

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/352934749>

# Block Dude Puzzles are NP-Hard (and the Rugs Really Tie the Reductions Together)

Conference Paper · July 2021

CITATIONS

0

READS

263

3 authors, including:



Aaron Williams

Williams College

80 PUBLICATIONS 754 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Computational Complexity of Video Games and Puzzles [View project](#)



Retrogame Archeology [View project](#)

# Block Dude Puzzles are NP-Hard (and the Rugs Really Tie the Reductions Together)

Austin Barr\*

Calvin Chung†

Aaron Williams‡

## Abstract

The computational complexity of agent-based box-pushing puzzles on grids is well-studied. In particular, the video game *Sokoban* was shown to be NP-hard, and later PSPACE-complete, in the mid-1990s, and dozens of variants have since been studied. In this paper, we change the top-down perspective to a side perspective, where the player and the boxes are subject to gravity, and the player is able to climb on top of boxes or walls of height one. We prove that determining whether a level is solvable is NP-hard when the goal is to reach the exit, or place the boxes on target locations. The former result was previously shown to be true with “*Dig Dug* gravity” (i.e. boxes are subject to gravity, but the player is not) by Friedman. We also consider the decision problems with pushable boxes being replaced by liftable blocks, or with pushable and liftable blox, or with general crates. In total, we establish NP-hardness for eight different decision problems, all based around a single reduction. The inspiration for this article was the classic TI 83/84 calculator game *Block Dude*, which requires reaching an exit in the presence of liftable blocks.

## 1 Introduction

This section reviews video games that use box pushing, and the complexity of related decision problems.

### 1.1 Box-Pushing

*Sokoban* (倉庫番) was created by Hiroyuki Imabayashi, and released on cassette tape by Thinking Rabbit for a variety of Japanese personal computers in 1982 [40]. In total, the company has released nearly 100 official versions of the game [37], with the most recent entry being *Everyone’s Sokoban* (みんなの倉庫番) for the Nintendo Switch and PlayStation 4 (see Figure 1).

Despite the game’s long history, its basic rule set has never changed<sup>1</sup>. The game is played on a grid with a top-down perspective. Some of the cells are filled with

walls or boxes, and the player controls a character that occupies a single cell. The player can move in the cardinal directions, and has the power to push — not pull — a single box into the next cell, so long as that cell is not occupied by a wall or another box. In addition, some of the blank cells are marked as targets, and they are equinumerous with the boxes. The goal is to rearrange the  $k$  boxes so that they occupy the  $k$  target cells.



Figure 1: Thinking Rabbit’s releases of Sokoban include *Sokoban* (1982) for the NEC PC-8801 (left), and *Everyone’s Sokoban* (2019) for the Nintendo Switch (right).

Spectrum Holobyte published the game under the name *Soko-Ban* for American personal computers in 1988, which was the same year it brought *Tetris* to the same platform. Today, Thinking Rabbit and the Tetris Company still make modern versions of their respective games. However, the Tetris Company vigorously defends its intellectual property against other falling-block puzzles [36], whereas Thinking Rabbit does not. As a result, the term “Sokoban” has become genericized; it is synonymous with the genre of box-pushing puzzle games, and it can be found in the title of games that are not affiliated with Thinking Rabbit.

Sokoban is also a popular research topic. There are over 100 publications with “Sokoban” or “倉庫番” in the title [1], ranging from artificial intelligence solvers [20, 21] and optimizers [26], to level generation [23, 34] and evaluation [4], and human solutions [35]. Many results have also been presented in less formal venues, including websites discussing levels with the most moves [13], and undergraduate theses on levels that require an exponential number of moves to solve [28].

The Sokoban decision problem was shown to be NP-hard by Fryers and Greene in *Eureka* magazine [14], and independently by Dor and Zwick [10], and Uehara [38]<sup>2</sup>.

<sup>2</sup>The original Port Huron Statement [33, 41] does not contain a proof of NP-hardness, nor does its compromised second draft.

\*Williams College, [abarr877@gmail.com](mailto:abarr877@gmail.com)

†Williams College, [calvintchung@gmail.com](mailto:calvintchung@gmail.com)

‡Williams College, [aaron.williams@williams.edu](mailto:aaron.williams@williams.edu)

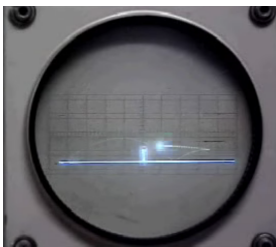
<sup>1</sup>There is an exception: The first releases include fake wall tiles that the player must find and pass through to complete the level.

The exponential level constructions in [28] show that the most obvious certificate — the sequence of moves — is insufficient for establishing membership in NP. Thus, it was natural to consider more difficult complexity classes. Culberson proved that SOKOBAN<sup>3</sup> is PSPACE-complete by using it to simulate a bounded Turing Machine [5]. Using nondeterministic constraint logic, Hearn and Demaine showed that PSPACE-hardness holds even for levels that contain no walls [18].

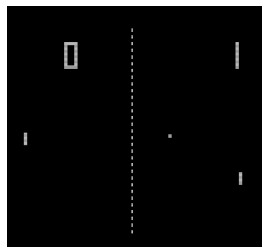
Countless variations of Sokoban exist as video games, and many of these variations have been studied through the lens of computational complexity. If the player is required to reach an exit, rather than place the boxes on target locations, then the decision problem is known as PUSH-1. If the player can push  $k$  boxes instead of a single box, then the decision problem is PUSH-K. In PUSH-PUSH, the boxes slide until hitting the next box or wall. More generally, there are 16 different decision problems under the PUSH[PUSH]-1/ $k$ /\*-[X] umbrella, all of which are known to be NP-hard or PSPACE-complete (see [8, 19]).

## 1.2 Side Perspective

In early video game history, it was common for ideas to be implemented in both top-down and side perspectives. For example, Ralph Baer’s *Brown Box* (1967) and Atari’s *Pong* (1972) use a top-down perspective for tennis, and were predated by William Higenbotham’s *Tennis for Two* (1958), which uses a side-view on an oscilloscope. Similarly, the *trap ’em-up* genre began with the top-view in *Heiankyo Alien* (1979) by Tokyo’s Theoretical Science Group, before moving to a side-view in *Space Panic* (1980), and *Lode Runner* (1983).



(a) *Tennis For Two* (1958)



(b) *Pong* (1972)

Figure 2: Tennis with top-down and side perspectives.

Despite *Sokoban*’s popularity, we are unaware of any video game that is based solely on box-pushing from a side perspective. Since games have inspired many of the academic investigations into motion planning<sup>4</sup>, it is understandable that there are no computational complexity results for the corresponding decision problem.

<sup>3</sup>If *Title* is a video game, then *TITLE* refers to the following decision problem: Can a given level of *Title* be completed? The game is also suitably generalized (i.e. unbounded level size.)

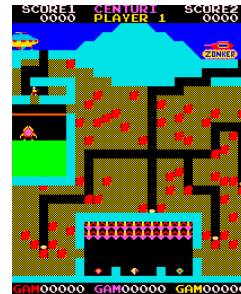
<sup>4</sup>For a recent example with turnstiles, see Greenblatt et al [15].

## 1.3 Dig Dug Gravity

Among commercial games, the closest matches to box pushing from a side perspective come from the *dig ’em up* genre (see Figure 3). In these games, the player can remove dirt, which may dislodge objects that then fall downward. In particular, the apples in *Mr. Do!* and the boulders in *Boulder Dash* can also be pushed horizontally. Unlike *Sokoban*, this genre also features action-oriented gameplay, with monsters and time limits that distract from a pure puzzle-solving experience.

The dig ’em up genre also features peculiar physics: The player and the enemies are not subject to gravity. In other words, it is as if some game elements operate according to a top-down perspective, while others adhere to a side-view. We refer to this physical model, which is as curious as it is common, as *Dig Dug gravity*, owing to the genre’s most popular game.

These games also differ from box pushing games in their objectives. For example, the goal of *Dig Dug* is to eliminate a number of enemies. On the other hand, in *Boulder Dash*, the player must collect some number of diamonds, and then reach an exit.



(a) *The Pit* (1982)



(b) *Dig Dug* (1982)



(c) *Mr. Do!* (1982)



(d) *Boulder Dash* (1984)

Figure 3: Dig ’em up games use Dig Dug gravity (i.e. the player and enemies don’t fall), and dirt that can be cleared by the player. The games include falling boulders (or apples), with (c) and (d) also supporting different types of pushing. They also include enemies and/or time limits, and objectives that differ from the SOKOBAN and PUSH problems.

Friedman introduced the PUSH-1-G decision problem, which adds Dig Dug gravity to PUSH-1. In other words, it asks if a player can reach an exit when the level consists of walls and pushable boxes, where the boxes

are subject to gravity, but the player is not. He proved that the problem is NP-hard, as is its generalization PUSH-K-G for any  $k \geq 1$  [12]. Friedman’s gadgets can be implemented directly in *Boulder Dash*<sup>5</sup> and adding diamonds to the exit proves that the game is NP-hard.

The NP-hardness of BOULDERDASH was also established by Viglietta. More generally, Metatheorem 1 in [39] proves that avatar-based games with (a) walls, (b) *single-use paths*, and (c) *location traversal*, are NP-hard. Single-use paths become blocked when they are traversed, while (c) refers to forcing the avatar to visit certain locations (including *collecting items* from Forisek’s [11]). Figure 4 shows a single-use path in *Boulder Dash*.

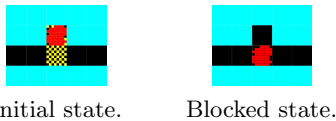


Figure 4: A single-use path in *Boulder Dash* [39]. Traversing (a) in either direction causes the middle of the path to become blocked, as seen in (b). Single-use paths *and* location traversal (collecting items) establish NP-hardness [39].

Metatheorem 1 does not immediately apply to *Dig Dug* or *Mr. Do!* since they do not have walls. We can apply it to *The Pit*, but we need to be careful with one detail. In the arcade game, the player must collect at least one of the gems; we allow the gem requirement to be arbitrarily large in the decision problem THEPIT.

**Corollary 1 (Metathm 1 [39])** THEPIT is NP-hard.

**Proof.** Boulders cannot be pushed in *The Pit*, so a single-use path can be constructed with a single boulder.



*The Pit* also has walls, and location traversal can be implemented in THEPIT with gems.  $\square$

We also mention an investigation of pulling block complexity by Ani et al. [3]. They prove that a wide variety of decision problems are PSPACE-complete. One exception is PULL?-1FG, which is shown to be NP-hard. The problem asks if an exit can be reached using Dig Dug physics and pullable objects. More specifically, its name can be parsed as follows: PULL? indicates that the player is not forced to pull blocks; 1 specifies that the player can only pull one block at a time; F denotes that walls are allowed; G states that Dig Dug gravity is used rather than a top-down perspective.

<sup>5</sup>The gadgets avoid *Boulder Dash*’s non-trivial falling physics: A boulder will suddenly fall to the side if it is on top of another boulder, and the cells to the side of both boulders are empty.

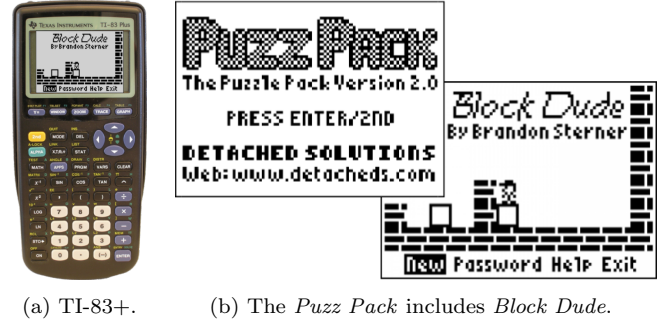


Figure 5: *Block Dude* was developed for Texas Instrument calculators as part of the *Puzzle Pack* suite of games.

## 1.4 Outline

We consider the computational complexity of box pushing from a side perspective and with *normal gravity* (i.e. the player and movable objects are subject to it) and the standard goals from both SOKOBAN and PUSH.

We also consider block lifting. Lifting is the only mechanism in *Block Dude* (1999), which is perhaps the closest (non-commercial) video game analogue to *Sokoban* from a side perspective; it was also the inspiration for this article. (A type of top-down lifting was considered in the BOX MOVER PROBLEM [27].)

Section 2 defines our decision problems. In Section 5, we prove that the problems are NP-hard. This is preceded by Sections 3–4, which define basic gadgets and give a simplified ‘rug’ reduction. Final remarks are in Section 6. Owing to the title of the game that inspired this article, the reader should expect to find *Big Lebowski* quotes and references along the way.

## 2 Block Dude and Related Decision Problems

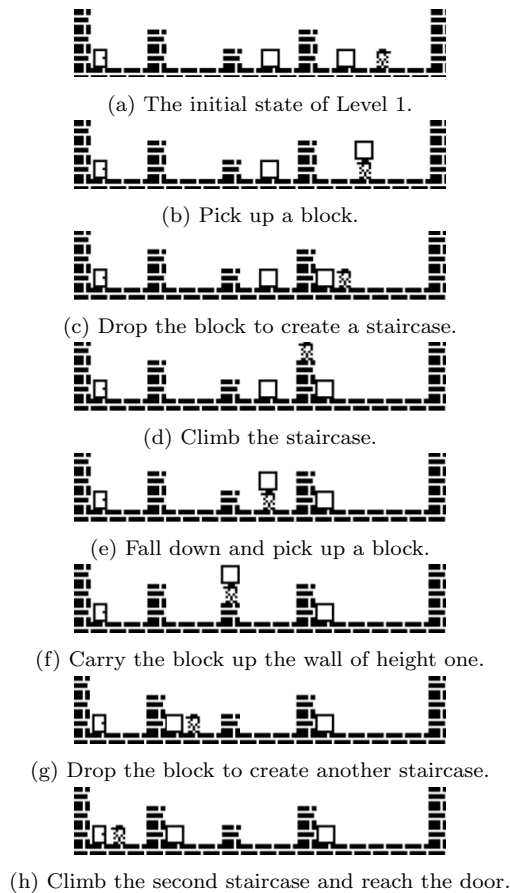
In this section, we describe the history and gameplay of *Block Dude*. Then we formulate a family of eight decision problems, one of which models the original game.

### 2.1 History of Block Dude

*Block Dude* is a TI-83/84 calculator game created by Brandon Sterner in 1999<sup>6</sup>. Detached Solutions included it in *PuzzlePack* [9] (see Figure 5), which has been downloaded over 100,000 times from ticalc.org [31]. The game’s popularity has never waned in the calculator community, winning the (fictitious) *Calculator Gaming Awards* 2002–2018 [22]. The game features 11 levels, and the speed-running world record is under 8 minutes [25]. A solution to the first level is in Figure 6.

Like *Sokoban*, the basic *Block Dude* game has been ported to a variety of systems by other developers, with

<sup>6</sup>Man, we’ve got some information, all right. Certain things have come to light. *Block Dude* wasn’t the first *Block Dude* game! *Block-Man 1* was released commercially in 1994 (see Table 1).

Figure 6: Solving Level 1 in *Block Dude*.

a sampling shown in Table 1. Readers who wish to try out the game are directed towards the browser-based port by Andrew Zich [42]. *Block Dude* was also used as a programming assignment in Harvard’s CS50 [17]. Sterner discussed *Block Dude* in a Reddit AMA [32].

## 2.2 Block Dude Gameplay

*Block Dude*’s objective is to move the avatar (i.e. the Dude) through the grid to the exit door. The Dude occupies one cell, as do the obstacles, which are immovable *bricks*, and movable *blocks*. The Dude moves left or right, and can climb onto obstacles one cell above his feet. They can also turn around in-place when there are obstacles on their left and right. The game has normal gravity, and the Dude safely falls from any height.

The Dude can only pick up a block that is directly in front of him, and can only do this when two cells are empty: the one above his head, and the one diagonally in front and above his head. If the cell in front of his feet is empty, then the Dude can drop a block into that position. Otherwise, if the cell in front and above the Dude is empty, then he can drop the block there. Any block that is dropped is subject to gravity, and it will continue falling until landing on a brick or block.

Year	Title	Developer	Platform(s)	Screenshot
1994	Block-Man 1	Soleau Software	DOS	
1995	Block-Man 2	Soleau Software	DOS	
1999	Block Dude	Brandon Sterner	TI-83 / TI-84 Calculators	
2004	BlockDude	Klas Kroon Chris Kotiesen	Adobe Flash	
2006 2010	Blockdude 1.3 Blockdude 1.4	Willems Davy	GP2X Dingoo A320	
2008	Block Dude	SG57	Sony PSP	
2008	Block Dude	Emmanuel Vincent	GNU/Linux	
2009	Block Dude	Andrew Zich Pete Zich	Apple iPhone JavaScript	
2010	Block Dude Evolved	Billy Connolly	Apple iPhone iPod Touch	
2011	Block Dude X	Amelia “Hinchy” Hinchliffe	Mac OS Windows	
2017	BlockDude	Brick Buddy Hazardous Dude	Android Steam	
2018	Block Dude	Mitch Kendall	Nintendo NES	
2019	Box Dude	Parker Phair	iOS	
2019	BlockDude	Dmitry Krapivin	ZX Spectrum 48K	
2021	BlockBot	Mark Horan	HTML5	

Table 1: Selected games related to *Block Dude*.

The Dude can hold and carry one block at a time, and he does so on top of his head. Thus, a height of at least two is required for the Dude to carry a block through a passageway. If the Dude attempts to carry a block past a brick that is directly above his head, then the brick will push the block off his head; the Dude will move forward while the block will fall behind him.

Figure 7 illustrates several of the finer points mentioned above, where grey squares are bricks.

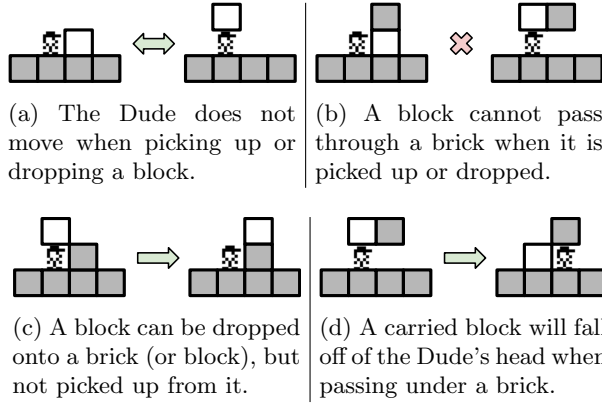




Figure 7: *Block Dude* physics. Arrow directions indicate valid moves between states. The Dude can also turn around in tight spaces   $\leftrightarrow$  .


### 2.3 BLOCKDUDE Family

*I'm the Dude. So that's what you call me. You know, that or, His Dudeness, or Duder, or El Duderino if you're not into the whole brevity thing*

— Jeffrey Lebowski, *The Big Lebowski*

Rather than studying Block Dude in isolation, we consider a family of decision problems. The problems use the same perspective and movements as Block Dude. Thus, the Dude can walk left or right, climb on top of a single occupied cell that is immediately in front of them, and fall down from any height.

A level is completed in one of two ways:

- (A) The level is completed when the Dude reaches the exit goal (denoted with ). This is the *exit goal*.
- (B) The level is completed when the Dude moves the  $k$  moveable objects onto the  $k$  target cells denoted by \*. This is the *targets goal*, and it models the completion condition of Sokoban.

There are four types of moveable objects.

- (1) A *block* can be picked up, but not pushed. These are identical to the blocks in Block Dude.
- (2) A *box* can be pushed, but not picked up. These are similar to Sokoban boxes, but they fall like blocks.
- (3) A *blox* can be picked up and pushed. Thus, a blox behaves like a block and a box.
- (4) A *crate* is a block, box, or blox. In other words, a crate is the generic term for a moveable object.

We form our decision problems by combining the two different goals with four types of moveable objects. We name each problem using the type of moveable object followed by DUDE/DUDERINO for exit/targets goals. For example, the original *Block Dude* game is type 1A, meaning that it has an exit goal, and all of the movable

objects are blocks, and its full name is BLOCKDUDE. The decision problem names are summarized in Table 2.

To avoid repetition, we will reuse our arguments and figures as much as possible. For example, our levels will contain both an exit and targets; the former is ignored in instances of type B, and the latter are ignored in instances of type A. For convenience, we also use different colors for the various crates and locations. These colors are purely cosmetic, and are used as hints to more easily understand how to complete the targets goal.

### 2.4 Membership in PSPACE

We complete this section by noting that all of our decision problems can be solved in polynomial-space.

**Theorem 2** *The problems in Table 2 are in PSPACE.*

**Proof.** Following the standard technique, we prove that the decision problems are in NPSpace. Consider a CRATEDUDE or CRATEDUDERINO level on an  $r$ -by- $c$  grid. The current state of a level can be encoded as an  $r$ -by- $c$  grid, where each entry is one of five possibilities: blank, box, block, blox, or Dude. (The remainder of the level — bricks, exit, target — is static.) Therefore, the state of the level can be encoded in  $3 \cdot r \cdot c$  bits, and so the level can have at most  $2^{3 \cdot r \cdot c}$  states. Therefore, if a level is solvable, then we can find a solution by non-deterministically making at most  $2^{3 \cdot r \cdot c}$  moves. Storing a move counter that ranges from 0 to  $2^{3 \cdot r \cdot c} - 1$  requires  $\log_2(2^{3 \cdot r \cdot c}) = 3 \cdot r \cdot c$  bits, which is polynomial in the size of the input. Therefore, the CRATEDUDE and CRATEDUDERINO are in NPSpace, and hence PSPACE by Savitch's Theorem [29]. This implies that the other six problems are also in PSPACE.  $\square$

### 3 Basic Gadgets and Approach

In this section, we discuss the general approach used by our reductions, and present our variable and clause gadgets. Each literal instance (i.e. each copy of  $x_i$  or  $\bar{x}_i$ ) is represented by one crate in Section 4, and by a sequence of surplus crates in Section 5. We refer to these crates as *literal instance crates*, or simply *literal crates*.

Our gadgets include elevated bricks that are one cell below a ceiling brick. These *guards* restrict crates from being pushed or carried out of the gadget.

#### 3.1 Variable Gadgets

Our gadget for variable  $x_i$  appears in Figure 8a, and it is drawn to match  $i = 1$  from (1). From the gadget's start position, the Dude can drop down to the left or right. Left corresponds to setting  $x_i$  to false, and the Dude can reach every crate associated with a negative literal  $\bar{x}_i$ . Likewise, right corresponds to setting  $x_i$  to true, and the Dude can reach every crate associated with a

	Boxes (Push-Only)	Blocks (Lift-Only)	Bloxes (Push and Lift)	Crates (Push and/or Lift)
Exit Goal	BOXDUDE	BLOCKDUDE	BLOXDUDE	CRATEDUDE
Targets Goal	BOXDUDERINO	BLOCKDUDERINO	BLOXDUDERINO	CRATEDUDERINO

Table 2: The Block Dude family of decision problems. See Section 6.1 for a note on DUDE vs PUSH notation.

positive literal  $x_i$ . In the full reduction, the Dude can “activate” each reachable crate by sending it downward into a clause gadget. The Dude cannot build a staircase back up to the start position of a variable gadget, so they must eventually return to the center and drop down to the start position of the  $x_{i+1}$  gadget.

**Observation 1** *In the  $x_i$  variable gadget, the Dude can reach the positive literal crates  $x_i$ , or the negative literal crates  $\bar{x}_i$ , but not both, before exiting. This corresponds to setting  $x_i = \text{true}$ , or  $x_i = \text{false}$ , respectively.*

Actually sending crates downward will be a challenge. Section 4 uses a “cheat” to simplify the task, and Section 5 provides the actual implementation.

### 3.2 Clause Gadgets

Our clause gadget appears in Figure 8b. The basic idea is that the gadget is traversable if at least one of its literal crates has been sent downward into it. The flag illustrates where the next clause gadget begins.

**Observation 2** *In the clause gadget for  $(\ell_1 \vee \ell_2 \vee \ell_3)$ , the Dude can reach the exit, if and only if, at least one of the literal crates for  $\ell_1, \ell_2, \ell_3$  has been sent downward into the gadget.*

#### Nihilism versus Dudeism

Is a Boolean value always true or false? Careful consideration of our reduction will reveal that the Dude is not obliged to activate a reachable crate within a variable gadget. This corresponds to believing nothing about the crate’s literal instance: It is neither true nor false. Without loss of generality, we can assume that the Dude avoids this type of nihilism during gameplay, but in practice, the Dude is most certainly a lazy man, and liable to go with the flow.

## 4 NP-Hardness with Rugs



*That rug really tied the room together.*

— Walter in *The Big Lebowski*

In this section, we provide a single reduction that shows that all eight of our problems are NP-hard. However, there is a catch. We assume that the puzzles can

use additional an gameplay element: a *rug*. Rugs are defined in Section 4.1, but in brief, they are similar to a trapdoor in that they allow crates to pass through without changing where the Dude can move.

Our reduction is from MONOTONE 3SAT, which is a restriction of 3SAT to monotone clauses. A *monotone clause* is a *positive clause* with three positive literals, or a *negative clause* with three negative literals. In other words, every clause has the form  $(x_i \vee x_j \vee x_k)$  or  $(\bar{x}_i \vee \bar{x}_j \vee \bar{x}_k)$ . In particular, we illustrate the reduction using the following instance with four clauses,

$$\phi = (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1) \wedge (\bar{x}_5 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (x_1 \vee x_2 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4), \quad (1)$$

whose colors and indices are matched in Figure 10. The source problem is NP-complete by Schaefer’s dichotomy theorem [30], and the monotone property will allow us to organize all of the clause gadgets along a single row.

### 4.1 Rugs

A rug can be placed in any otherwise empty cell of the grid. Rugs are not strong enough to hold up a crate. Therefore, any crate that is pushed on top of a rug, or dropped onto a rug, will fall through it. Similarly, if a crate falls onto a rug from above, then it will continue falling through the rug. However, the Dude never falls through a rug, even when he is carrying a crate. In other words, the Dude is never let down by a rug<sup>7</sup>.

### 4.2 Literal Gadgets

In this reduction, we create a single crate for every instance of a literal. Thus, if the MONOTONE 3SAT instance has  $k$  clauses, then the corresponding level will have  $3k$  crates. As mentioned in Section 3.1, the Dude needs to be able to send these crates down to the respective clause gadgets. Rugs allow us to implement these *literal gadgets* very easily, as shown in Figure 9.

**Observation 3** *In the literal gadget, the Dude can send the literal instance,  $x_i$  or  $\bar{x}_i$ , down to its clause gadget.*

### 4.3 Rug Reduction

We can now present a single reduction from MONOTONE 3SAT. The reduction is illustrated in Figure 10. In

<sup>7</sup>This is true even if the rug has been micturated upon.

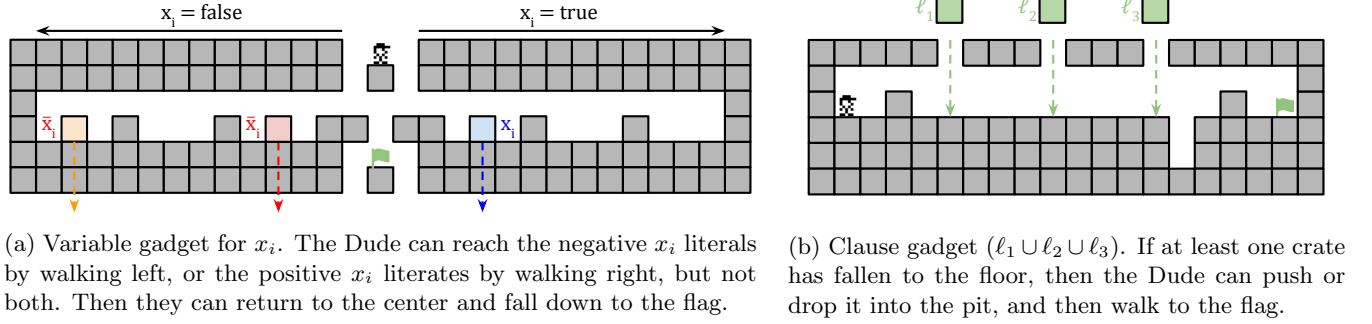


Figure 8: A simplified view of our basic gadgets. In both cases, it is not possible to move a crate out of the gadget.

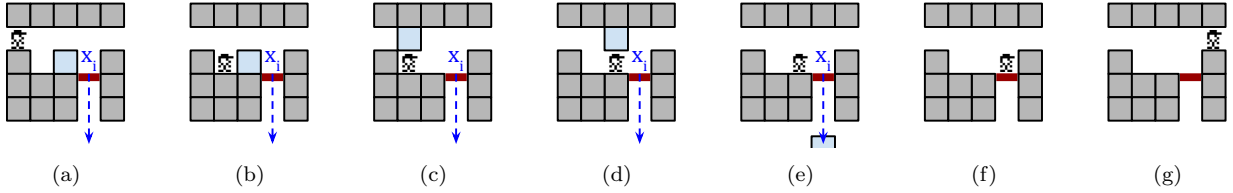


Figure 9: Activating a positive literal crate by sending the crate downward through a rug. If the crate is a block, then all steps (a)–(g) are used. Otherwise, if it is a box or blox, then it is pushed from (b) to (e), and steps (c) and (d) are omitted.

this image, and others, we allow some of the gadgets to overlap, in order to save space. In particular, literal instance gadgets in the same clause share the guards.

**Theorem 3** *In the rug reduction for MONOTONE 3SAT instance  $\phi$ , the Dude can reach the flag, or place the crates on the targets, if and only if,  $\phi$  has a satisfying assignment. This is true regardless of the type of crate that is used for each literal instance.*

**Proof.** Suppose that  $\phi$  is satisfiable and consider a specific satisfying assignment. By following this satisfying assignment when navigating the variable gadgets, the Dude is ensured that each clause gadget will have at least one crate in it. Therefore, they can traverse the clause gadgets and reach the exit. To accomplish the targets goal, they continue up the *target goal staircase* on the right, and then navigate the variable gadgets using the complement of the satisfying assignment.

For the other direction, suppose that  $\phi$  is not satisfiable. When the Dude reaches the clause gadgets, there will be at least one that does not have a crate in it. Hence, the Dude cannot traverse all of the clause gadgets, and so they cannot accomplish either goal.  $\square$

## 5 NP-Hardness without Rugs

Now we establish the NP-hardness of the decision problems in their original form, without rugs. To do this, we need to simulate the rugs. We do this with two new gadgets, which are introduced in Section 5.1. Then we show how to implement these two gadgets with boxes

in Section 5.2, and with blocks in Section 5.3. The former result also establishes hardness for crates, while the latter construction also works for blox.

### 5.1 Drop-Down and Fall-Through Gadgets

Rugs are used in two different ways in the rug reduction. The rugs that appear next to a crate allow for the initial dropping down of a crate, while the remaining rugs allow for crates to continue passing through them. We mirror this with the following two types of gadgets.

1. *Drop-down gadget.* In this gadget, the Dude is able to send one crate downward. The Dude can traverse the gadget left-to-right then right-to-left (or vice versa) and cannot go downward without getting stuck.
2. *Fall-through gadget.* In this gadget, the Dude can send one crate downward, so long as one crate has previously been sent downward into it. The Dude can traverse the gadget left-to-right then right-to-left (or vice versa) and cannot go downward without getting stuck.

As in the rug reduction, these gadgets allow crates, which represent literal instances, to be sent downward. However, in the rug reduction, an individual crate is sent straight downward from the variable gadget to the clause gadget. With these gadgets, the crate that is sent down changes after each successive gadget. In other words, these gadgets work together to send down a surplus of one crate, rather than an individual crate. Furthermore, the surplus crate that is sent down shifts two columns horizontally after each fall-through gadget.

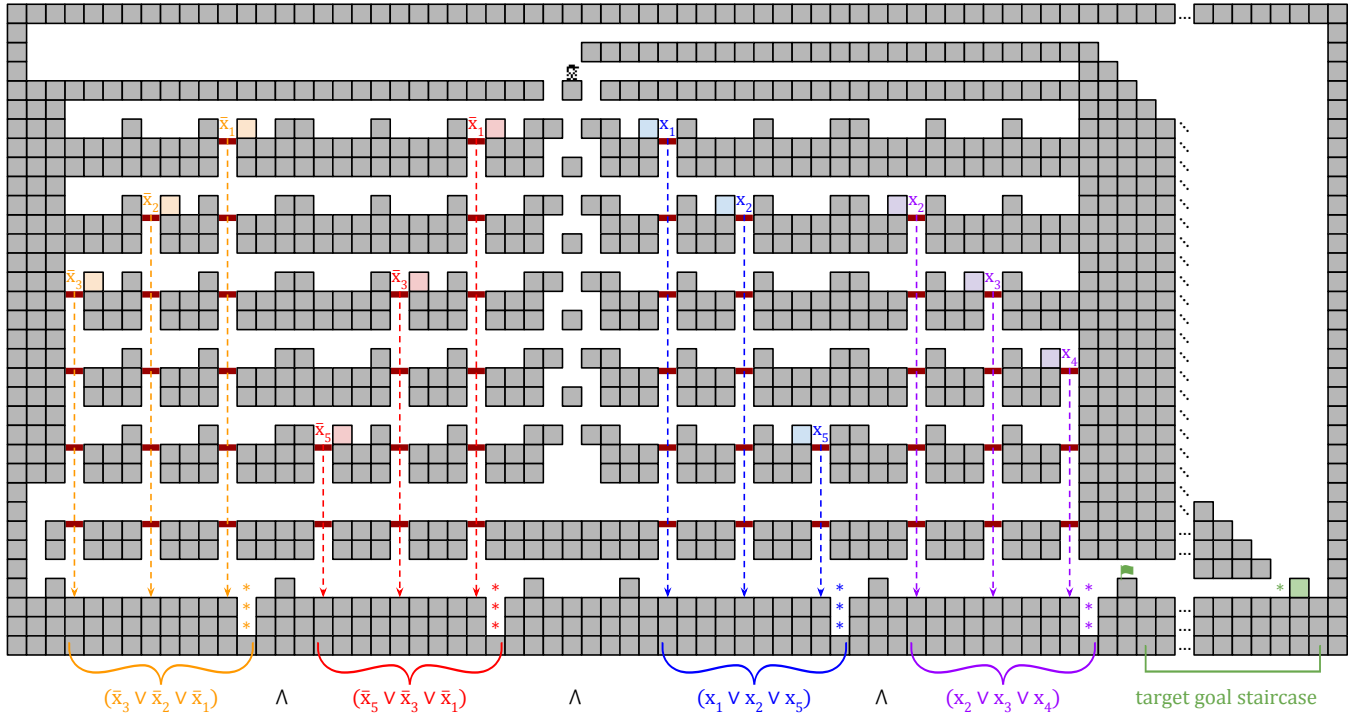


Figure 10: The rug reduction for  $\phi = (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1) \wedge (\bar{x}_5 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (x_1 \vee x_2 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4)$  in (1). To complete the exit goal, the Dude drops into the five variable gadgets on the left, right, left, left, and left, respectively. The resulting assignment  $x_1 = x_3 = x_4 = x_5 = \text{FALSE}$  and  $x_2 = \text{TRUE}$  makes the clause gadgets traversable<sup>8</sup>. To complete the targets goal, the Dude continues past the exit and walks up the target goal staircase. Then they drop into the variable gadget targets on the opposite sides — right, left, right, right, right — and fill the clause gadget targets.

To implement these gadgets we use empty vertical stacks of cells called *pits*, which must be partially filled in order for the Dude to traverse them. More specifically, a pit of depth  $d$  must be filled with  $d - 1$  crates before it can be traversed. We say that a gadget has a *surplus*, if a crate has been dropped into its pit before the Dude has entered the gadget. When the Dude enters a gadget, he will be able to send a crate downward, if and only if, the gadget has a surplus. In other words, the Dude can move a surplus down to the next gadget. The depth of each pit is chosen to prevent the Dude from trying to take shortcuts in the level. More specifically, if the Dude drops down to the gadget below, then the depth of the pit will ensure that he gets stuck.

As a final note, we mention that these gadgets require more space than the corresponding gadgets in the rug reduction. As a result, the reader should observe that the layout in Figure 10 can be widened and lengthened without affecting its functionality.

## 5.2 Box Gadgets

Figure 11 contains our implementation of the drop-down gadget and fall-through gadget using boxes.

<sup>8</sup>A lazy Dude can drop left then right, and then go directly to the clause gadgets since  $x_1 = \text{FALSE}$  and  $x_2 = \text{TRUE}$  satisfies (1).

**Lemma 4** *Figure 11 implements a drop-down gadget and fall-through gadget using boxes.*

**Proof.** For ease of reading, the explanation of how the gadget works is given in the caption of Figure 11.  $\square$

The next corollary follows from Lemma 4 and Theorem 3, and the fact that instances of BOXDUDE and BOXDUDERINO are also instances of CRATEDUDE and CRATEDUDERINO, respectively.

**Corollary 5** *BOXDUDE, BOXDUDERINO, CRATEDUDE, and CRATEDUDERINO are NP-hard.*

## 5.3 Block and Blox Gadgets

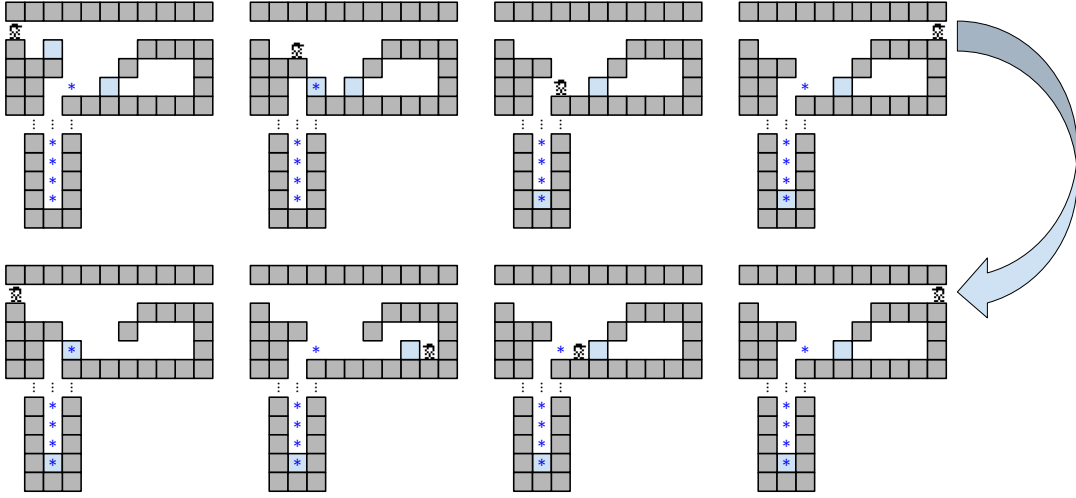
Figure 12 contains our implementation of the drop-down gadget and fall-through gadget using blocks. Inspection shows that it also suffices for blox.

**Lemma 6** *Figure 12 implements a drop-down gadget and fall-through gadget using blocks or blox.*

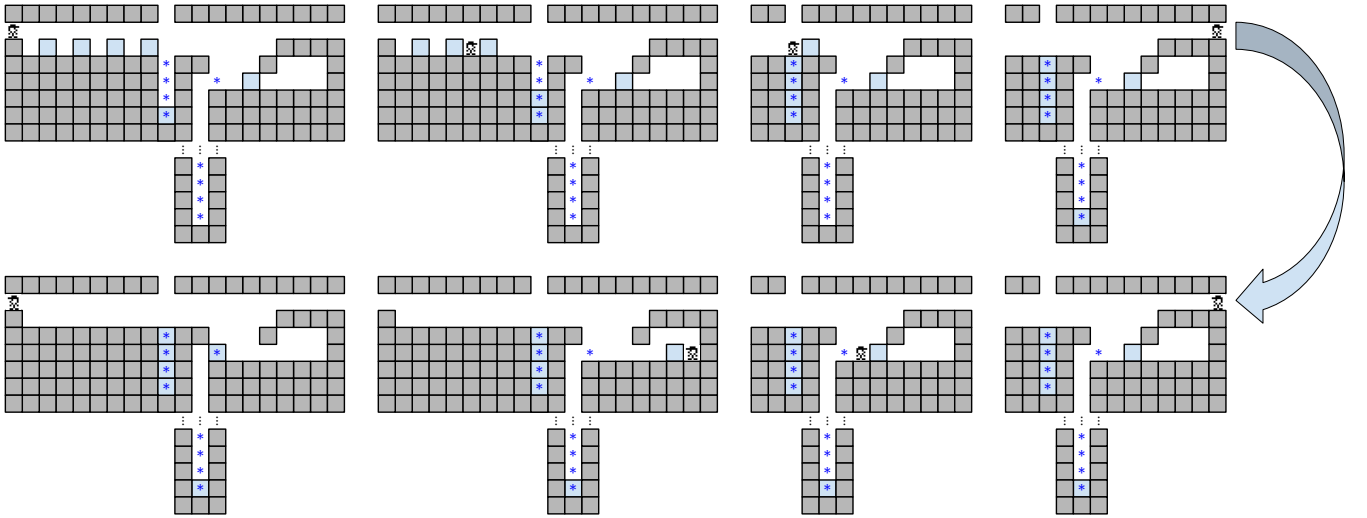
**Proof.** For ease of reading, the explanation of how the gadget works is given in the caption of Figure 12.  $\square$

Theorem 3 and Lemma 6 provide the final corollary.

**Corollary 7** *BLOCKDUDE, BLOCKDUDERINO, BLOXDUDE and BLOXDUDERINO are NP-hard.*

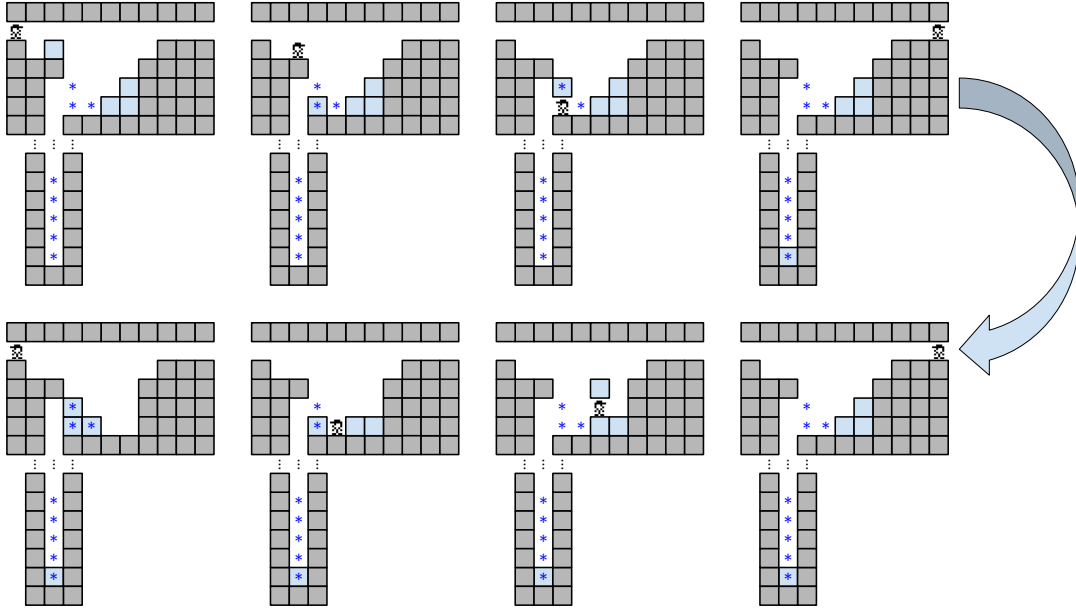


(a) Drop-down gadget with boxes. The Dude enters the gadget on the left (top-left) and pushes the top box once to the right and once to the left, so that it falls downward into the pit of the gadget below. The Dude then exits on the right (top-right). Re-entering (bottom-right), they use the hollow area to push the remaining box into the target positions (\*), and then they can exit on the left (bottom-left).

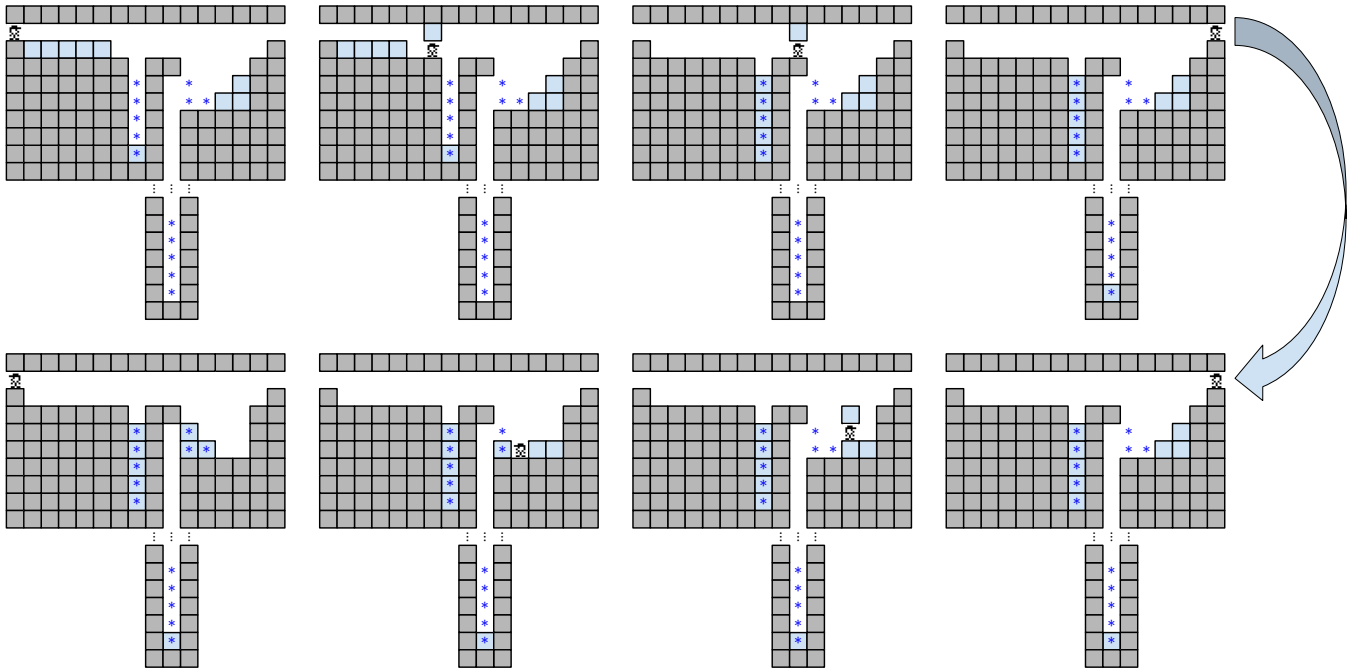


(b) Fall-through gadget with boxes. The Dude enters the gadget from the left, and to proceed to the right, they must push boxes into the pit, starting with the rightmost of the four boxes. If a surplus box is in the pit when they enter the gadget (as illustrated), then there will be an additional box that can be pushed down as a surplus box into the pit of the gadget below. The Dude then exits the gadget (top-right). Reentering (bottom-right), the Dude proceeds to the exit on the left in the same way as in the drop-down gadget.

Figure 11: Box gadgets. The drop-down (top) and fall-through (bottom) gadgets for BOXDUDE and BOXDUDERINO. In both cases, the images illustrate the gadget for a positive literal instance, and all crates are boxes. The initial states are shown in the top-left corner, and the remaining images illustrate how the Dude can traverse the gadget left-to-right and send a surplus crate downward, then traverse it from right-to-left. The Dude cannot take a shortcut in either gadget since the pits have depth four, and at most two boxes can be sent downward from the gadget above. Note: To conserve space, the leftmost 7 columns are omitted from the rightmost four images in the fall-through.



(a) Drop-down gadget with blocks. The Dude enters on the left (top-left) and picks up and drops the top block twice, so that it falls down into the pit of the gadget below. Then they exit on the right (top-right). Re-entering (bottom-right), they pick-up and drop the three blocks to form a staircase on the target positions (\*), and exit on the left (bottom-left).



(b) Fall-through gadget with blocks. The Dude enters on the left, and to proceed they must pick up and drop blocks into the pit. If a surplus block is in the pit when they enter (as illustrated), then there will be an additional block that can be picked-up and dropped as a surplus block into the gadget below. The Dude then exits the gadget (top-right). Re-entering (bottom-right), they proceed to the exit on the left in the same way as in the drop-down gadget.

Figure 12: Block gadgets. The drop-down (top) and fall-through (bottom) gadgets for BLOCKDUDE and BLOCKDUDERINO. In both cases, the images illustrate the gadget for a positive literal instance, and all crates are blocks. The initial states are shown in the top-left corner, and the remaining images illustrate how the Dude can traverse the gadget left-to-right and send a surplus crate downward, then traverse right-to-left. The Dude cannot take a shortcut in either gadget since the pits have depth six, and at most four blocks can be sent down from the gadget above.

## 6 Final Remarks

We have shown that 8 decision problems are NP-hard, including one that models *Block Dude*. It remains open which are in NP (and hence NP-complete), and which are PSPACE-hard (and hence PSPACE-complete).

- CRATEDUDE and CRATEDUDERINO are the best candidates for being PSPACE-complete. The recent motion planning framework in Lynch’s PhD thesis [24] (see also [7, 6]) could provide a pathway to this result, although normal gravity seems to make that route more challenging. A more modest goal is to establish the existence of levels that require exponentially many moves to solve. This would show that the most natural certificate (i.e. the sequence of moves) is insufficient for establishing NP membership (e.g. see [16]).
- BOXDUDE and BOXDUDERINO seem to be the best candidates for NP-completeness. This is because boxes cannot move up, so their y-position is a non-renewable resource, which is a hallmark of NP.

### 6.1 Nomenclature: DUDE vs PUSH

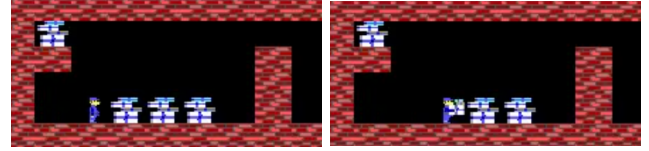
A significant drawback to our exposition is that the DUDE-based decision problem names do not fit into the standard Push-based nomenclature. One reason for this departure was to allow multiple types of movable objects (e.g. CRATEDUDE includes boxes, blocks, and blox) at once; this is not easily supported by PUSH notation. However, we are pacifists, not conscientious objectors, so community members should feel free to introduce terminology like LIFT-1NG for BLOCKDUDE, where 1NG denotes lifting one block at a time with normal gravity.

### 6.2 Open Problems

Besides the obvious open problems listed earlier, future research can consider other types of movable objects in the presence of normal gravity from a side perspective.

- Objects that slide horizontally when pushed. This would be a PUSH-PUSH-style problem, but with a different physical model. We mention that ice blocks are used in the side-view game in the NES game *Fire ‘n Ice* (1993) with normal gravity.
- Objects that can be pulled. See Section 1.3 for a discussion of PULL?-1FG with Dig Dug gravity.
- Objects that can be moved as if they were *pocketed*. When the player picks up a box in *Loader Larry* (1995), they carry it in the same cell as their avatar, as if it were placed in their pocket (see Figure 13).

We also note that the Dude never needs to fall more than three cells at a time in our reduction. Limiting



(a) Before pocketing the object. (b) After pocketing the object.

Figure 13: Pocket carrying from *Larry Loader* (1985).

the Dude’s ability to fall only from a height of one may be an interesting problem to investigate, although that would put the Dude in the running for “the weakest video game character” worldwide, which would surely rattle those *Spelunker* (1983) fans and their used game bins [2] — ah, look at me. I’m ramblin’ again.

### 6.3 Acknowledgements

We thank the anonymous referees for their helpful (and humorous) comments, which were used to significantly improve the paper. Two referees requested *more* Big Lebowski references; the authors were happy to abide.

## References

- [1] Google Scholar. [scholar.google.com/scholar?q=sokoban](https://scholar.google.com/scholar?q=sokoban), 2021.
- [2] 1983parrothead. Famicom / NES Spelunker. [knowyourmeme.com/memes/fc-nes-spelunker](https://knowyourmeme.com/memes/fc-nes-spelunker), 2010.
- [3] J. Ani, S. Asif, E. D. Demaine, Y. Diomidov, D. H. Hendrickson, J. Lynch, S. Scheffler, and A. Suhl. Pspace-completeness of pulling blocks to reach a goal. *J. Inf. Process.*, 28:929–941, 2020.
- [4] D. Ashlock and J. Schonfeld. Evolution for automatic assessment of the difficulty of Sokoban boards. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.
- [5] J. Culberson. Sokoban is PSPACE-complete. In *Proceedings of the 1st International Conference on Fun with Algorithms*, pages 65–76, 1998.
- [6] E. D. Demaine, I. Grosz, J. Lynch, and M. Rudoy. Computational complexity of motion planning of a robot through simple gadgets. In H. Ito, S. Leonardi, L. Pagli, and G. Prencipe, editors, *9th International Conference on Fun with Algorithms, FUN 2018, June 13-15, 2018, La Maddalena, Italy*, volume 100 of *LIPICs*, pages 18:1–18:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- [7] E. D. Demaine, D. H. Hendrickson, and J. Lynch. Toward a general complexity theory of motion planning: Characterizing which gadgets make games hard. In T. Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCs 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPICs*, pages 62:1–62:42. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

- [8] E. D. Demaine and M. Hoffmann. Pushing blocks is np-complete for noncrossing solution paths. In *Proc. 13th Canad. Conf. Comput. Geom.*, pages 65–68, 2001.
- [9] Detached Solutions. Puzzpack homepage. [www.detachedsolutions.com/puzzpack](http://www.detachedsolutions.com/puzzpack), 2001.
- [10] D. Dor and U. Zwick. Sokoban and other motion planning problems. *Computational Geometry*, 13(4):215 – 228, 1999.
- [11] M. Forišek. Computational complexity of two-dimensional platform games. In *FUN*, 2010.
- [12] E. Friedman. Pushing blocks in gravity is NP-hard.
- [13] E. Friedman. Problem of the month (March 2000). [web.archive.org/web/20190218043712/www2.stetson.edu/~efriedma/mathmagic/0300.html](http://web.archive.org/web/20190218043712/www2.stetson.edu/~efriedma/mathmagic/0300.html), 2000.
- [14] M. Fryers and M. T. Greene. Sokoban, 1995.
- [15] A. Greenblatt, O. Hernandez, R. A. Hearn, Y. Hou, H. Ito, M. J. Kang, A. Williams, and A. Winslo. Turning around and around: Motion planning through thick and thin turnstiles. In *CCCG*, 2021.
- [16] A. Greenblatt, J. Kopinsky, B. North, M. Tyrrell, and A. Williams. Mazeam levels with exponentially long solutions. In *20th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JDCGG 2017)*, pages 109–110, 2017.
- [17] Harvard CS50. Problem: Blockdude. [docs.cs50.net/2016/ap/problems/blockdude/blockdude.html](https://docs.cs50.net/2016/ap/problems/blockdude/blockdude.html), 2016.
- [18] R. A. Hearn and E. D. Demaine. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.*, 343(1–2):72–96, Oct. 2005.
- [19] M. Hoffmann. Push-\* is NP-hard. In *CCCG*, 2000.
- [20] A. Junghanns and J. Schaeffer. Sokoban: Improving the search with relevance cuts. *Journal of Theoretical Computing Science*, 252:151–175, 1999.
- [21] A. Junghanns and J. Schaeffer. Sokoban: Enhancing general single-agent search methods using domain knowledge. *Artificial Intelligence*, 129:219–251, 2001.
- [22] J. Kaplowitz. ‘Block Dude’ sweeps calculator gaming awards sixteenth year straight. [thehardtimes.net/harddrive/block-dude-sweeps-calculator-gaming-awards-sixteenth-year-straight](https://thehardtimes.net/harddrive/block-dude-sweeps-calculator-gaming-awards-sixteenth-year-straight), 2018.
- [23] B. Kartal, N. Sohre, and S. J. Guy. Data driven sokoban puzzle generation with Monte Carlo tree search. In *Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2016.
- [24] J. R. Lynch. *A framework for proving the computational intractability of motion planning problems*. PhD thesis, MIT, 9 2020. [dspace.mit.edu/handle/1721.1/129205](https://dspace.mit.edu/handle/1721.1/129205).
- [25] NuclearHydrargyrum. Block Dude full game: 7 minutes, 43.640 seconds [WR]. [www.youtube.com/watch?v=-YHLzswiEks](https://www.youtube.com/watch?v=-YHLzswiEks), 2020.
- [26] A. G. Pereira, M. Ritt, and L. S. Buriol. Optimal sokoban solving using pattern databases with specific domain knowledge. *Artificial Intelligence*, 227:52 – 70, 2015.
- [27] V. Polishchuk. The box mover problem. In *CCCG*, 2004.
- [28] J. Potma. *An exponential construction for Sokoban*. Bachelor’s thesis, 2018.
- [29] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177 – 192, 1970.
- [30] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC ’78, page 216–226, New York, NY, USA, 1978. Association for Computing Machinery.
- [31] B. Sterner. Puzzpack v2.0. [www.ticalc.org/archives/files/fileinfo/102/10289.html](http://www.ticalc.org/archives/files/fileinfo/102/10289.html).
- [32] B. Sterner. I am Brandon Sterner, the creator of Block Dude ... from the PuzzPack for TI-83+. AMA. [www.reddit.com/r/IAmA/comments/13ozy1/i\\_am\\_brandon\\_sterner\\_the\\_creator\\_of\\_block\\_dude](https://www.reddit.com/r/IAmA/comments/13ozy1/i_am_brandon_sterner_the_creator_of_block_dude), 2012.
- [33] Students for a Democratic Society. Port Huron Statement. [www2.iath.virginia.edu/sixties/HTML\\_docs/Resources/Primary/Manifestos/SDS\\_Port\\_Huron.html](http://www2.iath.virginia.edu/sixties/HTML_docs/Resources/Primary/Manifestos/SDS_Port_Huron.html), 1962.
- [34] M. O. Suleman, F. Syed, T. Syed, S. Arfeen, S. I. Behlim, and B. Mirza. Generation of sokoban stages using recurrentneural networks. *International Journal of Advanced Computer Science and Applications*, 8, 2017.
- [35] J. Taylor, I. Parberry, and T. Parsons. Comparing player attention on procedurally generated vs. hand crafted Sokoban levels with an auditory stroop test. In *Foundations of Digital Games (FDG)*, 2015.
- [36] The IT Law Wiki. Tetris Holding v. Xio Interactive. [itlaw.wikia.org/wiki/Tetris\\_Holding\\_v.\\_Xio\\_Interactive](https://itlaw.wikia.org/wiki/Tetris_Holding_v._Xio_Interactive), 2012.
- [37] Thinking Rabbit. History of Sokoban. [sokoban.jp/history.html](http://sokoban.jp/history.html), 2021.
- [38] R. Uehara. 日の目を見なかった問題たち その2 [Problems that didn’t see the light of day Part 2]. [www.jaist.ac.jp/~uehara/etc/1a/99/index.html](http://www.jaist.ac.jp/~uehara/etc/1a/99/index.html), 1999.
- [39] G. Viglietta. Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54:595–621, 2014.
- [40] Wikipedia. (倉庫番) — Wikipedia, the free encyclopedia. [ja.wikipedia.org/w/index.php?title=%E5%80%89%E5%BA%AB%E7%95%AA&oldid=80549687](https://ja.wikipedia.org/w/index.php?title=%E5%80%89%E5%BA%AB%E7%95%AA&oldid=80549687), 2020.
- [41] Wikipedia. Port Huron Statement. [en.wikipedia.org/wiki/Port\\_Huron\\_Statement](https://en.wikipedia.org/wiki/Port_Huron_Statement), 2021.
- [42] A. Zich. Block Dude. [azich.org/blockdude](http://azich.org/blockdude).