Journal of Information Processing Vol.21 No.3 405–412 (July 2013)

[DOI: 10.2197/ipsjjip.21.405]

Regular Paper

The Complexity of Tantrix Match Puzzles with Four Colors

Akihiro Uejima[1,a)] Fuhito Yanagitani[1] Shohei Tsukamoto[1]

Received: July 31, 2012, Accepted: January 11, 2013

Abstract: Tantrix Match is a puzzle in which hexagonal tiles are arranged within a hexagonal lattice board in which there are some fixed tiles. Each tile has painted lines in three of four possible colors, and it is required that all lines that touch on adjacent tiles must match in color. The aim of this research is to determine the computational complexity of this puzzle, and we prove that the generalized Tantrix Match is **NP**-complete by reduction from the circuit-satisfiability problem (Circuit-SAT).

Keywords: puzzles, Tantrix, computational complexity, NP-completeness

1. Introduction and Definitions

Tantrix[*1] is a hexagonal-tile-based abstract strategy puzzle game invented in 1988 by Mike McManaway of New Zealand. The hexagonal tiles each have three painted lines such that no two lines on a tile have the same color. There are four possible colors of the lines, which in the commercial product are *red*, *green*, *blue*, and *yellow*. There are also four different ways in which the lines are arranged on the tiles (see **Fig. 1**), thus the combination of four shapes of tiles and three colors chosen from among four possible colors give a total of 56 different tiles.

There are several different types of commercial Tantrix products, e.g., multi- or single-player versions, and versions that restrict the possible colors. In this paper, we focus on *Tantrix Match*, which was invented in 2009, and analyze its computational complexity. Tantrix Match is a single-play puzzle in which the number of possible colors is restricted to three, i.e., *red*, *green*, and *yellow*. This puzzle has a set of unique *stock tiles* and a hexagonal lattice board with pre-placed *clue tiles*, and it is played by arranging the stock tiles on the empty spaces in the given board according to the following rule (*basic constraint*): all touching lines must match in color. For an instance of Tantrix Match, the set of stock tiles is the *stock set*, the set of hexagonal cells in a board that do not have clue tiles form the *allocatable area*, and a member of the allocatable area is an *allocatable cell*. Note that the size of the stock set is equal to the size of the allocatable area.

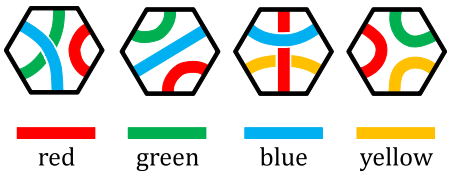


Fig. 1 Tantrix tiles.

One more rule for all Tantrix puzzles is that the arrangements of tiles cannot contain *holes*, which are places that do not contain tiles but that are completely enclosed by tiles. We call this condition the *hole constraint*. In Tantrix Match, this constraint is satisfied by the shape of the given boards.

In this paper, we consider the problem below as a generalization of Tantrix Match, and we prove that the problem without the hole constraint is **NP**-complete. Note that in the generalization, we may use tiles painted from within four colors. We assume that there exists a catalog consisting of all hexagonal tiles (for example, 56 tiles when using 4 colors) and their initial orientation. We note that the size of such a catalog is constant.

Tantrix Match Problem

Instance: A hexagonal lattice board B with clue tiles, an allocatable area A_B ($A_B \subseteq B$), and a stock *multiset* S ($|A_B| = |S|$),

Question: Is there an arrangement of all stock tiles in S on B that satisfies the basic constraint?

Instances of the Tantrix Match problem and their solutions are shown in **Figs. 2** and **3**. Throughout this paper, tiles with a white or gray background represent stock or clue tiles, respectively, and the blue hexagonal cells on the board B express the allocatable area A_B within B. If hexagonal cells b and b' on a board B have a common edge, b is a *neighbor* of b'. For a cell b, the set of all neighbors of b in B is denoted by $N_B(b)$, so each cell b has at most six neighbors in B (i.e., $|N_B(b)| \leq 6$).

An arrangement of all stock tiles in S on B can be expressed by a one-to-one mapping $f : A_B \to S$ and a rotation of each stock tile $f(b) \in S$ that is placed on a hexagonal cell b. If the mapping f is fixed as an input, i.e., the position of all stock tiles on A_B is given, then the only possibility to make S fit is to rotate each stock tile. A problem similar to this was considered in Refs. [1], [2]. M. Holzer and W. Holzer [1] invented the *Tantrix rotation problem*, which is the Tantrix Match problem with the stipulations that B has no clue tiles and a mapping $f : A_B \to S$ is fixed as an in-

[1] Osaka Electro-Communication University, Neyagawa, Osaka 572–8530, Japan

[a)] uejima@isc.osakac.ac.jp

[*1] Tantrix® is a registered trademark of Colour of Strategy Ltd. in New Zealand and of TANTRIX JAPAN in Japan, under the license of M. McManaway, the inventor.

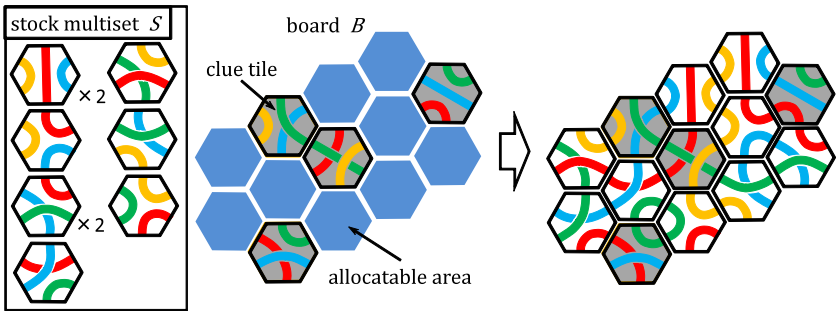


Fig. 2 Instance of a Tantrix Match problem.

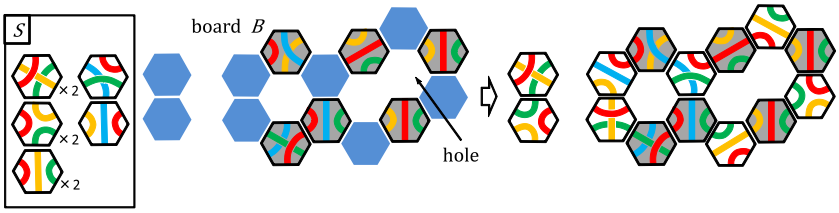


Fig. 3 Instance of a Tantrix Match problem without hole constraint.

put (note $A_B = B$ in this case). They proved in Ref. [1] that the Tantrix rotation problem with four colors (but without the hole constraint) is **NP**-complete, moreover in Ref. [2], Baumeister and Rothe showed that the three-color and two-color versions of the problem remain **NP**-complete.

It seems too restrictive that a mapping f is fixed prior to solving the Tantrix Match problem. Positioning the stock tiles on A_B by trial and error is one of the key factors that makes this puzzle entertaining. Consequently, we consider the Tantrix Match problem in which the player is allowed to position the stock tiles. We analyze the time complexity of the Tantrix Match problem as defined above, and we prove the following theorem.

Theorem 1 *The Tantrix Match problem with four colors available for the painted lines on the tiles, but without the hole constraint, is* ***NP****-complete.*

2. Proof of NP Membership

In this section, we state and prove Lemma 1 as the first step in proving our claim of **NP**-completeness, that is, Theorem 1.

Lemma 1 *The Tantrix Match problem is in* ***NP****.*

Proof. Consider a polynomial-time algorithm for verifying the candidates for solution. We suppose that the candidates for B, A_B, and S are expressed as the combination of a one-to-one mapping $f : A_B \to S$ and a rotation function $r : S \to \{0, 1, \ldots, 5\}$. For a mapping f and a rotation function r, this means that each tile $f(b) \in S$ is placed on a hexagonal cell b and rotated $r(f(b)) \cdot \pi/3$ radians in a clockwise direction, based on an initial orientation of $f(b)$ (we can see the initial orientation in $O(1)$ time by referring to the catalog data for the tile $f(b)$).

Given such f and r, we can easily check whether a mapping f is one-to-one in $O(|S|)$ time by a search of f, and we can verify the basic constraint in $O(|B|)$ time by checking for every hexagonal cell $b \in B$ that all touching lines between b and $N_B(b)$ match in color.

As mentioned in Section 1, whether the hole constraint is satisfied depends on the shape of a given board; hence, we do not consider how to check the hole constraint. Therefore, the above algorithm outputs "yes" if and only if the candidate satisfies the basic constraint, and it takes a total of $O(|B|)$ time.

In the remainder of this paper, we prove Lemma 2, i.e., we prove the **NP**-hardness of the Tantrix Match problem without the hole constraint.

Lemma 2 *The Tantrix Match problem with four colors available for the painted lines on the tiles, but without the hole constraint, is* ***NP****-hard.*

3. The Reduction

We prove Lemma 2 by reducing the circuit-satisfiability problem (Circuit-SAT for short), which is the problem of satisfying given Boolean circuits $C(v_1, v_2, \ldots, v_n)$ with n Boolean values $v_1, v_2, \ldots, v_n$, to the Tantrix Match problem. Circuit-SAT has already been shown to be **NP**-complete in Ref. [3]. For presenting a proof by construction, we describe some *gadgets* that act in circuit-like ways. Our goal is to simulate the behavior of a Boolean circuit by combining these gadgets. For a given Boolean circuit, we construct an instance (B, A_B, S) of a Tantrix Match problem corresponding to that circuit in polynomial time of the number of cells in the hexagonal lattice board.

We can create quite varied and complicated circuits by using only the basic logical gates AND and NOT (note that {AND, NOT} is universal), wires, and splits. In the reduction, we must be careful about wire crossing in the circuits. To construct the Tantrix Match puzzle from circuits, we consider the Boolean circuits restricted to a plane. For this purpose, we can replace the wire crossing by McColl's planar "cross-over" circuit [4] as shown in **Fig. 4**, which can be constructed by three XOR gates (note that a XOR gate can be expressed as the combination of

Table 1 List of stock tiles prepared for the gadgets.

	Wire Section 3.1	Input Section 3.2	Output Section 3.3	NOT Section 3.4	Split Section 3.5	AND Section 3.6	Bend Section 3.7
(a)	○	○				○	
(b)	○	○				○	
(c)				○		○	
(d)					○		○
(e)						○	
(f)					●	○	●
(g)					●	○	●

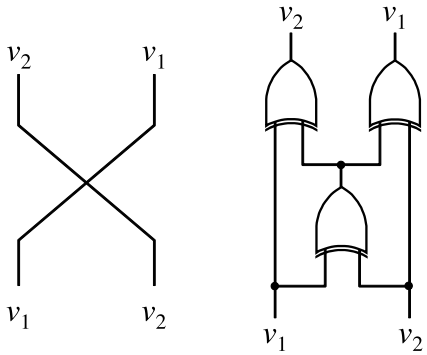


Fig. 4 McColl's planar "cross-over" circuit.

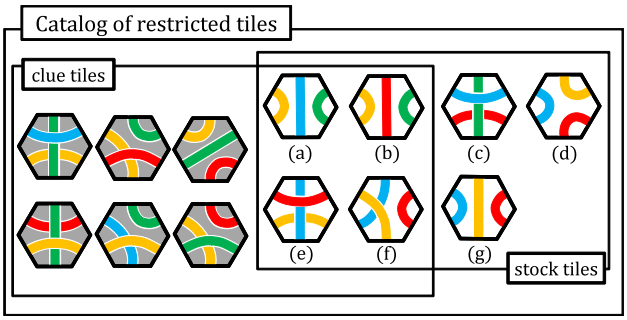


Fig. 5 Catalog of 13 types of tiles.

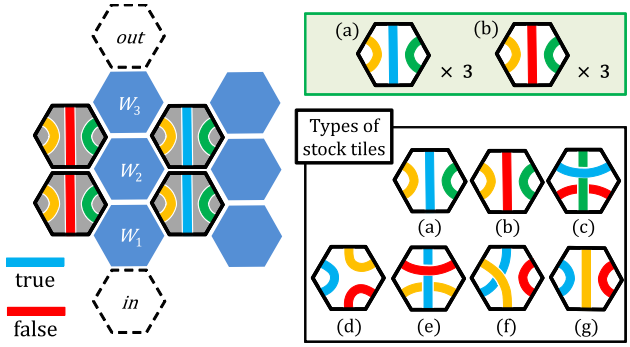


Fig. 6 Wire gadget.

AND and NOT gates). Our reduction, including the planarization of circuits, is similar to the reducing technique in Refs. [1], [2].

In the following, we consider the Tantrix Match problem in which the types of tiles set in the board B and included in the multiset S of stock tiles are restricted to the tiles listed in **Fig. 5**. We need only to consider the constructions of seven gadgets, which correspond to wires, inputs, outputs, NOT gates, splits, AND gates, and bended wires, and in the following subsections, we will construct such gadgets by using the 13 types of tiles illustrated in Fig. 5 as clue and stock tiles. **Table 1** shows information for the combination of types of stock tiles and the gadgets: (1) the tiles of types marked by ○ will be prepared as stock tiles for each gadget, and (2) if the row of a gadget has ● elements, then this means that the tiles of types marked by ● are not prepared as stock tiles for the gadget, but may be arranged on an essential part of the gadget by diverting from stock tiles prepared for the other gadgets.

After constructing all gadgets that we use for the reduction in the rest of this section, we will explain how to connect our several gadgets by illustrating an instance of the Tantrix Match problem, which simulates a simple Boolean circuit, in Section 4. When connecting between two gadgets except for the wire ones, we insert wire gadgets between them with the aim of combining them without a contradiction.

We will represent *true* by the color *blue* and *false* by the color *red*, and when we say that the input of a certain gadget is *blue* (resp. *red*), we will mean that the input of the corresponding logical gate carries the value of true (resp. false). The same holds for the output. For a hexagonal cell b in a sub-board $\tilde{B}$, a color sequence concerning a line touching b of a (pre-)placed tile on $b' \in N_{\tilde{B}}(b)$ is expressed as $InCOL_b = (c_0, c_1, \ldots, c_5)$ such that $c_i \in \{r, g, b, y, *, -\}$ for $i = 0, 1, \ldots, 5$ moving clockwise from the top position, and the ith mark c_i in $InCOL_b$ is expressed as $InCOL_b(i)$. The colors are represented by their initials r, g, b, y, and $*$ and $-$ mean suspense and nonexistence, respectively. Similarly, for a cell $b \in \tilde{B}$ on which a tile $f(b)$ is arranged, the order of colors of $f(b)$ on the six sides of b, starting at the top and moving clockwise, are represented as $OutCOL_b = [c'_0, c'_1, \ldots, c'_5]$, such that $c'_j \in \{r, g, b, y\}$ for $j = 0, 1, \ldots, 5$. We define $OutCOL_b(j)$ as the jth mark c'_j in $OutCOL_b$.

3.1 Wire Gadget

A wire gadget B_w of length three that corresponds to a wire in a circuit is shown in **Fig. 6**. In addition, we prepare stock tiles (a) and (b), as many in number as the length of the wire gadget. In the case of Fig. 6, three tiles each of types (a) and (b) are prepared as stock tiles for this sub-board. In the wire gadget, clue tiles of types (a) and (b) are fixed on both sides of allocatable cells W_i, and there exists a *red* (resp. *blue*) straight line, which parallel W_is, on clue tiles (a) (resp. (b)) at the left (resp. right) side of the W_is. We call these straight lines *side lines* of wire gadgets.

If the input color of the gadget is *blue*, i.e., the top color of a tile arranged on the cell marked by *in* is *blue*, then the color sequence of cell W_1 is $(*, y, -, b, -, g)$ due to the clue tiles on $N_{B_w}(W_1)$. Hence, the only stock tile on W_1 that fits the sequence is type (a).

In the similar way, we must place a type (a) tile on cell W_2 (resp. W_3) since the sequence of W_2 (resp. W_3) became

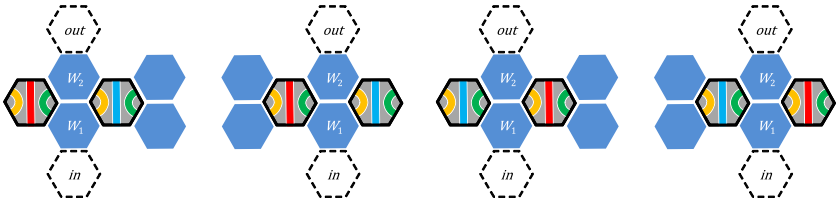


Fig. 8 Swapping *red* and *blue* colors of side lines on the wire gadget.

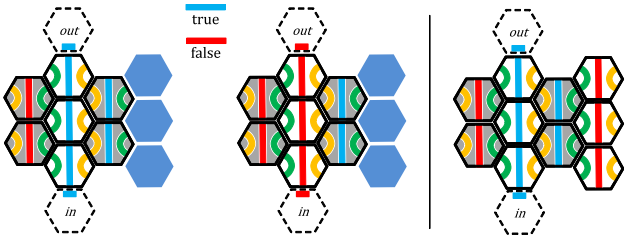


Fig. 7 Solutions to the wire gadget.

$(*, y, y, b, g, g)$ (resp. $(*, -, y, b, g, -)$) after placing the tile on W_1 (resp. W_2) as shown in the left figure of **Fig. 7**. As a result, the output color of the wire gadget is also *blue*. From the same discussion, if the input color is *red*, then the output is also *red* (see the center figure of Fig. 7).

After arranging the tiles on the allocatable cells W_i of the gadget, the remaining stock tiles can then be arranged on the unlabeled allocatable cells paralleling W_is. For example, when the input color of the wire gadget is *blue*, three tiles of type (b) remain after the above arrangements on W_is, we can arrange these tiles as shown in the right figure of Fig. 7.

The wire gadget shown in Fig. 6 has a *red* left-side line and a *blue* right-side line, and unlabeled allocatable cells on the extreme right. Then, we can easily construct variant wire gadgets about colors of side lines and layouts of the unlabeled allocatable cells as shown in **Fig. 8**. We note that the above discussion about arrangements of clue tiles also holds for these wire gadgets, since the color sequences of the allocatable cells for every wire one remain unchanged. Moreover, we can directly connect these variant wire gadgets (see **Fig. 9**), since the clue tiles on a wire gadget are not neighbors of the clue tiles on another wire gadget. Even if the allocatable cells on the outside in the wire gadgets connect as shown in the left figure of Fig. 9, we can arrange all clue tiles prepared for their wire gadgets on the allocatable cells of them in a similar way to the arrangements shown in the right figure of Fig. 7.

The combination of their variant wire gadgets is very useful for connecting several gadgets except for wire ones.

3.2 Input Gadget

An input gadget B_{IN} corresponding to an input in a circuit is shown in **Fig. 10**. In this gadget, we set one stock tile of each of types (a) and (b) in the same way as was done in the wire gadget of length three. We can then place the (a) or (b) tile on cell I of B_{IN}; thus the output color is only *blue* or *red* (see **Fig. 11**). The treatment of other cells in the gadget and the remaining tiles is similar to that of the wire gadgets. We remark that the stock tiles other than types (a) and (b) cannot be positioned on a side cell.

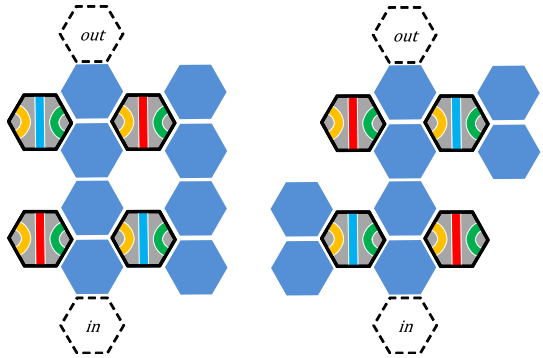


Fig. 9 Combination of the wire gadgets with different side colors.

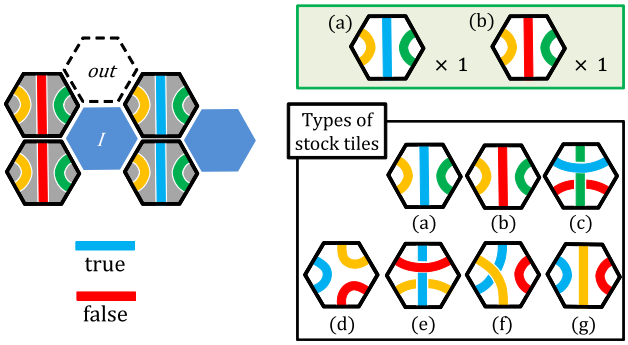


Fig. 10 Input gadget.

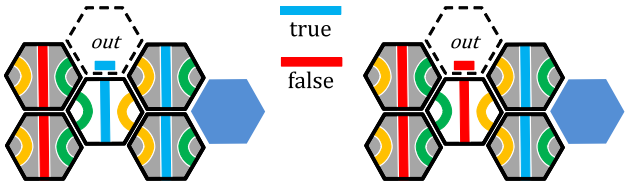


Fig. 11 Solutions to the input gadget.

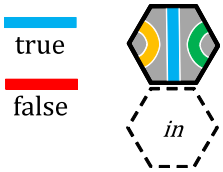


Fig. 12 Output gadget.

3.3 Output Gadget

The output of a circuit can be represented as shown in **Fig. 12**. If the input color is either *blue* or *red*, the basic constraint is satisfied only when the color is *blue*.

3.4 NOT Gadget

The NOT gadget B_{NOT}, corresponding to a NOT gate in a circuit, and one tile of type (c) that is set as a stock tile are shown in **Fig. 13**. Since the color sequence of a unique allocatable cell in

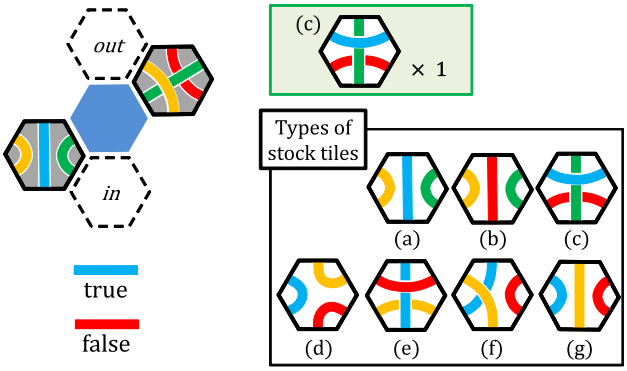


Fig. 13 NOT gadget.

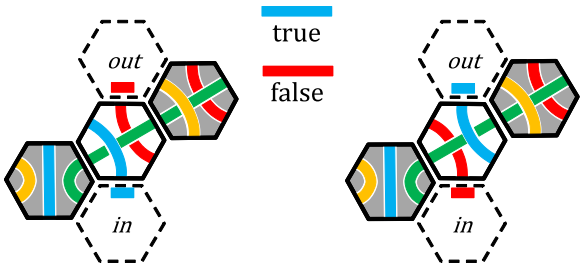


Fig. 14 Solutions to the NOT gadget.

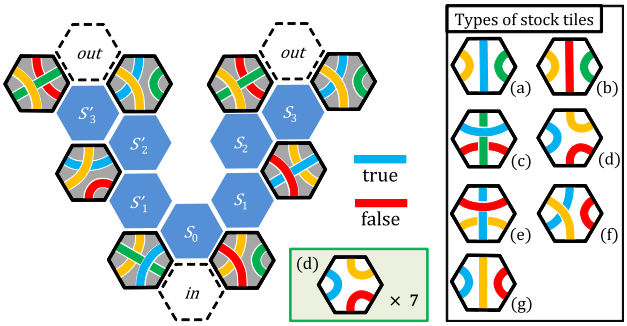


Fig. 15 Split gadget.

the gadget is $(*, g, -, *, g, -)$, the cell can have a stock tile of type (c). Hence, if the input color is *blue* (resp. *red*), then the output is *red* (resp. *blue*) as shown in **Fig. 14**.

3.5 Split Gadget

The split gadget B_s shown in **Fig. 15** simulates splitting a wire into two copies in a circuit. We prepare seven stock tiles of type (d) for a split gadget. A color sequence of S_0 is $(-, *, r, *, b, *)$, and if we assume that the input is *blue* or *red*, then a tile of type (d) is located on cell S_0.

If the input is *blue*, then the tile on S_0 is fixed, as shown in **Fig. 16**. The sequence of cell S'_1 is then $(*, -, y, y, -, r)$; thus the arrangement of the cell is fixed, and tiles of type (d) are also arranged on the subsequent cells S'_2 and S'_3 in the same way (see the left figure of Fig. 16). As a result, the left output is *blue*.

On the other hand, we can place either a tile of type (d) or type (g) on cell S_1 because its sequence is $(*, b, -, y, r, -)$. If a type (d) tile is placed on S_1 (refer to the right figure of Fig. 16), the sequence of S_2 is $(y, *, r, b, -, -)$. Then S_2 can only have a tile of type (d). Thus tiles of either type (d) or (g) can be placed on cell S_3. When a (g) tile is positioned on cell S_3, we must arrange a tile on the right cell with an *out* label such that the sequence

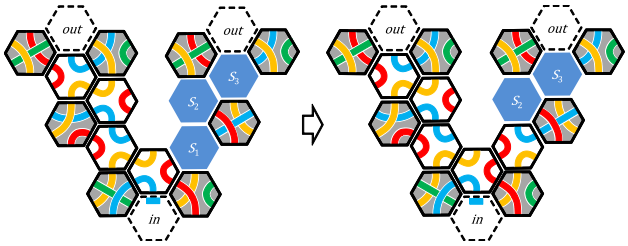


Fig. 16 Split gadget with *blue* input.

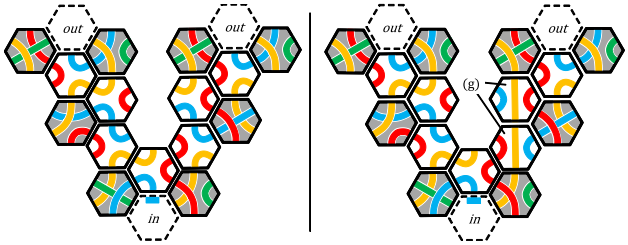


Fig. 17 Two solutions to the split gadget with *blue* input.

is $(-, -, y, y, g, -)$. However, there exist no stock tiles satisfying such a sequence. Given this situation, we must arrange a (d) tile on S_3. Therefore the right output is also *blue*, as shown on the left figure of **Fig. 17**.

For the case in which a tile of type (g) is placed on S_1, S_2 can only have a tile of type (g), since the sequence of S_2 is $(y, *, r, y, -, -)$. Then we can then arrange tiles of either type (d) or (g) on cell S_3. If a (g) tile is arranged on cell S_3, we have no possible tiles to be placed on the right cell with an *out* label, the sequence of which is $(-, -, y, y, g, -)$. Therefore, all the allocated cells of the gadget are fixed, as is shown in the right figure of Fig. 17, and both outputs are *blue*.

We consider the case where the input is *red*. For the left- and right-side sets of the allocatable cells in the split gadget, the color sequences are bilaterally symmetric. Hence, our discussion for this case is similar to the above discussion for the case where the input is *blue*, but with the sub-board reversed left to right. Since the input is *red*, the tiles on S_0 and S_1 are fixed as shown in **Fig. 18**. Then, S_2 can have a tile of either type (d) or (f). However, there exist no tiles satisfying the sequence of S_3 for placing an (f) tile. Hence, we can determine a tile arrangement for the right-side cells, as shown in Fig. 18.

For the left side, we can place a tile from among the types (d), (f), and (g) on S'_1 since the sequence of S'_1 is $(*, -, b, y, -, r)$. When choosing a tile of type (d) or (f), the sequence of S'_2 becomes $(y, -, -, r, b, *)$; otherwise it is $(y, -, -, y, b, *)$.

In the former case, we can place a tile of either type (d) or (e) on cell S'_2. However, if a tile of type (e) is placed on S'_2, there exist no tiles satisfying the sequence of S'_3. If a tile of type (d) is chosen for S'_2, the candidates for cell S'_3 are tiles of types (d) and (g). If a (g) tile is placed on cell S'_3, we cannot set a tile on the upper *out* cell. Therefore, a (d) tile is placed on S'_3 and the left output is *red* (see the left and center figures of Fig. 18).

In the latter case, a tile of type (g) is placed on cell S'_2 from the sequence of S'_2. Then, the sequence of S'_3 becomes $(*, b, b, y, -, r)$, and we can place only a tile of type (d) on cell S'_3 since another candidate (g) is excluded. The arrangement cor-

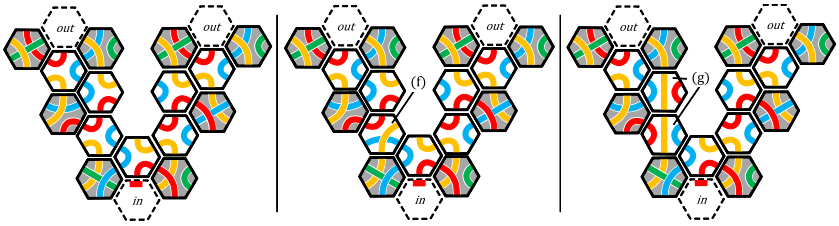


Fig. 18 Three solutions to the split gadget with *red* input.

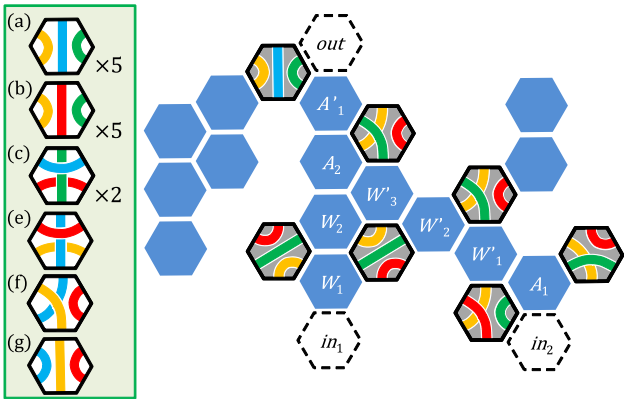


Fig. 19 AND gadget.

responding to this case is illustrated in the right figure of Fig. 18.

We thus have three solutions when the input is *red*, and the solutions shown in Fig. 18 represent a circuit in which a wire carrying a value of false is split into two copies. We notice that there exist solutions using all given tiles of type (d) for the cases of both *blue* and *red* inputs.

3.6 AND Gadget

Our AND gadget corresponds to an AND gate in a circuit, as illustrated in **Fig. 19**. Some of the unlabeled allocatable cells shown in the figure are places for arranging the remaining tiles after placing the labeled cells, such as the wire and input gadgets. For an AND gadget, we prepare stock tiles as shown in the green box in Fig. 19, i.e., five tiles each of types (a) and (b), two tiles of type (c), and one tile each of types (e), (f), and (g).

Suppose that the inputs from cells in_1 and in_2 are either *blue* or *red*. Because of the clue tiles on the neighbors of cells A_1 and A'_1, we can place on these cells only a tile of type (c) with a straight green line. As a result, the order of colors $OutCOL_{A_1}$ on A_1 by the (c) tile becomes either $[r, g, r, b, g, b]$ or $[b, g, b, r, g, r]$, and the order $OutCOL_{A'_1}$ for A'_1 is $[b, r, g, r, b, g]$ or $[r, b, g, b, r, g]$. Namely, if $InCOL_{A_1}(3) = b$ (resp. $InCOL_{A_1}(3) = r$), then $OutCOL_{A_1}(5) = b$ (resp. $OutCOL_{A_1}(5) = r$), which is a wire-like action. Also, if $InCOL_{A'_1}(3) = b$ (resp. $InCOL_{A'_1}(3) = r$), then $OutCOL_{A'_1}(0) = r$ (resp. $OutCOL_{A'_1}(0) = b$), which is a NOT-like action.

We consider a sequence of cells W_1 and W_2 as shown in Fig. 19. These cells are touched by green and yellow lines from the outside, so $InCOL_{W_1}(1) = g$ and $InCOL_{W_1}(5) = y$ for W_1, $InCOL_{W_2}(2) = y$ and $InCOL_{W_2}(4) = g$ for W_2. Thus we can arrange only tiles of type (a) and (b) on $W_1 and W_2$. As in wire gadgets, the sequence outputs the input color from cell in_1, such that $OutCOL_{W_2}(0) = b$ (resp. $OutCOL_{W_2}(0) = r$) for *blue* (resp. *red*) input. In a similar way, a sequence of cells W'_1, W'_2, W'_3 can also be assigned (a) and (b) tiles, and it then behaves as a wire gadget. We have thus fixed a tile arrangement on A_1, W_1, W_2, W'_1, W'_2, and W'_3 for each pair of inputs from in_1 and in_2 (see the left figures of **Figs. 20**, **22**, **24**, and **25**).

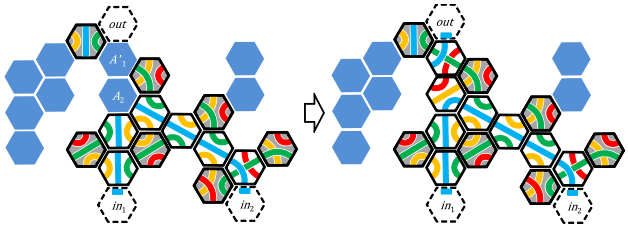


Fig. 20 AND gadget with inputs (*blue*, *blue*).

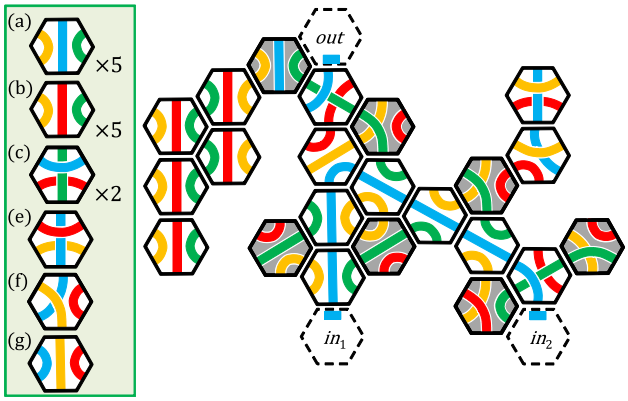


Fig. 21 Solution to the AND gadget with inputs (*blue*, *blue*).

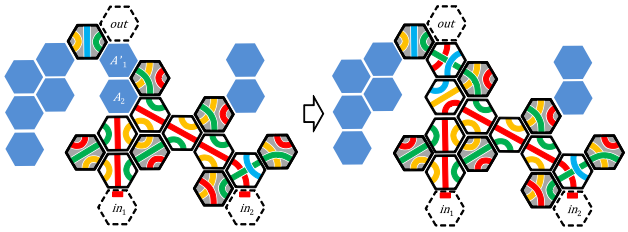


Fig. 22 AND gadget with inputs (*red*, *red*).

Because of the color sequence $InCOL_{A_2} = (*, y, b, b, -, -)$ for inputs (*blue*, *blue*), we must place a tile of type (g) on cell A_2 and one of type (c) on A'_1, as shown in Fig. 20. As a result, the output is *blue*. The remaining stock tiles for the AND gadget can then be arranged on the unlabeled allocatable cells, as shown in **Fig. 21**.

If the inputs are (*red*, *red*) (see Fig. 22), $InCOL_{A_2} = (*, y, r, r, -, -)$. With a tile of type (c) placed on cell A'_1, the allowable tiles on A_2 are limited to type (g). Hence, the arrangement of tiles on the labeled cells is fixed, as shown in the right figure of Fig. 22. The remaining stock tiles can be arranged on the unlabeled allocatable cells, as shown in **Fig. 23**.

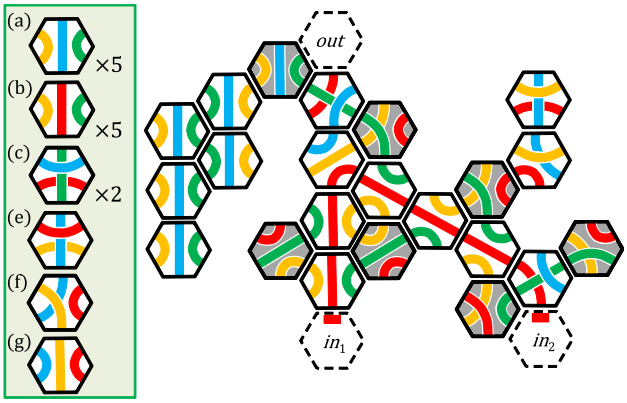


Fig. 23 Solution to the AND gadget with inputs (*red*, *red*).

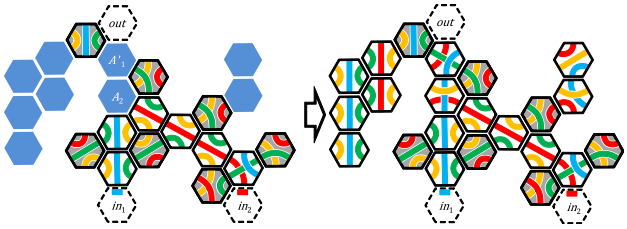


Fig. 24 AND gadget with inputs (*blue*, *red*).

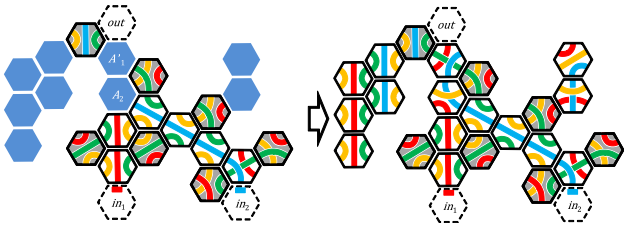


Fig. 25 AND gadget with inputs (*red*, *blue*).

If the inputs are (*blue*, *red*), then $InCOL_{A_2} = (*, y, r, b, -, -)$. Thus we can arrange the tiles on the labeled cells as shown in Fig. 24. The remaining stock tiles can be arranged on the AND gadget, as is shown in the right figure of Fig. 24.

If the inputs are (*red*, *blue*), then $InCOL_{A_2} = (*, y, b, r, -, -)$. In this situation, the arrangement of the tiles on the labeled cells is fixed, as shown in the left figure of Fig. 25, and the remaining tiles can be arranged on the AND gadget, as shown in the right figure of Fig. 25.

As a consequence, the output of the AND gadget for each pair of inputs works as the output of an AND gate in a circuit. The tiles placed on A_2 play an important role in the control of the output: a tile of type (g) is used in the case that the input is (*blue*, *blue*) or (*red*, *red*), type (e) when the input is (*blue*, *red*), and type (f) when the input is (*red*, *blue*). In addition, if there exist an AND gadget such that the input is not (*red*, *blue*) and a split gadget with the input *red*, then an arrangement of a tile of type (f) does not fixed on the unlabeled cells in the AND gadget. Thus the tile of type (f) may be exchanged for a tile of type (d) prepared as a stock tile for the split gadget, and we may obtain the arrangement on the split gadget shown in the center figure of Fig. 18. Similarly, if there exist two AND gadgets such that the inputs are neither (*blue*, *blue*) nor (*red*, *red*), an arrangement of tiles on a split gadget may be as shown in the right figures of Fig. 17 and Fig. 18.

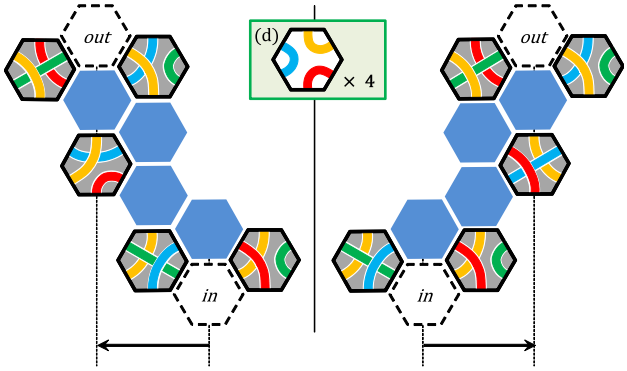


Fig. 26 Bend gadget.

3.7 Bend Gadget

Since wires must bend to connect logical gates in circuits, our construction needs to bend wire gadgets to route between our gate-like gadgets. In our reduction, we can put needed gadgets on a board, in which the parities of the horizontal positions of all the *in* and *out* cells of the gadgets are coincident, based on the construction of the above gadgets. **Figure 26** illustrates how to shift an *in* cell to the left or right by two cells. This bend gadget is constructed by using the left or right half of our split gadget.

When shifting an *in* cell to the left (or right) by $2p$ cells, we can construct the bended wires by alternating the p bend gadgets shown in the left (or right) figure of Fig. 26 and the $p - 1$ variant wire gadgets as shown in the left figure of Fig. 9 in Section 3.1.

4. Proof of Correctness of the Reduction

We are now ready to prove Lemma 2, with the help of the gadgets constructed in the previous section.

Proof of Lemma 2. Every gadget except the wire and output gadgets has clue tiles, which are on neighbors of the *in* and *out* cells, within the gadget. We consider such gadgets. For the upper right-and-left cells b with clue tiles of the *in* cells, $OutCOL_b(2) = g$ and $OutCOL_b(4) = y$, and for the lower right-and-left cells b' with clue tiles of the *out* cells, $OutCOL_{b'}(1) = g$ and $OutCOL_{b'}(5) = y$ (see **Fig. 27**). Moreover, these gadgets can be divided into four groups illustrated in Fig. 27 for the combination of color patterns in $OutCOL_b$s and $OutCOL_{b'}$s. Therefore, we can construct an instance of the Tantrix Match problem, which simulates a given instance of the Circuit-SAT, by combining our gadgets as shown in **Fig. 28**. The wire gadgets are expressed as cells with frames colored by orange in Fig. 28. For connecting between two gadgets except the wires without a contradiction, we insert the combination of variant wire gadgets between them and switch *red* and *blue* colors of their side lines on the wires connecting from the *out* cells on one gadget to the *in* cells on another gadget.

Because each component of a circuit can be replaced by a corresponding small gadget and the number of cells in the Tantrix Match problem is polynomial in the number of inputs for the Circuit-SAT problem, our reduction takes polynomial time of the number of inputs for the Circuit-SAT problem.

Let a satisfying truth-value assignment $(v_1, v_2, \ldots, v_n)$, $v_i \in \{0, 1\}$ for $i = 1, 2, \ldots, n$, for the input circuit C with n variables be given. Then we can construct an arrangement satisfying the

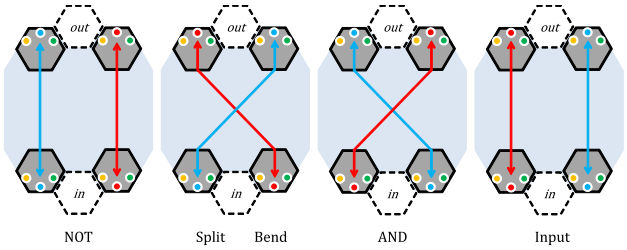


Fig. 27 Color patterns on neighbors of *in* and *out* cells on gadgets.

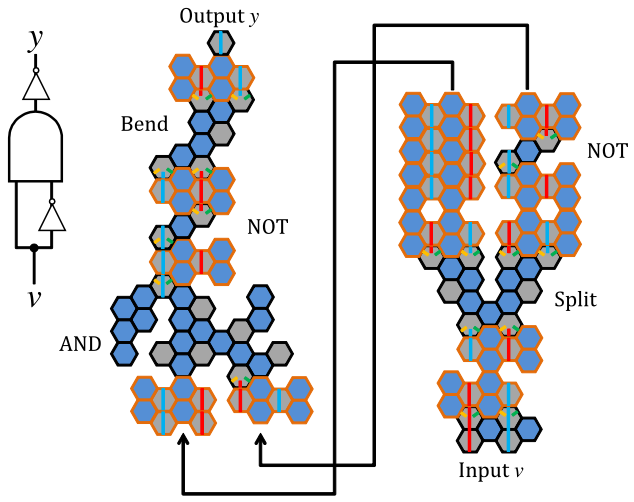


Fig. 28 The outline of the reduction for an instance of Circuit-SAT.

basic constraint for the Tantrix Match instance by choosing a local arrangement of the input gadgets corresponding to the truth assignment $(v_1, v_2, \ldots, v_n)$ and by adding the local arrangement to the other gadgets arising from arrangements of the input gadgets.

On the other hand, if there exists an arrangement satisfying the basic constraint for the Tantrix Match instance, then we can easily construct the satisfying truth-value assignment of C by following the local arrangement of the input gadgets, that is, choosing a tile of type (a) or (b) on the gadgets (recall Fig. 11 in Section 3.2).

5. Conclusions

We have shown that the Tantrix Match problem with four available colors but without the hole constraint is **NP**-complete by reducing the Circuit-SAT problem to this one. To simulate a Boolean circuit on a hexagonal lattice board with clue and stock tiles, we restricted the types of tiles used as clue and stock tiles (see Fig. 5 in Section 3). Since our constructed instance of the Tantrix Match problem may contain holes, the complexity of the Tantrix Match problem with the hole constraint is still open. It may also be interesting to analyze the complexity of this problem with three colors or less, using a proper subset of our tile restriction or an entirely different set of tiles. Moreover, we note that for each solution of the Circuit-SAT problem there are multiple solutions for the Tantrix Match: some gadgets may have local solutions in our gadgets, and the placing of tiles on allocatable cells without labels is not fixed. This means that the mapping from Circuit-SAT solutions to Tantrix Match solutions is not injective. As an area of future work, one might try to find a reduction such that there is an injective solution for the mapping.

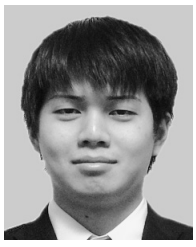
Acknowledgments This research was supported in part by Institute of Informatics, Osaka Electro-Communication University.

References

[1] Holzer, M. and Holzer, W.: Tantrix Rotation Puzzles are Intractable, *Discrete Applied Mathematics*, Vol.144, No.3, pp.345–358 (2004).
[2] Baumeister, D. and Rothe, J.: The three-color and two-color Tantrix rotation puzzle problems are NP-complete via parsimonious reductions, *Information and Computation*, Vol.207, No.11, pp.1119–1139 (2009).
[3] Cook, C.S.: The complexity of theorem proving procedures, *3rd ACM Symposium on Theory of Computing*, pp.151–158 (1971).
[4] McColl, W.: Planar crossovers, *IEEE Trans. Comput.*, Vol.30, No.2, pp.223–225 (1981).

Akihiro Uejima was born in 1975. He received a B.E. and M.E. from Information Systems Engineering, Department of Information and Computer Sciences, Toyohashi University of Technology in 1998 and 2000, respectively, and Dr. of Informatics degree from Department of Communications and Computer Engineering, Graduate School of Informatics at Kyoto University in 2005. Since 2005, he has been a lecturer in Department of Engineering Informatics, Osaka Electro-Communication University. His research interest is in graph theory, computational complexity. He is a member of the IEICE, IPSJ, the Operations Research Society of Japan, the Language and Automaton Symposium.

Fuhito Yanagitani was born in 1988. He received a B.Sc. from Department of Engineering Informatics, Faculty of Information and Communication Engineering, Osaka Electro-Communication University in 2011. From 2011, he is a student of Master course in Division of Information and Computer Science, Graduate School of Engineering at Osaka Electro-Communication University.



Shohei Tsukamoto was born in 1990. He received a B.Sc. from Department of Engineering Informatics, Faculty of Information and Communication Engineering, Osaka Electro-Communication University in 2012.