Journal of Information Processing Vol.20 No.3 719–722 (July 2012)

[DOI: 10.2197/ipsjjip.20.719]

Regular Paper

Rolling Block Mazes are PSPACE-complete

Kevin Buchin[1,a)] Maike Buchin[1]

Received: August 30, 2011, Accepted: December 16, 2011

Abstract: In a rolling block maze, one or more blocks lie on a rectangular board with square cells. In most mazes, the blocks have size $k \times m \times n$ where k, m, n are integers that determine the size of the block in terms of units of the size of the board cells. The task of a rolling block maze is to roll a particular block from a starting to an ending placement. A block is rolled by tipping it over one of its edges. Some of the squares of the board are marked as forbidden to roll on. We show that solving rolling block mazes is PSPACE-complete.

Keywords: constraint logic, puzzle, rolling block maze

1. Introduction

Rolling block mazes are a popular class of puzzles invented by Richard Tucker[*1]. In a rolling block maze, one or more blocks lie on a rectangular board with square cells. In most mazes, the blocks have size $k \times m \times n$ where k, m, n are integers that determine the size of the block in terms of units of the size of the board cells. Also there are mazes with U-shaped blocks. Typically, the dimensions of a block are small. The first and most rolling block mazes use blocks of size $2\times1\times1$. The task of a rolling block maze is to *roll* a particular block from a starting to an ending placement. A block is rolled by tipping it over one of its edges. Furthermore, some of the squares of the board are marked as forbidden to roll on. An example of a rolling block maze is shown in **Fig. 1**. In this maze, a $2 \times 1 \times 1$ block is standing upright on the square labeled 'Start' and needs to be rolled to stand upright on the square labeled 'Goal'. Forbidden squares are marked by a brick pattern.

Rolling block mazes differ in the number of blocks on the board. Many mazes use only one block that has to be rolled from *start* to *goal*. Other rolling block mazes, like Erich Friedman's mazes[*2], use several blocks. In these mazes only one particular block has to be rolled to the goal, while the other blocks are rollable obstacles.

In Section 2 we analyze the algorithmic complexity of rolling block mazes. Our main result is that solving rolling block mazes with several blocks is PSPACE-complete. This stands in contrast to rolling block mazes with only one block, which can be solved in linear time. Related puzzles and open problems are discussed in Section 3.

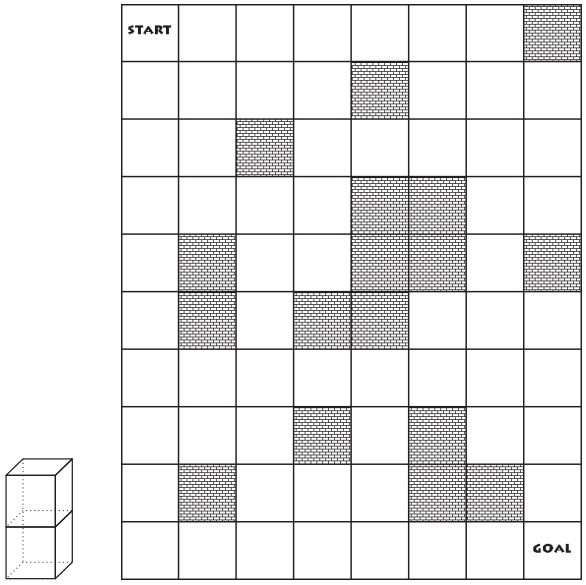


Fig. 1 Rolling block maze by Robert Abbott. The objective is to roll a $2 \times 1 \times 1$ block standing upright on 'start' to stand upright on 'end'. The block may only roll onto white squares. Figure used with permission by Robert Abbott.

2. Complexity

Rolling block mazes with only one block can be solved in linear time in the size of the board, as briefly discussed in Section 2.1. In Section 2.2 we recall the *Constraint Logic* framework [2], which we use in Section 2.3 to show our main result: that solving rolling block mazes with several blocks is PSPACE-complete.

2.1 Mazes with Only One Block

Observation. *Rolling block mazes with only one block can be solved using linear time and space.*

Proof. Consider the following graph that encodes all possible states of the maze. The vertices of the graph are the possible placements of the block on the board. An edge exists between two

[1] Department of Mathematics and Computer Science, Technical University Eindhoven, North Brabant, Netherlands
[a)] k.a.buchin@tue.nl

[*1] For examples and the history of rolling block mazes see Robert Abbott's website http://www.logicmazes.com/rb/column.html (accessed 2011-12-28)

[*2] See again Robert Abbott's website and Erich Friedman's website http://www.stetson.edu/~efriedma/rolling/ (accessed 2011-12-28)

such placements if one of the placement results from the other by one roll of the block. The size of this graph is linear in the size of the board, independent of the size of the block. The maze can be solved by searching this graph for a path from the starting to the ending placement. This can be done for instance using breadth first search in time linear in the number of edges of the graph, which is linear in the size of the board. □

The graph used in the above proof has been analyzed further [9].

2.2 Constraint Logic Framework

The constraint logic framework was introduced by Demaine and Hearn [2] and has been used to prove PSPACE-hardness and other complexity results for various puzzles and games. Constraint logic consists of a class of games played on planar directed graphs. The edges in the graph have a weight of 1 or 2, all vertices have degree 3. There are two types of vertices: *AND*-vertices, which have two incident edges of weight 1 and one incident edge of weight 2, and *OR*-vertices, which have three incident edges of weight 2.

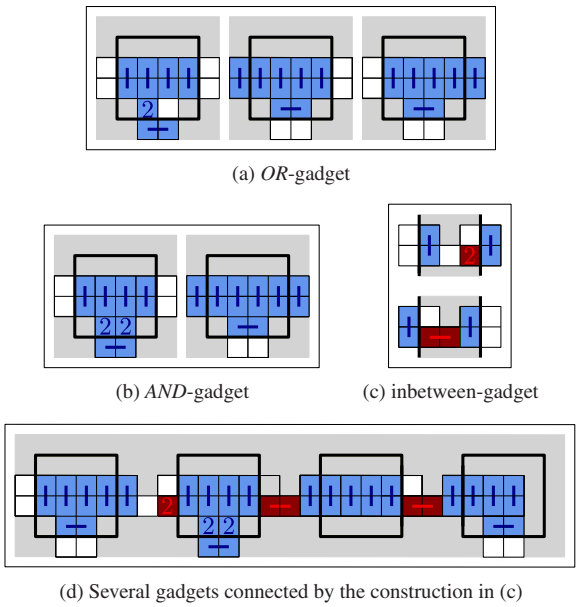
In a game of constraint logic, players reverse edges of the graph. At all times, the sum of the weights of ingoing edges at every vertex has to remain at least 2. For *OR*-vertices this means that an ingoing edge can be reversed only if both other edges are ingoing. For *AND*-vertices this means that an ingoing weight-1 edge can be reversed if the weight-2 edge is ingoing, while an ingoing weight-2 edge can be reversed if both weight-1 edges are ingoing.

One player has the goal to reverse a given edge. If there is only one player and edges can be reversed arbitrarily often, the problem of deciding whether a given edge can be reversed is PSPACE-complete [6]. Using this framework it suffices to construct gadgets for *AND*- and *OR*-vertices (called *AND*- and *OR*-gadgets in the following) and gadgets for connecting them (called *wires*). The resulting planar directed graph can be assumed to be laid out on a polynomial-size grid. Therefore only wire-gadgets for vertical and horizontal connections and for right-angle bends are needed. Furthermore, we can assume that the graph is laid out on the grid such that the weight-1 edges of any *AND*-vertex are both horizontal or both vertical. Also, the edge that needs to be reversed can be assumed to be an ingoing weight-2 edge of an *AND*-vertex (see Lemma 4 in Ref. [6]).

2.3 PSPACE-completeness

Theorem. *Rolling block mazes with several blocks are PSPACE-complete. This holds already for mazes where all blocks have size* $2 \times 1 \times 1$.

Proof. Rolling block mazes can easily be solved nondeterministically with polynomial space. Since PSPACE= NPSPACE holds [10], rolling block mazes are therefore in PSPACE. It remains to prove that solving rolling block mazes is PSPACE-hard. The gadgets for the reduction from constraint logic are shown in **Fig. 2**. All blocks in the gadgets have size $2 \times 1 \times 1$. They are shown in blue or red (mid-gray or dark-gray in grayscale printouts). The cells indicated by '2' are standing



(a) *OR*-gadget

(b) *AND*-gadget

(c) inbetween-gadget

(d) Several gadgets connected by the construction in (c)

Fig. 2 Gadgets for the PSPACE-hardness reduction. A line across two blocks indicates a lying $2\times1\times1$ block. A '2' indicates a standing $2 \times 1 \times 1$ block.

blocks. Additional to the gadgets mentioned in the previous section, we also use an *inbetween-gadget* as connectors between the other gadgets. The figure only shows one orientation for each gadget. Other orientations can be obtained by rotating the complete gadget.

The *AND*- and *OR*-gadget both have the same 'T'-shape and the (core of the) gadget is the area inside the black square, adjacent to which are three pairs of adjacent squares where the gadgets are connected to other gadgets. An edge (of the planar directed graph of the constraint logic instance) is directed inward if and only if a block lies on the corresponding region. Recall that all adjacent edges of an *OR*-gadget have weight 2, whereas an *AND*-gadget has one adjacent edge of weight 2 and two adjacent edges of weight 1. In Fig. 2 (b) the edge of weight 2 is facing downward. To prove the theorem, we need to show that the *AND*- and *OR*-gadgets always have an inward weight of 2 and flipping of edges (switching between states) is always possible. Furthermore we have to show that the orientations can be propagated using the wire-gadgets.

Figure 2 (a) shows the three main states of the *OR*-gadget, i.e., three configurations of the *OR*-gadget in which one of the edges is directed inward. Observe that the downward edge can only point outward, if either the left or right edge is pointing inward. Thus, at least one edge needs to point inward. It is possible to switch between the states, for instance as follows: To go from the left state of the *OR*-gadget to the middle state, first the two upper left blocks are rolled one space to the left. Then, the lower two blocks are rolled up. To go from the middle state to the right state, all blocks in the upper row are rolled one space to the right.

Figure 2 (b) shows the two main states of the *AND*-gadget, i.e., a configuration where the edge of weight 2 is directed inward and a configuration where both edges of weight 1 are directed inward. Observe that the bottom edge (of weight 2) can only point outward, if both the left and right edge are pointing inward. Thus, the inward weight is always at least 2. It is possible to switch be-

tween the states, for instance as follows: To go from the left state to the right state, first the two upper left blocks are moved one space to the left and the two upper right blocks are moved one the the right. Then the two standing blocks are each tipped on the two squares above. Finally the lowest block can be rolled up.

Figure 2 (c) shows how to connect two gadgets (*AND*-, *OR*- or wire-gadgets) using an inbetween-gadget. This construction ensures that the two outer blocks of the neighboring gadgets cannot both be rolled outside the gadgets. This is prevented by the further block, which has a different orientation than the outer blocks, and which always covers (exactly) one square where one of the outer blocks would need to roll on. Furthermore this construction prevents a block of a gadget to "leave" its gadget, because the outer blocks cannot roll over a space of one square and cannot be made to stand up. Note that a (red) block of the connection between two gadgets may "enter" one of these gadgets but it cannot leave this gadget in any other exit.

The wire-gadgets (horizonal, vertical and bend) are the same as *OR*-gadgets but with only two exists. Figure 2 (d) shows an *OR*-gadget, an *AND*-gadget, a horizonal wire, and a bend wire connected by inbetween-gadgets.

Using the gadgets described and analyzed above, we can thus construct a rolling block maze for any given instance of constraint logic. We assume that the instance is given by a planar directed graph laid out on a polynomial-size grid, and by a weight-2 edge of an *AND*-vertex that needs to be oriented outward (see Section 2.2). The instance of constraint logic is solvable if and only if the *AND*-gadget corresponding to the *AND*-vertex can be switched from the left state in Fig. 2 (b) to the right state. This is exactly the case if the lowest block (in the figure) can be rolled upwards. We therefore declare this block as the block that has to be rolled from its starting placement to its ending placement, the two squares directly above it. Now the instance of constraint logic is solvable if and only if the rolling block maze is. The number of gadgets used is bounded by the size of the grid. Since every gadget uses only a constant number of blocks, the reduction can be done in PSPACE. □

3. Related Puzzles and Open Problems

Similar to rolling block mazes are *rolling cube mazes* [*3]. Here, all blocks are cubes and faces of the cube and squares of the board show a number. A cube may roll on a square if its number on top coincides with the number on the square (depending on the puzzle: either before or after rolling). The complexity of rolling cube mazes has been studied for the case of a single cube [1]. Depending on the rules of the puzzle, the complexity varies between polynomial time solvable and NP-complete.

An open question, which initiated this research and which remains open, is to find rolling cube mazes with several cubes that are PSPACE-complete. However, for these constructing the *AND*-gadget is more difficult than for rolling blocks. The difficulty comes from the small size ($1 \times 1 \times 1$) of the cubes. The *AND*-gadget requires that rolling one cube into the gadget should force two cubes out. A possible way to achieve this is to add special kinds of squares or extra rules. For example, one could use numbered *grease spots* [*4]. A grease spot is a square where a die may slide over. We use a grease spot with a number where a die slides over if it does not match the number on the grease spot but the number behind the grease spot. Now if the cube that we want to roll into the *AND*-gadget has such a grease spot in front of it and does not show the corresponding number then it needs two free squares to roll into the gadget: the grease spot and the square behind the grease spot. Oriel Maxime [*5] suggested instead adding the rule that no two adjacent dice may have the same number on top.

Rolling cube and block mazes are related to the puzzle *Rush Hour* and sliding block puzzles. In Rush Hour pieces have size 1×2, 1×3, 2×1, and 3×1. The pieces do not roll but slide. Any block can slide either only vertically or only horizontally depending on their orientation. Rush Hour is PSPACE-complete [4], which even holds if the pieces are restricted to sizes 2×1 and 1×2 [11]. But as for rolling cube puzzles with several cubes the complexity of Rush Hour with 1×1 pieces (each piece labeled as either vertically or horizontally slidable) remains open. *Subway shuffle* [7] is a generalization of Rush Hour with 1×1 pieces played on a graph. For subway shuffle the complexity is also still open. In contrast 1×1 sliding block puzzles, i.e., puzzles in which the pieces can slide vertically and horizontally, are known to be polynomial time solvable. Solving sliding block puzzles with larger blocks is again PSPACE-complete [8].

A further puzzle that is similar to rolling block mazes is TipOver. This is known to be NP-complete [5]. The difference in TipOver that reduces the complexity of the puzzle is that every block may be "tipped" only once. See Refs. [3], [7] for more puzzles and their complexity.

Acknowledgments We would like to thank Robert Abbott, Andrea Gilbert, Robert Hearn and Oriel Maxime for discussing (PSPACE-complete) puzzles with us, and Robert Abbott for giving permission to include one of his great puzzles.

References

[1] Buchin, K., Buchin, M., Demaine, E.D., Demaine, M.L., El-Khechen, D., Fekete, S.P., Knauer, C., Schulz, A. and Taslakian, P.: On Rolling Cube Puzzles, *Proc. 19th Canadian Conference on Computational Geometry* (*CCCG*), pp.141–144 (2007).

[2] Demaine, E.D. and Hearn, R.A.: Constraint Logic: A Uniform Framework for Modeling Computation as Games, *IEEE Conference on Computational Complexity*, pp.149–162 (2008).

[3] Demaine, E.D. and Hearn, R.A.: Playing Games with Algorithms: Algorithmic Combinatorial Game Theory, *Games of No Chance 3*, Albert, M.H. and Nowakowski, R.J. (Eds.), Mathematical Sciences Research Institute Publications, Vol.56, pp.3–56, Cambridge University Press (2009).

[4] Flake, G.W. and Baum, E.B.: Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants," *Theoretical Computer Science*, Vol.270, No.1–2, pp.895–911 (2002).

[5] Hearn, R.A.: TipOver is NP-complete, *The Mathematical Intelligencer*, Vol.28, pp.10–14 (2006).

[6] Hearn, R.A. and Demaine, E.D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation, *Theoretical Compututer Science*, Vol.343, pp.72–96 (2005).

[7] Hearn, R.A. and Demaine, E.D.: *Games, puzzles and computation*,

[*3] See again Robert Abbott's website, http://www.logicmazes.com/rc/cubes.html

[*4] Grease spots in rolling cube mazes were suggested by Robert Abbott.

[*5] Personal communication

A K Peters (2009).
[8] Hopcroft, J.E., Schwartz, J.T. and Sharir, M.: On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the Warehouseman's Problem, *International Journal of Robotics Research*, Vol.3, No.4, pp.76–88 (1984).
[9] Jiang, M.: Flipping Triangles and Rectangles, *Proc. 17th Internat. Computing and Combinatorics Conference* (*COCOON*), Fu, B. and Du, D.-Z. (Eds.), Lecture Notes in Computer Science, Vol.6842, pp.543–554, Springer (2011).
[10] Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities, *Journal of Computer and System Sciences*, Vol.4, No.2, pp.177–192 (1970).
[11] Tromp, J. and Cilibrasi, R.: Limits of Rush Hour Logic Complexity, *Arxiv report* arXiv:cs/0502068v1 (2005).

Kevin Buchin received his Ph.D. in computer science from the Free University Berlin in Germany. He was a postdoctoral researcher at Utrecht University, Netherlands, and the Technical University Eindhoven, Netherlands. Currently he is an assistant professor at the Technical University Eindhoven. His main research interest lies in geometric and graph algorithms with the focus on geographic analysis and visualization.

Maike Buchin received her Ph.D. in computer science from the Free University Berlin in Germany. She was a postdoctoral researcher at Utrecht University, Netherlands, and is currently an assistant professor at the Technical University Eindhoven, Netherlands. Her research interest lies mostly in geometric algorithms with the focus on algorithms for geographic applications.