The hardness of Witness puzzles*

Irina Kostitsyna[1], Maarten Löffler[2], Max Sondag[1], Willem Sonke[1], and Jules Wulms[1]

1 **Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
`[i.kostitsyna | m.f.m.sondag | w.m.sonke | j.j.h.m.wulms]@tue.nl`

2 **Dept. of Information and Computing Sciences, Utrecht University, The Netherlands**
`m.loffler@uu.nl`

1 Introduction

The Witness is a puzzle video game that was produced by Thekla [1]. Since its release in 2016 the game has been critically acclaimed, and has been praised for its intelligence and astonishing visuals. The Witness contains 9 principal types of puzzles, and a number of hidden "environmental" puzzles, totaling to an amount of 664 puzzles spread over the open world of the game. A player is invited to explore the world and to deduce the rules of various puzzles they encounter.

Most of the puzzles in the game are based on a square grid, requiring the player to connect a starting point to an end point with a simple grid path while satisfying certain constraints, such as splitting the grid cells of different colors, passing through a set of given points, and drawing polyomino shapes comprising of a set of tetris blocks, among others. For example, the puzzle depicted in Figure 1 asks to connect the bottom-left corner of the grid to the top-right corner with a grid path separating the white square from the black squares.

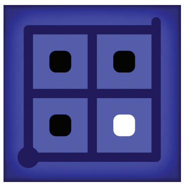

Figure 1 An example of a *Black and White Squares* puzzle from the Witness. (Image from [1].)

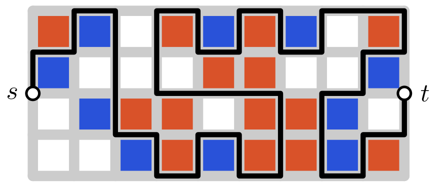

Figure 2 An example of a solved colored squares puzzle. All red squares are separated from the blue squares by the path.

In this short paper we resolve the complexity of two out of the nine types of puzzles in the game, namely the *Black and White Squares* and the *Multicolor Squares* puzzles. We show that in a restricted setting these types of puzzles can be solved in polynomial running time, but that in general they are NP-complete.

To formalize the setting, consider a rectangular grid of size $w \times h$, which we interpret as a graph $G = (V, E)$, where $V = [0, 1, \dots, w] \times [0, 1, \dots, h]$ and two vertices (a, b) and (c, d) are connected by an edge in E if and only if $a = c$ and $|b - d| = 1$, or $|a - c| = 1$ and $b = d$. G is a planar graph whose natural embedding decomposes the plane into $(w - 1)(h - 1)$ square faces that we call *cells*, and one unbounded outer face. Cells are colored by a color c from a

* Research on the topic of this paper was initiated at the 2nd Workshop on Applied Geometric Algorithms (AGA 2016) in Vierhouten, The Netherlands, supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208. M.L., M.S. and W.S. are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 614.001.504 (M.L.) and 639.023.208 (M.S. and W.S.). J.W. is supported by the Netherlands eScience Center (NLeSC) under grant number 027.015.G02.

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

set $C \cup \{\square\}$. Cells colored by $\square$ are called *empty*. Furthermore, two vertices s and t from V lying on the outer face are selected as the start and end point respectively. The decision version of the puzzle asks whether there exists a simple path P from s to t in the graph G, whose embedding on the grid splits the grid into several connected components of cells, that each contain cells of only one color from C and possibly some empty cells. We call such a path P a color-separating path. An example of a puzzle and its solution is given in Figure 2.

Related work. The Witness puzzle is closely related to Sheep and Wolves, a puzzle where one has to build a fence that separates sheep from wolves on a grid introduced by Dave Tuller.[1] In his version, however, some cells contain additional clues describing the number of fences directly incident to the cell. This puzzle type is known, among other names, as Slither Link, which was proven to be NP-hard by Yato [9]. The study of the computational complexity of puzzle games is as old as the notion of NP-completeness itself [8], and has a rich history, which is beyond the scope of this short note to discuss; instead, we refer the interested reader to the excellent surveys by Demaine and Hearn [3] and by Kendall *et al.* [4].

The object of the puzzles in The Witness, to find a curve separating white from black cells on a grid, is also closely related to the problem of finding a grid-aligned approximation of a shape: given a region overlaid by a grid, we wish to find a curve that has all cells that lie completely inside the shape inside, and all cells that lie completely outside the shape outside, with applications ranging from early computer graphics (casting characters to low-resolution screens) [7] to geographical information systems. Often, one has additional objectives, such as minimizing the symmetric difference or another distance measure of the two shapes. This area remains an active domain of research [2, 6].

2 Algorithms

First we consider the problem every cell is colored either red or blue. We will give an algorithm to solve this problem in time linear in the size of the grid $O(wh)$.

Observations. We can make a couple of basic observations about the path that will help us in understanding the problem. We let a *border* be defined as a maximal set of connected edges between an area of red cells and blue cells. Furthermore we let p_1 and p_2 be the paths from s to t along the edges of the outer face (see Figure 3). We observe that if a color-separating path P exists, then it must go through all the edges of all borders to separate the colors. Moreover as soon as the path P encounters a vertex on some border, it cannot deviate from the border. If it could deviate then only one endpoint of an edge e would be on P as P must be simple, and thus the colors adjacent to e would not be separated. It follows that the path through a border must start at either p_1 or p_2, as that is where the endpoints of the borders are. Large parts of any potential path P are thus fixed.

We categorize the borders into two types: *crossings* (that start on p_1 and end on p_2, or vice versa) and *bumps* (that both start and end on either p_1 or p_2). These are depicted in Figure 4. As a special case, borders that start or end in s or t, respectively, are considered bumps. A third possible type would be an *island* (that does not have any vertices on p_1 or p_2). These however cannot exist if a simple color-separating path P exists, as P would need to contain all edges around the island, which results in a cycle. Furthermore, bumps cannot be nested if there is to be a solution, as any color-separating path needs to traverse all of those bumps, which requires the path to be non-simple.

[1] https://www.amazon.com/Challenge-Brain-Logic-Puzzles-Mensa/dp/1402714491

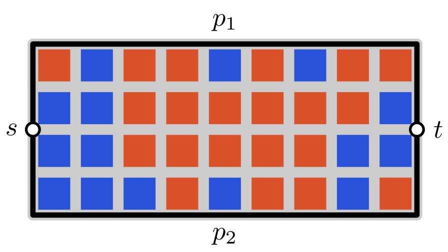

Figure 3 Paths p_1 and p_2 go from s to t along the outer face.

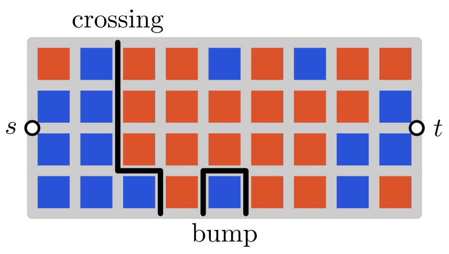

Figure 4 An example of a crossing and a bump are shown in black.

Witty algorithm. The idea of the algorithm is to construct a directed graph G' that admits an st-path if and only if there is a simple color-separating path in our input grid. It works as follows. We cut the grid along all crossings, producing a sequence of pieces $\mathcal{G} = [g_1, g_2, \ldots, g_k]$, ordered such that s is in g_1, t is in g_k, and for all $1 < i < k$, g_i shares a border with g_{i-1} and g_{i+1}. For each piece we remove all vertices (and adjacent edges) that are part of its adjacent crossings, except for the endpoints on p_1 and p_2. As we have subdivided G along all crossings, the only borders left within the pieces are bumps.

For each grid $g \in \mathcal{G}$, we now create a small directed graph g'. Let s_1, s_2 be the vertices on p_1 and p_2, respectively, that are closest to s, and similarly let t_1, t_2 be the vertices on p_1 and p_2 that are closest to t. For g_1 it holds that $s = s_1 = s_2$, and in g_k, $t = t_1 = t_2$. For each grid g we put vertices representing s_1, s_2, t_1, t_2 into g'.

Grids g_i and g_{i+1} used to be connected to each other by a crossing in the original grid. We know that any separating path P must use this crossing and thus we connect t_1 of g'_i to s_2 of g'_{i+1} and s_1 of g'_i to t_2 of g'_{i+1} to form G'. It then remains to determine how P can route within the grids $g \in \mathcal{G}$.

As each border must be followed from start to end by P, the path in a grid g_i must always start at $s_{1,2}$ and end at $t_{1,2}$. We add an edge between s_x and t_y in g' if there exists a color-separating path in g_i starting at s_x and ending at t_y. The existence of such paths can be determined by checking two conditions on g_i. Firstly, if there are bumps on both p_1 and p_2, then there must be at least two grid cells between the bumps on p_1 and p_2 such that the path can pass between them. Secondly, there must be enough vertices between bumps on p_1 and p_2 such that we can exit/enter p_1 and p_2 to go to the other side (and back again if needed). We need to enter/exit each path at most once. Figure 5a shows an example of such a path: Graph g_2 has an extra vertex after the bump on p_2, and enough space between the bumps on p_1 and p_2. A path starting at the bottom left can separate all bumps at the bottom, route back to the top left and separate all bumps at the top to end in the top right.

The graph G' now exactly represents all possible paths P can take. The answer to our original problem is thus reduced to whether there exists a path from s to t in G'.

▶ **Lemma 1.** *There exists a simple color-separating path P from s to t in a fully 2-colored grid G iff there exists a simple path from s to t in G'.*

More colors. In Lemma 1 we assumed that the grid contained only two colors. The same algorithm works using any number of colors, if we make some small adjustments. We construct graph g' to characterize P, and we use the fact that P should always follow the borders. However, if we have more than two colors, borders no longer have to be paths. Three/four faces with distinct colors can share a vertex v. Whenever this happens we know that there is no color-separating path, because the path would need to visit v more than once. We thus know that all borders are paths, and these borders can only be crossings or

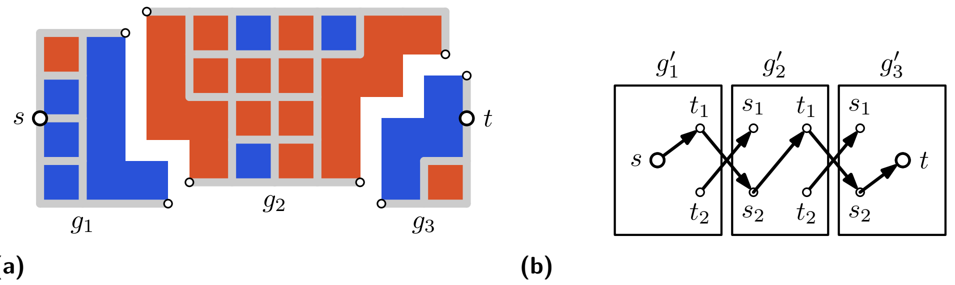

Figure 5 (a) Sequence of graphs created by removing crossings. (b) The graph that is created to decide whether there is a simple path that separates squares of different colors.

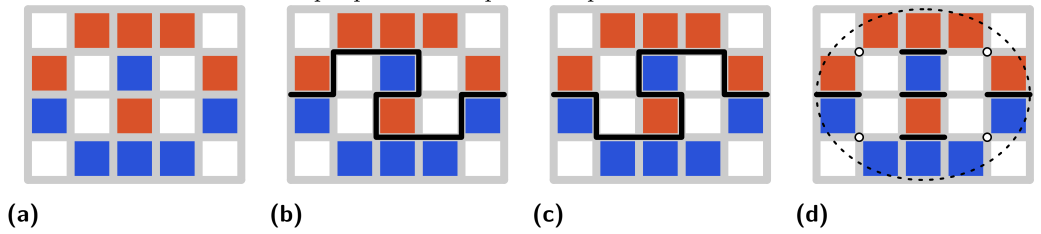

Figure 6 An egg. (a) The input. (b) One possible internal state of a path traversing the egg. (c) The other possible state. (d) The fixed edges and sockets of the egg. The egg outline is drawn in black, for easy recognition when used in later constructions.

bumps. We can therefore recolor the grid using only colors from $\{r, b\}$, such that on both sides of each crossing we get a different color. Assume that r is used to color cells between two crossings (or between s or t and a crossing) then b can be used to color the bumps (or vice versa). We can now apply Lemma 1 to find out whether there is a color separating path.

Running time. We can find the at most $O(wh)$ crossing borders in $O(wh)$ time by walking over p_1 and following the outline of every color to check if it hits p_2. As the total size of all pieces combined is bounded by $O(wh)$, we can thus create the graph G' in $O(wh)$. Finally we can determine if a path from s to t exists in the graph G' in $O(wh)$ with a simple DFS.

▶ **Theorem 2.** *Deciding whether there exists a simple color-separating path P from s to t in a fully-colored grid G can be done in $O(wh)$ time.*

3 Hardness

In this section we show that deciding whether there exists a simple color-separating path P from s to t in a grid G with 2 colors is NP-hard if the grid contains empty squares. To prove this we provide a reduction from planar rectilinear 3-SAT [5].

Eggs. Our reduction is built from several gadgets. Our most basic gadget consists of a 5×4 grid with 12 colored squares, which we call an *egg*. The gadget is illustrated in Figure 6a. Note that five edges of G inside the egg must be on P, because the adjacent cells are of different colors. There are then only two possible ways how P can traverse the egg (note that it is not possible to connect the edges to the boundary of the gadget using multiple paths), depicted in Figures 6b and 6c. Two pair vertices of G are used by only one of the two paths, we refer to those vertices as the *sockets* of the egg (see Figure 6d).

Variables. Variables are built by connecting many eggs together at their sockets: if one egg uses a socket, the adjacent egg cannot use the same socket. We define an *egg snake* of length

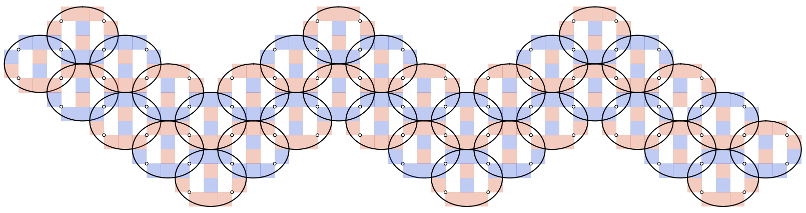

Figure 7 An egg snake of length 3.

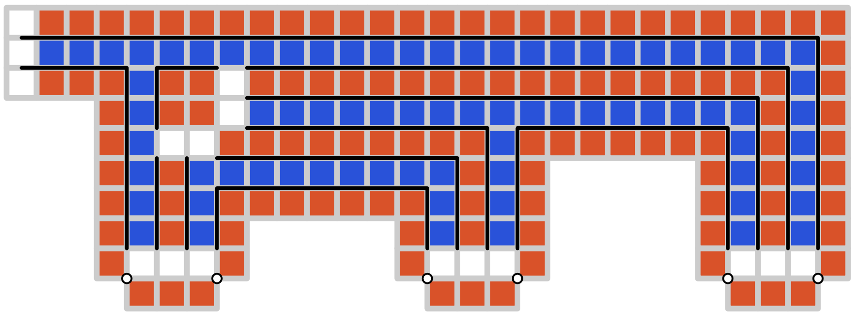

Figure 8 A clause comb. There are 6 sets of adjacent empty cells; one at the handle, two ornaments and three teeth.

k to be an arrangement of $12k - 2$ eggs, as depicted in Figure 7.

If x_i is `true`, then all top left and bottom right sockets (in the orientation shown in Figure 6d) of every egg are used, otherwise all bottom left and top right sockets of every egg are used. Note that, by the nature of their arrangement, all eggs in a snake must be in the same state. We globally place all variable egg snakes in a horizontal line; see Figure 9.

Clauses. We represent each clause of the SAT instance by a *comb* gadget, consisting of long strips of adjacent blue and red squares; see Figure 8. The exact shape of the comb is flexible: horizontal and vertical stretches can be made longer as required. The outside of the comb is covered by red squares, except for a single blue square in the top left which we refer to as the *handle* of the comb. Any color-separating path P must enter through the handle, collect (surround) all blue squares, and then leave again through the same gap.

There are five places in a comb where a choice can be made: three *teeth* and two *ornaments*, each can be found in Figure 8 as a set of adjacent empty cells. We can think of the choice to make as filling each tooth and ornament with either red or blue squares; this then fully determines the course of P. However, not all choices lead to valid paths: the two ornaments cannot both be set to blue because they would cause P to touch itself, and the left ornament and left tooth or the right ornament and right tooth cannot both be set to blue because this would create a red island. Similarly, we argue that exactly two out of the five teeth/ornaments must be blue and three must be red, otherwise there will be either a red or a blue island. It follows that at least one of the teeth must be blue.

Now, we observe that when a tooth is blue, it causes the path to run lower than when a tooth is red. We have a left and a right *socket* on each tooth of the comb, which we will let overlap with sockets of eggs. We connect a variable to the left socket for positive variables as the top right socket is not used in the case, and the right socket for negative variables for the same reason. Figure 9 shows a small example of an instance, leaving out most of the puzzle details, but showing how a color-separating path can be routed. The remaining space between variable and clause gadgets is filled by a grid of empty cells.

Satisfiability. We now show that we can efficiently find a solution to a planar 3-SAT formula

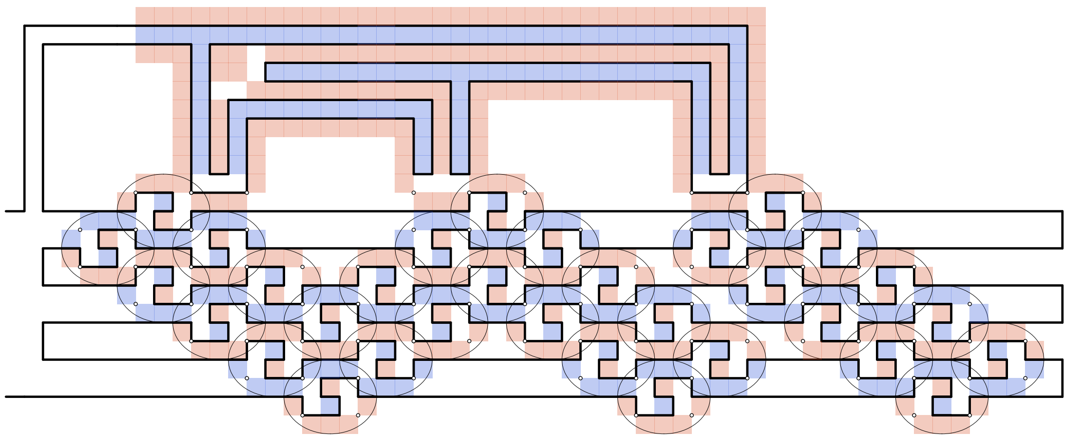

Figure 9 An example of a small satisfiable instance. The left egg snake represent the variable x and the right egg snake represent the variable y. The clause represent the statement $x \vee \neg x \vee \neg y$. The color-separating path shows us that setting x to true and y to false satisfies the clause.

if we can efficiently find a color-separating path in the constructed instance for such a formula.

To find a color-separating path, we thread six horizontal paths through all variables. The ends of the topmost and bottommost of these paths will be routed through the clause gadgets above and below the variable line respectively. Finally, we connect the six paths through the variables into a single path to create a color-separating path.

▶ **Theorem 3.** *Deciding whether there exists a simple color-separating path P from s to t in a grid G with two colors is NP-hard if the grid contains empty squares.*

References

1 The Witness. `http://the-witness.net/`, 2010. Accessed on 2018-01-07.

2 Quirijn W. Bouts, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Willem Sonke, and Kevin Verbeek. Mapping polygons to the grid with small Hausdorff and Fréchet distance. In *Proc. 24th Annual European Symposium on Algorithms (ESA)*, pages 22:1–22:16, 2016.

3 Erik D. Demaine and Robert A. Hearn. Playing games with algorithms: Algorithmic combinatorial game theory. In Michael H. Albert and Richard J. Nowakowski, editors, *Games of No Chance 3*, volume 56 of *Mathematical Sciences Research Institute Publications*, pages 3–56. Cambridge University Press, 2009.

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

5 Donald E Knuth and Arvind Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992.

6 Maarten Löffler and Wouter Meulemans. Discretized approaches to schematization. In *Proc. 29th Canadian Conference on Computational Geometry (CCCG)*, pages 220–225, 2017.

7 Nadia Magnenat-Thalmann and Daniel Thalmann. *New Trends in Computer Graphics*. Springer, 1st edition, 1988.

8 Edward Robertson and Ian Munro. NP-completeness, puzzles and games. *Utilitas Mathematica*, 13:99–116, 1978.

9 Takayuki Yato. On the NP-completeness of the Slither Link puzzle. *IPSJ SIGNotes ALgorithms*, 84(2000-AL-074):25–31, 2000. In Japanese.