Journal of Information Processing Vol.25 730–734 (Aug. 2017)

[DOI: 10.2197/ipsjjip.25.730]

Regular Paper

The Building Puzzle Is Still Hard Even in the Single Lined Version

Kazuya Haraguchi[1,a) Ryoya Tanaka[1]

Received: October 25, 2016, Accepted: May 16, 2017

Abstract: The Building puzzle (a.k.a., the Skyscraper) is a Latin square completion-type puzzle like Sudoku, KenKen and Futoshiki. Recently, Iwamoto and Matsui showed the NP-completeness of the decision problem version of this puzzle, which asks whether a given instance has a solution or not. We provide a stronger result in the present paper; it is still NP-complete to decide whether we can complete a single line of the grid (i.e., a $1 \times n$ or an $n \times 1$ subgrid) without violating the rule.

Keywords: computational complexity, Latin square completion type puzzle, Building puzzle, Skyscraper

1. Introduction

Let us begin with the rule of the *Building puzzle*. We illustrate a puzzle instance in **Fig. 1** (a). For a natural number n, let $[n] = \{1, \ldots, n\}$. In this puzzle, we are given an $n \times n$ grid of *cells*, along with some numbers in $[n]$ written around the grid. We refer to a row and a column in the grid simply as a *line*. A line has two ends. A number around the grid is called a *Building number*. It is placed next to an end of a line. We say that a line has a Building number b on its end if b is written next to the end. Since there are $2n$ lines and a line has two ends, there are at most $4n$ Building numbers.

We are asked to fill all n^2 cells with integers in $[n]$ so that;

- the integers altogether form an $n \times n$ Latin square (i.e., in each line, every integer in $[n]$ appears exactly once), and that;
- what we call the *Building condition* is satisfied.

Let us explain what is the Building condition. Suppose that a building is constructed in every cell so that the number of floors is the integer assigned to the cell. The condition requires that, for every Building number b, one should see exactly b buildings when he or she looks up at the buildings on the line from the end where b is placed. The point is that we cannot see any lower buildings behind a higher building. In Fig. 1 (b), we show a (complete) solution to the instance of Fig. 1 (a).

Recently, Iwamoto and Matsui[3] showed the NP-completeness of the decision problem version of the Building puzzle. In our terminology, the problem is summarized as follows.

Building Puzzle
Input: An $n \times n$ Building puzzle instance and an $n \times n$ partial Latin square S.
Question: Is there a solution to the instance that is an extension of S?

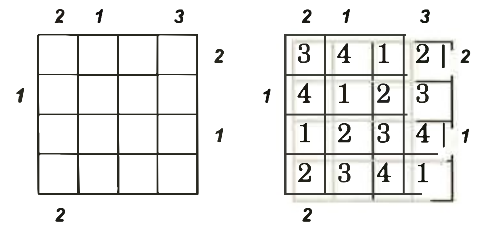

Fig. 1 (a) a 4 × 4 Building puzzle instance; (b) a solution to the instance.

In the present paper, we provide a stronger result on the complexity of the Building puzzle. Concentrating on one line that has a building number b on one of its ends, we consider the following question; can we complete all the empty cells in the line so that the Building condition with respect to b is satisfied?

For the sake of simplicity, we take up the first row and assume that b is on the left end. We can ignore empty cells in the second row to the n-th row. This is because, if we could complete the empty cells in the first row anyhow, it could be accomplished without assigning any integer to the empty cells in the second row to the n-th row.

We call the problem Single Lined Building Puzzle (SLBP), which is summarized as follows.

Single Lined Building Puzzle (SLBP)
Input: An $n \times n$ partial Latin square S and a Building number b on the left end of the first row.
Question: Is it possible to fill the empty cells in the first row with integers in $[n]$ so that the following two conditions are satisfied?

- *all-different condition*; i.e., every integer in $[n]$ appears exactly once (along with S).
- Building condition with respect to b; i.e., exactly b buildings are seen from the left end.

A *solution to the SLBP instance* (S, b) is a complete assignment of integers in $[n]$ to the empty cells in the first row that satisfies

[1] Otaru University of Commerce, Otaru, Hokkaido 047–8501, Japan
[a)] haraguchi@res.otaru-uc.ac.jp

the above two conditions.

The following theorem is the main contribution of the paper.

Theorem. *The problem SLBP is NP-complete.*

The remainder of the paper is devoted to the proof.

The theorem shows that it is computationally hard to complete even a single line, which is a stronger claim than [3]. It may also suggest the essential hardness of the Building puzzle, compared with other Latin square completion-type puzzles, such as the partial Latin square extension (PLSE) problem and Sudoku. Similarly to the Building puzzle, their decision problem versions require the decision as to whether a given partial solution can be extended to a complete solution under their own constraints, which are known to be NP-complete [1], [4]. However, as opposed to the Building puzzle, their single-lined versions can be solved in polynomial time by means of bipartite perfect matching. The Building puzzle is NP-hard, even in the single-lined version.

2. Proof of the Theorem

The problem SLBP is in NP since the size of a solution is at most n and we can check in polynomial time whether it is feasible or not.

We give a reduction from a variant of SAT, the problem called Cubic Monotone Not-All-Equal (2, 3)-SAT, which is known to be NP-complete [2].

2.1 Preliminaries

We introduce the problem Cubic Monotone Not-All-Equal (2, 3)-SAT. Let $X = \{x_1, \ldots, x_N\}$ denote a set of N Boolean variables. For a variable $x \in X$, x is called the *positive literal* and $\bar{x}$ is called the *negative literal*. A *clause* is a subset of literals over X. A *truth assignment* for X is denoted by $\tau : X \to \{\mathrm{T}, \mathrm{F}\}$. Given a true assignment τ, if $\tau(x) = \mathrm{T}$, then the positive literal x takes true and the negative literal $\bar{x}$ takes false. If $\tau(v) = \mathrm{F}$, then x takes false and $\bar{x}$ takes true. A clause is called *not-all-equal under* τ if there are two literals in the clause that take different truth values under τ. The problem Cubic Monotone Not-All-Equal (2, 3)-SAT is defined as follows.

Cubic Monotone Not-All-Equal (2, 3)-SAT

Input: A set $X = \{x_1, \ldots, x_N\}$ of N Boolean variables and a collection $C = \{C_1, C_2, \ldots, C_M\}$ of M clauses over X such that:

(monotone) all literals over C are positive;

(cubic) each variable appears as a literal in exactly three clauses;

(2,3) each clause C_b contains either two or three literals, i.e., $|C_b| \in \{2, 3\}$.

Question: Is there a truth assignment $\tau : X \to \{\mathrm{T}, \mathrm{F}\}$ such that every clause is not-all-equal under τ?

A SAT instance is *NAE-satisfiable* if there is a truth assignment under which every clause is not-all-equal. We partition C into $C^{(2)}$ and $C^{(3)}$ so that $C^{(h)}$ $(h \in \{2,3\})$ is the subcollection of h-clauses (i.e., those that contain h literals); $C^{(h)} = \{C_b \in C : |C_b| = h\}$.

Concerning the $n \times n$ grid, we denote the cell in the i-th row and in the j-th column by (i, j). For an empty cell (i, j) of a partial Latin square, if an integer k appears neither in the row i nor in the column j, then we say that k is *assignable to* (i, j), or equivalently, that (i, j) has k as a *candidate*.

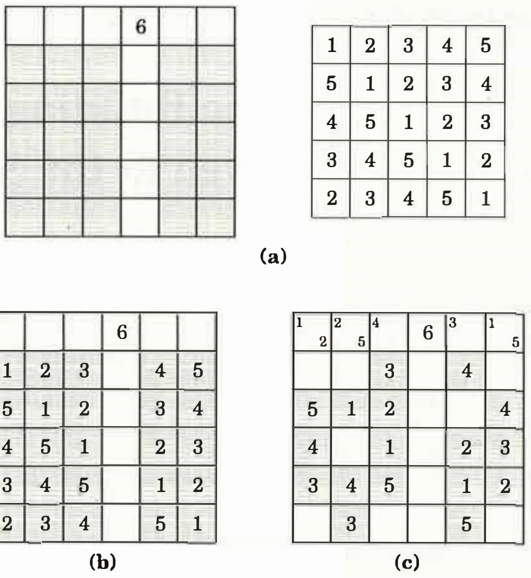

Fig. 2 Construction of a partial solution S ($n = 6$, $N = 2$): **(a)** the $n \times n$ partial Latin square such that the integer n is assigned to $(1, n - N)$ and an arbitrary $(n-1) \times (n-1)$ Latin square; **(b)** the $(n-1) \times (n-1)$ Latin square is spread over shaded cells; **(c)** some integers are removed, and small digits in the first row indicate the candidates that the empty cells have.

An *independent set* in a graph is a subset of vertices such that no two of them are adjacent to each other. The largest cardinality is called the *independence number*, and an independent set that achieves this number is called a *maximum independent set*, which we abbreviate into an MIS.

2.2 Overview

We are ready to transform a given SAT instance into an SLBP instance (S, b). The SLBP instance is built on the $n \times n$ grid with $n = 10|C^{(2)}| + 30|C^{(3)}| + N + 1$, where the Building number b is set to $b = 5|C^{(2)}| + 16|C^{(3)}| + 1$. Clearly, the size of the SLBP instance is polynomial with respect to that of the SAT instance.

We initialize the partial Latin square S as follows.

- We assign n (i.e., a highest building) to $(1, n - N)$.
- We leave every $(1, j)$ $(j \neq n - N)$ and every $(i, n - N)$ $(i \neq 1)$ empty.
- Taking an arbitrary $(n-1) \times (n-1)$ Latin square that has integers in $[n-1]$, we spread it over the second row to the n-th row, and the first column to the n-th column except the $(n - N)$-th column.

The initialization is illustrated in **Fig. 2** (a) and (b).

At this point, no value is assignable to every empty cell $(1, j)$. We will make a certain $k \in [n-1]$ assignable to $(1, j)$. To do this, we have only to remove k from the column j of S. Hence, we can let $(1, j)$ have arbitrary integers in $[n-1]$ as the candidates, by removing values from the grid appropriately.

We remove integers from S so that the following conditions are satisfied in the first row:

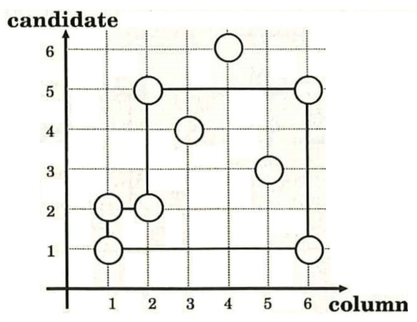

Fig. 3 The CC-graph constructed from the partial solution in Fig. 2 (c).

(I) Each empty cell has either one or two candidates.
(II) Each integer in $[n-1]$ appears as a candidate either once or twice over the first row.
(III) If an integer k appears as a candidate once, say in $(1, j)$, then $(1, j)$ has k as the only candidate. Otherwise, that is, if k appears as a candidate twice, say in $(1, j)$ and $(1, j')$, then both cells have two candidates respectively.

Figure 2 (c) illustrates these conditions. (I) We see that each empty cell has either one or two candidates. (II) The integers from 1 to 5 appear as candidates once or twice. (III) The integers 3 and 4 appear only in $(1,5)$ and $(1,3)$, respectively. These two cells have exactly one candidate. On the other hand, the integers 1, 2 and 5 appear twice over the remaining empty cells, that is $(1,1)$, $(1,2)$ and $(1,6)$. These cells have two candidates.

We do not construct S explicitly. Instead, we construct a graph that represents all appearing candidates, which we call the *column-candidate graph* (*CC-graph*). In the CC-graph, there is a vertex $v_{j,k}$ whenever a cell $(1, j)$ has a candidate k, and there is an edge between $v_{j,k}$ and $v_{j',k'}$ whenever they are "incompatible"; we say that two different vertices $v_{j,k}$ and $v_{j',k'}$ are *incompatible* if we cannot assign k to the cell $(1, j)$ and k' to the cell $(1, j')$ at the same time. This occurs when and only when either $j = j'$ or $k = k'$ holds. We also include the isolated vertex $v_{n-N,n}$ in the graph, the vertex for the highest building at the cell $(1, n-N)$. A vertex is regarded as an integral point on the 2D plane and an edge is drawn along with a grid line. In **Fig. 3**, we show the CC-graph that is constructed from the example of Fig. 2 (c).

Due to the conditions (I) to (III), the degree of each vertex $v_{j,k}$ is either zero or two. Hence, the CC-graph consists of isolated vertices and cycles. The length of every cycle is even since it consists of an alternation of horizontal and vertical edges.

Claim 1. *The independence number of the CC-graph is n.*

Proof. For every column $j \in [n]$, if there is only one vertex, then it is isolated, and it is included in every MIS. Otherwise, i.e., if there are two vertices, they are included in an even cycle, and thus one of them is included in every MIS. Since exactly one vertex is chosen from a column and there are n columns, the independence number is n. □

Claim 2. *Let I denote the collection of MISs in the CC-graph for a certain S, and A denote the collection of all-different assignable assignments of integers in $[n]$ to the n cells in the first row. Then I and A have one-to-one correspondence.*

Proof. Let us denote an arbitrary MIS by $\{v_{1,k_1}, \ldots, v_{n,k_n}\}$. Each k_j $(j \in [n] \setminus \{n-N\})$ is assignable to an empty cell $(1, j)$, whereas the integer $k_{n-N} = n$ is already assigned to $(1, n-N)$ by S. From the construction of S, k_j does not appear in the second row to the n-th row of the column j, and the integers $k_1, \ldots, k_n$ are all-different. Hence, we have an all-different assignable assignment, by assigning k_j to $(1, j)$. The converse is immediate. □

Following this claim, the remaining task is to show the way to construct the CC-graph from the given SAT instance so that the SAT instance is NAE-satisfiable iff there is an MIS in the CC-graph such that the Building condition is satisfied.

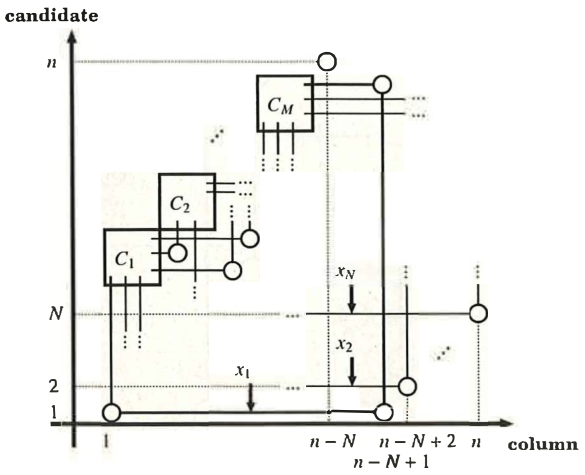

Fig. 4 Overview of the CC-graph layout.

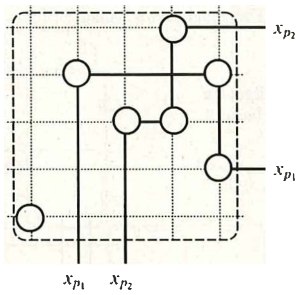

Fig. 5 The 5×5 component $P(x_{p_1}, x_{p_2})$.

2.3 Construction of the CC-Graph

In **Fig. 4**, we overview how we lay out the CC-graph on the 2D plane. We construct the CC-graph so that there are exactly N cycles, each of which is the "gadget" for a Boolean variable in the SAT instance. Two or three cycles cross each other intricately in a certain part of the 2D plane, which is the gadget for a 2- or 3-clause. The clause gadgets are indicated by bold squares.

The cycle for a Boolean variable x_p passes a vertex at $(n-N+p, p)$; see the lower-right part of Fig. 4. It goes into and out of the clause gadgets that include x_p, with hor ontal and vertical edges being alternated. Figure 4 assumes that x_1 appears in C_1, C_M and a certain other clause.

The M clause gadgets are allocated in a stair-like way, as in Fig. 4. The gadgets are different between 2-clauses and 3-clauses, but both of them are built by connecting copies of a certain component. The component is a subgraph in a 5×5 subgrid, which is shown in **Fig. 5**. Denoted by $P(x_{p_1}, x_{p_2})$, the component contains a part of the cycle for x_{p_1} and a part of the cycle for x_{p_2}. Note that

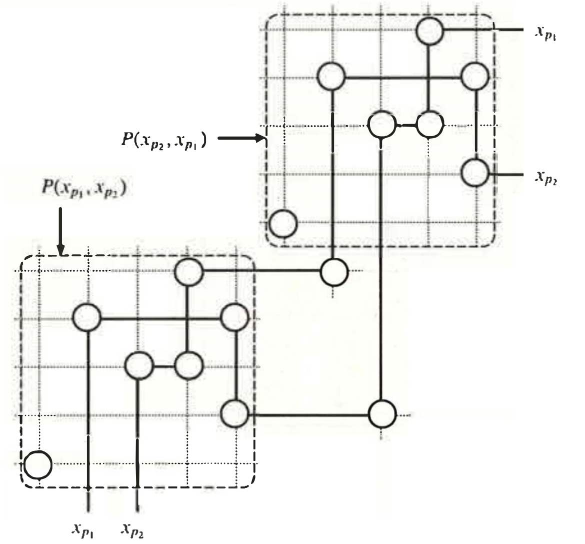

Fig. 6 The gadget for a 2-clause $\{x_{p_1}, x_{p_2}\}$.

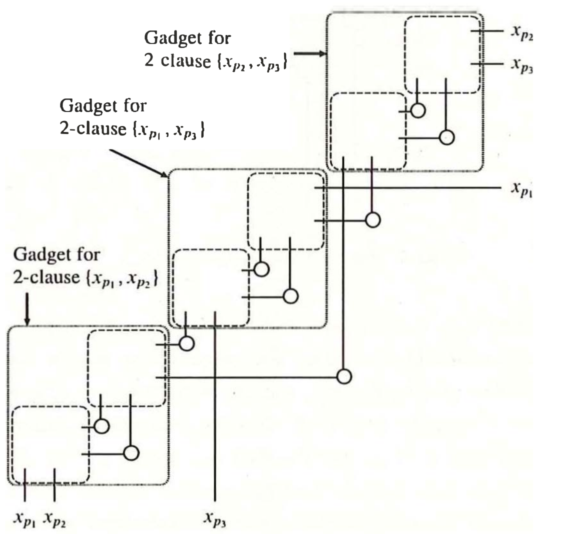
Gadget for 2 clause $\{x_{p_2}, x_{p_3}\}$
x_{p_2}
x_{p_3}
Gadget for 2-clause $\{x_{p_1}, x_{p_3}\}$
x_{p_1}
Gadget for 2-clause $\{x_{p_1}, x_{p_2}\}$
x_{p_1} x_{p_2}
x_{p_3}

Fig. 7 The gadget for a 3-clause $\{x_{p_1}, x_{p_2}, x_{p_3}\}$.

the lower-left vertex is isolated.

The gadget for a 2-clause $\{x_{p_1}, x_{p_2}\}$ is a subgraph in a 10×10 subgrid, which is constructed by connecting two copies of the component, $P(x_{p_1}, x_{p_2})$ and $P(x_{p_2}, x_{p_1})$, in the way of **Fig. 6**.

The gadget for a 3-clause $\{x_{p_1}, x_{p_2}, x_{p_3}\}$ is a subgraph in a 30×30 subgrid, which is constructed as follows; First we decompose the clause into three 2-clauses, that is $\{x_{p_1}, x_{p_2}\}$, $\{x_{p_1}, x_{p_3}\}$ and $\{x_{p_2}, x_{p_3}\}$. Then we connect the gadgets for the three 2-clauses in the way of **Fig. 7**. The resulting gadget is contained in a 30×30 subgrid as it is made of three 2-clause gadgets.

We see that the CC-graph is contained in the $n \times n$ grid, where $n = 10|C^{(2)}| + 30|C^{(3)}| + N + 1$.

Obviously, the partial Latin square S that corresponds to the CC-graph constructed in this way satisfies the conditions (I) to (III).

2.4 Connection between SAT and SLBP

Let us establish one-to-one correspondence between a truth assignment and an MIS. An MIS is the union of MISs over the connected components. An isolated vertex belongs to every MIS. An even cycle has two MISs. A horizontal edge in the cycle is either on the upper or lower side of the rectilinear polygon that the cycle makes. For example, in Fig. 3, we see three horizontal edges. The edges $(v_{1,2}, v_{2,2})$ and $(v_{2,5}, v_{6,5})$ are on the upper side of the polygon, while $(v_{1,1}, v_{6,1})$ is on the lower side. Concerning the MISs, one easily sees the following;

- One MIS consists of the left endpoints of the upper-sided horizontal edges, and the right endpoints of the lower-sided horizontal edges; in Fig. 3, it is $\{v_{1,2}, v_{2,5}, v_{6,1}\}$.
- The other MIS consists of the right endpoints of the upper-sided horizontal edges, and the left endpoints of the lower-sided horizontal edges; in Fig. 3, it is $\{v_{2,2}, v_{6,5}, v_{1,1}\}$.

Associating the MISs with the truth values, we call the former the true MIS, and the latter the false MIS. Since one of them belongs to an MIS of the entire CC-graph independently from cycle to cycle, there are 2^N MISs in the CC-graph, each of which corresponds to a truth assignment.

The clause gadgets are composed of copies of the 5×5 component of Fig. 5. The clause gadgets are allocated in a stair-like way, and within each gadget, copies of the component are allocated also in a stair-like way. Observe that the "buildings" we can see from the left end are only ones in the clause gadgets and the highest building at $(1, n - N)$. We cannot see any other building.

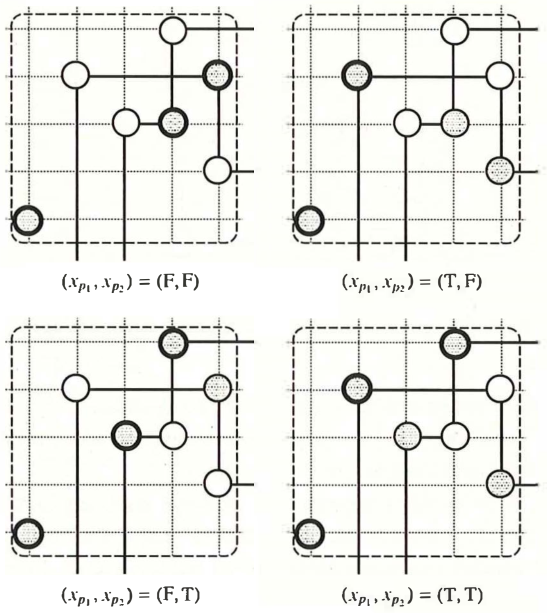
$(x_{p_1}, x_{p_2}) = (\mathrm{F}, \mathrm{F})$
$(x_{p_1}, x_{p_2}) = (\mathrm{T}, \mathrm{F})$
$(x_{p_1}, x_{p_2}) = (\mathrm{F}, \mathrm{T})$
$(x_{p_1}, x_{p_2}) = (\mathrm{T}, \mathrm{T})$

Fig. 8 Which buildings can be seen in the 5×5 component.

For a component $P(x_{p_1}, x_{p_2})$, **Fig. 8** shows which buildings can be seen when we set $(x_{p_1}, x_{p_2}) = (\mathrm{F}, \mathrm{F})$, (T, F), (F, T) and (T, T), respectively. In each figure, the shade indicates vertices in the corresponding MISs. A vertex corresponds to a building. In particular, an upper vertex corresponds to a higher building. Recall that a building hides any lower buildings on the right side from a viewer on the left side. Buildings that can be seen by the viewer are indicated in boldface. The point is that we see two buildings only when $(x_{p_1}, x_{p_2}) = (\mathrm{T}, \mathrm{F})$, and in the other cases, we see three buildings.

Then in **Table 1**, we show the number of buildings that can

Table 1 The number of buildings that can be seen in a clause gadget.

(2-clause)

x_{p_1}	x_{p_2}	Number
F	F	6
F	T	5
T	F	5
T	T	6

(3-clause)

x_{p_1}	x_{p_2}	x_{p_3}	Number
F	F	F	18
F	F	T	16
F	T	F	16
F	T	T	16
T	F	F	16
T	F	T	16
T	T	F	16
T	T	T	18

be seen in a clause gadget for every truth assignment. At least 5 (= 3 + 2) buildings are seen in a 2-clause gadget, and at least 16 (= 5 + 5 + 6) buildings are seen in a 3-clause gadget. Observe that buildings of the smallest number are seen when and only when the clause is not-all-equal under the assignment.

Over the first row, at least $5|C^{(2)}| + 16|C^{(3)}| + 1$ buildings are seen regardless of a truth assignment, where the last term is due to the highest building at $(1, n - N)$. This lower bound is equal to b. It is tight and achieved only by a truth assignment such that buildings of the smallest number are seen in the respective gadgets. Therefore, there is a solution to the SLBP instance iff there is a truth assignment for the SAT instance under which every clause is not-all-equal.

To construct the SLBP instance, we have only to lay out the clause gadgets in the manner of Fig. 4 and then to connect them by introducing the variable gadgets. The transformation time is clearly polynomial with respect to the size of the SAT instance.

References

[1] Colbourn, C.: The complexity of completing partial Latin squares, *Discrete Applied Mathematics*, Vol.8, pp.25–30 (1984).

[2] Dehghan, A., Sadeghi, M.-R. and Ahadi, A.: On the Complexity of Deciding Whether the Regular Number is at Most Two, *Graphs and Combinatorics*, Vol.31, No.5, pp.1359–1365 (online), DOI: 10.1007/s00373-014-1446-9 (2015).

[3] Iwamoto, C. and Matsui, Y.: Computational Complexity of Building Puzzles, *IEICE Trans. Fundamental of Electronics, Communications and Computer Sciences*, Vol.E99-A, No.6, pp.1145–1148 (2016).

[4] Yato, T. and Seta, T.: Complexity and Completeness of Finding Another Solution and Its Application to Puzzles, *IEICE Trans. Fundamentals*, Vol.E86-A, No.5, pp.1052–1060 (2003).

Kazuya Haraguchi received his B.E., Master of Informatics, and Doctor of Informatics from Kyoto University, in 2001, 2003 and 2007, respectively. He is currently with the Department of Information and Management Science, Faculty of Commerce, Otaru University of Commerce. His interests include discrete algorithms, discrete optimization, and their application to artificial intelligence, operations research and recreational mathematics.

Ryoya Tanaka received his Bachelor of Commerce from Otaru University of Commerce in 2017. He currently works for The Japan Post Bank.