Sto-Stone is NP-Complete

Addison Allen * Aaron Williams †

Abstract

Sto-Stone is a paper-and-pencil puzzle created by Japanese publisher Nikoli. A puzzle consists of an m-by-n grid whose squares are partitioned into connected 'rooms' each of which may have an associated number. The solver shades in squares of the grid, which form maximal 'stones' based on orthogonal connectivity. The goal is to shade squares so that (a) each room contains one stone, (b) individual stones do not cross between rooms, (c) numbered rooms contain a stone with exactly that number of squares, and (d) when the stones are "dropped" downward they perfectly fill the bottom half of the grid. We show that Sto-Stone is NP-complete. This is also true when rule (d) is weakened or omitted.

1 Introduction

This article proves the NP-completeness of a new paper-and-pencil puzzle by Japanese publisher Nikoli. The puzzle is *Sto-Stone* (ストストン) and it was introduced in *Puzzle Communication magazine* Volume 156 [1].

When discussing individual grid squares we use *adjacent* and *connected* to mean orthogonally adjacent and orthogonally connected, respectively.

1.1 Rules of the Puzzle

Sto-Stone is played on an m-by-n grid where m is even. The grid's squares are partitioned into connected "rooms" and the *size* of a room is its number of squares. A room may have a positive number w written in one of its squares, and in this case its *required weight* or *requirement* is w. A grid with these properties is a *board*.

The solver interacts with the puzzle by shading individual squares. The shaded squares partition into *stones* based on connectivity. In other words, any two shaded squares that are adjacent belong to the same stone. The *weight* of a stone is its number of shaded squares. The goal is to create stones subject to the following rules:

(S1) There is exactly one stone in each room. That is, in each room there are shaded squares and these squares are connected.

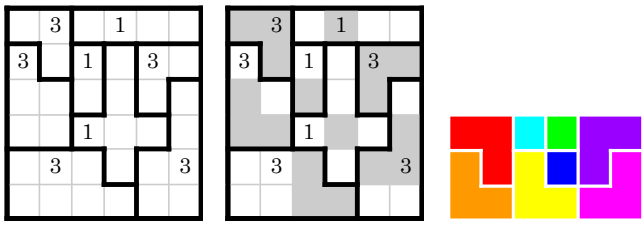
(S2) Shaded squares in different rooms are not adjacent. That is, stones can't be inside more than one room.

(S3) Rooms with requirement w have a weight w stone. That is, a room labeled w has w shaded squares.

(S4) When all stones are "dropped" downward they fill the bottom half of the grid with no gaps.

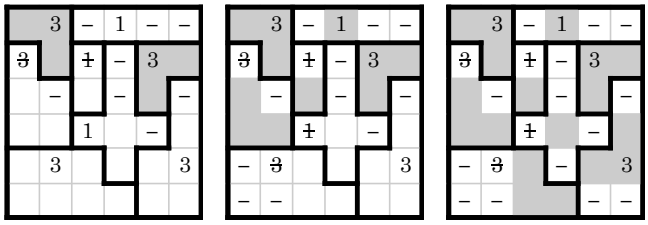
Rule (S4) requires clarification. When stones are dropped they move down as if influenced by gravity. Stones do not change shape when they are dropped, and all room boundaries are ignored during this time.

Figure 1 has a sample puzzle and Figure 2 illustrates the solving process. Figure 1 (c) visually verifies (S4), and - denotes a square that cannot be shaded.



(a) Board. (b) Solution. (c) Drop check.

Figure 1: The corrected version of Sto-Stone Puzzle 4 from Puzzle Communication Volume 162 [2].



(a) (S3) forces the above stones; (S2) contributes the −. (b) (S3) forces the above stones; (S2) and (S4) contribute the −. (c) (S3) forces the bottom-middle stones; (S4) forces the bottom-right.

Figure 2: Solving the Sto-Stone puzzle in Figure 1.

1.2 Drop Rules: Stone, Sand, Silt

We refer to (S4) as the *stone drop rule*. We also define a weaker *sand drop rule* as follows.

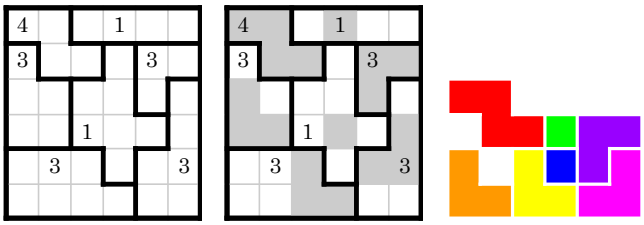
(s4) There are $\frac{m}{2}$ shaded squares in every column.

*Bard College at Simon's Rock, Massachusetts, aallen15@simons-rock.edu

†Bard College at Simon's Rock, Massachusetts, awilliams@simons-rock.edu

Notice that (s4) differs from (S4) in that it ignores the shape of the stones. In other words, the shaded squares are dropped independently like individual grains of sand. We refer to the lack of a drop rule as the *silt drop rule*. In other words, the shaded squares linger in the air like fine grains of silt.

The drop rule in Sto-Stone is somewhat unusual among Nikoli puzzles, and it has led to some initial confusion among puzzle designers, solvers, and academics. Figure 1 actually contains a *corrected* version of Sto-Stone Puzzle 4 from Puzzle Communication Volume 162 [2]. The originally published puzzle shown in Figure 3 can only be solved with the weaker drop rules.



(a) Board. (b) Solution with respect to SAND but not STONE. (c) Drop rule (S4) fails, but (s4) would pass.

Figure 3: The original version of Sto-Stone Puzzle 4 from Puzzle Communication Volume 162 [2].

The error was announced by `@nikoli_official` on twitter [8]. In response, `@postpostdoc` posted an early version of this article [9]. However, the authors did not properly understand rule (S4) at that time, and only established the hardness of (S1)-(S3) with (s4). The early version also allowed 'empty' rooms with no stone, and $w = 0$ requirements, which we now believe are invalid.

1.3 Decision Problems

We formalize three different puzzles based on the type of drop rule that is used. Each of these puzzles has an associated decision problems that takes a board B as input and answers 'yes' or 'no' depending on whether it can be solved using the rules for that puzzle.

- Nikoli's *Sto-Stone puzzle* uses rules (S1)-(S3) and drop rule (S4). The decision problem is STONE(B).
- The *Sto-Sand puzzle* uses rules (S1)-(S3) and drop rule (s4). The decision problem is SAND(B).
- The *Sto-Silt puzzle* uses rules (S1)-(S3) and no drop rule. The decision problem is SILT(B).

If STONE(B) is 'yes', then SAND(B) is 'yes'. Similarly, if SAND(B) is 'yes', then SILT(B) is 'yes'. Figure 3 gave an example board B in which STONE(B) is 'no' and both SAND(B) and SILT(B) are 'yes'.

All three decision problems are in NP because shading an m-by-n board can be done with $m \cdot n$ binary guesses, and each rule can be checked in $O(mn)$-time.

Remark 1 *The decision problems* STONE, SAND, *and* SILT *are all in NP.*

1.4 Popularity

Nikoli is currently promoting three new puzzles including *Sun or Moon* (月か太), *Pencils* (ペンシルズ), and Sto-Stone (ストストン). During a November 2017 poll held on twitter by `@nikoli_official`, the Sto-Stone puzzle ranked behind Sun or Moon in popularity. However, this has changed in a more recent poll from May 2018, as seen in Figure 4.

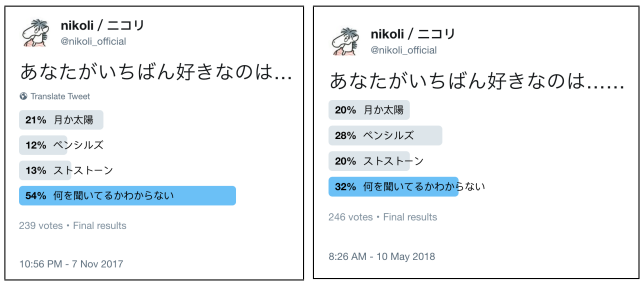


Figure 4: The popularity of three new Nikoli puzzles, where the bottom option translates to "I do not know".

Establishing the hardness of Nikoli puzzles has also been a popular pursuit in academia. An excellent resource on this general topic is *Games, Puzzles, and Computation* by Hearn and Demaine [6].

1.5 Outline

The article is organized as follows. Section 2 defines the NP-complete problem that we will use as a source problem. Section 3 introduces our gadgets and other preliminaries. Sections 4, 5, and 6 proves that SILT, SAND, and STONE are NP-complete, respectively. Section 7 concludes with final remarks and open problems.

2 Source Problem

This section defines the satisfiability problem used in our reduction. We also describe a slight variation to its standard representation.

2.1 Planar Monotone Rectilinear 3SAT

A *(Boolean) variable* is a variable that can be assigned TRUE or FALSE. If x_i is a variable, then its *positive literal* is x_i, and its *negative literal* is $\neg x_i$. A Boolean formula ϕ is in *3 conjunctive normal form (3CNF)* if it equals $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ where each *clause* C_i has the form $(\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$ and each $\ell_{i,j}$ is a literal.

A clause is *positive* or *negative* if it has only positive or negative literals, respectively. The 3CNF formula ϕ is *monotone* if each clause is either positive or negative.

A 3CNF formula is *planar* if the bipartite incidence graph of variables and clauses is planar. A *rectilinear embedding* of a planar monotone 3CNF formula is a drawing on a grid with the following properties:

- Variables and clauses are horizontal line segments.
- Vertical line segments connect variables to clauses.
- Variable line segments are on the same horizontal line called the *variable line*.
- Positive clauses are above the variable line, and negative clauses are below.

Rectilinear embeddings are drawn with their horizontal line segments vertically extended as in Figure 5.

The decision problem PLANAR MONOTONE RECTILINEAR 3SAT (PMR3SAT) takes a rectilinear embedding of a planar monotone 3CNF formula ϕ as input. A *'yes' instance* occurs when the variables can be assigned so that ϕ evaluates to TRUE. In this case, ϕ is *satisfiable*. Otherwise, ϕ is a *'no' instance* and is *unsatisfiable*. For brevity, we often refer to the input of PMR3SAT as the Boolean formula ϕ as opposed to a rectilinear embedding of it. Theorem 1 is by de Berg and Khosravi [4].

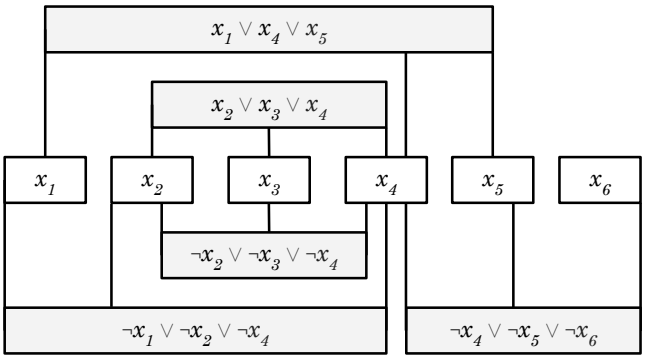


Figure 5: A 'yes' instance of PMR3SAT with $\phi = (x_1 \vee x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_4 \vee \neg x_5 \vee \neg x_6)$.

Theorem 1 ([4]) PMR3SAT *is NP-complete.*

When working with PMR3SAT we assume that the variables are ordered from left-to-right as $x_1, x_2, \ldots, x_n$ in the embedding. We also arrange each clause C as $(x_i \vee x_j \vee x_k)$ or $(\neg x_i \vee \neg x_j \vee \neg x_k)$ with the distinct indices satisfying $i < j < k$, and we refer to x_i, x_j, and x_j as the *left*, *middle*, and *right* literals in C, respectively.

2.2 Bent Representation

We will find it helpful to make the following cosmetic adjustments to the input to the PMR3SAT problem:

- Shrink each clause line by moving its left end and right end closer together by any small amount;
- Connections from clauses to positive left literals are redrawn as ⌜ lines. Similarly, negative left literals, positive right literals, and negative right literals are redrawn with ⌞, ⌝, and ⌟ lines, respectively.

We refer to this modified embedding as *bent rectilinear representation* since two-thirds of the connecting lines have a 90^o bend. Figure 6 shows the result of adjusting Figure 5 in this way.

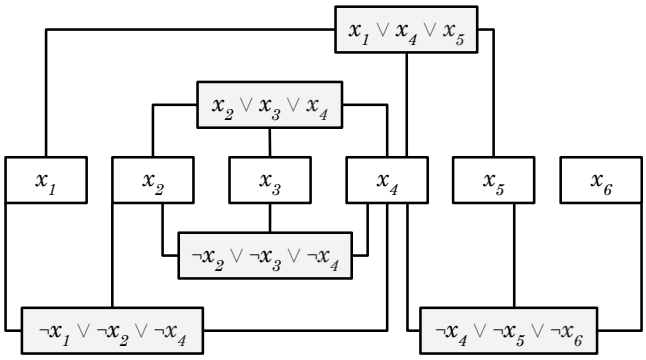


Figure 6: A bent embedding of Figure 5.

3 Gadgets and Preliminaries

In this section we introduce some conventions and terminology, and then present gadgets for Sto-Silt.

3.1 Grid Parity

A square in location (x,y) of the grid is *even* or *odd* based on the sum $x + y$, where the top-left square is in location (1, 1). In other words, the grid has an underlying even/odd checkerboard pattern.

Each room we create will have size at least two, so it will contain at least one even and one odd square. Therefore, we can use the following convention to make our figures more readable: If a room has required weight w, then w is written in a square whose parity is the same as w. In other words, odd requirements are written in odd locations, and even requirements are written in even locations. This convention extends back to Figure 1.

3.2 Rooms

In a partially shaded board B a room with a requirement w is *satisfied* if it has a stone of weight w, and otherwise it is *unsatisfied*. Furthermore, a room is *unsatisfiable* if it is impossible to satisfy the room by shading in additional squares while respecting the rules.

Rooms with size s and requirement w have *type* $s.w$. Our reductions will be primarily restricted to the following special room types.

- A room of type 2.1 is a *binary room*. A binary room can be satisfied in two ways.
- A room of type 3.2 is a *trinary room*. A trinary room can be satisfied in two ways.

- A room of type 3.1 is a *ternary room*. A ternary room can be satisfied in three ways.
- A room with no requirement is a *wild-card room*. These rooms do require at least one shaded square.

In the following subsections we will create gadgets that propagate decisions made at certain binary rooms to other binary rooms. When discussing these gadgets we use the following terminology and conventions.

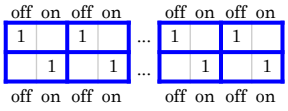
- An *input room* is a horizontal binary room with an *'on' square* to the right of an *'off' square*. A *positive* or *negative* input room has its 'on' square in an even or odd position, respectively.
- An *output room* is a horizontal or vertical binary room with specified 'on' and 'off' squares.

Input and output rooms are coloured red and blue, respectively. These rooms are 'on' if a stone is in their 'on' square, and are 'off' if a stone is in their 'off' square.

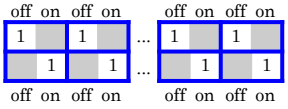
To simplify our figures we assume that empty regions on a board are wild-card rooms which are not drawn.

3.3 Variable Gadget

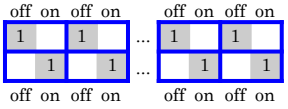
A variable gadget is designed to be satisfiable in one of two ways. Furthermore, this choice must be duplicatable so that it can be passed to any number of clause gadgets. To accomplish these goals we create a cycle of binary rooms. Our *variable gadget of width w* consists of a *positive row* with w positive input rooms, and below it is a *negative row* with w negative input rooms, as shown in Figure 7 (a). The top-left square is always placed on an even grid location.



(a) Variable gadget.



(b) Positive state.



(c) Negative state.

Figure 7: The variable gadget in (a) can be satisfied in exactly two ways (b)–(c).

Shading a single square anywhere in the gadget forces the entire gadget to be satisfied in a particular manner. The precise behavior and state of the gadget is defined Remark 2 and illustrated in Figure 7 (b)–(c).

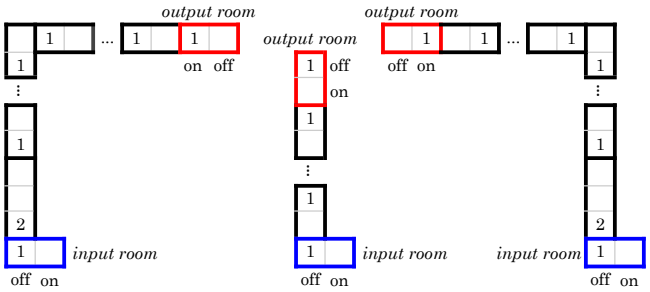
Remark 2 *In a solved Sto-Silt board, a variable gadget must be satisfied in one of two ways:*

- *Its* positive state *has positive input rooms 'on' and negative input rooms 'off'.*
- *Its* negative state *has positive input rooms 'off' and negative input rooms 'on'.*

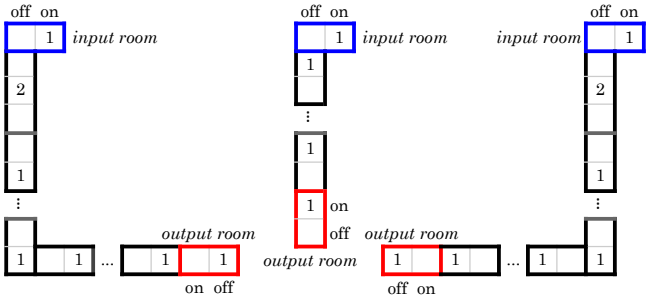
3.4 Wire Gadget

A wire gadget propagates the choice made in one binary room to another binary room. More specifically, the wire ensures a relationship between two specific squares on the board. Remark 3 outlines the main property of the wire gadget that we will construct.

Remark 3 *Binary input and output rooms that are connected by a wire in a solved Stone-Silt board have the following properties. If the input room is 'off', then the output room is also 'off'. If the input room is 'on', then the output room can be 'on' or 'off'.*



(a) Three types of positive wires.



(b) Three types of negative wires.

Figure 8: Wire gadgets connect an input room to an output room, and are paths of binary rooms and at most one trinary room. Shading the 'off' square of an input room forces the shading of the 'off' square in the connected output room.

Remark 3 is 'weak' since it only guarantees one direction, but this will be sufficient for our reduction. Now we define our wires with Figure 8 providing illustrations.

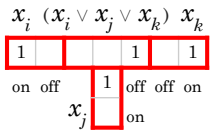
- A *positive/negative wire* is a path of binary and trinary rooms starting from the 'off' square of a positive/negative input room and ending at the 'on' square of a positive/negative output room.

A wire is *straight* if it travels vertically from an input room to a vertical output room. The other wires proceed vertically from an input room, then make a single right-turn or left-turn, and travel horizontally to a horizontal output room. The straight wires only use binary rooms, whereas the turning wires leave their output room with a single trinary room and then consist

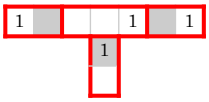
of binary rooms. As a result, the straight and turning wires reach their respective input rooms on opposite parity squares. All wire types are illustrated in Figure 8 and in each case it is easy to verify Remark 3.

3.5 Clause Gadget

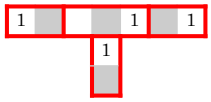
We base our clause gadgets on a ternary *clause room* and three binary output rooms. Each clause room is horizontal with output rooms adjacent to its left and right squares. In the *positive clause gadget* another output room is adjacent to the bottom of its middle square, whereas in the *negative clause gadget* it is adjacent to the top of its middle. The behavior of the gadget is given in Remark 4 and shown in Figures 9 and 10.



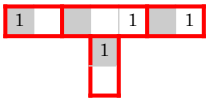
(a) Positive clause.



(b) Unsatisfiable if all 'off'.

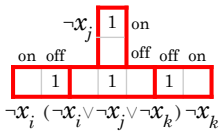


(c) Satisfiable if x_j is 'on'.

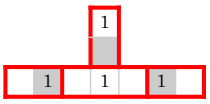


(d) Satisfiable if x_i is 'on'.

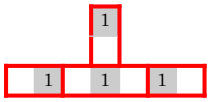
Figure 9: The positive clause gadget is satisfiable if and only if at least one input room is 'on'.



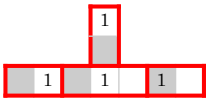
(a) Negative clause.



(b) Unsatisfiable if all 'off'.



(c) Satisfiable if x_j is 'on'.



(d) Satisfiable if x_i is 'on'.

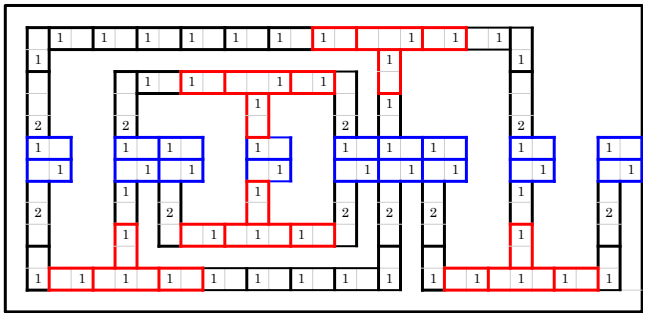
Figure 10: The negative clause gadget is satisfiable if and only if at least one input room is 'on'.

Remark 4 *In a solved Sto-Silt board, a clause gadget is satisfiable if and only if at least one of its adjacent output rooms is 'on'.*

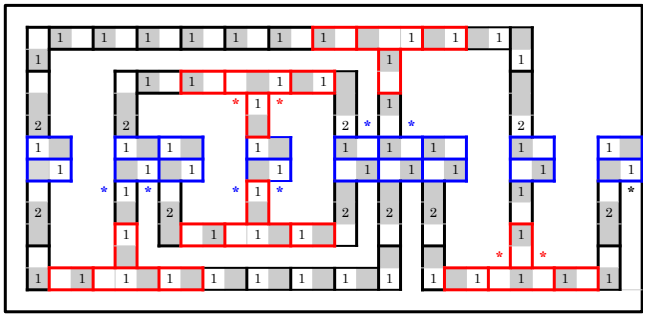
Note: In Theorem 2's proof we always satisfy clauses by shading their middle square if their middle wire is 'on'.

4 NP-Completeness of Sto-Silt

In this section we reduce PMR3SAT to SILT. An example of the reduction based on Figure 6 appears in Figure 11.



(a) The board $S(\phi)$ where wild-card rooms fill the area.



(b) A solution to $S(\phi)$ via $x_4 = x_5 =$ FALSE and $x_1 = x_2 = x_3 = x_6 =$ TRUE. The marked square * can be shaded in the outer wild-card room without violating (S2). Similarly, *'s and *'s mark all suitable squares along the straight wires.

Figure 11: The reduction of the PMR3SAT instance ϕ from Figure 6 to SILT(ϕ).

Suppose ϕ is an instance of PMR3SAT with p positive clauses and z negative clauses. Our reduction creates a board $B = S(\phi)$ whose rows are organized as follows:

- Row 1 is empty.
- Rows $2, 4, \ldots, 2p$ contain positive clause gadgets.
- Rows $2p + 3$ and $2p + 4$ contain variable gadgets.
- Rows $2p+7, 2p+9, \ldots, 2p+2z+5$ contain negative clause gadgets.
- Row $2p + 2z + 6$ is empty.

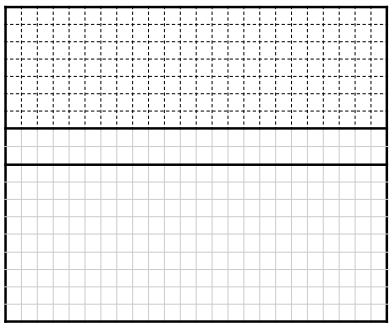
Now suppose that ϕ has p_i clauses with positive literal x_i, and z_i clauses with negative literal $\neg x_i$ for all i. The variable gadgets are sized and positioned as follows:

- Column 1 is empty.
- The variable gadgets are placed side-by-side starting from column 2 with two columns between them. Each x_i gadget is $max(p_i, n_i)$ input rooms wide.

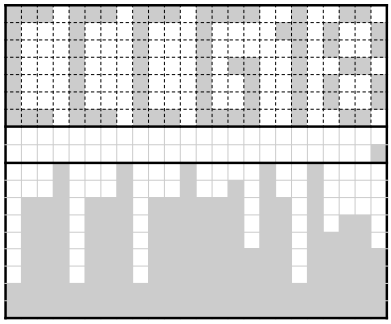
The width of the variable gadgets allow us to connect wires to distinct input rooms for each literal. In particular, the wire connected to the middle literal of a clause travels straight vertically to the middle of the corresponding clause gadget. Similarly, left and right literals enter the left and right sides of their clause gadgets.

Theorem 2 SILT *is NP-complete.*

Proof. Let ϕ be an instance of PMR3SAT. Remarks 3 and 4 imply that the variable, wire, and clause gadgets



(a) A 7-by-24 board B is extended to 16-by-24 board B' with 2-by-24 and 9-by-24 wild-card rooms.



(b) If SILT(B) is 'yes', then fill the added wild-card rooms to satisfy (S4), so SAND(B') and STONE(B') are 'yes'.

Figure 12: Reduction from SILT to SAND to STONE.

of $B = S(\phi)$ can be satisfied if and only if ϕ is satisfiable. The remaining detail is to show that the wild-card rooms of $S(\phi)$ can also be satisfied when ϕ is satisfiable. Since these rooms have no requirement we can satisfy them by shading any single square subject to (S2). See Figure 11 (b) for examples of the arguments below.

By the empty rows and column in B there is a wild-card room surrounding the gadgets. In this outer room we shade the rightmost column in row $2p+2$ or $2p+5$.

All other wild-card rooms border a straight wire.

- If this wire is on, then without loss of generality we can assume that its clause gadget is satisfied by shading its middle cell. Therefore, we can shade a square next to this clause gadget.
- If this wire is off, then we can shade a square next to its variable gadget.

Therefore, B is solvable if and only if ϕ is satisfiable. Theorem 1 and Remark 1 complete the proof. □

5 NP-Completeness of Sto-Sand

Now we prove that SAND is NP-complete by a reduction from SILT. Our strategy is to add rows to a given board so that the sand drop rule (s4) can be satisfied regardless of how the other squares are shaded. See Figure 12.

Theorem 3 SAND *is NP-complete.*

Proof. Suppose B is an m-by-n board that is an input to STONE. We create board B' of size $(2m+2)$-by-n by adding $m+2$ rows to the bottom of B. The additional rows are organized into two wild-card rooms as follows:

- A room of size 2-by-n is added below B.
- A room of size $(m+2)$-by-n is then added below.

Suppose that SILT(B) is a 'yes' instance. We now show that SAND(B') is 'yes'. We shade the top m rows of B' in any way that proves that SILT(B) is 'yes'. Then we shade the additional wild-card rooms as follows:

- The 2-by-n room has a single shaded square in its bottom-right corner.
- The bottom row of the larger room is fully shaded. If there are s shaded squares in kth column of B, then $m-s+1$ additional squares are shaded in its kth column from the bottom up. The only exception is the rightmost column which has one fewer square shaded.

This satisfies (S1)-(S3) and (s4), so SAND(B') is 'yes'.

Suppose that SILT(B) is a 'no' instance. In this case there is no way to satisfy rules (S1) – (S3) in the top m rows of B', hence, SAND(B') is 'no'.

Theorem 2 and Remark 1 complete the proof. □

6 NP-Completeness of Sto-Stone

Now we prove that STONE is NP-complete. We do this by analyzing the previous two reductions and showing that they create boards that can be solved using stones of width 1. In this context (s4) and (S4) are equivalent.

Theorem 4 STONE *is NP-complete.*

Proof. Let ϕ be an instance of PMR3SAT. Let $B = S(\phi)$ and B' be created as in Sections 4–5. We claim that ϕ is satisfiable if and only if STONE(B') is 'yes'.

Suppose that ϕ is satisfiable. By the proof of Theorem 2, SILT(B) is solvable using stones of width 1. By the proof of Theorem 3, this is also true for SAND(B'), except for the bottom stone which is already "bottom justified". Therefore, STONE(B') is also 'yes'.

Conversely, if ϕ is unsatisfiable, then SILT(B) is 'no' by Theorem 2, and so STONE(B') is also 'no'. □

7 Final Remarks

A *numberless Sto-Stone puzzle* is a Sto-Stone puzzle with no requirements. In other words, (S3) is ignored. What is the complexity of numberless Sto-Stone?

Jack Lance Puzzles [7] has several numberless examples. Numberless versions of other Nikoli puzzles have also been considered. For example, Shakashaka [5] and its numberless version [3] are both NP-complete. We note that *numberless Sto-Silt* is in P since (S1) and (S2) are satisfied by an empty board.

We thank the referees, one whom suggested *parameterized Sto-Stone* where the bottom k rows must fill.

References

[1] Puzzle Communication Nikoli Vol. 156. Sto-stone number 1, September 2016.

[2] Puzzle Communication Nikoli Vol. 162. Sto-stone number 4, March 2018.

[3] Aviv Adler, Michael Biro, Erik D. Demaine, Mikhail Rudoy, and Christiane Schmidt. Computational complexity of numberless Shakashaka. In *CCCG*, 2015.

[4] Mark de Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In My T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics*, pages 216–225, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[5] Erik Demaine, Yoshio Okamoto, Ryuhei Uehara, and Yushi Uno. Computational complexity and an integer programming model of Shakashaka. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E97.A(6):1213–1219, 2014.

[6] Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation*. A. K. Peters, Ltd., Natick, MA, USA, 2009.

[7] Jacob Lance. Puzzle 143 – stostone. `https://jacoblance.wordpress.com/2016/09/19/puzzle-143-stostone/`, September 2016.

[8] @nikoli_official. `https://twitter.com/nikoli_official/status/973516077970767873`, March 2018.

[9] @postpostdoc. `https://twitter.com/postpostdoc/status/973539335839469570`, March 2018.