Theoretical Computer Science 677 (2017) 31–40

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# SNOWMAN is PSPACE-complete ☆

Weihua He [a], Ziwen Liu [b], Chao Yang [c,*]

[a] *Department of Applied Mathematics, Guangdong University of Technology, Guangzhou, China*
[b] *School of Software Engineering, South China University of Technology, Guangzhou, China*
[c] *School of Mathematics, Sun Yat-Sen University, Guangzhou, China*

ARTICLE INFO

*Article history:*
Received 11 November 2016
Received in revised form 16 February 2017
Accepted 13 March 2017
Available online 18 March 2017
Communicated by A. Fink

*Keywords:*
Combinatorial puzzles
Computational complexity
PSPACE-complete
Sokoban
Nondeterministic constraint logic

ABSTRACT

SOKOBAN is one of the most studied combinatorial puzzle game in the literature. Its computational complexity was first shown to be PSPACE-complete in 1997. A new proof of this result was obtained by Hearn and Demaine (2005) [8], by introducing the Nondeterministic Constraint Logic (NCL) problem. Since then, NCL has been used to prove the PSPACE-completeness of several other puzzles including a few SOKOBAN variants, by many authors. In this paper, we show that SNOWMAN, a new SOKOBAN-like puzzle game released in 2015, is PSPACE-complete by reduction from NCL.

© 2017 Elsevier B.V. All rights reserved.

## 1. Introduction

The computational complexity of many combinatorial puzzles have been determined over the last few decades. Before the age of computers, sliding block puzzles including the Fifteen Puzzle had been popular for almost a century. Puzzles flourished after the advent of personal computers. SOKOBAN is one of the most representative puzzle in the Computer Age. One of the reasons that people are attracted by puzzles is that puzzles are hard. Finding the shortest solution for the generalized Fifteen Puzzle was shown to be NP-hard [1,2]. Another popular sliding block puzzle Rush Hour is PSPACE-complete [3]. Peg solitaire is NP-complete [4]. Generalized Pete's Pike and Lunar Lockout are PSPACE-complete [5,6].

The computer puzzle game SOKOBAN was invented in Japan in 1981. It was first shown to be PSPACE-complete by Culberson [7] in 1997, by emulating every instance of Linear Bounded Automata (LBA) by a SOKOBAN level. Hearn and Demaine [8] gave a new proof for the PSPACE-completeness of SOKOBAN in 2005. They first defined a problem called Nondeterministic Constaint Logic (NCL) and proved NCL is PSPACE-complete. Then they showed SOKOBAN is PSPACE-complete by reduction from NCL.

*A Good Snowman Is Hard To Build* (in the rest of this paper, the game will be denoted by SNOWMAN) is a SOKOBAN-like computer puzzle game created by Hazelden, Davis and Roth. The game was published in 2015 for several operating systems, including Android [9], iOS [10], Windows and Mac OS X [11]. The mechanism of the game will be introduced and extended in Section 2, and the computational complexity of this puzzle is determined in this paper.

☆ This work was supported by the National Natural Science Foundation of China (No. 11201496 and No. 11601093), and the Guangdong Natural Science Foundation (No. 2015A030313222).

* Corresponding author.
*E-mail addresses:* yangchao0710@gmail.com, yangch8@mail.sysu.edu.cn (C. Yang).

http://dx.doi.org/10.1016/j.tcs.2017.03.011
0304-3975/© 2017 Elsevier B.V. All rights reserved.

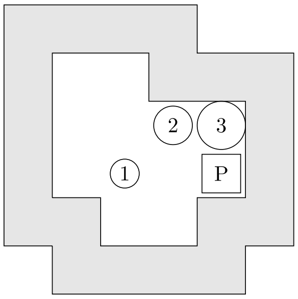

**Fig. 1.** A SNOWMAN level.

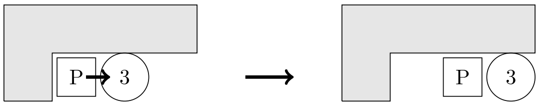

**Fig. 2.** Push like SOKOBAN.

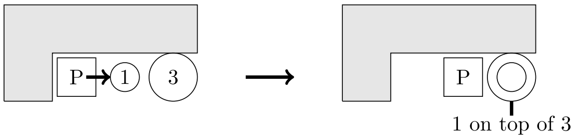

**Fig. 3.** Push a snowball against a stack of snowballs, in this case the stack consists of only one snowball.

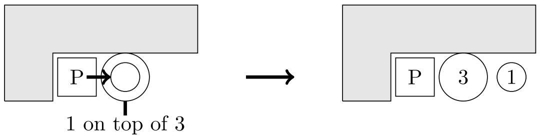

**Fig. 4.** Push to knock off the topmost snowball of a stack.

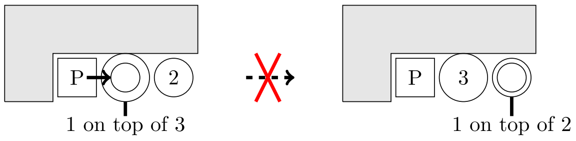

**Fig. 5.** Forbidden push.

## 2. SNOWMAN-*k* and NCL

The goal of the original SNOWMAN puzzle is to control a player in a 2-dimensional maze to build snowmen by pushing snowballs of three different sizes. The maze is a grid of squares, each square may be occupied by an unmovable wall, a player, a snowball, or a stack of snowballs. The player can move around or push snowballs horizontally or vertically. Fig. 1 shows a simple SNOWMAN level. The gray area represents the walls, the square represents the player, and the circles represent the snowballs of different sizes. Snowballs labeled with bigger number are larger.

The basic mechanism of SNOWMAN is the same as that of the classic puzzle SOKOBAN. Only one snowball can be pushed at a time, regardless of the size of the snowball. A snowball can be pushed either to an empty adjacent location, or to the top of a stack of snowballs provided the snowball being pushed is smaller than the topmost one of the stack.

A stack of three snowballs forms a snowman. If a stack of snowballs doesn't form a snowman yet, the topmost snowball can be knocked off to the other side of the player by pushing the stack, as long as the other side is unoccupied.

The three different kinds of push are illustrated in Fig. 2, Fig. 3 and Fig. 4. A typical SNOWMAN level has one player and $3n$ snowballs, with equal number of snowballs of each size, and the level is solved if $n$ snowmen have been built.

Note that it is not allowed to push a snowball from one stack to another directly (Fig. 5).

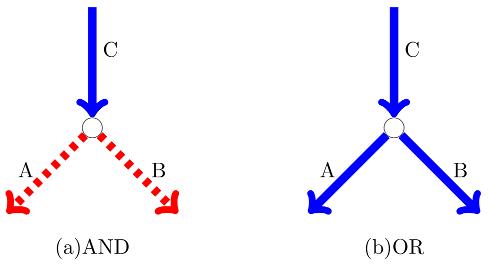

**Fig. 6.** NCL (a) an AND vertex, (b) an OR vertex.

The original SNOWMAN puzzle also introduced a mechanism that allows the snowballs to become larger when rolling over snow-covered ground. We will ignore this and assume that the sizes of all the snowballs remain unchanged during the game. Also, the knocking-off push (Fig. 4) is not necessary in our reduction. So we are in fact studying a subproblem of the original SNOWMAN puzzle. We will show that even with these restrictions, the corresponding decision problem is as hard as the classic puzzle SOKOBAN.

Meanwhile, we consider a generalized version of the original SNOWMAN puzzle, which is called SNOWMAN-$k$ with $k \geq 2$. In SNOWMAN-$k$, there are snowballs of $k$ different sizes, which will be denoted by integers from 1 to $k$. Bigger numbers represent larger snowballs. By *a set of snowballs*, we mean $k$ snowballs of different sizes (not necessarily form a stack). As the original SNOWMAN, only a smaller snowball can be stacked on top of a larger one in SNOWMAN-$k$. A snowman is built by making a stack of $k$ snowballs at any location of the maze.

**Definition 1** *(SNOWMAN-$k$ problem).*
INPUT. A SNOWMAN-$k$ level with $n$ sets of snowballs.
OUTPUT. Is there a sequence of allowed moves to build $n$ snowmen by these snowballs?

The main result of this paper is the following theorem.

**Theorem 1.** *For any fixed $k \in \mathbb{N}$, $k \geq 2$, SNOWMAN-$k$ is PSPACE-complete.*

We shall prove Theorem 1 in the next section by reduction from the NCL, which was introduced by Hearn and Demaine [8]. There are several slightly different versions of decision problems for NCL, and all of them are PSPACE-complete. For our purpose, we just need the version for the configuration-to-configuration planar NCL.

An instance of the configuration-to-configuration planar NCL problem is defined on a 3-regular planar directed graph, which is called an NCL graph. Each edge of the graph has a weight of either 1 or 2. Moreover, the NCL graph is consisted of only two kinds of vertices, the AND vertices and the OR vertices. An AND vertex is incident with two edges of weight 1 and one edge of weight 2, and an OR vertex is incident with three edges of weight 2 (Fig. 6).

A configuration of the NCL graph is a specific orientation of the edges, and the configuration is valid if and only if the sum of weights of the incoming edges is at least 2 at each vertex. A valid move for the NCL graph is the change of the orientation of any edge, by keeping the configurations valid before and after the change. Given an NCL graph, together with a valid initial configuration and a valid target configuration for that graph, the NCL problem asks whether there is a sequence of valid moves from the initial configuration to the target configuration such that all intermediate configurations are valid.

**Definition 2** *(NCL problem).*
INPUT. A planar NCL graph with an initial configuration and a target configuration.
OUTPUT. Is there a sequence of valid moves leading the NCL graph from the initial configuration to the target configuration?

**Theorem 2** *([8]). NCL is PSPACE-complete.*

## 3. The main result

First we show that SNOWMAN-$k$ is in PSPACE. Let $t$ be the total number of grid squares of a SNOWMAN-$k$ level. For each square, there are at most $2^k + 1$ different possible states, because there are $2^k$ different kinds of stacks of snowballs (including the empty stack), plus one state that the square is occupied by the player. Therefore, the total number of different possible configurations of the level with $t$ squares is at most $(2^k + 1)^t$. Obviously, if a level is solvable, it can be solved in at most $(2^k + 1)^t$ steps. So we can try out all different ways to solve a level in a nondeterministic Turing Machine and stop at $(2^k + 1)^t$ steps. The space used in this nondeterministic algorithm has two parts, the first part is the space

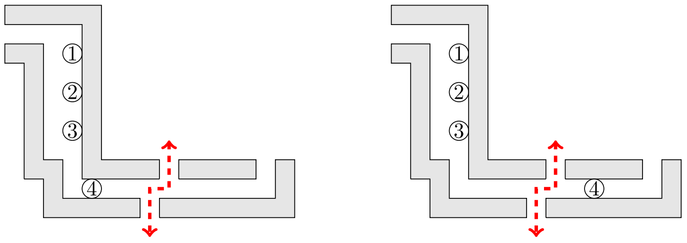

**Fig. 7.** A segment of an edge gadget. (Left) matched (right) mismatched.

for storing of the $t$ squares of the current configuration of level, and the second part is the space for the step counter, which requires $\log_2(2^k+1)^t = t\log_2(2^k+1)$ bits. Thus the overall size of the space is linear to $t$, the size of the level. This shows the problem SNOWMAN-$k$ is in NPSPACE. By Savitch's Theorem (see for example pages 279–281 of [12]) that NPSPACE=PSPACE, SNOWMAN-$k$ is also in PSPACE.

To show that SNOWMAN-$k$ is PSPACE-complete, we shall construct a SNOWMAN-$k$ level for every instance of the NCL, such that the NCL instance is solvable if and only if the corresponding SNOWMAN-$k$ level is solvable.

The SNOWMAN-$k$ levels we shall construct are open access levels, that is, the player can access and push any snowball in the maze without pushing any other snowballs. And the level is constructed by piecing together gadgets which emulate the directed edges, the AND vertices and the OR vertices of the NCL graph.

### *3.1. The edge gadget*

An edge gadget is a segmented tunnel. As we shall see, the main idea in constructing the edge gadget is to constrain the movement of the snowballs. By the rules of the SNOWMAN-$k$ puzzle, a snowman can be built at any location of the level. But in our edge gadget, snowmen can only be built at the specific corner locations in each segments.

A segment should contain at least, in order, a tunnel of breadth 1, a corner, a tunnel of breadth 2, another corner, a tunnel of breadth 1, a passage, a tunnel of breadth 1 and a corner. And a set of snowballs are placed inside the segment.

A segment is illustrated in Fig. 8 for SNOWMAN-4, but the mechanism can be easily applied to SNOWMAN-$k$. The *passage* of the segment is illustrated by the dashed line, which allows the player to go freely from one side of the edge to the other, thus helps making the level open access as we just mentioned. Another feature that makes the level open access is that each vertex gadget will have at least one entrance open, which will be explained in the next subsection.

In the segment, all but the largest snowballs are placed inside the tunnel of breadth 2. The largest snowball can be pushed across the passage. Base on the position of the largest snowball, the segment is in either *matched* configuration or *mismatched* configuration (see Fig. 7). Obviously, a snowman can only be built at the corner location from the snowballs inside that segment.

**Definition 3.** A segment is said to be *matched* if all the snowballs are in the same side of the passage. Otherwise it is *mismatched*.

A $m$-segment edge gadget is constructed by the concatenation of $m$ segments. The two ends of the tunnel would be connecting to the entrances of the vertex gadgets. Fig. 8 illustrates a 2-segment edge gadget. We will never need more than 3 segments in our reduction.

In the edge gadget, the segments can change between matched or mismatched configuration in a chain-reaction manner. In Fig. 8, the left segment can switch to mismatched configuration only if the right segment has been switched to mismatched configuration first, otherwise the level cannot be solved anymore.

**Definition 4.** An edge gadget is in *matched configuration* (*mismatched configuration*) if all of its segments are matched (mismatched).

Therefore, the edge gadget is always blocking the entrance of one of its two end-vertex gadgets, and leaving the other entrance open. Before building the snowmen, the edge gadget can switch between these two configurations, which we will call the *blocking configuration* or *nonblocking configuration*, with respect to a vertex gadget. Once all the snowmen have been built, the configuration can never be changed again. This forces us to put off the snowmen-building process to the final stage of solving the puzzle.

**Definition 5.** An edge gadget is said to be *blocking* a vertex gadget, if the entrance to the vertex gadget from this edge is blocked by snowballs. Otherwise, it is *nonblocking*.

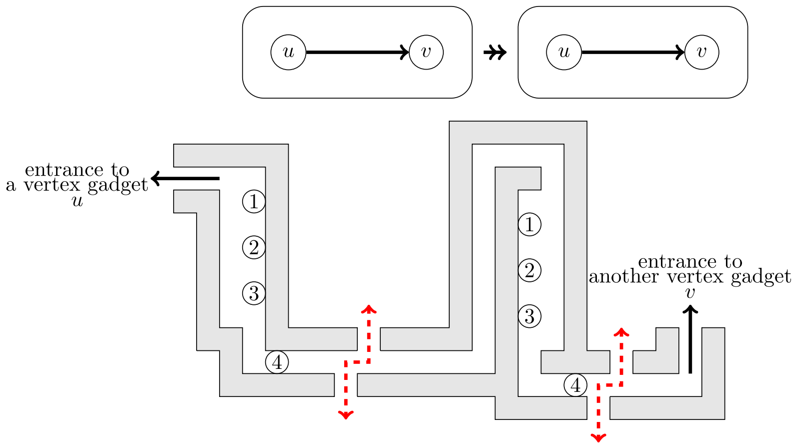

**Fig. 8.** An 2-segment edge gadget in matched configuration, and blocking the entrance to the vertex gadget on the left.

So the blocking and nonblocking configurations emulate the two possible orientations of the edge. An edge gadget emulates a directed edge pointing from vertex $u$ to vertex $v$ if it is blocking the vertex gadget of $u$ and is nonblocking with respect to the vertex gadget of $v$, as we shall see in more detail in the next subsection. In other words, blocking means pointing away, nonblocking means pointing in. Also, matched configurations correspond to correctly oriented edges, and mismatched configurations correspond to edges that have to be reoriented. In Fig. 8, the NCL edge that the gadget is emulating is shown on the top. The top left is the initial configuration, and the top right is the target configuration.

Note that the edge gadget is very versatile, and the tunnel can be very long and can turn corners if needed, so there would be no difficulty in connecting two vertex gadgets with an edge gadget.

In this subsection, we introduced the standard version of the edge gadget which can be used to emulate edges of either weight 1 or weight 2. The weight is not determined by the edge gadget itself but by the way how the edge gadgets are connected at the vertex gadget. We would need a minor variant of the standard version in constructing the AND gadgets in the next subsection. This variant is only used in some of the edge gadgets of weight 1.

## *3.2. The OR gadget and the AND gadget*

Three edge gadgets can be joined to form an OR vertex, as illustrated in Fig. 9. The OR vertex is represented by the region enclosed by the dashed rectangle. It is a room-like region surrounded by walls with three entrances, and each entrance is connected to an edge. Both edge $A$ and edge $B$ are in blocking configuration, with respect to this OR vertex gadget, and the edge $C$ is nonblocking. Obviously, either edge $A$ or edge $B$ must switch to nonblocking before edge $C$ changing to blocking. Otherwise, all the three entrances of the OR vertex gadget have been blocked, and the level cannot be solved anymore. This emulates an OR vertex of the NCL graph such that the edge $A$ and edge $B$ are pointing away from the vertex, and edge $C$ is pointing to the vertex.

**Remark 1.** Here, we view the OR gadget as an empty area connected to 3 edges. We can also define the OR gadget as 3 segments joined together at one end. The OR gadget defined in the latter way is larger, as it includes the first segment of each connecting edge as part of the OR gadget. The same viewpoint applies to the AND gadgets too.

In a slightly different way, three edges can be joined to form an AND vertex, as illustrated in Fig. 10 and Fig. 11 for two different cases. The AND vertex is the region enclosed by the dashed rectangle, with only two entrances. The two entrances are connecting to the edge $B$ and the edge $C$. Instead of being attached to the region of the AND vertex gadget directly, the edge $A$ is attached to the first passage of the edge $B$. By *first passage*, we mean the passage of the edge gadget $B$ which is nearest to the AND gadget region. Therefore, both the edge $A$ and the edge $B$ must switch to nonblocking configuration before the edge $C$ can switch to blocking configuration. The snowballs labeled *key snowball* in Fig. 10 and Fig. 11 play a very important role in making the AND gadgets work properly, to which we will give more explanation soon.

Now, we can give a short summary of how to construct a SNOWMAN-$k$ level from a given instance of the NCL problem. For each directed edge of the NCL graph, if the initial orientation is the same as the target orientation, the corresponding edge gadget should be in matched configuration initially, otherwise the edge gadget should be in mismatched configuration. Moreover, if a directed edge of the NCL graph is pointing from vertex $u$ to vertex $v$ initially, then the corresponding edge gadget should be blocking the vertex gadget for $u$ and nonblocking with respect to the vertex gadget for $v$.

Therefore, at each vertex gadget, an incident blocking edge gadget may be either matched or mismatched. The same goes for a nonblocking edge gadget. This could be done pretty easily if the edge gadget is representing an edge of weight 2, or is representing an edge of weight 1 but is connecting to the AND gadget region indirectly. It is a little tricky for an edge

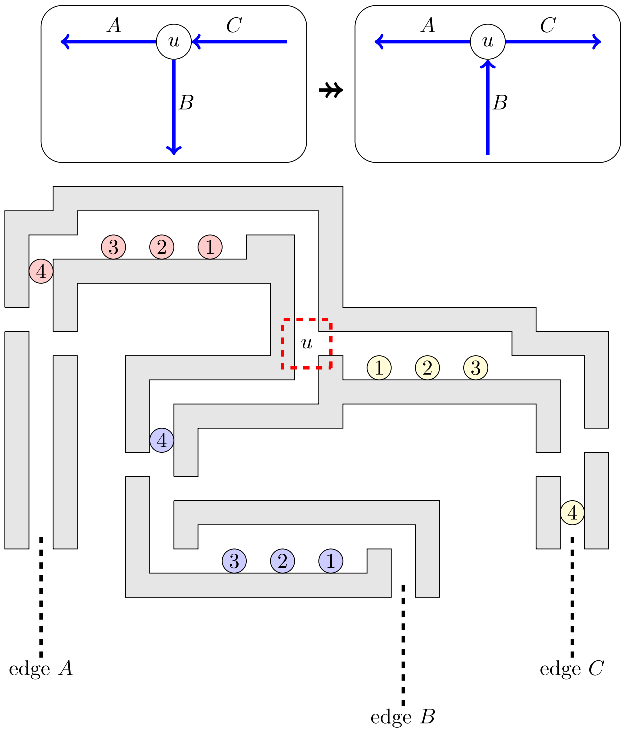

**Fig. 9.** OR gadget.

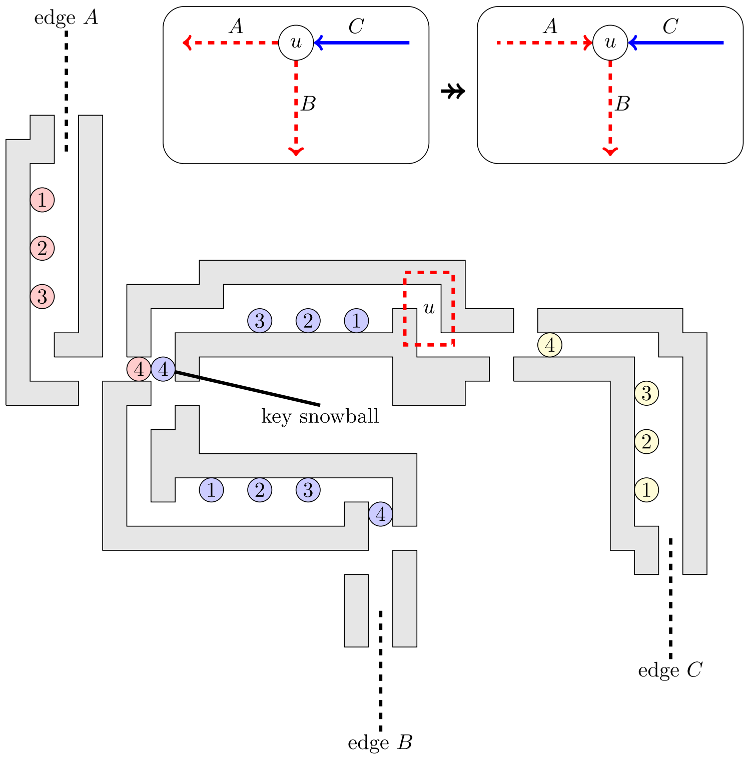

**Fig. 10.** AND gadget, with edge $B$ blocking and matched.

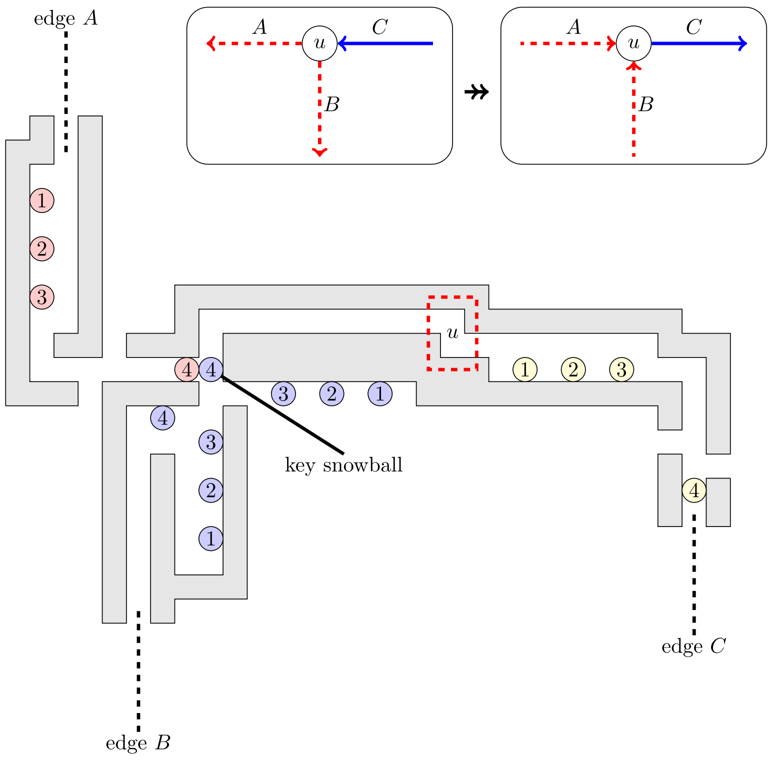

**Fig. 11.** AND gadget, with edge $B$ blocking and mismatched.

gadget of weight 1 connecting to the entrance of the AND gadget directly, so we have to construct the gadgets differently for two cases, as shown in Fig. 10 and Fig. 11. In Fig. 10, the edge $B$ is both blocking and matched, while in Fig. 11, the edge $B$ is both blocking and mismatched.

In both cases, the AND vertex gadget has the following two features, thus emulates the AND vertex of the NCL graph perfectly.

1. Edge $B$ can be switched to nonblocking configuration by pushing the key snowball just one step downwards, but not two steps, so it still blocks one side of the first passage (and the other side is blocked by edge gadget $A$). To achieve this, in Fig. 10, the key snowball cannot be pushed two steps downwards because that is a corner location. And in Fig. 11, one more set of snowballs are added to avoid the key snowball being pushed two steps downwards. If the key snowball is pushed two steps downwards, there is no way to build two snowmen from the two sets of snowballs in edge $B$, and the level becomes unsolvable. So it is actually a minor variant of the edge gadget introduced in the previous subsection. To see why this additional set of snowballs is needed, we present a failed AND gadget constructed from the standard edge gadgets in Fig. 12. In this failed construction, edge $B$ can be switched to nonblocking configuration by pushing the key snowball two steps downwards. As a result, edge $B$ could not change orientation any more, but then edge $C$ can switch to blocking configuration without edge $A$ being switched to nonblocking configuration, and the level may still be solvable. Thus the gadget fails to emulate the AND logic, and does not work as intended.
2. The edge $A$ and edge $B$ can switch to nonblocking configuration independently. In both Fig. 10 and Fig. 11, edge gadget $B$ is connected to the AND vertex room directly, it can be switched to nonblocking configuration without disturbing the edge gadget $A$. We can also switch edge gadget $A$ to nonblocking configuration while keeping the edge gadget $B$ in blocking configuration, by pushing the key snowballs one step upwards first. Note that by pushing the key snowball one step upwards, the AND gadget remains essentially unchanged.

**Remark 2.** The AND gadgets may alternatively be defined to include a few segments of its incident edges. This will eliminate the need for a variant of the edge gadget.

## 3.3. *Piecing together*

**Proof of Theorem 1.** We have already shown that SNOWMAN-$k$ is in PSPACE at the beginning of this section. To prove that SNOWMAN-$k$ is PSPACE-complete, we give a polynomial time reduction from NCL to SNOWMAN-$k$. The reduction converts

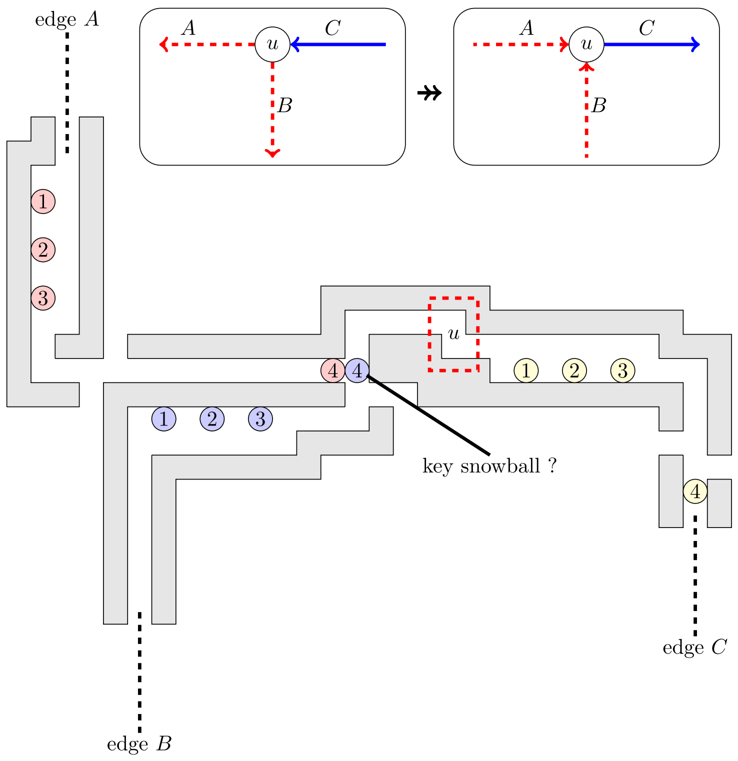

**Fig. 12.** A failed AND gadget, with edge $B$ blocking and mismatched.

an instance of the NCL problem to a SNOWMAN-$k$ level, by using the gadgets described in the previous subsections. Because we are considering planar NCL graph, there will be no edge crossings, and the construction of SNOWMAN-$k$ level should be straight forward. The AND gadgets may need to be taken care of, as we have introduced two cases in the previous subsection. But all the edges of weight 1 in an NCL graph induce a subgraph of disjoint cycles, therefore it suffices to show that one cycle of AND vertex gadgets can be constructed. See Appendix A (Figs. A.13 and A.14) for a complete SNOWMAN-2 level built from an instance of NCL.

The solving process of the SNOWMAN-$k$ level can be divided into two stages. In the first stage, the player tries to set all the edge gadgets to the matched configurations. If this cannot be achieved, the level is unsolvable. If all edge gadgets have been set to the matched configuration, proceed to the second stage. In the second stage, each set of snowballs can be pushed to build a snowman easily because each vertex gadget has at least one open entrance, and the level is solved.

Therefore, a SNOWMAN-$k$ level is solvable if and only if the first stage can be done successfully, which in turn is equivalent with the solvability of the corresponding instance of the NCL problem. □

## 4. Conclusion

In this paper, we prove that the SNOWMAN-$k$ problem is PSPACE-complete for every fixed integer $k \geq 2$ by reduction from the NCL problem. The NCL problem has been proved to be a very efficient tool in studying the computational complexity of puzzles by many different authors. Pereira et al. show two interesting variants of SOKOBAN, the variant PULL which the player can only pull boxes, and the variant PUSHPULL which the player can pull as well as push, are both PSPACE-complete in [13], by reduction from NCL. Using the same method, the food-eating-snake puzzle NIBBLER [14] and the rolling block puzzle [15] were shown to be PSPACE-complete. Our result gives yet another application of the NCL problem.

Though the NCL problem is generally applicable in showing the PSPACE-completeness for puzzles, it still requires considerable efforts in designing sophisticated gadgets in the specific puzzle. In our case, the most sophisticated gadget is the AND gadget with a directly attached edge gadget of weight 1 in mismatched configuration (Fig. 11).

We would like to point out that without fixing $k$, the number of different sizes of snowballs, the SNOWMAN problem is still in the class PSPACE. In other words, in a SNOWMAN level, we do not require to have equal number of snowballs of different sizes, and we solve a level by building snowmen with different heights and without leaving behind a single snowball. As a consequence of Theorem 1, SNOWMAN is also PSPACE-complete.

**Definition 6** *(Snowman problem).*
Input. A Snowman level with snowballs of different sizes.
Output. Is there a sequence of allowed moves to use up all the snowballs by building snowmen?

**Theorem 3.** Snowman *is PSPACE-complete.*

We conclude by making a few comments on future work. An interesting variant of the original Sokoban puzzle is to allow the player to push more than one box at a time ([13]). How can we define similar variant that the player can push more than one snowballs (not stacks) at once for Snowman properly? Does the problem become more difficult? Can we still determine its computational complexity by reduction from Ncl? What is needed to modify in our gadgets to prove that the new variant is also PSPACE-hard?

## Acknowledgements

The authors would like to thank the two anonymous reviewers for their valuable comments.

## Appendix A. A simple example

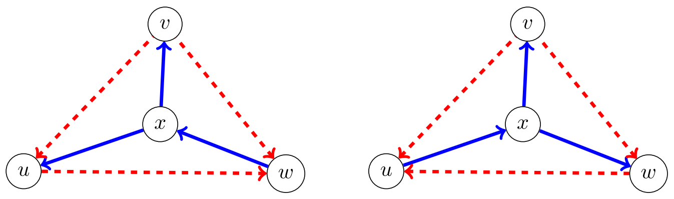

**Fig. A.13.** Ncl: initial configuration on the left, target configuration on the right.

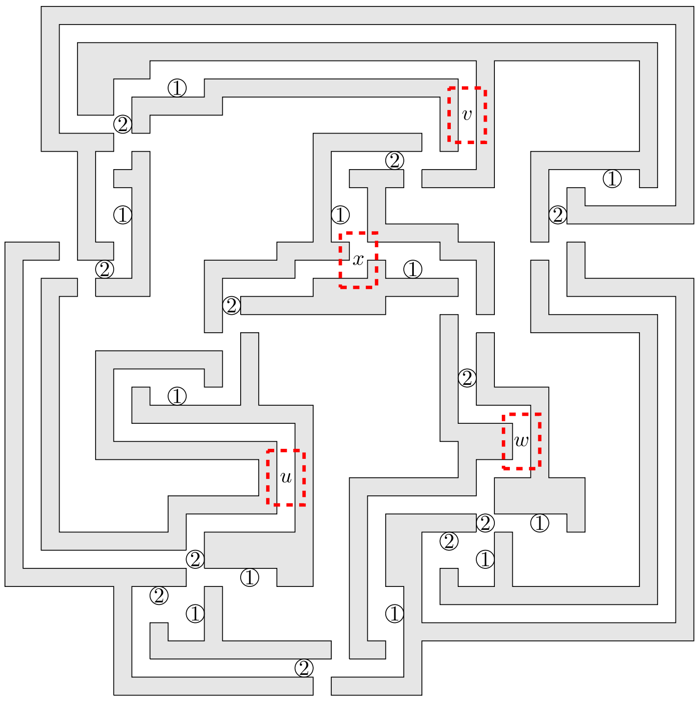

**Fig. A.14.** A Snowman-2 level emulating the above Ncl instance.

## References

[1] O. Goldreich, Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard, in: O. Goldreich (Ed.), Studies in Complexity and Cryptography, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 1–5, http://dl.acm.org/citation.cfm?id=2028116.2028118.
[2] D. Ratner, M. Warmuth, The $(n^2-1)$-puzzle and related relocation problems, J. Symbolic Comput. 10 (2) (1990) 111–137, http://dx.doi.org/10.1016/S0747-7171(08)80001-6, http://www.sciencedirect.com/science/article/pii/S0747717108800016.
[3] G.W. Flake, E.B. Baum, Rush Hour is PSPACE-complete, or "why you should generously tip parking lot attendants", Theoret. Comput. Sci. 270 (1) (2002) 895–911, http://dx.doi.org/10.1016/S0304-3975(01)00173-6, http://www.sciencedirect.com/science/article/pii/S0304397501001736.
[4] R. Uehara, S. Iwata, Generalized Hi–Q is NP-complete, IEICE Trans. E73-E (2) (1990) 270–273, http://hdl.handle.net/10119/4709.
[5] J.R. Hartline, R. Libeskind-Hadas, The computational complexity of motion planning, SIAM Rev. 45 (3) (2003) 543–557, http://dx.doi.org/10.1137/S003614450139517, http://epubs.siam.org/doi/abs/10.1137/S003614450139517.
[6] B. Meyer, Generalized Pete's Pike is PSPACE-complete, Theoret. Comput. Sci. 613 (2016) 115–125, http://dx.doi.org/10.1016/j.tcs.2015.11.036, http://www.sciencedirect.com/science/article/pii/S0304397515011226.
[7] J.C. Culberson, Sokoban Is PSPACE-Complete, Tech. rep. TR 97-02, Department of Computing Science, University of Alberta, 1997.
[8] R.A. Hearn, E.D. Demaine, PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation, Theoret. Comput. Sci. 343 (1) (2005) 72–96, http://dx.doi.org/10.1016/j.tcs.2005.05.008, http://www.sciencedirect.com/science/article/pii/S0304397505003105.
[9] A. Hazelden, B. Davis, R. Roth, A good snowman is hard to build — Google Play. Online; accessed 6 August 2016, https://play.google.com/store/apps/details?id=com.agoodsnowman, 2015.
[10] A. Hazelden, B. Davis, R. Roth, A good snowman is hard to build — Apple iTunes. Online; accessed, 6 August 2016, https://itunes.apple.com/us/app/a-good-snowman-is-hard-to-build/id1040930654?ls=1&mt=8, 2015.
[11] A. Hazelden, B. Davis, R. Roth, A good snowman is hard to build — Steam Store. Online; accessed 6 August 2016, http://store.steampowered.com/app/316610/, 2015.
[12] M. Sipser, Introduction to the Theory of Computation, PWS Publishing Company, 1997.
[13] A.G. Pereira, M. Ritt, L.S. Buriol, Pull and PushPull are PSPACE-complete, Theoret. Comput. Sci. 628 (2016) 50–61, http://dx.doi.org/10.1016/j.tcs.2016.03.012, http://www.sciencedirect.com/science/article/pii/S030439751600205X.
[14] M.D. Biasi, T. Ophelders, The complexity of Snake, in: E.D. Demaine, F. Grandoni (Eds.), 8th International Conference on Fun with Algorithms, FUN 2016, in: LIPIcs. Leibniz Int. Proc. Inform., vol. 49, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2016, pp. 11:1–11:13, http://drops.dagstuhl.de/opus/volltexte/2016/5884.
[15] K. Buchin, M. Buchin, Rolling block mazes are PSPACE-complete, J. Inf. Process. 20 (3) (2012) 719–722, http://dx.doi.org/10.2197/ipsjjip.20.719.