Computational complexity and bounds for Norinori and LITS

Michael Biro [*] Christiane Schmidt [†]

Abstract

Norinori (aka Dominnocuous) and LITS (aka Nuruomino) are pencil-and-paper puzzles played on $m \times n$ square grids. In this paper we show that both Norinori and LITS are NP-complete and that their associated counting problems are #P-complete. Furthermore, we display $m \times n$ boards for each game that have unique solutions using the minimal number of polyomino regions.

1 Introduction

Norinori and *LITS* are a pair of related pencil-and-paper puzzles, made popular by the Japanese publisher Nikoli [3, 4]. The games are each played on an $m \times n$ square grid that has been partitioned into connected polyomino regions. To solve each puzzle, the player is required to place black squares in the polyomino regions to satisfy certain conditions.

In Norinori, the solver places black squares in the polyominos such that the final board satisfies the following two properties:

1. Each black square has exactly one black neighbor.
2. There are exactly 2 black squares in each polyomino region.

See Figure 1(a)/(b) for an example Norinori board and its solution. When discussing placing black squares, it is often useful to think of the player as placing black dominos as a basic move, with no two dominos adjacent, see Figure 2(a). A domino may span two polyomino regions, see Figure 1(b).

In LITS, the solver places black squares in the regions such that the final board satisfies the following properties:

1. The black squares form a connected polyomino.
2. Each polyomino region contains a connected black tetromino.
3. No two congruent tetrominos are adjacent.
4. Black squares may not build 2×2 squares.

See Figure 1(c)/(d) for an example LITS board and its solution. For LITS, it is useful to think of the player as placing one of the four legal black tetrominos, see Figure 2(b), which resemble the letters in

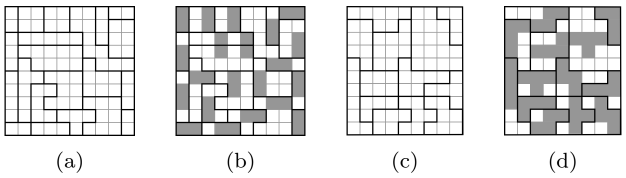
Figure 1: Example Norinori board (a) and LITS board (c) with solution in (b) and (d), respectively.

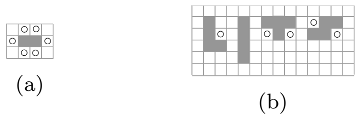
Figure 2: Basic shapes for Norinori (a) and LITS (b), circles give white squares that may not be filled in.

LITS. A variant of LITS, without the condition that no two congruent tetrominos are adjacent, was considered by McPhail [2] and shown to be NP-complete.

In this paper we will show that the problem of solving Norinori and LITS boards is NP-complete and that the problem of counting the number of solutions to a Norinori or LITS board is $\#P$-complete. For our reductions, we will use the PLANAR 1-IN-3-SAT PROBLEM, a well known NP-complete and $\#P$-complete problem [1].

Definition 1 *An instance F of the PLANAR 1-IN-3-SAT problem is a Boolean formula in 3-CNF consisting of a set $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of m clauses over n variables $\mathcal{V} = \{x_1, x_2, \ldots, x_n\}$. Clauses in F contain variables and negated variables, denoted as* literals. *A clause is satisfied if and only if it contains exactly one* `true` *literal, and the formula F is true if and only if all its clauses are satisfied. The variable-clause incidence graph G is planar and it is sufficient to consider formulae where G has a rectilinear embedding.*

In addition, we explore combinatorial questions on both puzzles: the smallest number of regions in an $n \times m$ board that has a unique solution. We show that for most boards only 3 regions are required.

2 NP-completeness of Norinori

Theorem 1 *Determining if a Norinori board is solvable is NP-complete and counting the number of solutions is #P-complete.*

Proof. The proof is by reduction from PLANAR 1-IN-3-SAT. Given an instance F of planar 1-in-3-

[*]Department of Mathematics and Statistics, Swarthmore College, `mbiro1@swarthmore.edu`.
[†]Communications and Transport Systems, ITN, Linköping University, `christiane.schmidt@liu.se`.

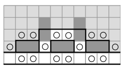

Figure 3: Gadget to fix the 2 filled-in squares in each whitespace (here, for clarity, indicated in light gray): at one corridor we add the region in U-shape.

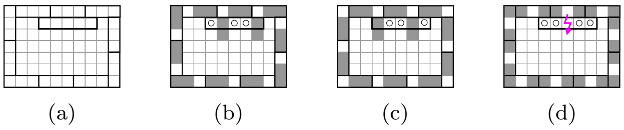

Figure 4: (a) Variable loop, with the two feasible solutions (b)/(c). We associate the solution in (b) and (c) with a truth setting of "false" and "true", respectively. The 2×5 region in the center face ensures that for (b) and (c) the filled-in squares are fixed, and it renders the third solution for the loop of the 1×3 rectangles infeasible (d).

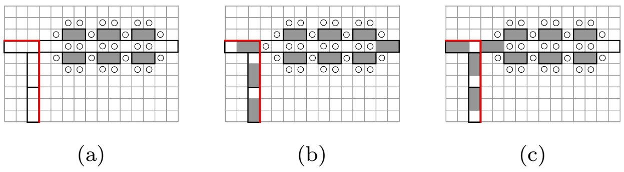

Figure 5: (a) Corridor gadget with enforced white/black pixels. The connected variable gadget is located within the red boundary. If the variable is set to "false", only a 2×1-block pushed to the end of the corridor is a feasible fill-in (b). If the variable is set to "true", it is possible to place a 2×1-block directly at the connection to the variable loop, leaving the last pixel of the corridor white (c).

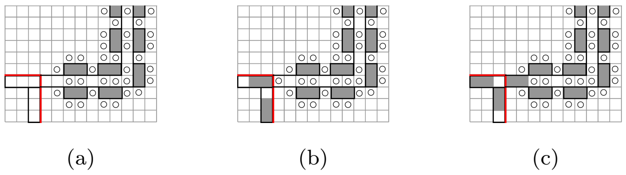

Figure 6: (a) The bend gadget with enforced white and black pixels. The connected variable gadget is located within the red boundary. (b) "false", (c) "true'.

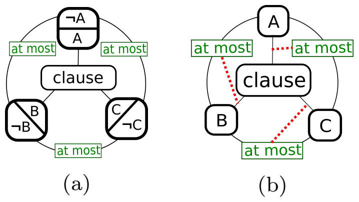

Figure 7: 1-in-3 gadget construction for Norinori (a) and LITS (b). The red, dotted lines indicate the connectors from the "at most"-gadget.

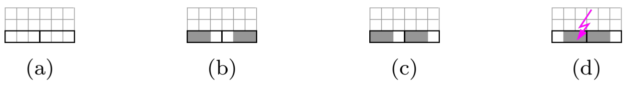

Figure 8: (a) The at-most gadget: corridors from two negated variables enter. If both corridors enter with a setting of "true" (variable setting of "false") (b), or if the corridors have different truth settings (c), there exists a feasible solution. (d) If both corridors enter with a variable setting of "true" the board cannot be completed.

SAT with incidence graph G, we show how to turn a rectilinear planar embedding of G into a Norinori board B such that a solution to B yields a solution to F, thereby showing NP-completeness. Furthermore, there will be a one-to-one correspondence between solutions of B and solutions of F, showing #P-completeness.

We begin by constructing a representation of variables and their negations, then show how to propagate and bend the variable values using 'wires' and combine the wires to form clauses. Note that throughout, the constructed gadgets have a single solution for any given variable assignment, which will make this reduction parsimonious.

The polyomino regions not incorporated into our gadgets will not affect the gadget solutions, as for any open region we use a **face gadget**, shown in Figure 3, to force the region to contain two black squares, which disallows any others. Therefore, there can be no interference between our gadgets and the polyomino regions surrounding them.

The basic variable gadget is created out of 1×3 polyominos, and forms a **variable loop**, shown in Figure 4. Each variable loop has two possible solutions, corresponding to setting the variable as "true" or "false", and the domino placement is completely determined by the variable assignment. (A third solution placing the dominoes over the region boundaries

From each variable loop, we can propagate the variable value, creating a **corridor gadget**, shown in Figure 5. Note that a "false" assignment for a variable forces a domino to be placed at the far end of the corridor, while a "true" assignment leaves open the possiblity of placing a domino at either end. This will be accounted for in the final clause gadget.

By choosing the appropriate place to connect the corridor gadget to the variable loop, we can generate wires for both the variable value and its negation. That is, no separate negation gadget is needed. In addition, we can connect several corridor gadgets to the variable loop. Furthermore, we can create 90° turns in the corridor gadget using the **bend gadget**, shown in Figure 6.

To combine the corridor gadgets into clauses, we use the **1-in-3 gadget** in Figure 7(a), where the **at-most** and the **clause gadget** is shown in Figure 8 and 9, respectively. The 1-in-3 gadget uses two negated copies of each variable assignment and combines them with the at-most gadgets. This forces at most one of the variables' truth assignments to be true, while the center clause gadget requires at least one true assignment. Together, this forces exactly 1 true assignment, giving a solution to the instance F.

Given a solution to an $m \times n$ Norinori board, it can obviously be verified in polynomial time. □

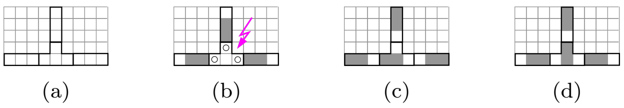
Figure 9: Norinori clause gadget (a): if all variables do not fulfill the clause, the clause region cannot be filled (b). (c), (d): feasible solutions for three and one variable fulfilling the clause.

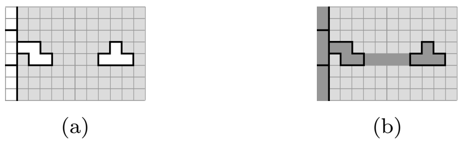
Figure 10: Gadget for each face of the arrangement: at one corridor of the whitespace (here, for clarity, indicated in light gray) we add a T in distance 4 from an S (a), this enforces the placement of an I to connect the T.

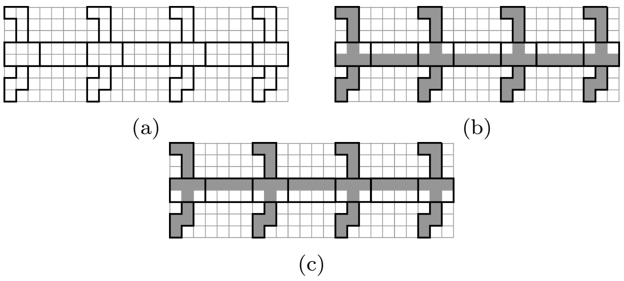
Figure 11: Variable gadget (a), with the two possible feasible solutions (b),(c). We associate one with a truth setting of “false” (b) and the other with “true” (c).

3 NP-completeness of LITS

Theorem 2 *Determining if a LITS board is solvable is NP-complete and counting the number of solutions is #P-complete.*

Proof. The proof is again by reduction from PLANAR 1-IN-3-SAT. The structure of the LITS reduction is the same as in the previous proof for Norinori. The properties of a final LITS board enforce unique feasible solutions for the following gadgets.

We begin by noting that the polyomino regions not incorporated into our gadgets will not affect the gadget solutions, as for any open region we use a **face gadget**, shown in Figure 10, to force the region to contain a connecting 4×1 tetromino, which disallows any others. Therefore, there can be no interference between our gadgets and the polyomino regions surrounding them.

The basic **variable gadget** is created out of a repeating pattern of polyominos, shown in Figure 11. Each variable gadget has two possible solutions, corresponding to setting the variable as ”true” or ”false”, which are completely determined by the tetromino placement. The variables are connected by 4×1 tetrominos, similar to the face gadget. This ensures that the resulting set of black squares can be connected.

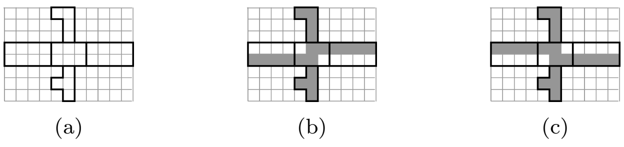
Figure 12: (a) The NOT gadget. (b),(c) The wires connected by the NOT gadget always satisfy opposite truth assignments.

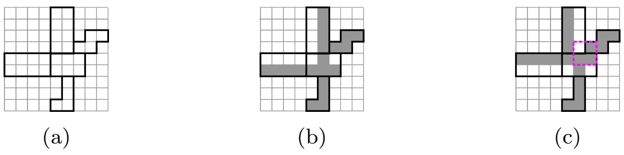
Figure 13: (a) The bend gadget. As the 3x2 rectangle is adjacent to an enforced L, S and I (length of the 2x4 rectangle), it must be filled in with a T. In (b) the other T would not connect to the incoming I; the other outgoing I would leave the S unconnected. In (c) the other T would not connect to the incoming I; the other outgoing I would result in a filled 2x2 square shown in dashed pink.

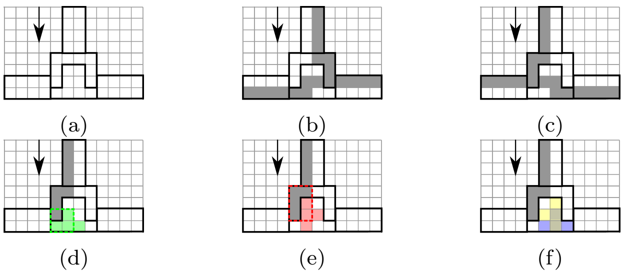
Figure 14: (a) The split gadget, with the two different truth settings (b), (c). The central shape (adjacent to an L and an I) must contain either a S or a T. No position of the T is possible (d)-(f).

We can propagate the variable assignment, creating a **corridor gadget**, by linearly repeating the pattern in the variable gadget. Negating a variable corresponds to inserting a **NOT gadget** into the corridor, as in Figure 12. To create 90° turns in the corridor gadget, we use the **bend gadget**, shown in Figure 13, and to create copies of the variable assignment we use the **split gadget**, shown in Figure 14.

To combine the corridor gadgets into clauses, we use the **1-in-3 gadget** in Figure 7(b), where the **at-most** and the **clause gadget** is shown in Figure 15 and 16, respectively. The 1-in-3 gadget combines two copies of each variable’s assignment with the others using the at-most gadgets. This forces at most one of the variables’ truth assignments to be true, while retaining the connectivity condition. Then, the center clause requires at least one true assignment and together these force exactly 1 true assignment, giving a solution to the instance F.

Given a solution to an $m \times n$ LITS board, it can obviously be verified in polynomial time. ☐

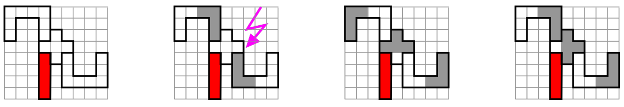
(a) (b) (c) (d)

Figure 15: (a) At-most gadget: the two C-shaped regions connect to the variable corridors (as in the clause gadget); the red I is a connector (corridor of (enforced) I- and T-shapes that connects to an S or L of a corridor on that face), as indicated by the red lines in Fig. 7(b). If both variables have a truth setting fulfilling the clause (b), no T in the central cross can be placed without filling a 2x2-square. If both variables do not fulfill the clause (c), or one of them does (d), a T can be placed.

4 Boards with unique solutions

Definition 2 *Define $U_N(n,m)$ to be the minimal number of regions among all $n \times m$ Norinori boards with unique solutions. Similarly, define $U_L(n,m)$ to be the minimal number of regions among all $n \times m$ LITS boards with unique solutions. Note that $U_N(n,m)$ and $U_L(n,m)$ need not exist for all n,m, in which case we say $U_N(n,m) = 0$ or $U_L(n,m) = 0$.*

Theorem 3 *The following values for $U_N(n,m)$ hold:*

1. *$U_N(n,1) = 0$ for $n \not\equiv 2 \mod 3$*
2. *$U_N(n,1) = \frac{n+1}{3}$ for $n \equiv 2 \mod 3$*
3. *$U_N(n,2) \leq \lceil \frac{n}{4} \rceil$ for $n \geq 3$*
4. *$U_N(n,m) = 3$ for all $n \geq 5, m \geq 3$.*

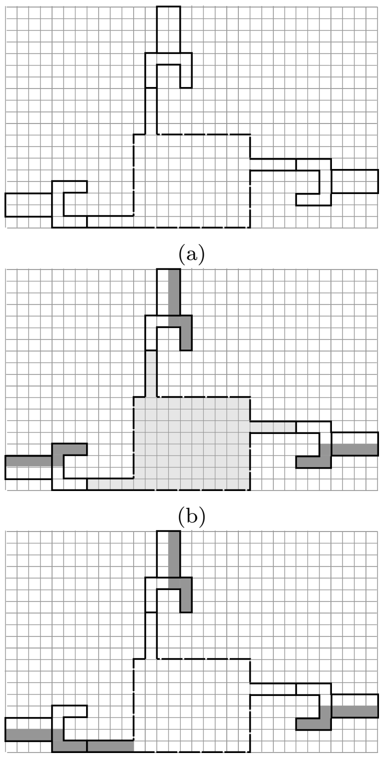
(a)
(b)
(c)

Figure 16: (a) The clause gadget: if all variables have truth settings that do not fulfill the clause (b), no tetromino in the light gray region can be connected; if at least one variable has a truth setting fulfilling the clause (c), an I can connect to the other tetrominoes.

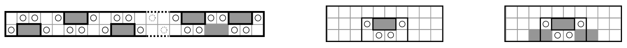
(a) (b) (c)

Figure 17: (a) For $m = 2$, $\lceil \frac{n}{4} \rceil$ regions can have a unique solution. (b)/(c) For $m \geq 3$ and $n \geq 5$ there exist 3 polyomino regions, such that the board has a unique solution.

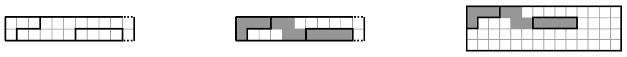
(a) (b) (c)

Figure 18: For $m \geq 2$ and $n \geq 10$ there exist 3 polyomino regions (a), such that the board has a unique solution (b). An example for m=4, n=14 (c).

Proof. For the case of an $n \times 1$ board, examine the leftmost domino in the completed board. Since the board has a unique solution, it must consist of the leftmost two squares, as otherwise we could move it left without making the board infeasible. Furthermore, the leftmost region can consist of either 2 or 3 squares, as otherwise we could move the domino right without making the board infeasible. Therefore, we generate an $(n-3) \times 1$ board by removing the leftmost three squares and, if necessary, modifying the second leftmost region by removing its leftmost square. The resulting board has the same unique solution as the original, restricted to its squares. We can then repeat the process until there is only one region and verify that the only $n \times 1$ board with one region that has a unique solution is the 2×1 board. Therefore, the only $n \times 1$ boards with a unique solution have $n \equiv 2 \mod 3$. Moreover, we remove one domino per three squares, so $U_N(n,1) = \frac{n+1}{3}$ in this situation.

For the other cases, see Figure 17 for constructions achieving the given bounds, with the observation that $U_N(n,m) > 2$ for $n \geq 5, m \geq 3$. ☐

Theorem 4 *$U_L(n,m) = 3$ for all $n \geq 10, m \geq 2$. In other words, 3 regions suffice to completely determine an $n \times m$ LITS board, as long as $n \geq 10$ and $m \geq 2$.*

Proof. See Figure 18. ☐

References

[1] M. E. Dyer and A. M. Frieze. Planar 3DM is NP-complete. *Journal of Algorithms*, 7(2):174–184, 1986.

[2] B. McPhail. Metapuzzles: Reducing SAT to your favorite puzzle. *CS Theory Talk*, 2007.

[3] Nikoli. http://www.nikoli.com/en/puzzles/norinori/, NIKOLI Co., Ltd. Accessed December 6, 2016.

[4] Nikoli. http://www.nikoli.co.jp/en/puzzles/lits.html, NIKOLI Co., Ltd. Accessed December 6, 2016.