzles by changing some special pieces. However, while the goal of the flip over puzzle is to flip *all* pieces, the goal of the flying block puzzle is to arrange a *specific* piece to the goal position. This difference requires different techniques.

The counterpart of Theorem 2 is as follows:

Theorem 7. *The flying block puzzle is PSPACE-complete even with all the following conditions:*

(1) *the frame is a rectangle of constant height without holes,*
(2) *every piece is a rectangle of size* $1 \times k$ *with* $1 \le k \le 3$*, and*
(4) *there is only one vacant unit square.*

Proof. In the proof of Theorem 2, we use an L-tetromino as the unique asymmetric piece. In the flying block puzzle, we do not need to use this trick. Instead of that, the goal of the resulting flying block puzzle is just to move any specific I-tromino in the wire or corner gadget in Fig. 5 corresponding to e_t, which plays the same role of the flip of the L-tetromino in the proof of Theorem 2. □

The counterpart of Theorems 3 and 6 is as follows:

Theorem 8. *The flying block puzzle is NP-hard even with all the following conditions:*

(1) *the frame is a rectangle of height* 2*,*
(2) *every line-symmetric piece is a rectangle of height* 1*, and*
(3) *there is only one non-rectangle piece, which is line-symmetric.*

Proof. The basic idea is the same as the proof of Theorem 3: We reduce from the 3-PARTITION problem. We let the initial feasible packing S be as in Fig. 15, while the goal feasible packing $\mathcal{T}$ is obtained by reversing S horizontally. The goal of this puzzle is to move the rectangle piece of size $1 \times mB$ to the right side. To do that, all other pieces of size $1 \times a_i$ should be moved to the m vacant spaces of size $1 \times B$.

When the instance of the 3-PARTITION problem is a yes-instance, we can solve the obtained instance of the flying block puzzle as follows: First, pack the pieces of size $1 \times a_i$ into the m rectangular holes of size $1 \times B$ according to the solution of the 3-PARTITION problem, and then move the piece of size $1 \times mB$ to the right side.

Now we assume that the flying block puzzle given in Fig. 15 has a solution. We focus on the turn that the rectangular block of size $1 \times mB$ is moved to the goal position. It is easy to see that it can be done only if the other rectangular blocks of size $1 \times a_i$ with $1 \le i \le 3m$ are moved to the m rectangular holes of size $1 \times B$. Then this packing of $3m$ blocks into m holes gives us a solution of the original instance of the 3-PARTITION problem, which completes the proof. □

We note that the claim in the proof of Theorem 8 still holds even if each piece can be moved at most once. In this case, the NP-hardness of Theorem 8 is changed to NP-completeness since we have a witness of polynomial-length for a yes-instance. On the other hand, the case discussed in Theorem 8 can be PSPACE-complete since the problem is solvable in PSPACE. In this sense, Theorem 8 is the counterpart of both Theorems 3 and 6.

The counterpart of Theorem 4 is as follows:

Theorem 9. *The flying block puzzle with constant flies is NP-complete even if it satisfies the following conditions:*

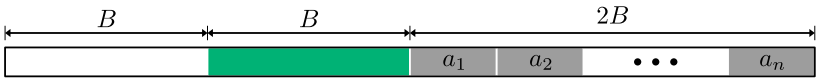

Fig. 16. The reduction from PARTITION.

(0) *each piece can be moved at most once,*
(1) *the frame is a rectangle without holes, and*
(2) *every piece is a rectangle of size* $1 \times k$ *with* $1 \le k \le 3$.

Proof. We apply the same technique of the proof of Theorem 7 to Theorem 4. □

Interestingly, the reduction of the proof of Theorem 5 from the Hamiltonian cycle cannot be extended to the flying block puzzle since we do not need to "visit" every piece to flip. Therefore, the computational complexity of the flying block puzzle with the conditions in Theorem 5 is open.

We now turn to the flying block puzzles with the frame of height 1. While this case is trivially "yes" in any instance of the flip over puzzle since every piece is a rectangle of height 1, we show weak NP-hardness of the flying block puzzle if the frame is of height 1 and each piece can be moved at most once. On the other hand, when we can move the pieces any number of times, we have a polynomial-time algorithm to solve it.

Theorem 10. *The flying block puzzle with constant flies is weakly NP-complete even if each piece can be moved at most once and the frame is a rectangle of height* 1.

Proof. We present a polynomial-time reduction from PARTITION [7], which asks, given n positive integers $a_1, \dots, a_n$ with $\sum_{i=1}^{n} a_i = 2B$, whether there exists a partition of $\{1, \dots, n\}$ into two sets A and A' such that $\sum_{i \in A} a_i = \sum_{i \in A'} a_i = B$. This problem is weakly NP-complete. Namely, the problem is NP-complete if each number is given as its binary representation.

For a given instance $\langle a_1, a_2, \dots, a_n \rangle$ of PARTITION, we construct an instance of the flying block puzzle as follows. It consists of a frame F and a set of pieces $\mathcal{P} = \{P_1, P_2, \dots, P_{n+1}\}$. The frame F is a $1 \times (4B)$ rectangle. For $1 \le i \le n$, the piece P_i is a $1 \times a_i$ rectangle. The special piece P_{n+1} is a $1 \times B$ rectangle. The initial feasible packing of $\mathcal{P}$ to F is given in Fig. 16. The goal is to move P_{n+1} to the right end of F.

First assume that $\langle a_1, a_2, \dots, a_n \rangle$ is a yes-instance of PARTITION. That is, there is a partition of $\{1, \dots, n\}$ into A and A' such that $\sum_{i \in A} a_i = \sum_{i \in A'} a_i = B$. We first move the pieces P_i with $i \in A$ to the leftmost vacant space of size $1 \times B$ and then we move and pack the pieces P_i with $i \in A'$ to left as much as possible in the vacant space of size $1 \times 2B$. Now we can move P_{n+1} to the goal position. Note that each piece moves at most once in this process.

Next assume that there exists a sequence of moves that eventually places P_{n+1} to the goal position. Let us consider the step right before P_{n+1} moves to the goal position in F. We set A to the indices of the pieces placed in the left side of P_{n+1} in F at this moment and A' to $\{1, \dots, n\} \setminus A$. At this moment, there has to be a vacant space of $1 \times B$ rectangle in the goal position. Therefore, $\sum_{i \in A'} a_i \le B$ holds. On the other hand, since P_{n+1} is in the initial position now, $\sum_{i \in A} a_i \le B$ holds too. Since the sum of $\sum_{i \in A} a_i$ and $\sum_{i \in A'} a_i$ is $2B$, both of them have to be exactly B. □

Theorem 11. *The flying block puzzle can be solved in polynomial time when the frame is a rectangle of height* 1.

Proof. Let F be the frame of size $1 \times f$, and $\{Q, P_1, \dots, P_n\}$ be the set of pieces, where Q is the special piece to be moved to the given goal position. Let a_i denote the length of P_i for each i. We assume without loss of generality that $a_1 \le a_2 \le \cdots \le a_n$. Let a_0 be the length of Q. Now let $S = f - \sum_{i=0}^{n} a_i$, which gives us the total vacant space in F. Using S, we divide the set of pieces into two groups. A piece P_i (or Q) is *short* if $a_i \le S$, and *long* if $a_i > S$. We assume that the pieces $P_1, \dots, P_{n'}$ are short, and $P_{n'+1}, \dots, P_n$ are long. Let $\mathcal{X}$ be the set of long pieces. (The set $\mathcal{X}$ may contain Q.)

Observe that two long pieces cannot exchange their relative positions in F since the working space S is too small to swap them. Therefore, the ordering in $\mathcal{X}$ cannot be changed by any flies of pieces. On the other hand, when at least one of them is short, two pieces can exchange their relative positions in F in linear number of flies without changing the relative positions of the other pieces.

Now we focus on the special piece Q. If Q is in $\mathcal{X}$, the relative position of Q in $\mathcal{X}$ is fixed. On the other hand, if Q is short, the number of possible orderings of $\mathcal{X} \cup \{Q\}$ is $|\mathcal{X}| + 1$. In the latter case, the algorithm checks all of these $|\mathcal{X}| + 1$ cases. Hereafter, we assume that the ordering of $\mathcal{X} \cup \{Q\}$ is fixed. (If Q is short and not in that position in $\mathcal{X} \cup \{Q\}$, then we first move Q to that relative position.)

When we put Q at the goal position, the frame F is divided into two parts by Q. Let f_L and f_R be the length of the left and right parts of F divided by Q, respectively. In the ordering of $\mathcal{X} \cup \{Q\}$, let $\mathcal{X}_L$ and $\mathcal{X}_R$ be the sets of long pieces in the left and right of Q in F, respectively. Let $L = f_L - \sum_{P_i \in \mathcal{X}_L} a_i$ and $R = f_R - \sum_{P_i \in \mathcal{X}_R} a_i$, which give us the total spaces we can use for placing short pieces in the left and right sides in F divided by Q at the goal position.

If we have already $L < 0$ or $R < 0$, the answer is "no". On the other hand, if $L \ge 0$ and $R \ge 0$, we can decide that the answer is "yes". In order to answer "yes", we have to show that the set of short pieces can be divided into two sets such that the total lengths

of them are less than or equal to L and R, respectively. That is, for the set A of indices of short pieces, we have to construct a subset $A \subseteq \{1, 2, \ldots, n'\}$ so that $\sum_{i \in A} a_i \leq L$ and $\sum_{i \in \bar{A}} a_i \leq R$, where $\bar{A} = \{1, 2, \ldots, n'\} \setminus A$. This set A can always be constructed as follows. We greedily add as many elements P_i in $\{P_1, P_2, \ldots, P_{n'}\}$ as possible to A while keeping $\sum_{i \in A} a_i \leq L$. Then, we have $L - \sum_{i \in A} a_i < S$ since $a_i \leq S$ for every short piece P_i. Therefore, we have $L - S < \sum_{i \in A} a_i \leq L$. On the other hand, $\sum_{i \in A} a_i + \sum_{i \in \bar{A}} a_i = \sum_{1 \leq i \leq n'} a_i = L + R - S$. Thus we obtain $\sum_{i \in \bar{A}} a_i = L + R - S - \sum_{i \in A} a_i < L + R - S - (L - S) = R$. Therefore, this instance of the flying block puzzle has a feasible arrangement of the whole pieces such that the special piece Q is placed at the goal position. □

5. Complexity of C2C jumping block puzzles

In this section, we consider the C2C jumping block puzzle. The problem asks to determine if we can reconfigure the pieces from $\mathcal{S}$ to $\mathcal{T}$ by a sequence of flips. We note that pieces in $\mathcal{S}$ and $\mathcal{T}$ are not labeled, and hence the congruent pieces are not distinguished.

5.1. Tractable cases

We first show that the C2C jumping block puzzle is polynomial-time solvable in some simple cases.

Observation 12. *The C2C jumping block puzzle is polynomial-time solvable if the frame F is of height* 1.

Proof. Let s be the total number of empty unit squares in a feasible packing. A piece of size $1 \times k$ is said to be *long* if $k > s$; otherwise, it is *short*. Clearly, it is not possible to swap the order of two long pieces, because this would require creating an empty space longer than s while keeping all pieces in the frame.

However, it is always possible to create a contiguous empty space of length s between any two pieces without changing the ordering of the pieces. Thus, it is possible to pick up any short piece and insert it between any two other pieces. It follows that the initial feasible packing $\mathcal{S}$ can be reconfigured to the goal feasible packing $\mathcal{T}$ if and only if the orderings of the long pieces in $\mathcal{S}$ and $\mathcal{T}$ are the same. □

Observation 13. *The C2C jumping block puzzle is polynomial-time solvable if all pieces are rectangles of size* 1×2.

Proof. We can reduce this problem to the Matching Reconfiguration problem, which is defined in a natural way on the matching problem: for a given pair of matchings, the problem asks if the initial matching can be transformed into the goal matching via a sequence of edge replacements such that all intermediate sets are also matching. It is shown to be polynomial-time solvable in [13, Proposition 2]. We construct a grid graph G on the unit squares of the frame F; the vertex set of G is the set of unit squares of F, and two unit squares are connected by an edge in G if and only if they are neighbors in F (or they share an edge in F). Each piece of size 1×2 covers exactly one edge of G, and therefore a feasible packing naturally corresponds to a matching of G. Moving a piece in the C2C jumping block puzzle corresponds to replacing an edge in a matching of G with a different edge, in such a way as to form a new matching of the same size. Thus an instance of the C2C jumping block puzzle has a solution if and only if the corresponding instance of the Matching Reconfiguration problem on G is a yes instance. Therefore, the C2C jumping block puzzle where all pieces are dominoes is polynomial-time solvable. □

We can extend Observation 13 to the variant with constant flips (cf. Section 3.3).

Theorem 14. *The C2C jumping block puzzle with constant flips is polynomial-time solvable if all pieces are rectangles of size* 1×2 *and each piece can be moved at most once.*

Proof. Let an initial feasible packing $\mathcal{S}$ and a goal feasible packing $\mathcal{T}$ be given. We view $\mathcal{S}$ as a set of dominoes $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ arranged in the frame F. Similarly, we view $\mathcal{T}$ as a set of dominoes $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ arranged in F. We construct the bipartite graph $G = (\mathcal{S} \cup \mathcal{T}, E)$ such that $\{S_i, T_j\} \in E$ if and only if the dominoes S_i and T_j share at least a unit square in F. Fig. 17 shows an example.

We say that a connected component of G is *balanced* if it has the same number of vertices in $\mathcal{S}$ and in $\mathcal{T}$; otherwise, it is *unbalanced*. More specifically, if a component has more vertices in $\mathcal{S}$ (respectively, in $\mathcal{T}$), it is said to be *$\mathcal{S}$-unbalanced* (respectively, *$\mathcal{T}$-unbalanced*).

We will prove that the puzzle can be solved by moving each piece at most once if and only if G satisfies at least one of the following conditions:

(1) G has no cycles, or
(2) G has at least one unbalanced connected component.

Since we can construct G and verify these conditions in polynomial time, we will conclude that the puzzle is polynomial-time solvable.

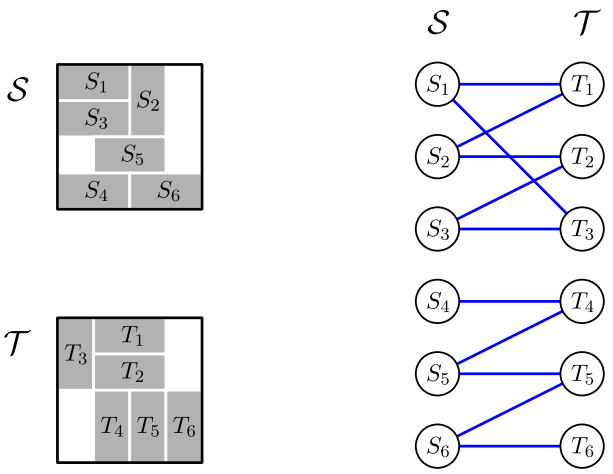

Fig. 17. An initial and a goal feasible packing (left) and the corresponding graph G (right).

Note that a solution to the puzzle can be described as a sequence of n pairs $((S_{i_1}, T_{j_1}), (S_{i_2}, T_{j_2}), \ldots, (S_{i_n}, T_{j_n}))$. This is because each piece can be moved at most once; thus, every time we pick up a piece $S_i \in \mathcal{S}$, we must immediately place it in an empty location corresponding to a piece $T_j \in \mathcal{T}$. In terms of the graph G, such a move is equivalent to erasing the vertex S_i and all its incident edges, and then erasing the vertex T_j, which is assumed to be isolated (if T_j is not isolated, then its location is not empty, and the move is not feasible). The puzzle is solvable if and only if there is a sequence of moves that eliminates all vertices of G.

Since all pieces are dominoes, G must have maximum degree at most 2. Therefore, each connected component of G is either a path (possibly consisting of a single isolated vertex) or a cycle. Note that all cycles in G must be balanced, because G is bipartite. In the following, the term "path" will be used to denote any connected component of G that is also a path.

We first observe that any balanced path $(T_{i_1}, S_{i_2}, T_{i_3}, \ldots, T_{i_{k-1}}, S_{i_k})$ can be eliminated via the sequence of moves (S_{i_2}, T_{i_1}), (S_{i_4}, T_{i_3}), $\ldots$, $(S_{i_k}, T_{i_{k-1}})$.

Similarly, a $\mathcal{T}$-unbalanced path, say $(T_{i_1}, S_{i_2}, T_{i_3}, \ldots, S_{i_{k-1}}, T_{i_k})$, can be reduced to the isolated vertex $T_{i_k} \in \mathcal{T}$ by the sequence of moves (S_{i_2}, T_{i_1}), (S_{i_4}, T_{i_3}), $\ldots$, $(S_{i_{k-1}}, T_{i_{k-2}})$.

On the other hand, assume that G contains an $\mathcal{S}$-unbalanced path, as well as an isolated vertex $T_j \in \mathcal{T}$. In this case, we can pair up T_j with an endpoint of the path; this results in a balanced path, which can be eliminated as above. (In the special case where the path consists of a single vertex $S_i \in \mathcal{S}$, we can simply pair up S_i and T_j to eliminate both.)

Since each move reduces both $\mathcal{S}$ and $\mathcal{T}$ by one vertex, the two sets always have the same size. Moreover, every unbalanced path with k vertices in $\mathcal{S}$ must have either $k-1$ or $k+1$ vertices in $\mathcal{T}$. It follows that the number of $\mathcal{S}$-unbalanced paths is always the same as the number of $\mathcal{T}$-unbalanced paths.

Assume that condition (1) is verified, i.e., every component of G is a path. By the above discussion, we can first eliminate all balanced paths, and then reduce each $\mathcal{T}$-unbalanced path to an isolated vertex in $\mathcal{T}$. Finally, we can pair up each $\mathcal{S}$-unbalanced path with an isolated vertex in $\mathcal{T}$ and eliminate both. Since there are as many $\mathcal{S}$-unbalanced paths as $\mathcal{T}$-unbalanced paths, this results in the elimination of all vertices, proving that the puzzle is solvable.

Assume now that condition (2) is verified, i.e., G contains some unbalanced paths. We will prove by induction on the number of cycles that the puzzle can be solved. If there are no cycles, then condition (1) is verified, and we know how to solve the puzzle. Suppose now that G contains a cycle C. Since there are unbalanced paths in G, there is at least one $\mathcal{T}$-unbalanced path. Firstly, we reduce this path to an isolated vertex $T_i \in \mathcal{T}$, and then we pair up T_i with any vertex of C lying in $\mathcal{S}$. This reduces C to a $\mathcal{T}$-unbalanced path; as a result, the number of cycles has decreased, and there are still unbalanced paths in G. Thus, the puzzle can be solved by the inductive hypothesis.

It remains to prove that, if all components of G are balanced and some of them are cycles, then the puzzle cannot be solved. We will prove it by induction on the number of vertices of G that do not lie on a cycle. The base case is when all components of G are cycles; this case is unsolvable because no move is possible, as there is no way to create an isolated vertex in $\mathcal{T}$ by removing a single vertex from $\mathcal{S}$. Suppose now that G contains some balanced paths, as well as some cycles. The only available move in this case is to pair up an endpoint $T_j \in \mathcal{T}$ of a balanced path with its adjacent vertex $S_i \in \mathcal{S}$. Since this move reduces the length of the path while keeping it balanced, the inductive hypothesis implies that the puzzle cannot be solved. □

5.2. PSPACE-complete and NP-hard cases

The extension of the observations above to more general cases, e.g., blocks of size 1×3, seems to be interesting (cf. [12]). In this context, we can obtain variants of Theorems 2 and 3 for this problem as follows.

Theorem 15. *The C2C jumping block puzzle is PSPACE-complete even with all the following conditions:*

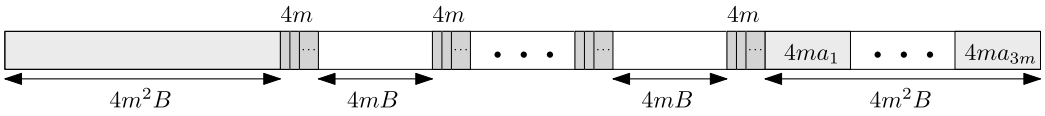

Fig. 18. Reduction to the shortest variant of the C2C jumping block puzzle. Each dark gray block consists of $4m$ pieces of size 1×1.

(1) *the frame is a rectangle of constant height without holes,*
(2) *every line-symmetric piece is a rectangle of size* $1 \times k$ *with* $1 \leq k \leq 3$,
(3) *there is only one asymmetric piece of size* 4, *and*
(4) *there is only one vacant unit square.*

Proof. In [10, Theorem 5.15], it is proved that the following variant of NCL is PSPACE-complete: its input consists of a constraint graph $G = (V, E, w)$ and two legal configurations C_s and C_t for G, and the problem asks to determine if there exists a sequence of legal configurations $C_s = C_0, C_1, \ldots, C_k = C_t$ such that C_i is obtained by reversing the direction of a single edge in C_{i-1} with $i \geq 1$.

Using the same proof of Theorem 2 for this variant of the NCL problem, we immediately obtain the theorem. □

Theorem 16. *The C2C jumping block puzzle is NP-hard even with all the following conditions:*

(1) *the frame is a rectangle of height* 2,
(2) *every line-symmetric piece is a rectangle of height* 1, *and*
(3) *there is only one non-rectangle piece, which is line-symmetric.*

Proof. The basic idea is the same as the proofs of Theorems 3 and 8: We reduce from the 3-PARTITION problem by reusing the gadgets in Fig. 15. The construction of an instance of the flying block puzzle from a given instance of the 3-PARTITION problem is given in Fig. 15. The goal is exchanging the rectangular block of size $1 \times mB$ with the other rectangular blocks of size $1 \times a_i$ with $1 \leq i \leq 3m$ (in any order). To do that, all pieces of size $1 \times a_i$ should be moved to the m vacant spaces of size $1 \times B$ before moving the rectangular block of size $1 \times mB$. Therefore, the instance of the 3-PARTITION problem is a yes instance if and only if the constructed instance of the flying block puzzle has a solution. □

Recall that the C2C jumping block puzzle is polynomial-time solvable if the frame F is of height 1 (Observation 12). Here we show that finding a shortest sequence is hard even in this setting.

Theorem 17. *Given an integer* k *and an instance of the C2C jumping block puzzle on a frame of height* 1 *without holes, it is NP-complete to decide whether there is a jumping sequence of length at most* k *from the initial feasible packing to the goal feasible packing.*

Proof. The proof of Observation 12 implies that a yes-instance of the C2C jumping block puzzle with a frame of height 1 admits a jumping sequence of length polynomial in the number of pieces as a yes-certificate. Therefore, we can assume that k is bounded by such a polynomial. This implies that the shortest variant belongs to NP. We show the NP-hardness by reducing 3-PARTITION to this problem.

Let $\langle a_1, \ldots, a_{3m} \rangle$ be an instance of 3-PARTITION. We construct the initial feasible packing as shown in Fig. 18, and set the goal feasible packing to the reversal of this configuration. We set $k = 6m + 1$.

First assume that $\langle a_1, \ldots, a_{3m} \rangle$ is a yes-instance of 3-PARTITION. Using a solution $A_1, \ldots, A_m$, we can move all pieces of size $1 \times 4ma_i$ to the m gaps of length $4mB$ in the middle. Then, we move the piece of size $1 \times 4m^2B$ to its goal position. Finally, we move the pieces of size $1 \times 4ma_i$ to their goal positions. We obtained the goal packing in $k = 6m + 1$ steps.

Conversely, assume that the C2C jumping block instance admits a jumping sequence of length at most $k = 6m + 1$ from the initial packing to the goal packing. We call each block of $4m$ consecutive pieces of size 1×1 in the initial packing a *wall*. We say that the pieces of size 1×1 are *small*, and the other pieces are *large*. Observe that, in the sequence we have to move each of the $3m + 1$ large pieces at least once, and thus we can move at most $3m$ small pieces. This implies that no wall can be completely removed. Thus, if A is the set of indices i of pieces of size $1 \times 4ma_i$ that are simultaneously put between two consecutive (possibly broken) walls at some step of the jumping sequence, then $\sum_{i \in A} 4ma_i \leq 4mB + 3m$, and thus $\sum_{i \in A} a_i \leq B$ holds. Observe that at the configuration right before moving the piece of size $1 \times 4m^2B$ beyond the leftmost wall, all pieces of size $1 \times 4ma_i$ are put into the m gaps in the middle. This gives a solution to the 3-PARTITION instance as before. □

6. Concluding remarks

In this paper, we investigate the computational complexities of the jumping block puzzles which form the token-jumping counterpart of the sliding block puzzles in the context of reconfiguration problems. In the context of combinatorial reconfiguration, the C2C jumping block puzzles leave some interesting open problems. In the reduction in the proof of Theorem 16, each piece is moved at most twice. On the other hand, as shown in Theorem 14, the C2C jumping block puzzle is polynomial-time solvable if all pieces

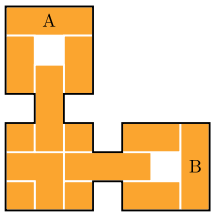

Fig. 19. A "simple" example of the flying block puzzle designed by Hideyuki Ueyama in [1] (with permissions): It requires 256 steps to exchange A and B.

are dominoes and each piece can be moved at most once. In Theorem 17, we show that finding a shortest sequence between two feasible packings is NP-complete as the $(n^2 - 1)$-puzzle. Can we then solve the C2C jumping block puzzle in polynomial time when each piece can be moved at most once?

We still have many other variants of the jumping block puzzles. We may allow the frame to have holes (or fixed obstacles). Another natural variant allows us to use (partially) colored blocks. In this framework, several puzzles were designed by Hideyuki Ueyama, and some of them can be found in [1]. One interesting example can be found in Fig. 19. It consists of 12 rectangle pieces and one L-tromino (i.e., all pieces are line-symmetric). There are two vacant unit squares. The goal of this puzzle is to exchange two I-trominoes with labels A and B. At a glance, it seems to be impossible. However, they can be exchanged in 256 steps. One issue in this example is that two I-trominoes are distinguished, while other blocks have no labels.

Another feature of this puzzle is the number of vacant unit squares. Intuitively, the puzzle is likely to become easier if we have many vacant unit squares. (In fact, some concrete examples can be found in [1]. In the examples, almost the same patterns are given except the number of unit squares. Removing unit squares, we can observe that the number of turns of the puzzle decreases.) On the other hand, one can increase the number of vacant unit squares without changing the solvability of a given instance by scaling up and then using only the "skeletons" of the pieces. Thus it would be interesting to find a parameterization that justifies our intuition above in some sense.

Other grid patterns are also natural variants of this puzzle. In fact, a commercial product based on *polyabolos*, which are figures obtained by gluing unit isosceles right triangles, is available as "Flip over Abolo" [14].

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ryuhei Uehara reports financial support was provided by Japan Advanced Institute of Science and Technology.

Data availability

No data was used for the research described in the article.

Acknowledgements

A part of this research is supported by JSPS KAKENHI Grant Numbers JP21K11752, JP20K11673, JP20H05964, JP20H05793, JP18K11169, JP18K11168, JP18H04091, and JP17H06287.

Appendix A. NP-completeness of the Hamiltonian cycle problem on 3-regular planar digraphs

In [19], it was shown that the *Hamiltonian cycle problem* on 3-regular planar digraphs where each vertex has indegree 1 or 2 is NP-complete. For short, we call a vertex in a digraph an (i, j)-*vertex* when it has indegree i and outdegree j. We here give slightly stronger results than the one in [19].

Theorem 18. *The* Hamiltonian cycle problem *on* 3*-regular planar digraphs G is NP-complete even with all the following conditions:*

(0) *every vertex is a* $(1,2)$*-vertex or a* $(2,1)$*-vertex,*
(1) *every edge from a* $(2,1)$*-vertex is joined to a* $(1,2)$*-vertex,*
(2) *every edge from a* $(1,2)$*-vertex is joined to a* $(2,1)$*-vertex,*
(3) *the number of* $(1,2)$*-vertices is equal to the number of* $(2,1)$*-vertices, and*
(4) *G is bipartite.*

Proof. Let $G = (V, E)$ be a digraph that satisfies the condition (0). By the result in [19], it is sufficient to construct a digraph $G' = (V', E')$ from G in polynomial time such that G' satisfies all the conditions and G has a Hamiltonian cycle if and only if G' has a Hamiltonian cycle.

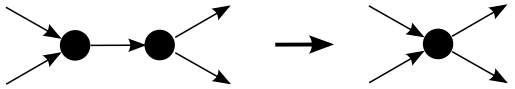

Fig. 20. Reduction from 3-regular digraph to 4-regular digraph.

We suppose that an edge $e = (u, v)$ joins two $(2,1)$-vertices. Then G contains an edge (w, v) for some $w \in V \setminus \{u\}$. When G has a Hamiltonian cycle, it has to use the edge (u, v) since u has no other outneighbor. Thus the edge (w, v) is redundant, and we can remove it from G. In other words, G has a Hamiltonian cycle if and only if $G - (w, v)$ has a Hamiltonian cycle, where $G - (w, v)$ is a digraph obtained from G by removing the edge (w, v). Now we focus on w in $G - (w, v)$. If w is a $(2,1)$-vertex, it has no outneighbor in $G - (w, v)$, thus we can decide G does not have Hamiltonian cycles. Otherwise, w and v are $(1,1)$-vertices in $G - (w, v)$. That is, the edges $(u, v), (v, v'), (s, w), (w, t)$ are in E for some v', s, t, and w and v have no other neighbors. Then we can obtain the digraph G'' by replacing the unique directed paths (u, v, v') and (s, w, t) in $G - (w, v)$ by directed edges (u, v') and (s, t), and it is easy to see that G has a Hamiltonian cycle if and only if G'' has a Hamiltonian cycle. When the digraph has edges joining two $(1,2)$-vertices, we can perform a symmetric reduction. We repeat this process until the digraph has no edge joining two $(2,1)$-vertices or two $(1,2)$-vertices.

Therefore, the Hamiltonian cycle problem on 3-regular planar digraphs is NP-complete even if we restrict to the digraphs G' of conditions (0), (1), and (2). For a digraph G' with conditions (0) (1), and (2), let A be the set of $(2,1)$-vertices and B be the set of $(1,2)$-vertices in G'. Then it is easy to see that G' is bipartite by these sets A and B since every edge joins a vertex in A and another in B. Now we consider the number of directed edges from A to B and ones from B to A. Since A is the set of $(2,1)$-vertices, the number of edges from A to B is $|A|$. On the other hand, since B is the set of $(1,2)$-vertices, the number of edges from B to A is $2|B|$. Then, since each vertex in A has indegree 2 and each vertex B has indegree 1, it is easy to see that $|A| = |B|$. Thus G' satisfies (0)-(4), and G has a Hamiltonian cycle if and only if G' has a Hamiltonian cycle. The reduction can be done in polynomial time, which completes the proof. □

By Theorem 18, we also have the following corollaries.

Corollary 19. *If G is a yes-instance that satisfies the conditions (0)–(4) in Theorem 18, then any Hamiltonian cycle of G uses every edge from a $(2,1)$-vertex to a $(1,2)$-vertex.*

Proof. By the condition (1) in Theorem 18, it follows. □

Corollary 20. *The* Hamiltonian cycle problem *on 4-regular planar digraphs is NP-complete even if the digraphs consist of only $(2,2)$-vertices.*

Proof. We can obtain the claim by contracting all edges in the condition (1) in Theorem 18 as shown in Fig. 20. □

We note that any 4-regular planar digraph in the proof of Corollary 20 satisfies one more property that two incoming edges (and hence two outgoing edges) are consecutive in clockwise ordering around the vertex as shown in Fig. 20 in the planar embedding. This property may be useful in other reductions.

References

[1] H. Akiyama, Board Puzzle Reader, Shin Kigen Sha, 2009 (in Japanese).
[2] M. Bonamy, M. Johnson, I. Lignos, V. Patel, D. Paulusma, On the diameter of reconfiguration graphs for vertex colourings, Electron. Notes Discrete Math. 38 (2011) 161–166.
[3] P. Bonsma, L. Cereceda, Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances, Theor. Comput. Sci. 410 (50) (2009) 5215–5226, https://doi.org/10.1016/j.tcs.2009.08.023.
[4] L. Cereceda, J. van den Heuvel, M. Johnson, Finding paths between 3-colourings, J. Graph Theory 67 (2011) 69–82.
[5] E.D. Demaine, M. Rudoy, A simple proof that the $(n^2 - 1)$-puzzle is hard, Theor. Comput. Sci. 732 (2018) 80–84.
[6] M. Gardner, The hypnotic fascination of sliding-block puzzles, Sci. Am. 210 (1964) 122–130.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability — A Guide to the Theory of NP-Completeness, Freeman, 1979.
[8] S.W. Golomb, Polyominoes, Princeton University Press, 1994.
[9] P. Gopalan, P.G. Kolaitis, E.N. Maneva, C.H. Papadimitriou, The connectivity of Boolean satisfiability: computational and structural dichotomies, SIAM J. Comput. 38 (2009) 2330–2355.
[10] R.A. Hearn, E.D. Demaine, Games, Puzzles, and Computation, A K Peters Ltd., 2009.
[11] R.A. Hearn, E.D. Demaine, PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constant logic model of computation, Theor. Comput. Sci. 343 (1–2) (2005) 72–96.
[12] T. Horiyama, T. Ito, K. Nakatsuka, A. Suzuki, R. Uehara, Packing trominoes is NP-complete, #P-hard and ASP-complete, in: The 24th Canadian Conference on Computational Geometry (CCCG 2012), 2012, pp. 219–224.
[13] T. Ito, E.D. Demaine, N.J. Harvey, C.H. Papadimitriou, M. Sideri, R. Uehara, Y. Uno, On the complexity of reconfiguration problems, Theor. Comput. Sci. 412 (2011) 1054–1065.
[14] N. Iwase, Flip over abolo, http://www.puzzlein.com/shop/e_shop.cgi?order=&mode=p_wide&G_NO=1139&IMG=./image/1139.jpg, 2021.
[15] M. Kamiński, P. Medvedev, M. Milanič, Shortest paths between shortest paths, Theor. Comput. Sci. 412 (2011) 5205–5210.

[16] M. Kamiński, P. Medvedev, M. Milanič, Complexity of independent set reconfigurability problems, Theor. Comput. Sci. 439 (2012) 9–15.
[17] Y. Liu, A. Morgana, B. Simeone, A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid, Discrete Appl. Math. 81 (1) (1998) 69–91.
[18] K. Makino, S. Tamaki, M. Yamamoto, An exact algorithm for the Boolean connectivity problem for k-CNF, Theor. Comput. Sci. 412 (2011) 4613–4618.
[19] J. Plesník, The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two, Inf. Process. Lett. 8 (4) (1979) 199–201.
[20] D. Ratner, M. Warmuth, The $(n^2 - 1)$-puzzle and related relocation problems, J. Symb. Comput. 10 (1990) 111–137.
[21] R. Uehara, S. Iwata, Generalized Hi-Q is NP-complete, Trans. IEICE E73 (2) (1990) 270–273, the proof is available at http://www.jaist.ac.jp/~uehara/pdf/phd7.ps.gz.
[22] T.C. van der Zanden, Parameterized complexity of graph constraint logic, in: IPEC 2015, in: LIPIcs, vol. 43, Dagsthul, 2015, pp. 282–293.