Zig-Zag Numberlink is NP-Complete

Aaron Adcock[1], Erik D. Demaine[2], Martin L. Demaine[2], Michael P. O'Brien[3], Felix Reidl[4], Fernando Sánchez Villaamil[4], and Blair D. Sullivan[3]

[1]Stanford University, Palo Alto, CA, USA, `aadcock@stanford.edu`
[2]Massachusetts Institute of Technology, Cambridge, MA, USA, {`edemaine,mdemaine`}`@mit.edu`
[3]North Carolina State University, Raleigh, NC, USA, {`mpobrie3,blair_sullivan`}`@ncsu.edu`
[4]RWTH Aachen University, Aachen, Germany, {`reidl,fernando.sanchez`}`@cs.rwth-aachen.de`

October 23, 2014

Abstract

When can t terminal pairs in an $m \times n$ grid be connected by t vertex-disjoint paths that cover all vertices of the grid? We prove that this problem is NP-complete. Our hardness result can be compared to two previous NP-hardness proofs: Lynch's 1975 proof without the "cover all vertices" constraint, and Kotsuma and Takenaga's 2010 proof when the paths are restricted to have the fewest possible corners within their homotopy class. The latter restriction is a common form of the famous Nikoli puzzle *Numberlink*; our problem is another common form of Numberlink, sometimes called *Zig-Zag Numberlink* and popularized by the smartphone app *Flow Free*.

1 Introduction

Nikoli is a famous Japanese publisher of pencil-and-paper puzzles, best known world-wide for its role in popularizing Sudoku puzzles [27]. Nikoli in fact publishes whole ranges of such puzzles, following rules of their own and others' inventions; see [23]. The meta-puzzle for us theoretical computer scientists is to study the computational complexity of puzzles, characterizing them as polynomially solvable, NP-complete, or harder [7]. In the case of Nikoli puzzles, nearly every family has been proved NP-complete: Country Road [10], Corral [5], Fillomino [29], Hiroimono [1], Hashiwokakero [2], Heyawake [9], Kakuro [24], Kurodoko [15], Light Up [21], Masyu [6], Nurikabe ([20, 8, 22]), Shakashaka [3], Slither Link ([28, 29]), Sudoku ([30, 14, 29]), Yajilin [10], and Yosenabe [12].

In this paper, we study the computational complexity of one Nikoli family of puzzles called *Numberlink* (also known as Number Link, Nanbarinku, Arukone, and Flow). A Numberlink puzzle consists of an $m \times n$ grid of unit squares, some of which contain numbers, which appear in pairs. The goal of the player is to connect corresponding pairs of numbers by paths that turn only at the center of grid squares, do not cross any other numbered squares, and do not cross any other paths. Furthermore, in most versions of the puzzle, the paths should together visit every grid square; in many puzzles, this constraint is in fact forced by any otherwise valid solution.[1] Figure 1 shows a simple example.

Although popularized and named by Nikoli, the history of Numberlink-style puzzles is much older [13, 26]. The earliest known reference is an 1897 column by Sam Loyd [17] (most famous for popularizing the 15 Puzzle in 1891 [25]). Figure 2 reproduces his puzzle, called "The Puzzled Neighbors". While the puzzle statement does not require visiting every square, his solution visits most of the squares, and a small modification to his solution visits all of the squares. The same

---

[1]This coverage constraint is absent from Nikoli's Numberlink website (`http://www.nikoli.co.jp/en/puzzles/numberlink.html`), but this may simply be an omission.

	(a) Puzzle		(b) Solution

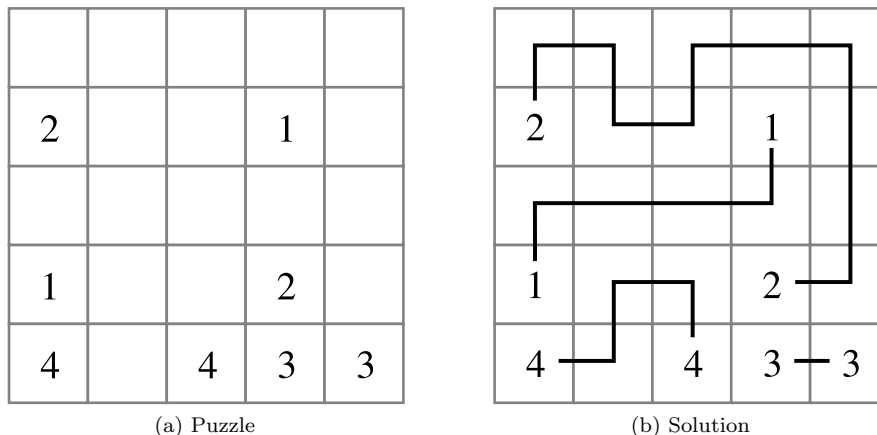Figure 1: Sample 5×5 instance with 4 terminal pairs.

puzzle later appeared in Loyd's famous puzzle book [18]. Another early Numberlink-style puzzle is by Dudeney in his famous 1931 puzzle book [4]. His puzzle is 8×8 with five terminal pairs, and while the puzzle statement does not require visiting every square, his solution does.

Some Numberlink puzzles place an additional restriction on paths, which we call *Classic Numberlink* (following the Numberlink Android app). Informally stated, restricted paths should not have "unnecessary" bends. Although we have not seen a formal definition, based on several examples, we interpret this restriction to mean that each path uses the fewest possible turns among all paths within its homotopy class (i.e., according to which other obstacles and paths it loops around). This version of Numberlink has already been shown NP-complete [16].

In this paper, we analyze the unrestricted version, which we call *Zig-Zag Numberlink* (again following the Numberlink Android app). Informally, this style of puzzle allows links to zig-zag arbitrarily to fill the grid. Personally, we find this formulation of the problem more natural, given its connection to both vertex-disjoint paths and Hamiltonicity in graphs. When we only allow for one pair of terminals this problem reduces to finding a Hamiltonian path in a grid given fixed start and end points, which is known to be solvable in polynomial time [11]. Unfortunately, for the case with several terminal pairs, as shown in Section 3, the hardness proof of [16] does not (immediately) apply to Zig-Zag Numberlink. Nonetheless, we construct a very different and intricate NP-hardness proof, inspired by an early NP-hardness proof from 1975 for vertex-disjoint paths [19].

2 Definitions

Next we formally define the puzzle Zig-Zag Numberlink.

Definition 1. *A* board $B_{m,n}$ *is a rectangular grid of* mn *equal sized* squares *arranged into* m *rows and* n *columns. We will identify the squares with an ordered pair consisting of their column and row position. The top left corner is defined to be position* $(1,1)$.

Definition 2. *A* terminal pair *is a pair of distinct squares. An* instance *of* Zig-Zag Numberlink *is a tuple* $\mathcal{F} = (B_{m,n}, \mathcal{T})$ *where* $B_{m,n}$ *is a* $m \times n$ *board and* $\mathcal{T} = \{(T_1, T_1'), \ldots, (T_t, T_t')\}$ *a set of terminal pairs. All terminals are distinct, that is, any square of the board may contain at most one terminal.*

Definition 3. *Two squares* (x, y) *and* (p, q) *are* adjacent *if either* $x = p$ *and* $|y - q| = 1$, *or* $y = q$ *and* $|x - p| = 1$. *A sequence of squares* $P = s_1, \ldots, s_k$ *is a* path *of length* k *if* s_i *is adjacent to* s_{i+1} *for* $i \in [1, k-1]$ *and* $s_i \neq s_j$ *for all* $i \neq j$. *Two squares are* linked *in* P *if they appear successively in* P.

2

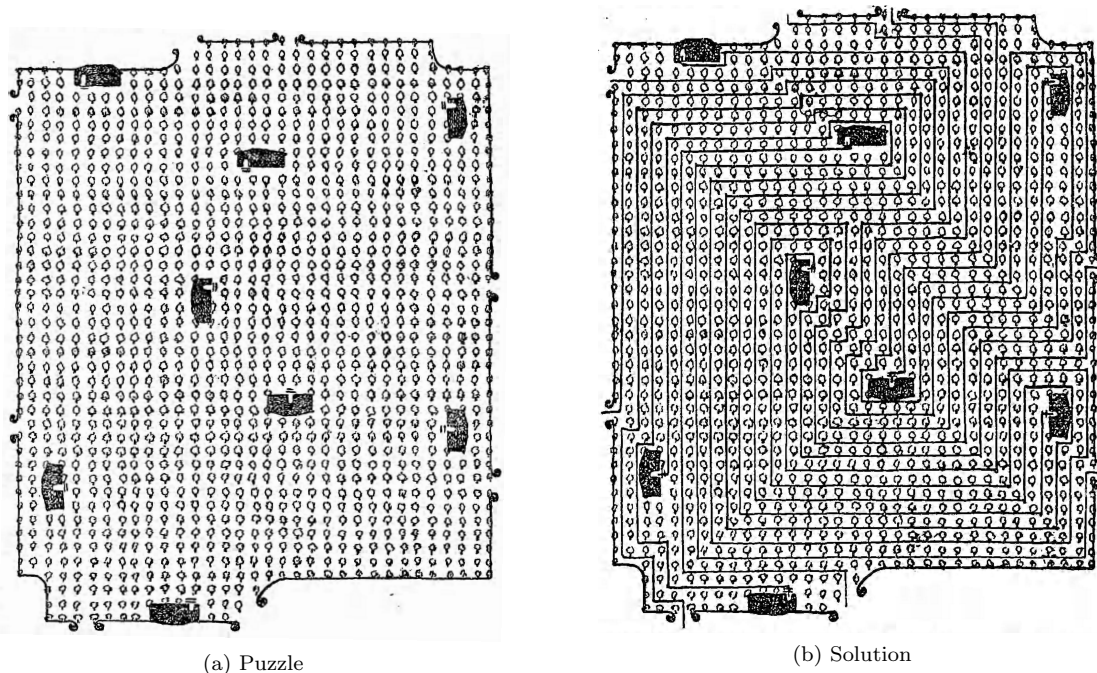|	
(a) Puzzle	(b) Solution

Figure 2: Sam Loyd's "The Puzzled Neighbors" from 1897 [17]. Each house should be connected by a path to the gate it directly faces, by noncrossing paths. Scans from `http://bklyn.newspapers.com/image/50475607/` and `http://bklyn.newspapers.com/image/50475838/`.

Definition 4. *A solution to a* Zig-Zag Numberlink *instance* $\mathcal{F} = (B, \mathcal{T})$ *is a set of paths* $\mathcal{S} = \{P_1, \ldots, P_t\}$, *where* $P_i = s_{i,1}, \ldots, s_{i,k_i}$, *so that:*

 (i) *Every terminal pair* (T_i, T_i') *is connected by path* P_i, *i.e.* $s_{i,1} = T_i$ *and* $s_{i,k_i} = T_i'$.

 (ii) *Each square in* B *is contained in exactly one path in* $\mathcal{S}$.

We call an instance of Zig-Zag Numberlink *solvable* if there exists a solution and *unsolvable* otherwise. We say that two squares are *linked* by a solution S if they are linked in some path $P \in \mathcal{S}$. In the case of $t = 1$, we will abuse the above notation and identify the solution by a single path.

Finally, we will talk about *parity*. To illustrate this concept, we will color the squares alternating black and white as on a checkerboard (cf. Figure 3) and assume that $(1, 1)$ is colored black. The important aspect of parity is that any path of a solution necessarily alternates between white and black squares. In the construction of our gadgets, parity helps to ensure that a combination of gadgets still allows a solution. As it will turn out, parity along with terminal position is crucial to determine whether instances with only one terminal pair are solvable.

3   NP-hardness

Our NP-hardness reduction for Zig-Zag Numberlink mimics the structure of a very early NP-hardness proof for vertex-disjoint paths among k terminal pairs in grid graphs by Lynch [19]. The important differences are that Lynch's reduction (1) allows obstacles (untraversable squares), and (2) does not require every traversable square to be covered by some path. The first issue is relatively easy to deal with because terminals serve as obstacles for all other paths. For the second issue, we replace all of Lynch's gadgets with more complicated gadgets to make it possible to cover every square of the grid in all cases.

Theorem 1. Zig-Zag Numberlink *is NP-complete for* k *terminal pairs in an* $n \times n$ *square.*
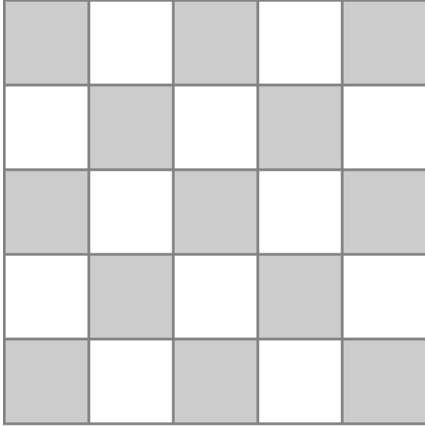
3

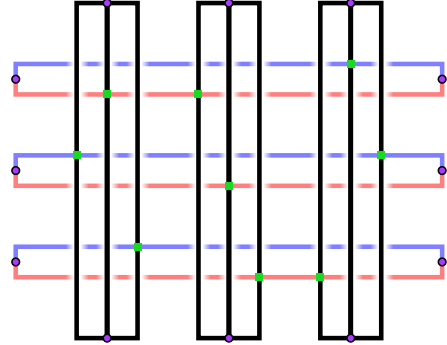Figure 3: Squares colored to illustrate parity.

Figure 4: High-level sketch of NP-hardness proof: two paths per variable, three paths per clause, and crossover gadgets (faded underpasses). Circles indicate terminals; squares indicate actual crossings. Whitespace indicates obstacles where paths cannot go.

Proof. ZIG-ZAG NUMBERLINK is in NP because the solution paths can be expressed in $O(n^2 \log k)$ space and checked in the same amount of time.

To prove NP-hardness, we reduce from 3SAT. Figure 4 illustrates the high-level picture. We construct one terminal pair (v_i', v_i'') for each variable v_i, and two candidate paths $V_{i,-}, V_{i,+}$ for connecting this pair, $V_{i,-}$ representing the false setting and $V_{i,+}$ representing the true setting. We construct one terminal pair (c_j', c_j'') for each clause c_j, and three candidate paths $C_{j,1}, C_{j,2}, C_{j,3}$ for connecting this pair, one per literal in the clause $c_j = c_{j,1} \lor c_{j,2} \lor c_{j,3}$. Effectively, for each clause literal $c_{j,k} = \pm v_i$ say, the corresponding literal path $C_{j,k}$ intersects just the variable path $V_{i,\mp}$ corresponding to the setting that does *not* satisfy the clause literal. Thus, setting the variable in this way blocks that clause path, and a clause must have one of its paths not blocked in this way (corresponding to satisfaction). It follows that any noncrossing choice of paths corresponds to a satisfying assignment of the 3SAT instance.

The reality is more complicated because, in a square grid, all variable paths will intersect all clause paths. However, we can simulate the nonintersection of two paths using the *crossover gadget* shown in Figure 8. The idea is to split the two variable paths $V_{i,-}, V_{i,+}$ into two classes of variable paths, top and bottom, with nonintersection of the paths forcing alternation between top and bottom classes. A variable starting with a top path corresponds to a false setting (darker in the figure), and starting with a bottom path corresponds to a true setting (lighter in the figure). The crossover gadget must work when the vertical clause path is either present (chosen to satisfy the clause) or absent (having chosen a different of the three paths), resulting in four total cases.

We simulate obstacles between paths using many pairs of terminals at unit distance (black in the figure). These *obstacle pairs* can be connected by a unit-length edge (as drawn in black) or by a longer path (drawn lighter, as a replacement for the black path, though the black path is still drawn). Such longer paths can only prevent choices for the variable/clause paths, so any nonintersecting set of paths still solves the 3SAT instance. Furthermore, by connecting the obstacle pairs by the lighter longer paths illustrated in the figure, in each of the four cases, we can turn any solution to the 3SAT instance into a valid solution to ZIG-ZAG NUMBERLINK (which in particular visits every square).[2]

---

[2]Our careful and deliberate placement and orientation of obstacle pairs to enable such "filling" in all cases is the main novelty to our proof. Lynch's crossover gadget has the same nonobstructed paths as our Figure 8a, but as his proof allowed obstacles, lacked the complexity of obstacle pairs and the four cases.

(a) Crossing (preventing) true, while still allowing false.

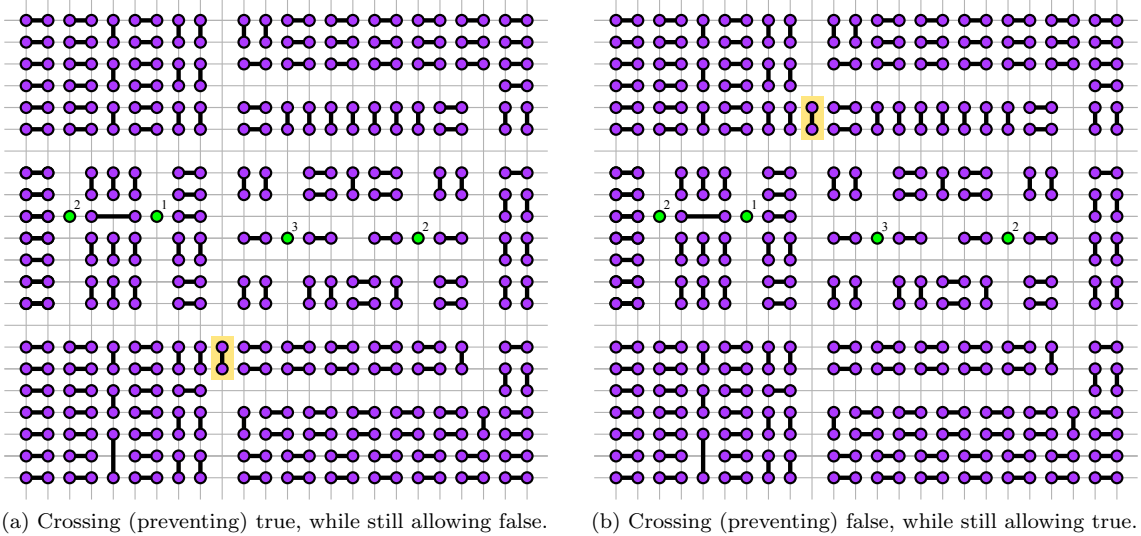(b) Crossing (preventing) false, while still allowing true.

Figure 5: Crossing gadget, with paths drawn only between obstacle pairs. The highlight indicates the unique added obstacle pair.

This crossover gadget necessitates nontrivial *crossing gadgets*, because the two variable paths in the high-level picture (Figure 4) have been replaced by alternation between two classes of paths. Figure 5 illustrates two gadgets for crossing (preventing) the false and true settings, respectively. Each of these gadgets adds just a single obstacle pair to the crossover gadget of Figure 8. These obstacles suffice to block the clause path in the prevented variable setting, but are consistent with the solutions in Figure 8 both with and without the clause path, so they still allow the other variable setting.

Finally, we can form the two-way branches on the left side v_i' and the right side v_i'', and the three-way branches at the top side c_j' and the bottom side c_j''), using the split gadget in Figure 6. This gadget allows the incoming path on the top to exit at either of the two bottom ports, while still covering all squares, provided the exit ports both have the same parity (color on the checkerboard). A key property of the crossover and crossing gadgets (and the reason for the strange fifth column) is that they have even numbers of rows and columns. As a result, when we build a grid of these gadgets, the top entrance ports all have the same parity, as do all the left entrance ports. Therefore the split gadgets can correctly connect to the ports on the left, right, top, and bottom sides of the grid. □

Figure 7 illustrates why the NP-hardness proof for the restricted game [16] does not immediately apply to ZIG-ZAG NUMBERLINK. The illustrated gadget, from Fig. 7 of [16], consists of a 1-in-4 SAT clause connected to four wires. Figure 7a shows the intended solution, which has one wire in the opposite state from the three other wires. But Figure 7b shows another possible solution in ZIG-ZAG NUMBERLINK, where the wires are all in the same state (or two in each state, depending on the definition of state). Thus the reduction does not immediately apply to the zig-zag case. An interesting question is whether the proof (which is rather different from ours) can be adapted to ZIG-ZAG NUMBERLINK by modifying the gadgets from [16].
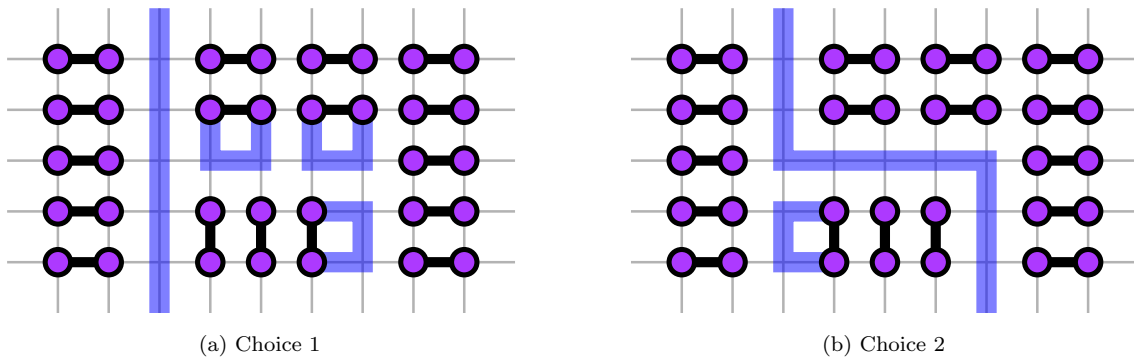
(a) Choice 1   (b) Choice 2

Figure 6: Split gadget.

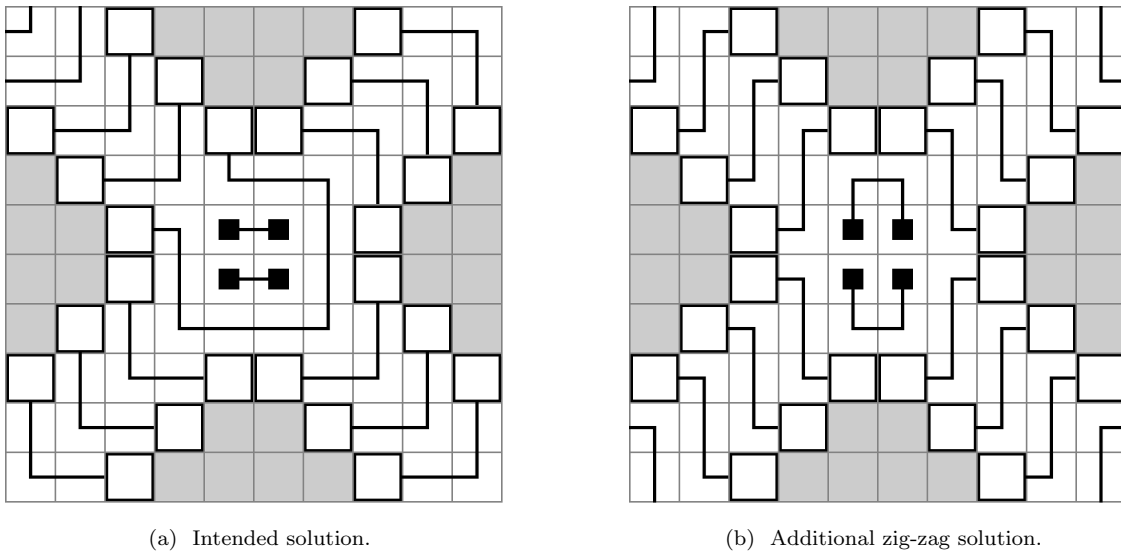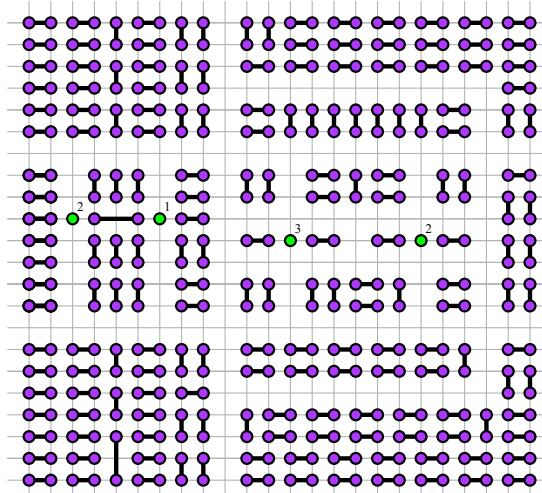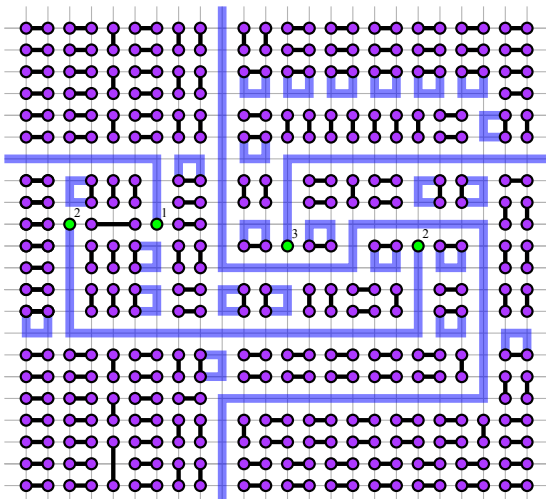(a) Intended solution.   (b) Additional zig-zag solution.

Figure 7: The 1-in-4 SAT clause from [16] does not immediately work for ZIG-ZAG NUMBERLINK. Shaded regions represent obstacles (made by obstacle pairs).
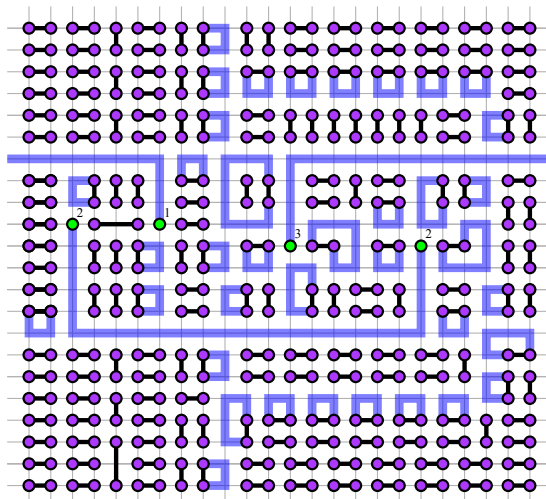
4   Open Questions

There is still more fun to be had out of this game. Is there a polynomial solution to the problem for more than one pair of terminals? Our reduction from 3SAT creates instances with a huge number of terminals. Therefore we cannot exclude the possibility that, for any constant number of terminals, the problem can be solved in polynomial time. If so, it would be especially interesting to know whether the problem is fixed parameter tractable, i.e., solvable in $f(k) \cdot n^{O(1)}$ time for some function f. It might be that more sophisticated tools like multiterminal flow algorithms could help answer these questions.
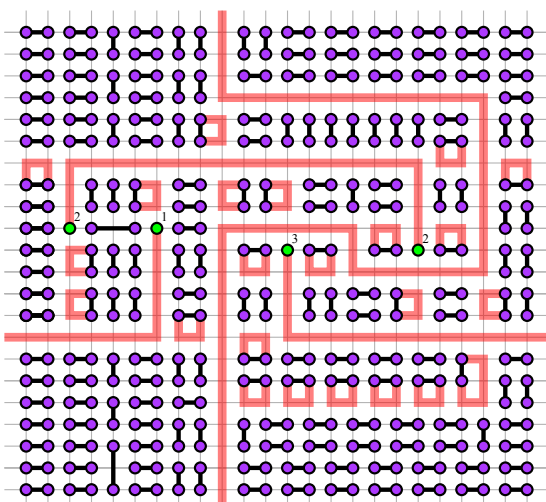
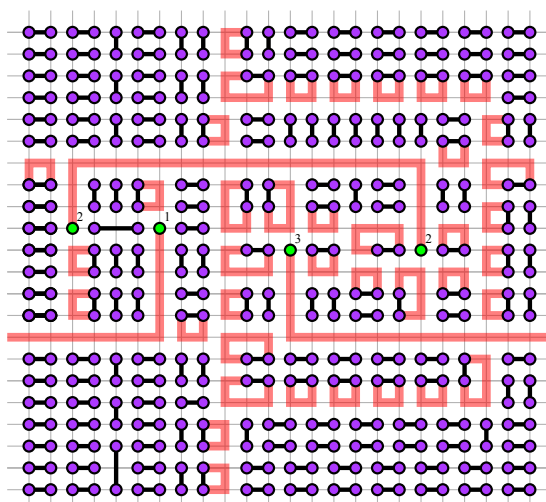(a) Crossover gadget with paths drawn only between obstacle pairs.

(b) False setting with clause path.

(c) False setting without clause path.

(d) True setting with clause path.

(e) True setting without clause path.

Figure 8: Crossover gadget.

7

References

[1] Daniel Andersson. HIROIMONO is NP-complete. In *Proceedings of the 4th International Conference on Fun with Algorithms*, volume 4475 of *Lecture Notes in Computer Science*, pages 30–39, 2007.

[2] Daniel Andersson. Hashiwokakero is NP-complete. *Information Processing Letters*, 109(19):1145–1146, 2009.

[3] Erik D. 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.

[4] Henry Ernest Dudeney. 428. Planning tours. In *536 Puzzles and Curious Problems*. Londres, 1931. Available at `http://jnsilva.ludicum.org/HMR13_14/536.pdf`.

[5] Erich Friedman. Corral puzzles are NP-complete. `http://www2.stetson.edu/~efriedma/papers/corral/corral.html`.

[6] Erich Friedman. Pearl puzzles are NP-complete. `http://www2.stetson.edu/~efriedma/papers/pearl/pearl.html`, August 2002.

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

[8] Markus Holzer, Andreas Klein, and Martin Kutrib. On the NP-completeness of the Nurikabe pencil puzzle and variants thereof. In *Proceedings of the 3rd International Conference on Fun with Algorithms*, pages 77–89, 2004.

[9] Markus Holzer and Oliver Ruepp. The troubles of interior design–a complexity analysis of the game Heyawake. In *Proceedings of the 4th International Conference on Fun with Algorithms*, volume 4475 of *Lecture Notes in Computer Science*, pages 198–212, 2007.

[10] Ayaka Ishibashi, Yuichi Sato, and Shigeki Iwata. NP-completeness of two pencil puzzles: Yajilin and Country Road. *Utilitas Mathematica*, 88:237–246, 2012.

[11] Alon Itai, Christos H Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[12] Chuzo Iwamoto. Yosenabe is NP-complete. *Journal of Information Processing*, 22(1):40–43, 2014.

[13] Ed Pegg Jr. Number Link: Beyond Sudoku. *The Mathematica Journal*, 10(3), August 2007.

[14] Graham Kendall, Andrew Parkes, and Kristian Spoerer. A survey of NP-complete puzzles. *ICGA Journal*, 31(1):13–34, 2008.

[15] Jonas Kölker. Kurodoko is NP-complete. *Journal of Information Processing*, 20(3):694–706, 2012.

[16] Kouichi Kotsuma and Yasuhiko Takenaga. NP-completeness and enumeration of number link puzzle. *IEICE Technical Report*, 109(465):1–7, March 2010.

[17] Sam Loyd. Sam loyd's puzzles: The fuzzled neighbors. *The Brooklyn Daily Eagle*, page 26, 28 February 1897. The solution appeared 14 March, 1897 on page 17.

[18] Sam Loyd. *Sam Loyd's Cyclopedia of 5,000 Puzzles, Tricks and Conundrums: With Answers.* Lamb Publishing Company, New York, 1914. Reprinted by Pinnacle Books in 1976. Available at `http://www.mathpuzzle.com/loyd/`.

[19] James F. Lynch. The equivalence of theorem proving and the interconnection problem. *ACM SIGDA Newsletter*, 5(3):31–36, September 1975.

[20] B. McPhail and J. D. Fix. Nurikabe is NP-complete. In *Poster Session of the 6th CCSC-NW Annual Northwestern Regional Conference of the Consortium for Computing Sciences in Colleges*, 2004.

[21] Brandon McPhail. Light Up is NP-complete. `http://people.cs.umass.edu/~mcphailb/papers/2005lightup.pdf`, February 2005.

[22] Brandon P. McPhail. Complexity of puzzles: NP-completeness results for Nurikabe and Minesweeper. Bachelor's Thesis, Reed College, December 2003.

[23] NIKOLI Co., Ltd. Web nikoli — enjoy pencil puzzles! `http://www.nikoli.co.jp/en/`.

[24] Takahiro Seta. The complexities of puzzles, CROSS SUM, and their ANOTHER SOLUTION PROBLEMS (ASP). Senior Thesis, University of Tokyo, February 2002.

[25] Jerry Slocum and Dic Sonneveld. *The 15 Puzzle.* Slocum Puzzle Foundation, June 2006.

[26] Wikipedia. Numberlink. `http://en.wikipedia.org/wiki/Numberlink`.

[27] Wikipedia. Sudoku. `http://en.wikipedia.org/wiki/Sudoku`.

[28] Takayuki Yato. On the NP-completeness of the Slither Link puzzle (in Japanese). *IPSJ SIG Notes*, AL-74:25–32, 2000.

[29] Takayuki Yato. Complexity and completeness of finding another solution and its application to puzzles. Master's thesis, University of Tokyo, January 2003.

[30] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 86(5):1052–1060, 2003.