Unstacking Slabs Safely in Megalit is NP-Hard

Conference Paper · July 2022

CITATIONS
0

READS
51

3 authors:

Kirby Gordon

1 PUBLICATION **0** CITATIONS

SEE PROFILE

Jacob Lezberg

Williams College

1 PUBLICATION **0** CITATIONS

SEE PROFILE

Aaron Williams

Williams College

98 PUBLICATIONS **1,100** CITATIONS

SEE PROFILE

Unstacking Slabs Safely in Megalit is NP-Hard

Kirby Gordon* Jacob Lezberg† Aaron Williams‡

Abstract

We consider the problem of safely unstacking rectangular boxes through the lens of *Megalit* (ASCII, 1991). In this Game Boy game, the player is confronted with a pile of 1-by-k and k-by-1 megaliths (slabs). The goal is to pull and push the slabs until they reach the floor, without falling more than one cell at a time. We prove that an associated problem is NP-hard. Along the way, we introduce the drop-ladders problem, and prove that the Game Gear game *Popils* (Tengen, 1991) is NP-hard.

1 Introduction

In the late-1990s, the quintessential box pushing game *Sokoban* (倉庫番) was shown to be NP-hard independently by Fryers and Greene [11], Dor and Zwick [8], and Uehara [21]. In this top-down game, the player controls an agent who must push m boxes onto m locations. Many papers have been written under the Push[Push]-1/k/*-[X] banner (e.g., [15, 7]), where the goal is to reach a location under various physical models. Pulling [2], pushing rows [13], and rotation [12] have been studied, and *Sokoban* was proven PSPACE-complete [6].

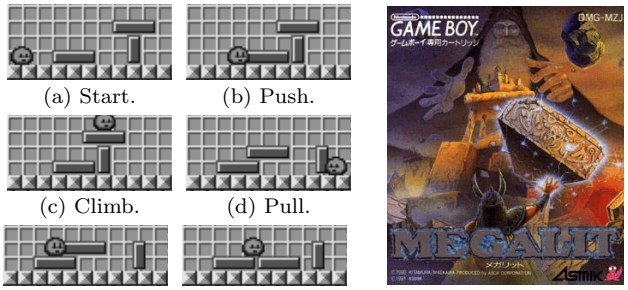
The 1990s also saw the establishment of NP-hardness for *Blocks World* [5]. In this grid-based problem, 1-by-1 blocks are stacked in columns and the goal is to unstack and restack the blocks to be in a particular state, and this led to ample subsequent research [14, 20, 16, 19].

In this paper, we consider a decision problem that has ingredients of both box pushing and *Blocks World*.

- *Side-view.* The two-dimensional world has gravity and a side perspective [18, 10] like *Block Dude* [4, 3].
- *Agent-based.* The player controls an agent who is vulnerable to falling objects, but not falling.
- *Unstacking.* Unlike *Sokoban* and *Blocks World*, the goal is to safely bring the boxes to the ground floor.
- *Fragility.* Unlike *Sokoban* and *Blocks World*, the blocks break when dropped more than one unit.
- *Push and Pull.* The agent can push and pull [17].

1.1 Inspiration and Outline

We were inspired by another 1990s artifact: *Megalit* (ASCII, 1991) for Nintendo's Game Boy (see Figure 1).



(a) Start. (b) Push. (c) Climb. (d) Pull. (e) Climb. (f) Push. (g) Box art (🇯🇵).

Figure 1: Solving a *Megalit* level (a)–(f).

- Section 2 formalizes our *Megalit* decision problem.
- Section 3 describes how any configuration of slabs can be flattened with the help of additional slabs.
- Section 4 introduces a toy problem called Ladders.
- Section 5 provides our reduction from Ladders.
- Section 6 proves that our reduction is correct.
- Section 7 concludes with final remarks.

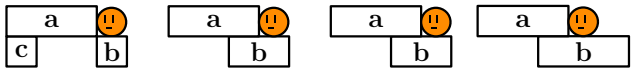
2 Megalit: Gameplay and Decision Problem

Megalit puzzles involve *slabs* and the following rules:

1. The playfield is a grid with bottom-left cell at $(0, 0)$.
2. The player may move left, move right, or jump. A jump is 3 units high. While jumping, the player may travel up to 2 units left or right.
3. If the player is horizontally adjacent to a slab and on stable footing (i.e. not in midair), they may grab the slab and push or pull it with them as they move.
   (a) If the player moves and falls off a slab while clutching a slab, then their grip is released.
   (b) Only the slab being grabbed is pushed or pulled; only one slab moves at a time.
   (c) Slabs will fall down due to gravity when unsupported. They cannot move upwards.
4. Slabs are horizontal 1-by-k or vertical k-by-1.
5. A slab $\mathbf{x}$ is *supported* by slab $\mathbf{y}$ when any cell of $\mathbf{x}$ is directly above a cell of $\mathbf{y}$ (see Figure 1d).
6. A level is failed if a slab falls ≥ 2 units or the player is crushed under a falling slab or trapped.
7. A cleared level has every slab touching the ground.

In the actual game, the slabs and player move horizontally in $\frac{1}{2}$-unit increments. In our decision problem we ignore this complication and use 1-unit moves. We also disallow pull and push moves that result in the player falling, which affects rule 3a as shown in Figure 2.

*Williams College, jacob.lezberg@gmail.com
†Williams College, kirbyjgordon@gmail.com
‡Williams College, aaron.williams@williams.edu



Can't push **a**. Can push **a**. Can't pull **a**. Can pull **a**.

Figure 2: In our Megalit decision problem the player must step onto a solid cell when performing a pull or push.

Our decision problem Megalit$(\mathcal{C}, p)$ asks if a configuration of supported slabs $\mathcal{C}$ can be flattened when the player starts at position p. Our main result is below.

Theorem 1 *The decision problem* Megalit *is NP-hard.*

We will specify a slab $\mathbf{s}$ by a 4-tuple, (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) are the bottom-left and top-right grid cells in $\mathbf{s}$, respectively. We also let cells($\mathbf{s}$) be the set of all cells in slab $\mathbf{s}$, and cells($\mathcal{C}$) be the union of cells($\mathbf{s}$) for $\mathbf{s} \in \mathcal{C}$. Finally, we normalize our co-ordinate system so that the minimum x and y coordinates of slabs in $\mathcal{C}$ are both zero. In other words, $(0, 0)$ is the bottom-left cell in the initial bounding box of the slabs.

3 Ramunto's Extraction Algorithm

In this section, we show that a configuration of slabs $\mathcal{C}$ can be safely flattened, so long as it is surrounded by other slabs. Our flattening process resembles pulling pizzas out of an oven using long wooden pizza trays known as *pizza peels*. Thus, we name our approach after our local pizza joint: *Ramunto's Brick House Pizza* [23].

3.1 Extractable Slabs

We define a slab $\mathbf{s} = (x_1, y_1, x_2, y_2)$ to be *extractable* if it has the following three properties:

X_1: There is no slab cell to the right of $\mathbf{s}$. That is, $\nexists \mathbf{t} \in \mathcal{C}$ with $(x, y) \in$ cells($\mathbf{t}$) and $x > x_2$ and $y_1 \leq y \leq y_2$.

X_2: There is no slab cell above $\mathbf{s}$. That is, $\nexists \mathbf{t} \in \mathcal{C}$ with $(x, y) \in$ cells($\mathbf{t}$) and $y > y_2$ and $x_1 \leq x \leq x_2$.

X_3: There is no gap to the right of $\mathbf{s}$ in the row below it. That is, if $\exists \mathbf{t} \in \mathcal{C}$ with $(x_2 + i, y_1 - 1) \in$ cells($\mathbf{t}$) for $i \geq 2$, then $\exists \mathbf{t}' \in \mathcal{C}$ with $(x_2+i-1, y_1-1) \in$ cells($\mathbf{t}'$).

For insight into this definition, the reader may skip ahead to Figures 4–5. Property X_3 ensures that $\mathbf{s}$ can be pulled along a series of slabs until it is transferred to a pizza peel; X_2 ensures that no slab $\mathbf{t}$ gets in the way when $\mathbf{s}$ is pulled to the right; X_1 ensures that $\mathbf{s}$ does not support any other slab $\mathbf{t}$ that could fall and break.

Lemma 2 *Every non-empty configuration of slabs $\mathcal{C}$ contains at least one extractable slab.*

Proof. We find an initial candidate slab, and then prove that it is extractable, or find a new candidate. During this process, we maintain a rectangular *search region* from (x_*, y_*) to (x^*, y^*), which are initially the bottom-left and top-right cells in cells($\mathcal{C}$), respectively.

If $\exists \mathbf{s} \in \mathcal{C}$ with $(x^*, y^*) \in$ cells($\mathbf{s}$), then $\mathbf{s}$ is extractable since all three properties are vacuously true. Otherwise, we let the first candidate be $\mathbf{s} = (x_1, y_1, x_2, y_2) \in \mathcal{C}$ that maximizes the minimum of $x^* - x_2$ and $y^* - y_2$, breaking ties by maximizing y_2. In other words, $\mathbf{s}$ is the first slab found by searching along down-right lines originating from the top row proceeding from right-to-left. After identifying the candidate, we reduce the search region.

- If $\mathbf{s}$ is horizontal, then we set $(x_*, y_*) = (x_1, y_1 + 1$ and $(x^*, y^*) = (x_2, y^*)$ (i.e., the cells above $\mathbf{s}$.
- If $\mathbf{s}$ is vertical, then we set $(x_*, y_*) = (x_1 + 1, y_1$ and $(x^*, y^*) = (x^*, y_2)$ (i.e., the cells right of $\mathbf{s}$).

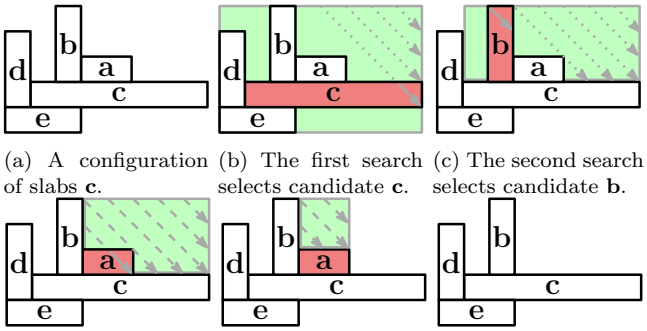
If a slab is found in the new search region, then it is the new candidate, and we repeat the process. Otherwise, $\mathbf{s}$ is our finalized candidate. Finally, we need to prove that $\mathbf{s}$ is extractable. We first assume that $\mathbf{s}$ is horizontal.

X_1: There is no slab $\mathbf{t}$ with a cell above $\mathbf{s}$ since the search that made $\mathbf{s}$ a candidate would have found $\mathbf{t}$.

X_2: There is no slab $\mathbf{t}$ with a cell to the right of $\mathbf{s}$ since nothing was found in the final search area.

X_3: There is no gap immediately below and right of $\mathbf{s}$ since the search that made $\mathbf{s}$ a candidate would find a slab $\mathbf{t}$ in the row immediately below $\mathbf{s}$.

When $\mathbf{s}$ is vertical the same points hold, but with the first two arguments interchanged. □



(a) A configuration of slabs **c**. (b) The first search selects candidate **c**. (c) The second search selects candidate **b**.

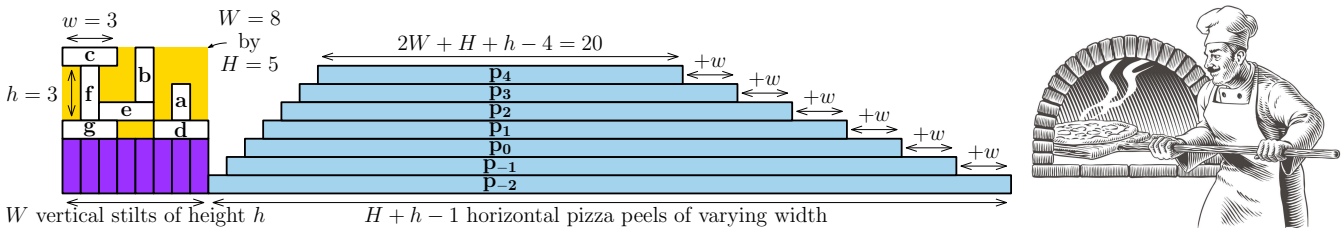
(d) The third search selects candidate **a**. (e) The fourth search confirms candidate **a**. (f) Ramunto's algorithm will extract **a**.

Figure 3: Lemma 2 considers successively smaller search regions (green) along arrows (gray) from the top-right. The confirmed candidate is extractable. The slabs are labeled by their extraction order during Ramunto's algorithm.

3.2 Pizza Oven Template

Let $\mathcal{C}$ be a configuration of slabs of width W and height H, with the maximum width and height of slabs being w and h, respectively. We surround this configuration slabs with additional slabs as follows.

- Vertical slabs of height h below $\mathcal{C}$.
- Horizontal slabs $\mathbf{p_i}$ for $-W + 1 \leq i \leq H - 1$. Each slab is $w + 1$ units wider than the one above, with the top slab $\mathbf{p_{H-1}}$ having width $2W + H + h - 4$. These slabs are organized into a left staircase immediately to the right of $\mathcal{C}$.



(a) The configuration $\mathcal{C}$ in the gold region, with purple vertical slabs and blue horizontal slabs (pizza peels) added, forms $\mathcal{C}' = \mathsf{pizza}(\mathcal{C})$. The peel lengths ensure support for each extraction.

(b) Slabs in $\mathcal{C}$ are extracted like pizzas, and (gently) dropped.

Figure 4: The demolition layout for our example configuration $\mathcal{C}$. The total width and height of $\mathcal{C}$ are $W = 8$ and $H = 5$, respectively, while the maximum width and height of a slab in $\mathcal{C}$ are $w = 3$ and $h = 3$, respectively.

We denote the resulting configuration $\mathcal{C}' = \mathsf{pizza}(\mathcal{C})$. This is illustrated in Figure 4, where the initial $\mathcal{C}$ form the slabs inside of the pizza oven.

3.3 Extraction Algorithm

This section's main result is illustrated in Figure 5.

Theorem 3 *If $\mathcal{C}$ is a configuration of slabs, then*

$$\mathsf{Megalit}(\mathcal{C}', p) = \mathit{yes},$$

where $\mathcal{C}' = \mathsf{pizza}(\mathcal{C})$ and p is the unique standing position immediately to the right of $\mathcal{C}$.

Proof. Algorithm 1 flattens $\mathcal{C}'$[1]. □

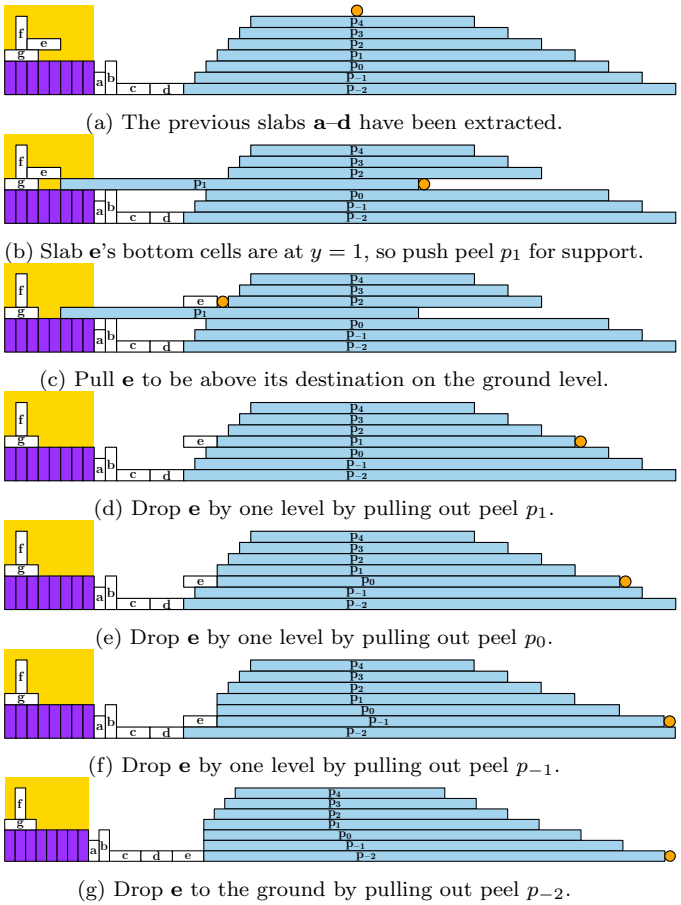
Algorithm 1 Ramunto's algorithm for flattening a configuration of slabs $\mathcal{C}' = \mathsf{pizza}(\mathcal{C})$ where p is the unique standing position immediately to the right of $\mathcal{C}$.

```
procedure RAMUNTOS(C')
  while |C| > 0 do
    C ← C − {s} for an extractable s = (x1, y1, x2, y2)
    push peel p_{x1} as close to s as possible
    pull s to be above the bottom peel p_{−W+1}
    pull peels p_{x1}, p_{x1−1}, ..., p_{−W+1} to lower s
    pull peels p_{−W+1}, p_{−W+2} ..., p_{H−1} to alignment
  end while
  pull peels p_{−W+1}, p_{−W+2}, ..., p_{H−1} to flatten them
end procedure
```

3.4 Flattening Goal to Target Location Goal

Theorem 3 helps us reduce the problem of flattening a configuration of slabs to reaching a particular location. For example, the player can complete Figure 4 so long as they can exit the initial gold region to the right, since Ramunto's algorithm will work regardless of where the slabs in $\mathcal{C}$ are located. We'll further refine this idea by surrounding the gold region with tall vertical walls, which ensures that the player can flatten the level if, and only if, they can reach the top-right cell in the region. In other words, we change the flattening goal into target location goal, and then a climbing goal. Climbing is further discussed in the following toy problem.

[1]Several temptingly simple greedy algorithms do not work.



(a) The previous slabs **a**–**d** have been extracted.

(b) Slab **e**'s bottom cells are at $y = 1$, so push peel p_1 for support.

(c) Pull **e** to be above its destination on the ground level.

(d) Drop **e** by one level by pulling out peel p_1.

(e) Drop **e** by one level by pulling out peel p_0.

(f) Drop **e** by one level by pulling out peel p_{-1}.

(g) Drop **e** to the ground by pulling out peel p_{-2}.

Figure 5: A snapshot of Ramunto's algorithm extracting slab **e** from configuration $\mathcal{C}$. After **e** reaches the ground in (g), the player realigns the peels as in (a) in order to extract **f**.

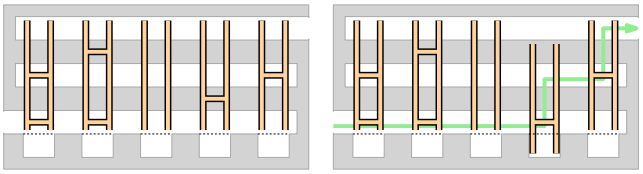
4 A Toy Problem

In this section, we introduce an NP-complete toy problem that resembles several well-established metatheorems [9] [22] [1]. The problem involves drop-ladders and we'll illustrate it by proving that *Popils* is NP-hard.

4.1 Drop-Ladders

A *drop-ladders* problem consists of ℓ ladders and $f + 1$ floors. Each *ladder* extends from the ground floor up

to the top floor and consists of some number of *rungs*. More specifically, there are $2f$ possible locations for a rung: on floor i and between floor i and $i+1$, for $1 \leq i \leq f$ (where $i = 1$ denotes the bottom floor). If a ladder contains a rung on floor i, then the player can *climb* from floor i to floor $i+1$ using this ladder; the rungs between floors cannot (immediately) be used to climb. In addition, when the player is on the bottom floor, they have the ability to lower any ladder by one unit. Lowering a ladder moves all of its rungs down one position, so a rung that was between floor i and $i+1$ moves to floor i, thus allowing the player to climb upward. Similarly, a rung on floor i moves between floor $i-1$ and floor i, eliminating its ability to help the player climb upward. The player starts on the bottom floor, and their goal is reach the top floor. Figure 6 provides an illustration.



(a) A Ladders problem. (b) A solution.

Figure 6: (a) A Ladders problem with $\ell = 5$ ladders and $f+1 = 3$ floors. It is a yes instance since the player can climb from the ground floor to the top floor using the rungs on the first ladder. Alternatively, they can drop the fourth ladder, then climb the fourth ladder and fifth ladder, as in (b). This level illustrates our reduction from ϕ in (1), with ladders for variables v_1–v_5 from left-to-right, and floors for $C_1 = (v_1 \vee v_2 \vee \neg v_4)$ and $C_2 = (v_1 \vee \neg v_2 \vee v_5)$ from bottom-to-top. The solution in (b) corresponds to the satisfying assignment $v_1 = v_2 = v_3 = v_5 = True$ and $v_4 = False$, with C_1 satisfied by $v_4 = False$ and C_2 satisfied by $v_5 = True$.

4.2 Ladders is NP-Complete

We now prove that Ladders is NP-complete.

Lemma 4 Ladders *is NP-complete.*

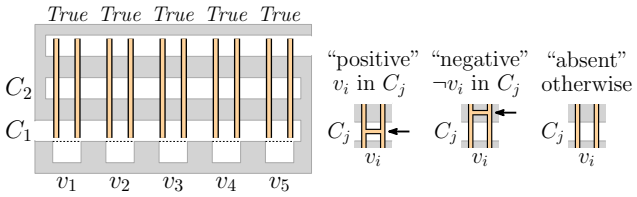
Proof. NP-hardness follows from a simple 3-SAT reduction, as illustrated in Figures 6–7 for the following:

$$\phi = C_1 \wedge C_2 = (v_1 \vee v_2 \vee \neg v_4) \wedge (v_1 \vee \neg v_2 \vee v_5). \quad (1)$$

In particular, lowering a ladder corresponds to changing the assignment of a variable from $True$ to $False$. Ladders is in NP since a suitable certificate specifies which ladders to drop and to climb. □

The following observation strengthens Lemma 4.

Observation 1 *It is possible to climb to floor i if and only if the variable assignment associated with the dropped ladders satisfies clauses $C_1, C_2, \ldots, C_i$.*



(a) Template for $n = 5$ variables and $m = 2$ clauses. (b) Rungs are added based on literal-clause membership.

Figure 7: Reducing 3-SAT to Ladders using (a) a template, with (b) added rungs. Specifically, if v_i is in clause C_j, then ladder i has a rung on floor j; if $\neg v_i$ is in clause C_j, then ladder i has a rung between floor j and floor $j+1$.

4.3 Application: Popils is NP-Hard

Another 1990s handheld puzzler is Tengen's *Magical Puzzle Popils* (JP 1991) / *Popils* (US 1992) for Sega's Game Gear. Each round follows the save-the-princess trope. Its mechanics and elements include the following:

- *Normal blocks.* Breakable by punching left or right, headbutting up, or kicking down.
- *Gold blocks.* Unbreakable and can be stood on.
- *Black blocks.* Empty and cannot be stood on.
- *Ladders.* The player can walk across or on top.

When the player breaks a normal block, all of the blocks stacked above in the same column will fall down one cell. The princess paces horizontally and never intentionally moves vertically; both characters are subject to gravity. The Popils decision problem generalizes the single-screen rounds to be arbitrarily large. See Figure 8.

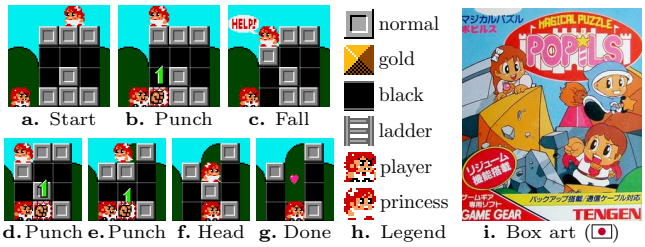
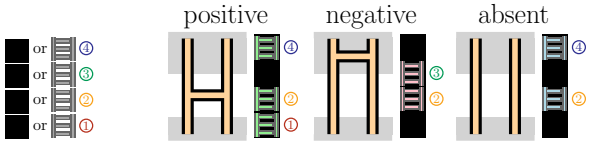


Figure 8: Round 1 in *Popils* with partial legend and box art.

We now illustrate Ladders with two reductions to Popils. The first uses the elements in Figure 8h and has simpler player movements, while the second omits gold blocks and has simpler *rung gadgets*. See Figures 9–10 and 11–12 for details, where ladders cells in the rung gadgets are tinted green, pink, or blue for readability.

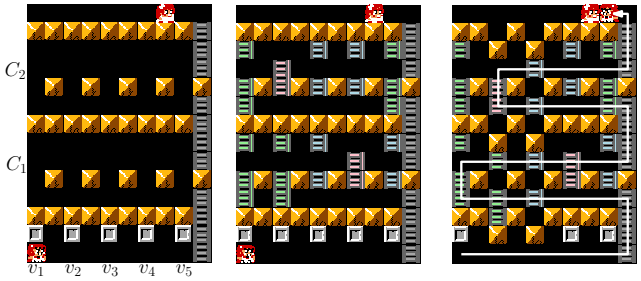
In both reductions, a drop-ladder occupies one column with a normal block at its base. The player starts in a *cellar* that appears below the first clause. The cellar is used to set the value of the variables. More specifically, the player can drop a drop-ladder by headbutting its normal block. Additional ladders on the right allow the player to exit a clause and enter the next clause, with the Princess pacing above the last clause. Each clause has a *lower-half* and an *upper-half*. To traverse a

clause, the player walks right-to-left on the lower-half, climbs a ladder associated with a satisfying literal, and then walks left-to-right on the upper-half.



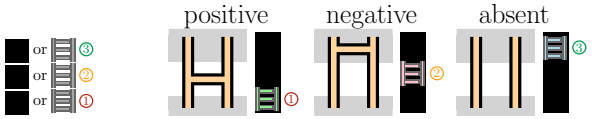
(a) Four cells. (b) Implementing the three gadgets.

Figure 9: Rung gadgets using four ladder or black square cells. To understand (b) note that ① and ② allow climbing undropped columns, while ② and ③ allow climbing dropped columns. The remaining ladder cells allow walking past (un)dropped columns on the lower and upper halves.



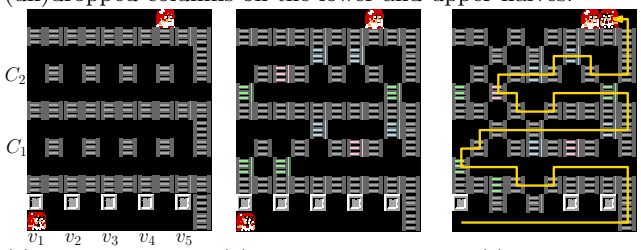
(a) Template. (b) Popils level for ϕ. (c) Solution.

Figure 10: Reducing Popils to Ladders for formula ϕ in (1) with $n = 5$ variables and $m = 2$ clauses. The provided solution involves headbutting and dropping the v_2 and v_3 columns, and then using $v_1 = T$ to satisfy $C_1 = (v_1 \vee v_2 \vee \neg v_4)$ and $v_2 = F$ to satisfy $C_2 = (v_1 \vee \neg v_2 \vee v_5)$.



(a) Three cells. (b) Implementing the three gadgets.

Figure 11: Rung gadgets using three ladder or black square cells. To understand (b) note that ① allows climbing undropped columns, while ② allows climbing dropped columns. The ladder in the absent case allows walking past (un)dropped columns on the lower and upper halves.



(a) $n = 5$ and $m = 2$. (b) Popils level for ϕ. (c) Solution.

Figure 12: Reducing Popils to Ladders for formula ϕ in (1). The provided solution follows the same format as Figure 10 but requires a more complicated path.

The second reduction gives the following theorem.

Theorem 5 Popils *is NP-hard with only ladders, normal blocks, and black blocks, even if the player can't fall.*

5 Reduction from 3-SAT

Now we describe our reduction from 3-SAT to Megalit. More specifically, we describe the configurations $\mathcal{C}$ that will be placed in the 'pizza oven' as per Section 3. We refer to these configurations as *haunted houses*. A sample is given in Figure 17. We define two new terms:

- A *cornerstone* is a lower corner cell of a slab. Vertical slabs have 1, while horizontal slabs have 2.
- A slab is *mobile* if the player can stand next to a cornerstone of the slab and push or pull it.

5.1 Primitive Gadgets: Tables and Chunks

A *table*[2] consists of a tabletop supported by a pair of legs, which are 2 or 3 units in height. This gadget appears in many variations, as seen in Figure 13. Only one leg is needed to support the tabletop, which allows the other leg to be pulled away. The player may pass underneath a table, by pulling and pushing the legs to reset them as needed, or over it (given that nothing on the tabletop obstructs passage). They may also re-purpose a leg as a climbing aid elsewhere in the level.

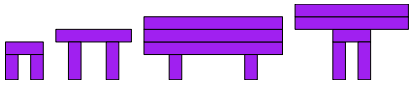


Figure 13: Tables come in many different sizes.

A *chunk* is a row of adjacent vertical slabs in any plural quantity. Notably, a chunk is extremely stable because only its two exterior slabs have exposed cornerstones. It follows that a chunk is completely immobile if the player cannot access either of these cornerstones.

5.2 Variable Towers

Variables are modeled by vertical structures called *variable towers* (see Figure 16). The base of each variable tower is a *pit of truth*, as shown in Figure 14a.

(a) True (default). (b) False (modified).

Figure 14: A pit of truth in its initial and modified state.

By pulling out the left leg of the underlying table, the player can jump up into the pit. To escape the pit, they must move the two horizontal slabs, as shown in Figure 14b. The net effect is that the tower is lowered by 2 units. This is analogous to dropping a ladder in the toy problem, or setting the associated variable to False. We note that a horizontally-mirrored version of the pit is used at the base of the last variable tower.

[2]Historically, this stone table structure is known as a *dolmen*.

Above a pit we stack literal gadgets for that variable. These gadgets come in three flavors (see Figure 15).

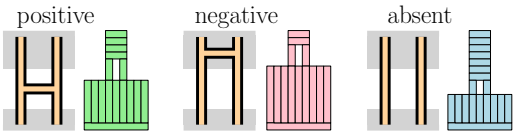


Figure 15: The three types of variable literal gadgets.

Each literal has a horizontal 9x1 slab at the base, then a chunk covering its full width, and finally a small table with multiple tabletop layers. Each gadget is 14 units tall, but the relative placement (and height) of the legs change based on the type of literal. As in our toy problem, the positive literals will be useful for climbing, unless the associated variable is negated, which causes its negative literals to become useful.

5.3 Scaffolding

Scaffolding towers between the variable towers enable climbing. A scaffolding tower is supported by a pair of *support gadgets* (yellow), which consist of a table with a tall vertical slab on top. The base of a scaffolding tower also includes a *pyramid gadget* (yellow), which allows the player to climb up to the rest of the scaffolding. See the bottom-middle of Figure 16.

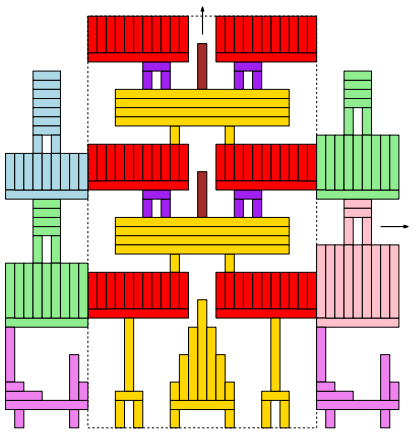


Figure 16: Illustrating two variable towers (left and right) with scaffolding (middle) between them.

A *scaffolding block* has two horizontal slabs supporting chunks of height 4 (red), each of which supports one leg of a large table (yellow). This table has 4 tabletop layers, and on top of it are two small tables (purple) on either side of a 5-unit vertical slab (maroon). This scaffolding tower can be seen in the center of Figure 16. If the player enters a scaffolding block from the bottom, they can access either side of the gadget, but cannot jump to the upper surface without a climbing aid (analogous to rungs from the toy problem). The mapping of these jumps to the satisfiability of the given problem is the discussion of Section 6.

6 Proof that Megalit is NP-Hard

The player must complete a constructed level in two stages – ascend to the top of the "haunted house," then flatten it using "pizza peels." We complete our proof of Theorem 1 by proving that the first stage can be completed if, and only if, the 3-SAT instance is satisfiable. More specifically, Observation 1 holds. The following lemmas prove the two directions.

Lemma 6 *Given a satisfiable instance of 3-SAT, the level generated by the reduction rules is climbable.*

Proof. Suppose we are given a satisfiable instance of 3-SAT and build a *Megalit* level by the procedure in Section 5. Next, we let the player dislodge the truth-setting slabs for each variable as would correspond to a satisfying assignment for the instance of 3-SAT. Since the given instance of the problem is satisfiable, each clause has at least one positive literal set to "true" or one negative literal set to "false."

In both of these cases, the corresponding gadget representing the variable literal is offset by exactly one unit above its neighboring scaffolding. We can see that the truth-assignment created this useful offset, just as it made the ladder rungs accessible in Lemma 4. As a result, the player is able to borrow an extra table leg from the literal and bring it into the scaffolding region close enough to the center that they can use it as a "ladder rung" to overcome the tall jump from the lower scaffolding surface to the upper one. See Figure 18.

Since this occurs for each clause, and therefore for each layer of scaffolding and literals, the player can jump all the way to the top of the house by moving laterally until they reach the satisfying literal, using its table leg to climb to the next layer, and repeating this process for each layer. In accordance with Observation 1, this climb is possible because the satisfying variable assignment provided a climbing aid on each "floor." □

Observation 2 *It is impossible to move a tabletop on an isolated table; see our physical model in Section 2.*

Observation 3 *Even when two tables are adjacent on the same surface, it is impossible to move either tabletop, provided the leg-heights of the tables are different.*

Lemma 7 *Given an unsatisfiable instance of 3-SAT, the level generated from the reduction has no escape routes from the haunted house.*

Proof. Suppose that we are given an unsatisfiable instance of 3-SAT. We must show that no alternative exits from the haunted house exist for the player. Referencing Figure 17, we will proceed with an examination of each gadget and then each zone where gadgets may interact with one another.

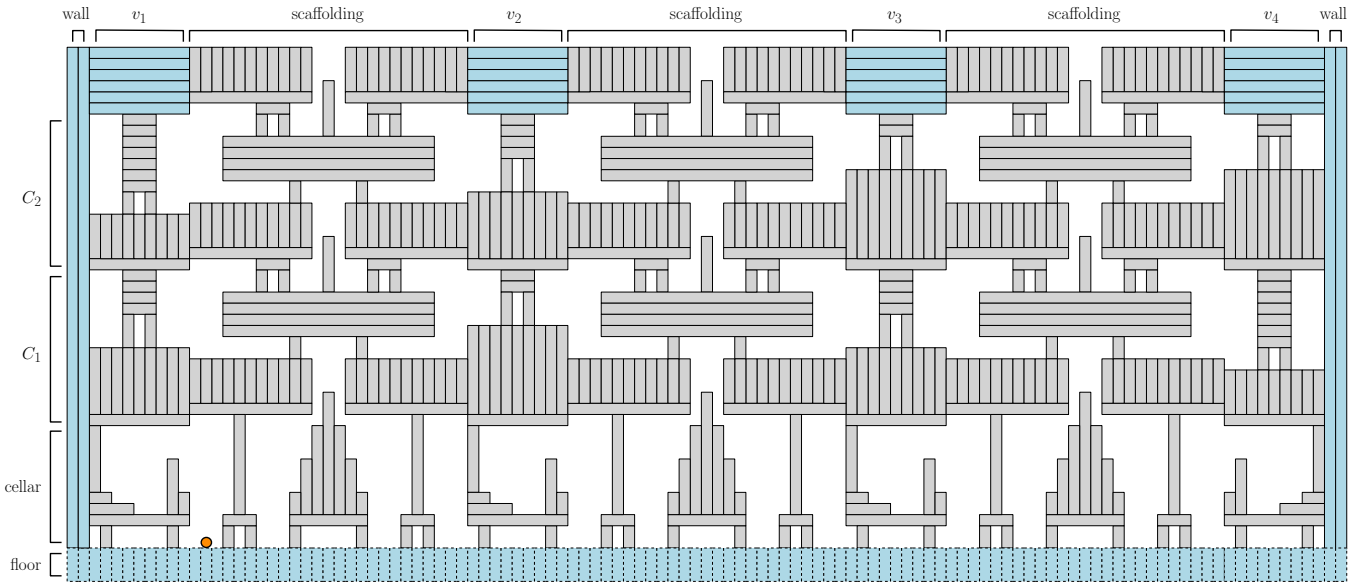


Figure 17: A full haunted house, with surrounding structure (walls and floor) and labeled regions, generated from the 3-SAT instance $\phi = C_1 \wedge C_2$ with $C_1 = (v_1 \vee \neg v_2 \vee v_3)$ (middle layer) and $C_2 = (v_2 \vee \neg v_3 \vee \neg v_4)$ (top layer). The player is shown at their starting location (orange circle) and the additions in blue ensure that the player may only exit the house via its top level. The added slabs on top of each scaffolding tower enable traversal along the "roof" and pairs of tall vertical slabs bookend the house as per Section 3.4. Note that the vertical "floor" slabs are much taller than shown here, and a set of pizza peels is present immediately to the right of this structure.

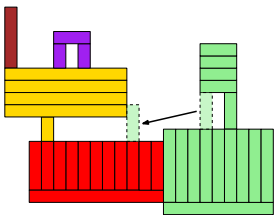


Figure 18: A +1 offset allows a table leg from the tower to be brought into the scaffolding region. The player cannot climb the scaffolding further without this aid.

The player has free lateral movement in the cellar and may encounter four notable structures:

1. *The Pit.* Due to the manner in which it traps the player, the pit has an isolating effect in all respects besides the intended 2-unit tower drop, and thus cannot affect the rest of the haunted house.
2. *Scaffolding Support.* Per Observation 3, the only way for the player to interact with the support's table is to pass beneath it. The mobility of the long vertical support slab is severely limited, as it cannot be safely dropped from the table and has only 1 unit of available movement left or right.
3. *Pyramid.* The aforementioned constraints on tables apply to the pyramid's base as well. The vertical slabs covering the tabletop's surface form a chunk, preventing access to each other's cornerstones.
4. *Thick Walls.* Double-slab walls blockade both sides of the house and are immobile.

None of these can interact with the larger structure except for the pit's intended transformation. Proceeding upwards, the player reaches a series of scaffolding/tower combinations. By itself, the scaffolding offers only two surfaces on which the player can stand and no mobile parts except the pairs of table legs. These legs cannot be dropped between scaffolding layers or surfaces within a layer because they would shatter. In an isolated tower, a variable literal offers the player only one surface on which to stand, from which the player can do nothing but pass underneath the table. At the edge of the house, the double-slab wall is again present, and equally immovable.

Finally, we consider interactions between the scaffolding and variable literals, characterized by the height of the literal relative to the scaffolding. For example, the surface of a true/negative literal is +3 units above the lower surface of the adjacent scaffolding.

- +3 (True/Negative): This offset is too high to safely drop a table leg from the tower's literal onto the scaffolding region. It is possible to use a scaffolding leg as a step down, creating an effective offset of +2, but there is no way to move it any closer to the scaffolding's center or cross the wide gap from there to the scaffolding's upper surface.
- +1 (False/Negative & True/Positive): As described in Lemma 6, this offset lets the player safely bring a table leg from the tower onto the scaffolding's lower surface, and use it as a ladder rung to reach the upper surface and continue climbing.
- −1 (False/Positive & True/Absent): Since slabs

may never move upwards (rule 3c), this table leg is stuck on the tower's surface. It is possible to drop a scaffolding leg into the tower section, but this offers no further productive moves. Note that the scaffolding leg *can* be used to replace the leg of a variable literal and lower the tower, but doing so will trap the player.

- -3 (False/Absent): The absent literal once again has no impact on the player's climbing ability.

Since no manipulations of slabs within a layer of scaffolding allow the player to climb higher without a positive evaluation of a variable, and there is no way to cause a collapse except for the 2-unit drop of a tower that occurs when escaping a pit, the player cannot escape the haunted house unless they create a positively-evaluated variable in each layer of scaffolding. This is equivalent to satisfying each clause in an instance of 3-SAT, but the given instance was unsatisfiable. Hence, the player cannot exit the haunted house. □

7 Open Problems

Sharpness. Is Megalit NP-complete or PSPACE-complete? Membership in NP is unclear since slabs can move back-and-forth and downward, but not upward.
Restrictions. Toward NP-completeness, one could consider slabs of constant size.
Generalizations. Toward PSPACE-completeness, one could add rectangular slabs, or immovable walls.
Physics. Megalit is not faithful to the physics of *Megalit*. In fact, $\frac{1}{2}$-unit moves and rule 3a invalidate Lemma 7. One could also consider center of gravity physics.
Fragility. Block pushing games and problems seldom consider the fragility of the objects being moved.
Unstacking. Other unstacking games could provide inspiration like *QBillion* (SETA, 1990) for Game Boy.

The authors would like to thank the anonymous referees whose feedback helped improve this paper.

References

[1] G. Aloupis, E. D. Demaine, A. Guo, and G. Viglietta. Classic nintendo games are (computationally) hard. *Theoretical Computer Science*, 586:135–160, 2015.

[2] 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.

[3] J. Ani, L. Chung, E. D. Demaine, Y. Diomidov, D. Hendrickson, and J. Lynch. Pushing blocks via checkable gadgets: PSPACE-completeness of Push-1F and Block/Box dude. In *Proceedings of the 11th International Conference on Fun with Algorithm (FUN 2022)*, volume 226 of *LIPIcs*, page 3:1–3:30, 2022.

[4] A. Barr, C. Chang, and A. Williams. Block Dude puzzles are NP-hard (and the rugs really tie the reductions together). In *Proceedings of the 33rd Canadian Conference on Computational Geometry, Dalhousie University, Halifax, Canada, August 10-12, 2021*, 2021.

[5] S. V. Chenoweth. On the NP-hardness of blocks world. In *AAAI*, pages 623–628, 1991.

[6] J. Culberson. Sokoban is PSPACE-complete. In *Proceedings of the 1st International Conference on Fun with Algorithm*, pages 65–76, 1998.

[7] 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.

[8] D. Dor and U. Zwick. Sokoban and other motion planning problems. *Computational Geometry*, 13(4):215 – 228, 1999.

[9] M. Forišek. Computational complexity of two-dimensional platform games. In *FUN*, 2010.

[10] E. Friedman. Pushing blocks in gravity is NP-hard. *Unpublished manuscript, March*, 2002.

[11] M. Fryers and M. T. Greene. Sokoban, 1995.

[12] 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.

[13] A. Greenblatt, J. Kopinsky, B. North, M. Tyrrell, and A. Williams. Mazezam levels with exponentially long solutions. In *20th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCGGG 2017)*, page 2, 2017.

[14] N. Gupta and D. S. Nau. On the complexity of blocks-world planning. *Artificial Intelligence*, 56(2-3):223–254, 1992.

[15] M. Hoffmann. Push-* is NP-hard. In *CCCG*, 2000.

[16] M. Johnson, C. Jonker, B. v. Riemsdijk, P. J. Feltovich, and J. M. Bradshaw. Joint activity testbed: Blocks world for teams (BW4T). In *International Workshop on Engineering Societies in the Agents World*, pages 254–256. Springer, 2009.

[17] A. G. Pereira, M. Ritt, and L. S. Buriol. Pull and pushpull are pspace-complete. *Theoretical Computer Science*, 628:50–61, 2016.

[18] M. Ritt. Motion planning with pull moves. *arXiv preprint arXiv:1008.2952*, 2010.

[19] L. She, S. Yang, Y. Cheng, Y. Jia, J. Chai, and N. Xi. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*, pages 89–97, 2014.

[20] J. Slaney and S. Thiébaux. Blocks world revisited. *Artificial Intelligence*, 125(1-2):119–153, 2001.

[21] 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/la/99/index.html, 1999.

[22] G. Viglietta. Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54:595–621, 2014.

[23] M. Willey. Ramunto's brick house pizza. . [Online; accessed 1-May-2022].