Complexity of 2D Snake Cube Puzzles

MIT Hardness Group* Nithid Anchaleenukoon† Alex Dang† Erik D. Demaine† Kaylee Ji†
Pitchayut Saengrungkongka†

Abstract

Given a chain of HW cubes where each cube is marked "turn 90°" or "go straight", when can it fold into a $1 \times H \times W$ rectangular box? We prove several variants of this (still) open problem NP-hard: (1) allowing some cubes to be wildcard (can turn or go straight); (2) allowing a larger box with empty spaces (simplifying a proof from CCCG 2022); (3) growing the box (and the number of cubes) to $2 \times H \times W$ (improving a prior 3D result from height 8 to 2); (4) with hexagonal prisms rather than cubes, each specified as going straight, turning 60°, or turning 120°; and (5) allowing the cubes to be encoded implicitly to compress exponentially large repetitions.

1 Introduction

Snake Cube [1] is a physical puzzle consisting of wooden unit cubes joined in a chain by an elastic string running through the interior of each cube. For every cube other than the first and last, the string constrains the two neighboring cubes to be at opposite or adjacent faces of this cube, in other words, whether the chain must continue straight or turn at a 90° angle. In the various manufactured puzzles, the objective is to re-arrange a chain of 27 cubes into a $3 \times 3 \times 3$ box.

To generalize this puzzle, we ask: given a chain of DHW cubes, where D, H, W are positive integers, is it possible to rearrange the cubes to form a $D \times H \times W$ rectangular box? We call this problem $D \times H \times W$ SNAKE CUBE. Previous results on its complexity include:

- Abel et al. [1] proved $8 \times H \times W$ SNAKE CUBE is NP-complete by reduction from 3-PARTITION.
- Demaine et al. [2] proved 2D SNAKE CUBE PACKING—deciding whether a chain of cubes can ***pack*** (but not necessarily fill) a $1 \times H \times W$ rectangular box where all cubes are constrained to align with the box—is NP-complete by reduction from LINKED PLANAR 3SAT. This result also holds for a closed chain [2].

Both [1] and [2] pose the (still) open problem of determining the complexity of $1 \times H \times W$ SNAKE CUBE:

Open Problem 1 (2D Snake Cube) *Is* $1 \times H \times W$ SNAKE CUBE *NP-hard?*

1.1 Our Results

In this paper, we prove NP-hardness of several variations of Open Problem 1:

- In Section 4, we prove NP-completeness of 2D SNAKE CUBE WITH WILDCARDS where at some cubes there is a free choice between straight or turn. This is motivated by a variant of the snake cube puzzle where a slit cut into a cube allows the chain to continue at a 90° or 180° angle.

  We also give an alternative proof that 2D SNAKE CUBE PACKING is NP-complete, simplifying [2].
- In Section 5, we prove that $2 \times H \times W$ SNAKE CUBE is NP-complete. This improves the result of Abel et al. [1] from $D = 8$ to $D = 2$.
- In Section 6, we prove NP-completeness of HEXAGONAL 2D SNAKE CUBE PACKING: deciding whether a chain of hexagonal prisms each specified as going straight, turning 60°, or turning 120° can be packed into a 60°, $H \times W$ parallelogram. Similar to [2], we extend this result to closed chains. One can view this as an improvement to [3] in that angles can be restricted to be in $\{60°, 120°\}$.
- In Section 7, we prove *weak* NP-hardness of 2D SNAKE CUBE, allowing the chain of cubes to be encoded to efficiently represent repeated sequences.

The first three results are reductions from NUMERICAL 3D MATCHING following a similar framework detailed in Section 3, while the last result is a reduction from 2-PARTITION. We introduce both base problems in Section 2.

Not all results are proven fully in this paper. All omitted details can be found in the full version of the paper.

*Artificial first author to highlight that the other authors (in alphabetical order) worked as an equal group. Please include all authors (including this one) in your bibliography, and refer to the authors as "MIT Hardness Group" (without "et al.").

†MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, {nithidan,alexdang, edemaine,kayleeji,psaeng}@mit.edu

2 Preliminaries

We define our exact problems in mathematical terms.

A ***box*** is the $D \times H \times W$ rectangular cuboid that the cubes of the snake-cube puzzle must fit into. This box can be visualized as a cubic grid where each cube occupies one space of the grid. A ***program*** is a length-k string of ***instructions*** $\mathcal{P} = p_1 \dots p_k$, where each instruction is either the character `T` or `S`. The ***chain*** is the corresponding sequence of adjacent cubes $(c_1, \dots, c_k)$ following the program such that each instruction p_i (where $i \in \{2, \dots, k-1\}$) constrains the angle between the 3 cubes c_{i-1}, c_i, c_{i+1} to be 90° for $p_i =$ `T` (i.e., a turn) and 180° for $p_i =$ `S` (i.e., a straight). A length-k ***segment*** refers to a length-k subchain where all cubes are constrained to form a straight line (e.g., the subchain following the instructions `TSSST` refers to a length-5 segment). If s is a sequence of instructions, let $(s)^k$ denote s repeated k times (e.g., $\texttt{T}(\texttt{ST})^3$ is equivalent to `TSTSTST`). The input to all problems is the box and program. In 2D Snake Cube with Wildcards, each instruction may also be a third character `*` denoting that the angle can be either 90° or 180°. The instructions in Hexagonal 2D Snake Cube Packing use three different characters introduced in Section 6.

2D Snake Cube with Wildcards, $2 \times H \times W$ Snake Cube, and Hexagonal 2D Snake Cube Packing are in NP, because verification only requires checking all constraints, which takes linear time with respect to the size of the box.

2.1 Reduction Base Problems

Given a multiset $A = \{a_1, a_2, \dots, a_n\}$ of positive integers, ***2-Partition*** is the problem of deciding whether there exists a partition of A into disjoint union $A_1 \sqcup A_2$ such that the sums of elements in A_1 and in A_2 are equal. This problem is known to be weakly NP-hard when the number a_i's are encoded in binary (thus may have exponential value) [4, Section A3.2].

For any given target sum t and sequences $(a_i)_{i=1}^n$, $(b_i)_{i=1}^n$, and $(c_i)_{i=1}^n$, each consisting of n positive integers, ***Numerical 3-Dimensional Matching (N3DM)*** is a problem to decide whether there exist permutations σ and π of set $\{1, \dots, n\}$ that satisfies $a_i + b_{\sigma(i)} + c_{\pi(i)} = t$ for all i. This problem is known to be NP-hard even when the numbers are encoded in unary [4, Section A3.2]. We refer to a solution to an instance of N3DM as a ***matching***.

Since we can transform an instance of N3DM by setting $a'_i = a_i + 4X$, $b'_i = b_i + 2X$, $c'_i = c_i + X$, and $t' = t + 7X$, for a large integer X (linear in t), the following proposition holds.

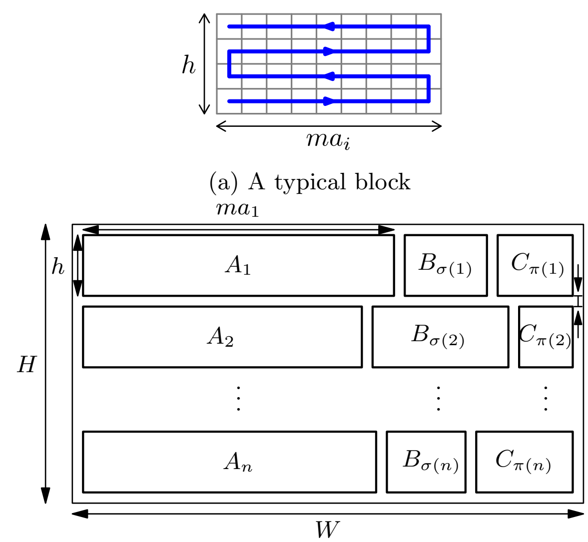
Proposition 2 N3DM *is NP-hard even when we assume that* $a_i \in (0.5t, 0.6t)$*,* $b_i \in (0.25t, 0.3t)$*, and* $c_i \in (0.125t, 0.15t)$ *for all* $1 \le i \le n$*.*

3 Overview of Reductions from N3DM

The reductions in Sections 4, 5, and 6 all share a very similar infrastructure, which we informally outline here. In this overview, we let $D = 1$. We explain how to adapt this framework to $D = 2$ in Section 5.

We reduce from the variant of N3DM in Proposition 2. Let $(a_i)_{i=1}^n$, $(b_i)_{i=1}^n$, $(c_i)_{i=1}^n$, and t be an instance of N3DM. We choose the following parameters: the gap width $g = \Theta(n)$, the height of the block $h = \Theta(n^2)$, and the width multiplier $m = \Theta(n^3)$.

The structure of the reduction is as follows. The dimensions of the box are $D \times H \times W = 1 \times (nh + (n+1)g) \times (mt + 4g)$. The numbers $(a_i)_{i=1}^n$, $(b_i)_{i=1}^n$, and $(c_i)_{i=1}^n$ are represented by ***block gadgets*** $(\langle A_i \rangle)_{i=1}^n$, $(\langle B_i \rangle)_{i=1}^n$, and $(\langle C_i \rangle)_{i=1}^n$, which are instructions that can generate ***blocks*** $(A_i)_{i=1}^n$, $(B_i)_{i=1}^n$, and $(C_i)_{i=1}^n$ of dimensions $1 \times h \times ma_i$, $1 \times h \times mb_i$, and $1 \times h \times mc_i$, respectively. Blocks typically consist of h segments as shown in Figure 1a, but details vary in different variants. In the instructions, each block gadget will be separated by a ***wiring gadget***, a sequence of instructions that allows connecting between two adjacent blocks no matter where they are in the grid.



(a) A typical block

(b) The high-level structure of the reduction

Figure 1: The reduction

If a matching exists (i.e., there exist two permutations σ and π of $\{1, 2, \dots, n\}$ such that $a_i + b_{\sigma(i)} + c_{\pi(i)} = t$ for all i), then (ignoring the wiring gadget) one can arrange the blocks into a perfect $1 \times nh \times mt$ rectangle by aligning each triple of blocks A_i, $B_{\sigma(i)}$, and $C_{\pi(i)}$ together in the same row. Since our box is slightly larger than $1 \times nh \times mt$, we can place the blocks such that there is a gap g between neighboring blocks and between each block and the boundary of the rectangular box. The gap g is chosen so there is sufficient space for a subchain following the wiring gadget to connect all the blocks. Wires detour around blocks and do not cross; the explicit algorithm will be given in Lemma 4. Finally, depending on the variant, there may be additional instructions at the end of the program to fill in the remaining space in the box. Figure 1b depicts the overall reduction structure.

In the other direction, we also need to show that the existence of a chain following the program forces the existence of matching, even if the block gadgets $\langle A_i \rangle$, $\langle B_i \rangle$, and $\langle C_i \rangle$ do not fold into perfectly aligned and evenly spaced blocks (e.g., if part of a subchain following $\langle B_i \rangle$ may go into gaps between subchains following $\langle A_i \rangle$). In the following subsection, we prove Lemma 3 that shows the existence of a chain following the program necessitates the existence of a matching, even if blocks do not fold ideally.

3.1 Segment Packing Lemma

We view each block as h segments; for instance, the block gadget $\langle A_i \rangle$ specifies h consecutive ma_i-segments. Thus, we have $3nh$ segments, h of each length $ma_1, \ldots, ma_n, mb_1, \ldots, mb_n, mc_1, \ldots, mc_n$ to pack into the box. This motivates the following "Segment Packing Lemma".

Lemma 3 (Segment Packing Lemma) *Let $(a_i)_{i=1}^n$, $(b_i)_{i=1}^n$, $(c_i)_{i=1}^n$, t be an instance of* N3DM *satisfying the conditions in Proposition 2. Let m and h be positive integers, and consider a $1 \times H \times W$ box where $W > mt$ and $nh < H < m$. Suppose there are $3nh$ segments of $3n$ types $A_1, \ldots, A_n$, $B_1, \ldots, B_n$, and $C_1, \ldots, C_n$. If all of the following are true, then there exists an N3DM matching:*

- *$W < m(t+1)$ and $H < nh + \frac{h}{40}$;*
- *for all $1 \le i \le n$, all segments of type A_i, B_i, and C_i have lengths ma_i, mb_i, and mc_i, respectively;*
- *there are exactly h segments of each type; and*
- *no two segments of the same type are more than h rows vertically apart (note that since $m > H$, all $3nh$ segments must lie horizontally in the box.).*

Proof. (Sketch) We call ma_i-segments A-segments, and analogously for B-segments and C-segments. From constraints in Proposition 2, each row of the box must be of one of the following four categories: (1) a *good row*, which contains exactly one A-segment, one B-segment, and one C-segment; (2) an *A-bad row*, which contains no A-segment; (3) a *B-bad row*, which contains one A-segment but contains no B-segment; and (4) a *C-bad row*, which contains one A-segment, one B-segment, but no C-segment. Let n_{good}, n_A, n_B, and n_C denote the number of good rows, A-bad rows, B-bad rows, and C-bad rows, respectively. Due to the constraints of Proposition 2 and $W < m(t+1)$, we count the number of A-, B-, and C-segments to derive the following inequalities:

$$n_A = H - nh < \tfrac{h}{40}$$
$$n_B \le 2n_A + \tfrac{h}{40} < \tfrac{3h}{40}$$
$$n_C \le 6n_A + 2n_B + \tfrac{h}{40} < \tfrac{13h}{40}.$$

Therefore, $n_A + n_B + n_C < h$.

Finally, we color each row by its residue modulo h. Thus, there are either n or $n+1$ of each color. Moreover, there exists color c that colors only good rows. Since segment of the same type are less than h rows apart, there is exactly one segment of each type colored c and exactly n rows of color c. For each row of color c, let ma_i, mb_j, and mc_k be the segment lengths. Then,

$$ma_i + mb_j + mc_k \le W < m(t+1) \implies a_i + b_j + c_k \le t.$$

Summing the inequality for each row of color c gives $nt \le nt$, so all inequalities must be equalities. Therefore, $a_i + b_j + c_k = t$ for each row of color c, forming a solution to the instance of N3DM. $\square$

3.2 Connecting Wires

This subsection concerns the wiring part. It guarantees that, if the gap is large enough, there exists a way to place wiring gadgets without crossing, regardless of the arrangement of blocks forced by a solution to the instance of N3DM. This lemma was adapted from [5, Lemma 5].

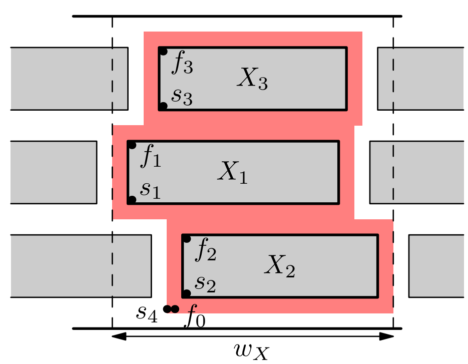


Figure 2: Example of the setup for wire packing when $n = 3$. Red area represents available space.

The setup for this lemma is depicted in Figure 2 and goes as follows: given a bounding box of size $H' \times w_X$ and locations of ***rectangles*** $X_1, X_2, \ldots, X_n$ with widths $x_1, x_2, \ldots, x_n$, respectively, and the same height h'. Each row contains at most one rectangle, but the rectangles are in arbitrary order from top to bottom. Note that the "rectangles" are not the same as blocks; a rectangle consists of squares, and a square is filled with 2×2 cubes which will be discussed further in Section 4 when applying this lemma to prove the existence of a chain. Define a wire connecting squares a and b to be a sequence of adjacent squares with the first and the last squares are adjacent to a and b, respectively.

Lemma 4 (Wire Lemma) *Assume the above setup with* $\min_i x_i > w_X/2$*, and all rectangles are at least* $g' \geq 100n$ *squares apart. Define the* ***available space*** *to be a set of squares in the extension of all rectangles on each edge by* $g'/2$*. For each* $i = 0, 1, \ldots, n$*, let* ℓ_i *be an even integer in* $[8nw_X, 12nw_X]$*. Let* s_i *and* f_i *be the bottom-left and top-left corners of rectangle* X_i*, and* f_0, s_{n+1} *be two chosen squares at the bottom-left of the available space. Then, one can draw* $n+1$ *disjoint wires* $W_0, \ldots, W_n$ *in the available space, where* W_i *has length exactly* ℓ_i *and* W_i *connects* f_i *to* s_{i+1} *for all* $i \in \{0, 1, \ldots, n\}$*. Furthermore, no two cells from different wires* W_i *and* W_j *are adjacent.*

Proof. (Sketch) We will briefly explain an algorithm to place the wires $W_0, \ldots, W_n$ inductively. First, mark squares $m_0 = f_0, m_1, \ldots, m_n, m_{n+1} = s_{n+1}$ in the same row in this order; all of these should be near the bottom-left of overall available space. We will construct wires $(U_i)_{i=1}^n$ and $(V_i)_{i=1}^n$ such that U_i connecting m_{i-1} to s_i, and V_i connecting m_i to f_i. Then, W_i is a concatenation of wire U_{i+1}, square m_i, and wire V_i for all $i \in \{1, \ldots, n-1\}$. Moreover, $W_0 = U_0$ and $W_n = V_n$. We also reserve space of width $40n$ squares above and below each rectangle and $10n$ squares on the left of each rectangle. The two main stages of placing wires are

(a) Place U_i and V_i without crossing U_1, V_1, …, U_{i-1}, V_{i-1}. This process is done inductively.

(b) Adjust the length of the wire W_i to be exactly ℓ_i by placing the remaining length U_i and V_i inside reserved space of rectangle X_i, which has size at least $40n \times x_i$; the space can fit a wire of length $> 20nw_X$, large enough to contain the extra length.

To accomplish (a), place U_i and V_i by following these steps simultaneously for each i.

(i) Create a sequence of squares from m_i to the top of the available space, following along the left gaps.

(ii) Draw the wire down to the same row as s_i between the wires we have placed in (i) and the left edges of all rectangles, and then draw the wire horizontally to s_i.

(iii) The current wire may cross U_j or V_j for some $j < i$ when they are horizontally connected to s_j or f_j. In this case, replace the current wires by making then go around other edges of rectangle X_j.

To justify the size of available space, each of U_i and V_i may contribute to at most 2 layers of wires on each edge of the block with a space of one square between each layer of wires. Combine this with the reserved space; we need available space with width $40n + 2 \cdot 2 \cdot (2n) < 50n$ on each edge of the rectangles.

The dominant contribution to the length of the wire occurs when the wires have to go around other rectangles since $w_X \gg nh' + (n+1)g'$. However, there are at most n blocks that a wire has to go around. Including all other distances, the sufficient length of a wire is $8nw_X$. □

4 Snake Cube Puzzles in $1 \times H \times W$ box

In this section, we consider the 2-dimensional variants of Snake Cube. We first consider 2D Snake Cube with Wildcards, where we allow the wildcard `*` that could be used as either `S` or `T`. We will prove the following:

Theorem 5 2D Snake Cube with Wildcards *is NP-hard.*

Section 4.1 will sketch the proof of Theorem 5. Then, in Section 4.2, we will explain how to modify this proof to give an alternative proof of the following, which was first proved in [2].

Theorem 6 2D Snake Cube Packing *is NP-hard.*

4.1 Proof with Wildcard Option

Given an instance of N3DM with target sum t, $(a_i)_{i=1}^n$, $(b_i)_{i=1}^n$, $(c_i)_{i=1}^n$, where $a_i \in (0.5t, 0.6t)$, $b_i \in (0.25t, 0.3t)$, and $c_i \in (0.125t, 0.15t)$ for all i (Proposition 2), we define these parameters to construct a string input to 2D Snake Cube with Wildcards.

$$\begin{aligned}
g &= \text{gap width} &&= 200n \\
m &= \text{multiplier of widths} &&= 30000n^3 \\
h &= \text{height of blocks} &&= 20000n^2 \\
H &= \text{height of the grid} &&= nh + (n+1)g \\
W &= \text{width of the grid} &&= mt + 4g
\end{aligned}$$

Then, construct block gadgets A_i, B_i, and C_i for all $1 \leq i \leq n$. The sequence for A_i is given below, and the sequences for B_i, C_i are analogous. These blocks will fold into rectangles of size $h \times ma_i$, $h \times mb_i$, or $h \times mc_i$.

$$\langle A_i \rangle = (\mathtt{S})^{ma_i - 1} (\mathtt{TT}(\mathtt{S})^{ma_i - 2})^{h-1}$$

The program is given by

$$\begin{aligned}
&\langle A_1 \rangle (*)^{16nW} \langle A_2 \rangle (*)^{16nW} \ldots (*)^{16nW} \langle A_n \rangle (*)^{16nW} \\
&\quad \langle B_1 \rangle (*)^{8nW} \langle B_2 \rangle (*)^{8nW} \ldots (*)^{8nW} \langle B_n \rangle (*)^{8nW} \\
&\quad \langle C_1 \rangle (*)^{4nW} \langle C_2 \rangle (*)^{4nW} \ldots (*)^{4nW} \langle C_n \rangle (*)^{4nW} (*)^{\ell},
\end{aligned}$$

where the number ℓ of `*`'s at the end is to make the length of the whole program exactly WH.

Chain $\Rightarrow$ Matching. Each block gadget $\langle A_i \rangle$, $\langle B_i \rangle$, $\langle C_i \rangle$ contains h segments that must be horizontal due to its length. Thus, we have $3nh$ segments of $3n$ types that fit the condition of Lemma 3. The lemma forces the existence of matching.

Matching $\Rightarrow$ Chain. Suppose there is a matching, so we can place all blocks as shown in Figure 1b.

Apply the wire lemma to connect all these blocks where each square in the lemma corresponds to 2×2 cubes in this construction. Hence, each block is at least $g' = g/2 = 100n$ squares apart, and the width of the bounding box is $w_X = W/2$ squares. For wires among A_i blocks, Lemma 4 implies that there exists a sequence of $8nw_X = 4nW$ squares of size 2×2 that connect all of A_i blocks. These squares can be filled with $16nW$ `*` cubes as demonstrated in Figure 3. For wires among B_i blocks, the block size is roughly half of A_i; thus, we can reduce the parameter w_X in the lemma by half. Similarly, the parameter is reduced to a quarter for C_i blocks.

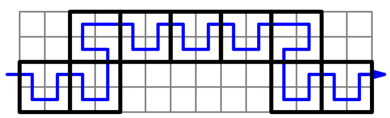


Figure 3: Example of filling a wire with `*` tiles.

The wire between A_n and B_1 is long enough to connect the following cubes in order: (1) the last cube of block A_n, (2) the cube at the square marked as f_n in the application of Lemma 4 to connect wires between A_i blocks, (3) the cube at the square marked as s_0 in the application of Lemma 4 to connect wires between B_i, and (4) the first cube of block B_1. This is always possible since gaps are all connected, and the length needed never exceeds $8nW + 4W + 4nW \leq 16nW$. The wire between B_n and C_1 can be placed similarly.

Lastly, notice that the construction we described so far is aligned with the 2×2 polygrid, and the remaining squares are connected because the blocks and wires are topologically equivalent to a path with no closed loop. From [6], there exists a Hamiltonian cycle in any connected shape aligned with 2×2 polygrid. Thus, the remaining wildcards can fill all the remaining space.

4.2 Proof Outline for Packing

To prove Theorem 6, the setup and the main proof are almost the same. The only difference is that we cannot use the wildcard `*`. To fix this, we make two modifications. First, we remove $(*)^\ell$ at the end because we do not need to fill the box. Second, we replace all $*$'s between block gadgets with an equally long string of `T`'s. By making squares in Lemma 4 correspond to 2×2 cubes, wiring gadgets can connect different block gadgets using only `T`'s. This is possible because not all cubes in the wires need to be used, and cubes outside the wires may be used. We can make the wire length exactly as specified by varying how the chain fills the wires at turns. More details are available in the full version of the paper.

5 Snake Cube Puzzles in $2 \times H \times W$ box

In this section, we outline our reduction from N3DM to $2 \times H \times W$ Snake Cube. A detailed proof can be found in the full version of the paper.

Theorem 7 *$2 \times H \times W$ Snake Cube is NP-hard.*

We follow the block and wire reduction infrastructure introduced in Section 3. Consider an instance of N3DM $(a_i)_{i=1}^n$, $(b_i)_{i=1}^n$, $(c_i)_{i=1}^n$, and t satisfying the conditions in Proposition 2. Define the parameters as follows.

$$g = 1200n, \quad h = 60000n^2, \quad m = 60000n^3,$$
$$H = n(h + 6g + 4) + 2, \quad W = 4g + mt + 6.$$

In this variant, the blocks generated by the block gadgets have depth 2 (e.g., block A_i is of size $2 \times h \times ma_i$). Like previously, the program consists of block gadgets separated by wire gadgets. However, unlike previous $1 \times H \times W$ variants, a second layer results in fewer constraints on the shape of subchains following the block gadget, so Lemma 3 no longer applies. To bypass this issue, we introduce additional instructions at the beginning of the program that specifies a ***shelf*** — the structure shown in Figure 4 that constrains the folding of the $\langle A_i \rangle$, $\langle B_i \rangle$, and $\langle C_i \rangle$ subchains. The shelf is designed so that it can only be made into a subchain the intended way. The program is given by

$$\begin{aligned}\langle \text{shelf} \rangle \langle A_1 \rangle (\texttt{T})^{96nW} \langle A_2 \rangle (\texttt{T})^{96nW} \ldots (\texttt{T})^{96nW} \langle A_n \rangle (\texttt{T})^{96nW} \\ \langle B_1 \rangle (\texttt{T})^{48nW} \langle B_2 \rangle (\texttt{T})^{48nW} \ldots (\texttt{T})^{48nW} \langle B_n \rangle (\texttt{T})^{48nW} \\ \langle C_1 \rangle (\texttt{T})^{24nW} \langle C_2 \rangle (\texttt{T})^{24nW} \ldots (\texttt{T})^{24nW} \langle C_n \rangle (\texttt{T})^{\ell},\end{aligned}$$

where $(\texttt{T})^\ell$ pads the string to length $2HW$ and $\langle A_i \rangle = (\texttt{S})^{ma_i - 1}(\texttt{TT}(\texttt{S})^{ma_i - 2})^{2h-1}\texttt{S}$; $\langle B_i \rangle$ and $\langle C_i \rangle$ are defined analogously.

Chain $\Rightarrow$ Matching. Since the gap to the right of each shelf is small ($g \ll m$), all cubes within a block must fit entirely within one row of the shelf. By a similar counting argument as in Lemma 3, there exists a row containing exactly one A_i, B_j, and C_k block in each shelf, and the corresponding a_i, b_j, and c_k must sum to t.

Matching $\Rightarrow$ Chain. Place all blocks as in Figure 1b where segments in the blocks fill the top and bottom layers alternately. The remaining grid can be partitioned into $2 \times 4 \times 4$ blocks of space. We cite a result from [1]:

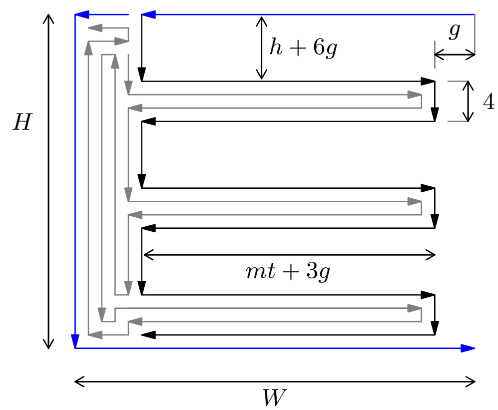


Figure 4: One layer of the "shelf" with 3 rows. The chain moves to the other layer at discontinuities.

given a sequence $(\mathtt{T})^8$ of cubes entering a $2 \times 2 \times 2$ block of space, the cube chain can exit from any face. Thus, traversing between $2 \times 2 \times 2$ blocks of space with $(\mathtt{T})^8$'s has the same movement freedom as traversing between cells in a 2D grid with wildcards. Thus, by grouping $2 \times 2 \times 2$ cubic blocks together, the remaining proof is equivalent to that in Section 4, except the wires are $\frac{3}{2}$ times long to allow for detours around the shelf.

6 Snake Cube Puzzles with Hexagonal Prisms

In this section, we consider a version of a 2D Snake cube with a chain of hexagonal prisms. When the prisms are represented by points, the movement patterns form a triangular grid. Thus, the problem becomes a triangular grid variant of the flattening fixed-angle chains problem in [2].

An infinite ***triangular grid*** is a two-dimensional lattice generated by vectors $v_1 = \binom{1}{0}$ and $v_2 = \binom{\cos 60°}{\sin 60°}$; each point represents a hexagonal prism. Two points in a triangular grid are ***adjacent*** if they are distance 1 apart. A ***60° parallelogram box*** of dimension $H \times W$ is the set of HW points obtained by translating the set $\{iv_1 + jv_2 : i \in \{1, \ldots, W\}, j \in \{1, \ldots, H\}\}$ by some lattice vector.

For this section, a ***program*** is a string that consists of only characters $\mathtt{S}$, $\mathtt{T}_{60}$, and $\mathtt{T}_{120}$, where $\mathtt{S}$ denotes straights, $\mathtt{T}_{60}$ denotes 60° turns (forming 120° angle), and $\mathtt{T}_{120}$ denotes 120° turns (forming 60° angle). We say that a chain $C = (p_1, p_2, \ldots, p_{|s|})$ (length $|s|$) satisfies s if and only if for every $i \in \{2, 3, \ldots, |s| - 1\}$, the angle between p_{i-1}, p_i, p_{i+1} is 180° if $s_i = \mathtt{S}$, 60° if $s_i = \mathtt{T}_{120}$, and 120° if $s_i = \mathtt{T}_{60}$. C is ***closed*** if and only if $p_1 = p_{|s|}$.

Theorem 8 *Both of the following problems are NP-complete.*

- BOUNDED TRIANGULAR PATH PACKING*: given a 60° parallelogram box B, a program $\mathcal{P}$, and two adjacent vertices u and v on a boundary of B, decide whether there is a chain connecting u and v satisfying $\mathcal{P}$.*
- TRIANGULAR CLOSED CHAIN*: given a program $\mathcal{P}$, decide whether there is a closed chain satisfying $\mathcal{P}$.*

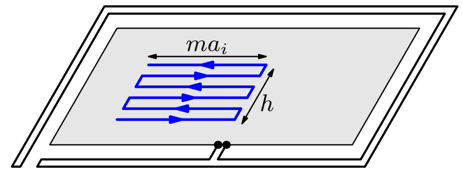


Figure 5: The frame gadget and an example block gadget inside.

To prove the first problem NP-Hard, we use the same reduction, except that block gadgets are 60° parallelograms shown in Figure 5. Then, we can reduce the first problem to the second problem, creating a ***frame*** gadget to force the chain by modulo a large prime condition similar to [2] shown in Figure 5.

7 Weak-NP-hardness of 2D Snake Cube Puzzle

In this section, we consider 2D SNAKE CUBE, where the chain must fill a $1 \times H \times W$ rectangle. However, we allow the instructions to be encoded using the shorthand notation, which keeps the inputs polynomial with respect to the input integers. Since this modification means the problem may no longer be in NP, this reduction only proves NP-hardness. For any set S, let $\sum S$ be the sum of its elements.

Let A be the multiset of positive integers, a 2-PARTITION instance. We select $H = 2|A| + 4$ and $W = 4 \sum A + 1$. The program comprises the ***caps*** at either end and $|A|$ ***layers*** in between, encoding each a_i in A sequentially. The ***swivel points*** join each gadget and allow the layers to flip horizontally. The orientation of each layer left or right corresponds to assigning each a_i to either partition (Figure 6).

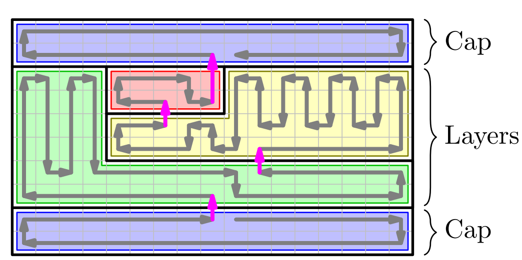


Figure 6: Chain for $A = \{1, 2, 1\}$, emphasizing the different gadgets, highlighting the swivel points (in bolded red), and demonstrating the 3 variants of layers.

7.1 Gadgets

The starting cap is the subsequence (the ending cap being the reverse):

$$(\mathtt{S})^{\frac{W-1}{2}-1}\mathtt{TT}(\mathtt{S})^{W-2}\mathtt{TT}(\mathtt{S})^{\frac{W+1}{2}-1}\mathtt{T}\dots.$$

Since $W > H$, the W-segments in the caps can only fit horizontally. They must be at the top and bottom since any other position would create an unfillable empty space. This forces the position of the swivel points joining the caps and the layers to be horizontally centered.

For each i, let $A_i = \{a_j : j \in \{1, \dots, i\}\}$, $w_i = 4\sum(A \setminus A_{i-1})+1$, $x_i = (w_i-1)/2$, and $h_i = 2|A\setminus A_{i-1}|$. There are 3 variants of the corresponding layer gadget.

If $4a_i \le x_i$ and $h_i > 2$, the layer is the subsequence (sections named for ease of discussion, see Figure 7):

$$\begin{aligned} \dots\mathtt{T}(\mathtt{S})^{x_i-1}\mathtt{T} &\qquad \text{“arm”} \\ (\mathtt{S})^{h_i-2}(\mathtt{TT}(\mathtt{S})^{h_i-3})^{4a_i-1}\mathtt{T} &\qquad \text{“padding”} \\ (\mathtt{S})^{x_i-4a_i+1}(\mathtt{T})^{2(2a_i-1)} &\qquad \text{“shift”} \\ (\mathtt{S})^{x_i-2a_i}\mathtt{TT}(\mathtt{S})^{x_i-2a_i+1}\mathtt{T}\dots &\qquad \text{“return.”} \end{aligned}$$

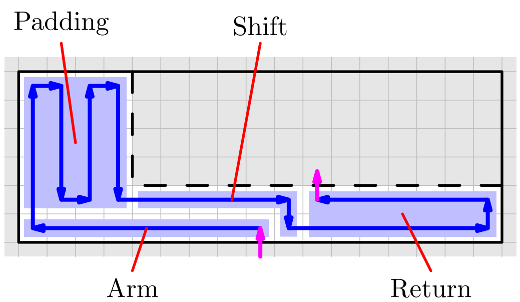


Figure 7: Sample layer gadget with $a_i = 1$, $w_i = 17$, $h_i = 6$ with labeled sections.

If $4a_i > x_i$ and $h > 2$, informally the padding *spills over* into the shift, resulting in these differences:

$$\begin{aligned} (\mathtt{S})^{h-2}(\mathtt{TT}(\mathtt{S})^{h-3})^{x_i}(\mathtt{TT}(\mathtt{S})^{h-2})^{4a_i-x_i-1} &\qquad \text{“padding”} \\ (\mathtt{T})^{2(2a_i-1-(4a_i-x_i-1))} &\qquad \text{“shift.”} \end{aligned}$$

If $h = 2$, informally the padding can be visualized as degenerating and subsuming the shift and return, resulting in these changes from the first variant:

$$\mathtt{T}(\mathtt{S})^{x_i}(\mathtt{T})^{2(2a_i-1)+1}\mathtt{S} \qquad \text{“padding.”}$$

Each layer gadget has a $w_i \times h_i$ space available to it and leaves behind a $w_{i+1} \times h_{i+1}$ space while displacing the swivel point horizontally by $2a_i$ left or right. To show this, we use induction starting from the first layer. Note that the arm and padding sections are all forced by space constraints. The shift section is forced since turning the chain outward in the subsequence of repeated turns (T) would leave behind a 1×1 space. This space can only be filled by the endpoints, which is impossible because their positions are forced by the cap gadgets. Then, the return section is also forced.

7.2 Reduction

If there exists a solution to 2-Partition, then construct all the gadgets and flip the layer gadgets so that arms for all numbers in A_1 point to the left, and those for numbers in A_2 point to the right. The horizontal displacements of the swivel points must sum to 0, so the last layer can connect to the upper cap.

If there exists a solution for 2D Snake Cube, then we have demonstrated the gadgets are forced to be constructed in the correct orientation. Since the last layer gadget connects to the upper cap gadget, the horizontal displacements of the swivel points must sum to 0. Reversing the above process produces a solution to the 2-Partition instance.

Acknowledgement

This work was conducted during open problem-solving sessions in the MIT class on Algorithmic Lower Bounds: Fun with Hardness Proofs (6.5440) in Fall 2023. We thank the other participants in the class — in particular, Papon Lapate, Benson Lin Zhan Li, and Kevin Zhao — for related discussions and for providing an inspiring atmosphere.

References

[1] Z. Abel, E. D. Demaine, M. L. Demaine, S. Eisenstat, J. Lynch, and T. B. Schardl, “Finding a Hamiltonian path in a cube with specified turns is hard,” *Journal of Information Processing*, vol. 21, no. 3, pp. 368–377, 2013.

[2] E. D. Demaine, H. Ito, J. Lynch, and R. Uehara, “Computational complexity of flattening fixed-angle orthogonal chains.” arXiv:2212.12450, 2022. https://arXiv.org/abs/2212.12450. Preliminary version in CCCG 2022.

[3] E. D. Demaine and S. Eisenstat, “Flattening fixed-angle chains is strongly NP-hard,” in *Proceedings of the 12th Algorithms and Data Structures Symposium (WADS 2011)*, pp. 314–325, August 15–17 2011.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness.* USA: W. H. Freeman & Co., 1990.

[5] Z. Abel and E. Demaine, “Edge-unfolding orthogonal polyhedra is strongly NP-complete,” in *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry (CCCG 2011)*, 2011.

[6] K. C. Cheung, E. D. Demaine, J. R. Bachrach, and S. Griffith, "Programmable assembly with universally foldable strings (moteins)," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 718–729, 2011.