Algorithms **2012**, *5*, 148-157; doi:10.3390/a5010148

OPEN ACCESS

algorithms

ISSN 1999-4893

www.mdpi.com/journal/algorithms

Article

Any Monotone Function Is Realized by Interlocked Polygons

Erik D. Demaine [1], Martin L. Demaine [1] and Ryuhei Uehara [2,*]

[1] Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, MA 02139, USA; E-Mails: edemaine@mit.edu (E.D.D.); mdemaine@mit.edu (M.L.D.)

[2] School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan

[*] Author to whom correspondence should be addressed; E-Mail: uehara@jaist.ac.jp; Tel.: +81-761-51-1220; Fax: +81-761-51-1149.

Received: 15 November 2011; in revised form: 13 March 2012 / Accepted: 14 March 2012 / Published: 19 March 2012

Abstract: Suppose there is a collection of n simple polygons in the plane, none of which overlap each other. The polygons are *interlocked* if no subset can be separated arbitrarily far from the rest. It is natural to ask the characterization of the subsets that makes the set of interlocked polygons free (not interlocked). This abstracts the essence of a kind of sliding block puzzle. We show that any monotone Boolean function f on n variables can be described by $m = O(n)$ interlocked polygons. We also show that the decision problem that asks if given polygons are interlocked is PSPACE-complete.

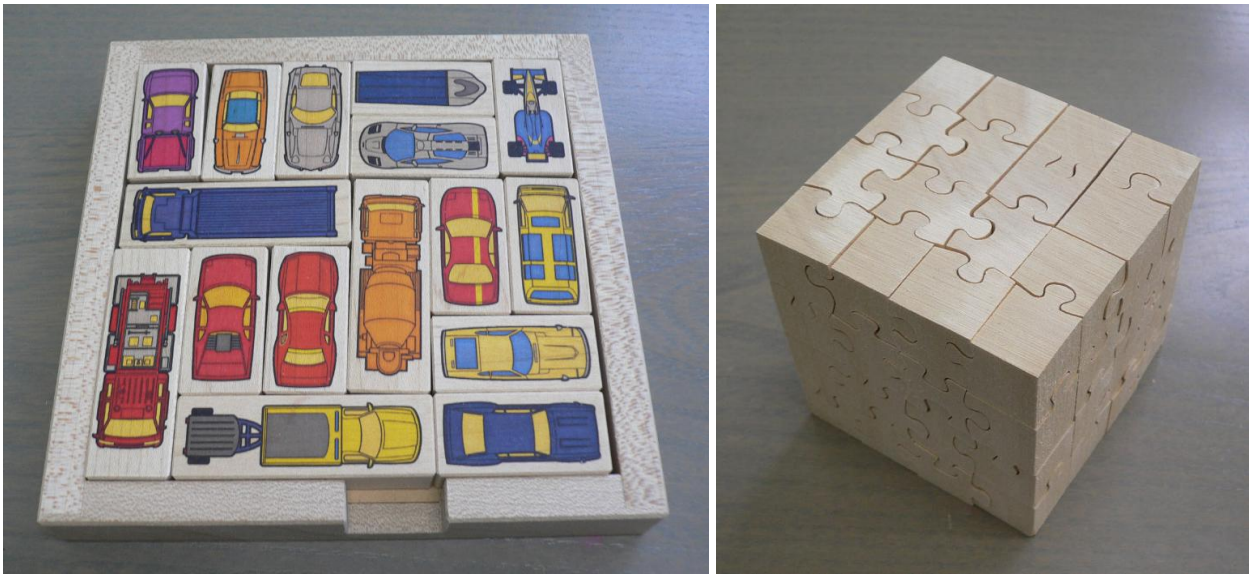
Keywords: computational complexity; interlocked polygons; monotone boolean function; sliding block puzzle

1. Introduction

Since Sam Loyd invented the famous 15-puzzle, sliding block puzzles have played an important role in mathematical recreations. There are many variations of sliding block puzzles (Figure 1), and they have been investigated widely (see a comprehensive handbook [1]). Recently, a new framework using games on graphs is developed, and many complexity results of these puzzles are settled [2]. In most of these puzzles, for a given initial state, we aim at finding a way to its goal state. Sometimes, the difficulty of the puzzles is changed if we change the initial state. For example, the 15-puzzle becomes much easier if we remove two or more pieces from (potentially) sixteen pieces. In the puzzles in Figure 1, if we remove

more pieces, the problems (getting out a specified car or disassembling the puzzle) become easier. That is, in a sliding block puzzle, fewer pieces mean easier in general. We investigate such a monotonicity of the sliding block puzzles.

Figure 1. Typical sliding block puzzles in 2D and 3D.

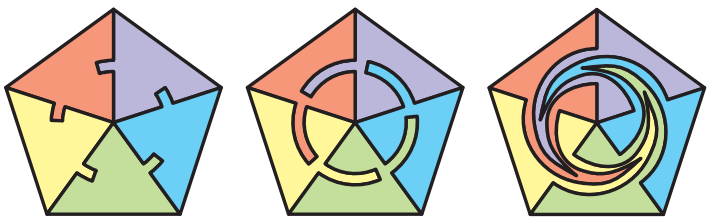


Suppose we have a collection of n simple polygons in the plane, none of which overlap each other. We call the polygons *interlocked* if no subset can be separated arbitrarily far from the rest. Otherwise, we say them *free*. If we remove a subset S of the polygons, the remaining polygons might or might not be interlocked.

Define $\delta(S) = 1$ if the remaining polygons become free after removing S, and $\delta(S) = 0$ if they remain interlocked. Clearly f is monotone: if $S \subseteq S'$, then $\delta(S) \leq \delta(S')$. (Removing more makes the polygons less interlocked.)

Then, which monotone Boolean functions f can be described by n interlocked polygons? Figure 2 shows some simple examples. For a specified n and k, this technique can design n interlocked polygons such that removing any k of them makes them all free.

Figure 2. Interlocked polygons freed by removing any one, two, or three of the polygons (from left to right).



Since the early period of computer science, monotone functions are well investigated, especially, in the context of the lower bound of circuit complexity (see, e.g., [3, Chapter 14.4]). In general, monotone Boolean functions are formed by AND and OR operations (without NOT). Such gates seem easy enough to build, but not without extra pieces that affect the function.

Then, can any monotone Boolean functions f on n variables be described by $m > n$ interlocked polygons, where f operates on a particular subset of the polygons? In this paper, we give an affirmative answer to this question:

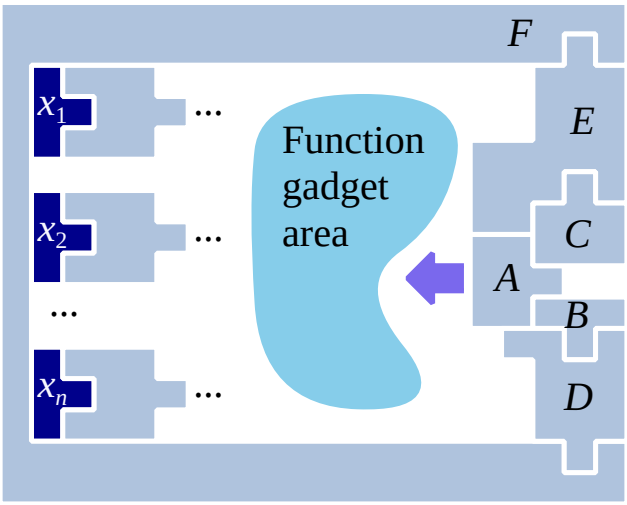
Theorem 1. *For any given monotone Boolean function f on n variables $x_1, \ldots, x_n$, there is a collection S of m simple polygons with $m = O(n)$ such that (1) S is interlocked, (2) a subset $S' \subset S$ of n simple polygons corresponding to the variables, and (3) $\delta(S'') = 1$ for $S'' \subseteq S'$ if and only if $f(x_1, \ldots, x_n) = 1$, where $x_i = 1$ indicates that the simple polygon corresponding to x_i is in S''.*

We next turn to the sliding block puzzles again. Using a new framework in [2], several sliding block puzzles are shown to be PSPACE-complete. In such a puzzle, the problem asks whether a specified piece can reach the goal or not. More basic problem is to determine if a given collection of polygons is interlocked or not. In this paper, we generalize the basic problem to a new sliding block puzzle; the *exploding sliding block puzzle* is a puzzle that asks if all polygons of a given collection of polygons can be free. Combining the idea in Theorem 1 and the gadgets in [2], we show that the exploding sliding block puzzle is also PSPACE-complete. This result holds even if all pieces are rectangles but one nonconvex piece. This is optimal since any collection of convex polygons is explodable.

2. Construction

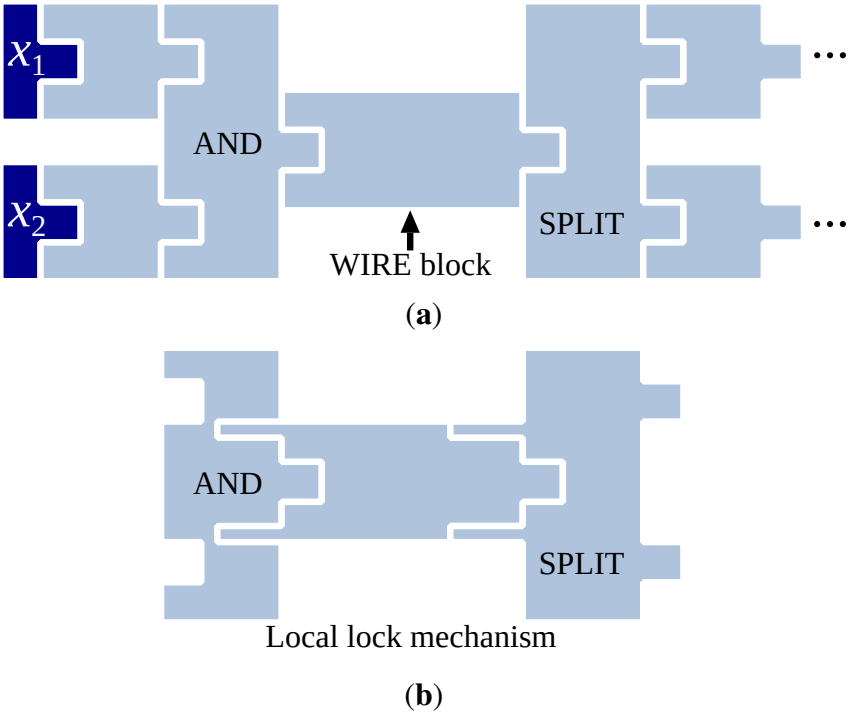
The proof of Theorem 1 is done by a construction of the desired interlocked polygons for any monotone Boolean function f on n variables $x_1, \ldots, x_n$. The outline of the construction is depicted in Figure 3. The polygons B, C, D, E, and F form the outer frame. When polygon A is at the initial position as in the figure, they are interlocked. However, once we slide A left, B can be plugged out, and all of C, D and E can be removed in this order, and then all polygons can be moved from inside of F to its outside, and they will be free. Hence we will construct the gadgets inside of this "frame". Each of the variables $x_1, \ldots, x_n$ corresponds to a polygon as depicted in Figure 3. If we remove the polygon x_i (we sometimes identify a variable/operator and the corresponding gadget) since $x_i = 1$, the input spread to right side. We note that each of "input" *pulls* the polygons one unit length that may be different from typical reductions.

Figure 3. Outline of the construction.



In this frame, to complete the construction of the gadgets that realize a monotone function f, we have to design five kinds of gadgets; (1) WIRE for wire to transfer a signal; (2) AND for "and" operation between two variables; (3) OR for "or" operation between two variables; (4) SPLIT to split a signal to two or more signals; (5) TURN to turn a signal; and (6) CROSS to cross two independent signals. Among them, WIRE, AND, and SPLIT are relatively simple to realize. In Figure 4(a), the operation (x_1 and x_2) is computed by the AND block, and the signal is sent to right by the wire blocks, and split by the SPLIT block. (In fact, SPLIT is essentially the same as AND, which is used backward way.) In each move, each block moves one unit length. One may wonder what happens if some blocks are moved out from our assumption. For example, the wire block between x_1 and AND in Figure 4(a) can be moved to up after removing x_1. But we can avoid these unexpected cases; the blocks can be locked locally if we attach some long arms as shown in Figure 4(b). To simplify, we omit the arms in the other figures.

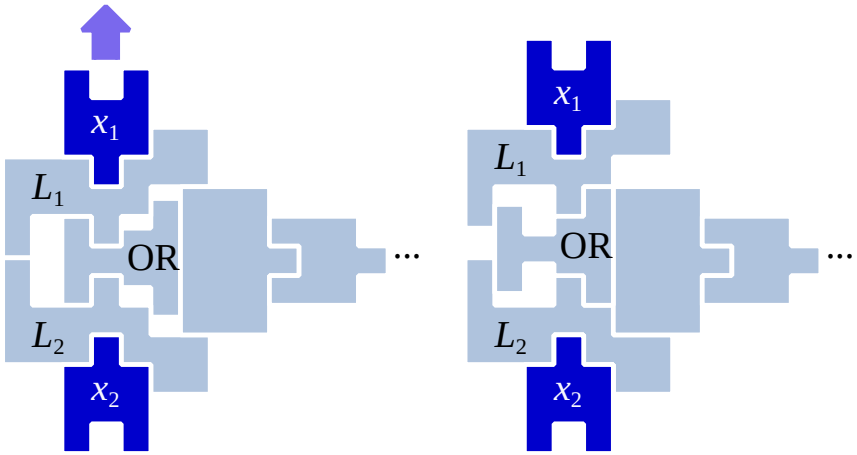
Figure 4. The gadgets for WIRE, AND, and SPLIT.



The OR gadget is designed as in Figure 5. The initial position is depicted in Figure 5(a). If one of x_1 and x_2 moves a unit length far from the block OR, OR can be moved to left as in Figure 5(b). (In the figure, x_1 moves to up.) We note that even if both x_1 and x_2 move a unit length far from the block OR, OR is still locked in with the blocks L_1 and L_2. Otherwise, moving just x_1 may have some bad influence to the different variable x_2 and vice versa; intuitively, this mechanism prevents to bounce back the signal of x_1 to x_2 through this gadget.

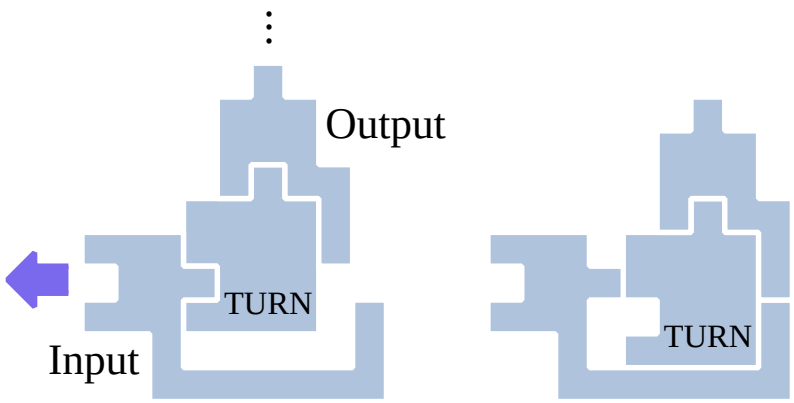
The TURN gadget is designed as in Figure 6. The initial position is depicted in Figure 6(a). When the left block moves to left, the blocks can move as in Figure 5(b).

Figure 5. OR gadget.



(**a**) (**b**)

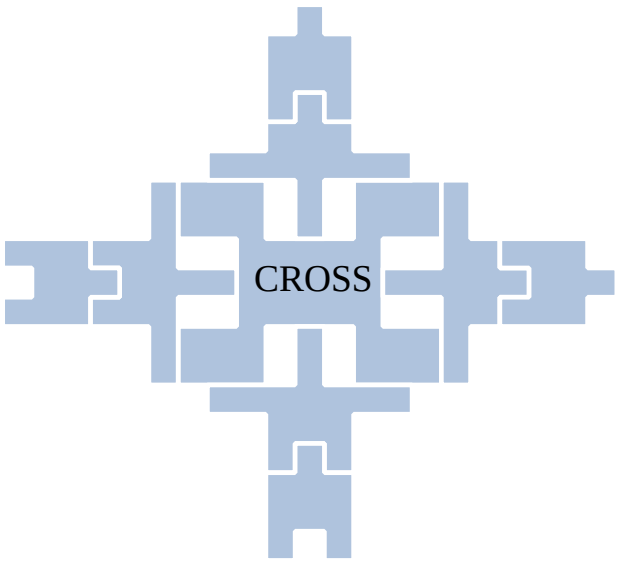
Figure 6. TURN gadget.



(**a**) (**b**)

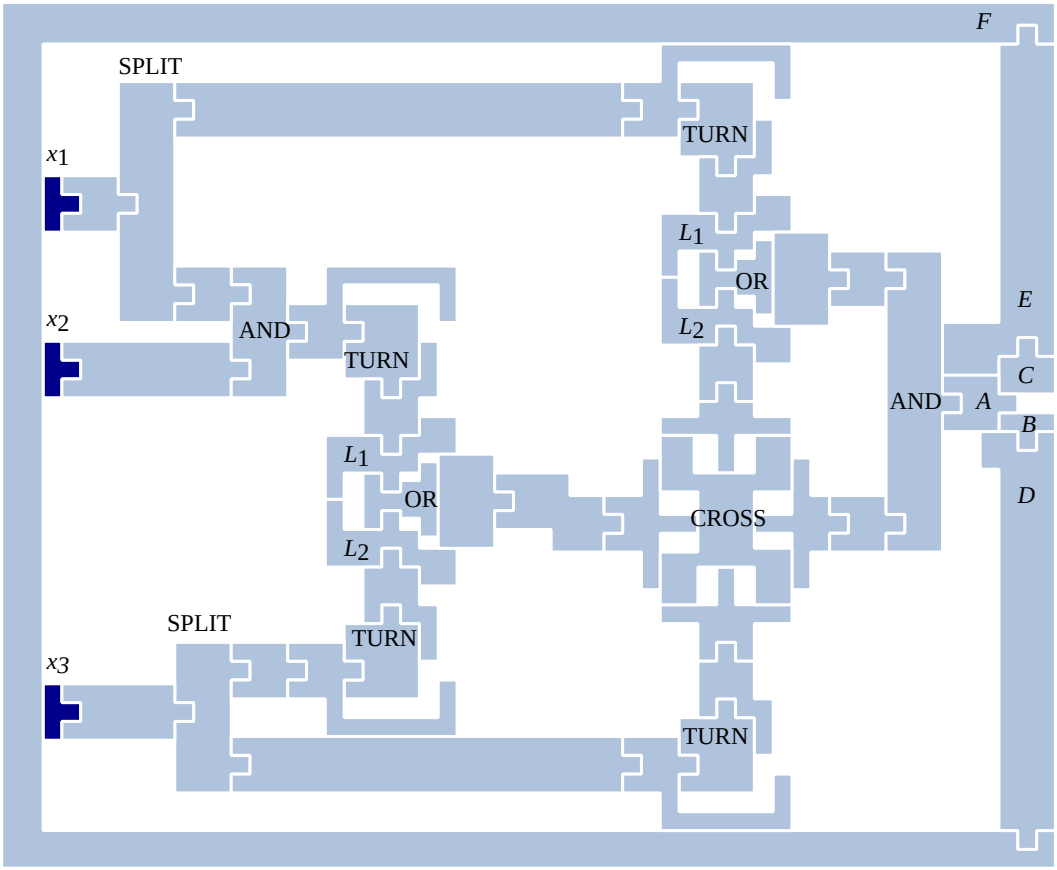
In many reductions, the crossover of the wires is the key gadget (see [2] for the details). But in this problem, the crossover can be realized by a simple gadget in Figure 7.

Figure 7. CROSS gadget.



Using the gadgets, we can complete the construction of the polygons that computes $f(x_1, \ldots, x_n)$, and the final output is connected to the polygon A in Figure 3. A simple example of the construction for $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$ is shown in Figure 8. (We note that this example is illustrative, though it can be simplified logically.) Some blocks are stretched in a trivial way to adjust their size.

Figure 8. A construction for $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$.



Proof. (of Theorem 1.) For any given monotone Boolean function f on n variables $x_1, \ldots, x_n$, we can construct the interlocked polygons S as described above. Let α be any assignment of $x_1, \ldots, x_n$, and $S(\alpha) \subset S$ be the set of polygons that consist of the polygons x_i with $x_i = 1$ in α. When we remove the polygons x_i in $S(\alpha)$ from S, the removal is propagated as described above. Then all the interlocked polygons can be free if and only if $f(x_1, \ldots, x_n) = 1$ in the assignment. It is easy to see that the other conditions in the main theorem are satisfied in the reduction. □

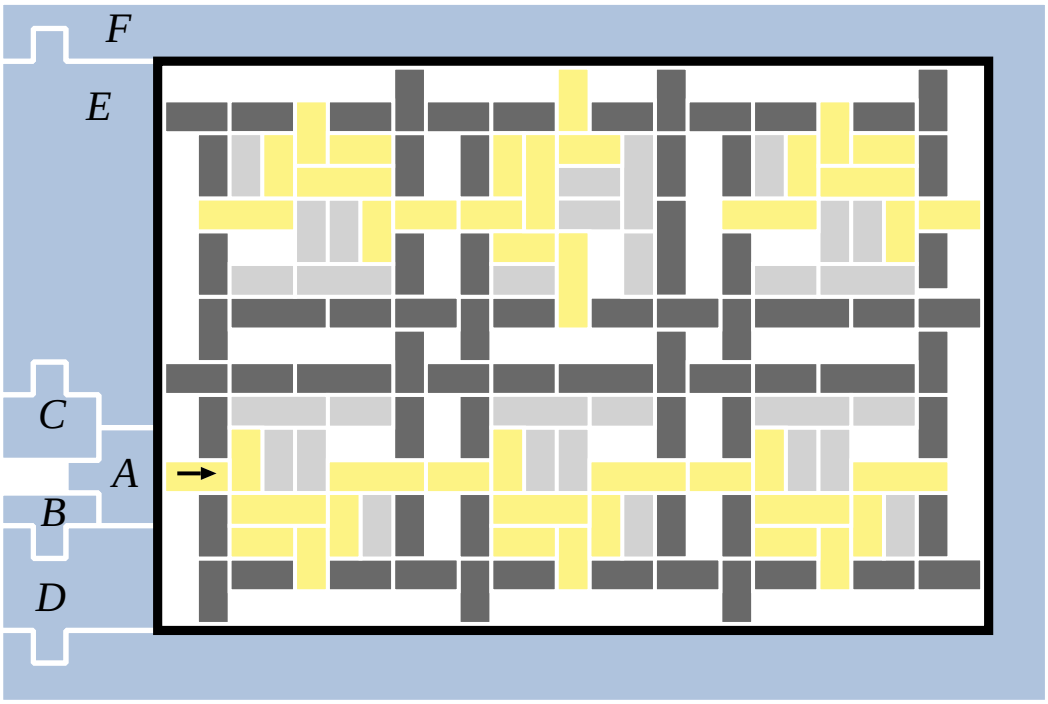
3. Exploding Sliding Block Puzzle

Now we turn to the the exploding sliding block puzzle. In this problem, for a given collection of polygons, we are asked if all blocks can be free. Combining the idea in Theorem 1 and the gadgets in [2], we have the following theorem.

Theorem 2. *The exploding sliding block puzzle is PSPACE-complete.*

Proof. It is PSPACE-complete to decide whether a given block on the boundary can move right in a collection of 1×2 rectangles in a big rectangle [2, Section 9.3]. See Figure 9 for a typical construction (in the figure, we use 1×2 rectangles and 1×3 rectangles to simplify, but 1×3 rectangles are not necessary by using more complicated construction); all small blocks drawn in solid lines are the collection, and a big solid rectangle indicates the frame of the puzzle. In the construction, it is PSPACE-complete to decide the block labeled "→" can move right. Now we replace the frame by the set of polygons that form the frame in Figure 3 as in Figure 9. (We also pad extra blocks to fill the gap in the figure; see [2, Section 9.3] for further details of the construction.) Then the exploding sliding block puzzle has a solution if and only if the block labeled "→" can move right. This completes the proof of PSPACE-completeness of the problem. □

Figure 9. An exploding sliding block puzzle.

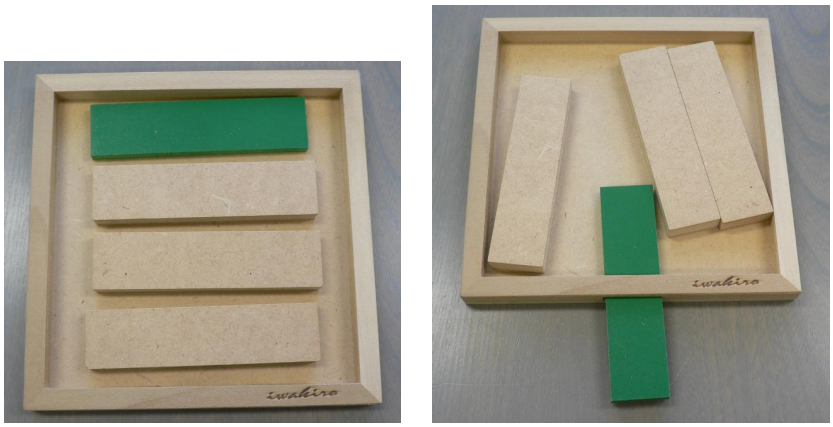


The exploding sliding block puzzle is a special case of a general decision problem:

Corollary 3. *It is PSPACE-complete to decide whether a given collection of polygons is interlocked or not.*

In the proof of Theorem 2, almost all blocks are rectangles. This motivates us to investigate the minimum number of nonrectangular blocks to make these problems PSPACE-complete. In fact, there exists an interesting and difficult puzzle that essentially consists of only four rectangles in a C-shape frame (Figure 10) (This "Rectangular Jam" puzzle won a prize at the IPP Puzzle Design Competition in 2005. See [4] for further details.). The following theorem gives the answer to the question; it is enough to have only one nonrectangular block to make the problems PSPACE-complete, and it is optimal.

Figure 10. A difficult puzzle that consists of four rectangles and one nonrectangle frame.



Theorem 4.

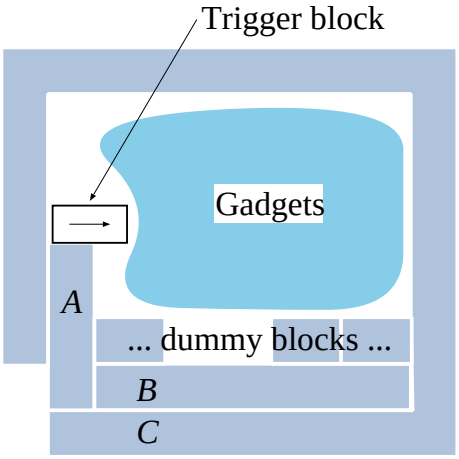
(1) When all blocks are rectangles, every instance of the exploding sliding block puzzle has a solution, that is, any collection of rectangles is free.

(2) When all blocks are rectangles except one block, the exploding sliding block puzzle in Theorem 2 and the decision problem in Corollary 3 are PSPACE-complete.

Proof.

(1) Any convex shapes (in any dimension) can be separated by simultaneous explosion, as proved by de Bruijn (see [5, page 2] for a comprehensive references). When we only consider orthogonal arrangements of rectangles, we have a simpler proof. We take any rectangle that touches the bounding box (which exists). Slide it out to infinity, and repeat. (Update the bounding box each time.)

Figure 11. Frame that requires only one nonrectangular piece.



(2) We use the frame in Figure 11 instead of one in Figure 3. Suppose that the trigger block moves to right. After moving the block A to up in the frame, we can remove the block B from inside of C. Then we can remove all dummy blocks. Using this room, each of the blocks of size 1×2 in

the gadgets can be rotated and slid out from C one by one. Finally, the block A can be slid out after a rotation. Therefore, we have the claim. □

4. Concluding Remarks

In Figure 2, there is no space in the interlocked polygons. Although we can pad some extra polygons in Figure 8, some spaces around the gadgets and inside of the gadgets cannot be padded to move the polygons. Especially, the spaces in the CROSS gadget seem to be essential. The interlocked polygons without any space in them are future work.

In our reduction, some gadgets can be glued into one piece. Typically, a sequence of wire blocks can be replaced by one long block, and this long block can be glued to the output of the last gadget. However, these blocks essentially work as a function, and they do not represent variables. Thus it is an open problem whether we can construct any monotone Boolean function on all the pieces as in Figure 2.

We may have a weaker goal for this problem; we are given a function of n pieces, and a "robustness" k. Then we want to construct $m = O(n + k)$ polygons so that, n of which are special and $m - n$ of which are extra, even if we remove up to k extra pieces, falling apart is determined by which special pieces we remove. Wires can be made robust in this way by a similar idea in Figure 2. However, the CROSS gadget seems to be difficult to make it robust.

In Theorem 4, we showed that some sliding block puzzles are PSPACE-complete even if all pieces are rectangles but one nonrectangular piece. In the rush hour puzzle (left puzzle in Figure 1), rectangular blocks are placed axis-parallel, and each block can move only one direction (along its long axis). In such a situation (or other restrictions), we do not know if one nonrectangular block is enough to make the problems in Theorem 2 and Corollary 3 PSPACE-complete. This is a nice future work; the number of nonrectangular blocks might measure the difficulty of the puzzles.

Acknowledgements

This work was initiated at the 25th Bellairs Winter Workshop on Computational Geometry, co-organized by Erik Demaine and Godfried Toussaint, held on February 6–12, 2010, in Holetown, Barbados. We thank the other participants of that workshop—Greg Aloupis, Brad Ballinger, Nadia Benbernou, Prosenjit Bose, David Charlton, Sébastien Collette, Mirela Damian, Karim Douïeb, Robin Flatland, Ferran Hurtado, John Iacono, Krishnam Raju Jampani, Anna Lubiw, Vera Sacristan, Vida Dujmović, Stefan Langerman, Pat Morin, Diane Souvaine—for providing a stimulating research environment.

References

1. Berlekamp, E.R.; Conway, J.H.; Guy, R.K. *Winning Ways for Your Mathematical Plays*; A K Peters Ltd.: Natick, MA, USA, 2001–2003; Volume 1–4.
2. Hearn, R.A.; Demaine, E.D. *Games, Puzzles, and Computation*; A K Peters Ltd.: Natick, MA, USA, 2009.
3. Leeuwen, J. *Handbook of Theoretical Computer Science*; Elsevier Science Publishers: Amsterdam, The Netherlands, 1990.

4. Rectangular Jam. Available online: http://home.r01.itscom.net/iwahiro/main/eng_contents/eng_-intro.html (accessed on 17 March 2012).
5. Demaine, E.D.; Langerman, S.; O'Rourke, J.; Snoeyink, J. Interlocked open and closed linkages with few joints. *Comput. Geom. Theory Appl.* **2003**, *26*, 37–45.