Regular Paper

Tetris is NP-hard even with $O(1)$ rows or columns

Sualeh Asif[1,a)]   Michael Coulombe[1,b)]   Erik D. Demaine[1,c)]
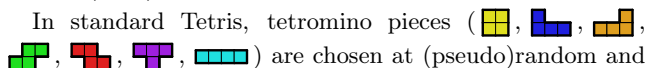Martin L. Demaine[1,d)]   Adam Hesterberg[1,e)]   Jayson Lynch[1,f)]   Mihir Singhal[1,g)]

Abstract:
We prove that the classic falling-block video game *Tetris* (both survival and board clearing) remains NP-complete even when restricted to 8 columns, or to 4 rows, settling open problems posed over 15 years ago [2]. Our reduction is from 3-Partition, similar to the previous reduction for unrestricted board sizes, but with a better packing of buckets. On the positive side, we prove that 2-column Tetris (and 1-row Tetris) is polynomial. We also prove that the generalization of Tetris to larger k-omino pieces is NP-complete even when the board starts empty, and even when restricted to 3 columns or 2 rows or constant-size pieces. Finally, we present an animated Tetris font.

Keywords: complexity, hardness, video games, font

1. Introduction

Tetris is among the best-selling [9], and perhaps best-known, video games ever. Since its invention by Alexey Pajitnov 35 years ago in 1984, over 80 versions have been developed on nearly every platform [10]. Perhaps most famous is the Nintendo Game Boy edition, which was bundled with the Game Boy in the USA, resulting in 35 million copies sold [5]. The most recent editions — Tetris Effect for PS4 and PC including VR (2018) and Tetris 99 for Nintendo Switch (2019) — prove Tetris's sustained popularity.

In standard Tetris, tetromino pieces (□, ▪, ▪, ▪, ▪, ▪, ▪) are chosen at (pseudo)random and fall from the top of a 10-wide 20-tall rectangular playfield. While 10 is the typical width of most Tetris implementations, the height varies between 16 and 24, and some editions change the width to anywhere between 6 and 20 [6]. The player can rotate each piece by $\pm 90°$ and/or slide it left/right as it falls down, until the piece "hits" another piece beneath it and the piece freezes. If any rows are completely full, they get removed (shifting higher rows down), and then the next piece starts falling from the top.

To make this game easier to analyze from a computational complexity perspective, the *perfect-information* Tetris *problem* [2] asks, given an initial board state of filled cells and a sequence of pieces that will arrive, whether the pieces can be played in sequence to either survive (not go above the top row) or clear the entire board. (See Section 2 for precise game rules.) This problem was proved NP-hard for arbitrary board sizes in 2002 [2], and more recently for the generalization to k-omino pieces for various k [3].

Our results. In this paper, we analyze the following special cases of Tetris; refer to Table 1.

(1) c-column Tetris, where the playfield has exactly c columns (and an arbitrary number of rows). The original Tetris paper [2] asked specifically about the complexity of c-column Tetris for $c = O(1)$, motivated by standard Tetris where $c = 10$.

In Section 4, we prove that it is NP-complete to survive or clear the board in c-column Tetris for any $c \geq 8$. This result includes the width of most Tetris variants, including the already small Tetris Jr. ($c = 8$), but excludes one variant, Tetris Wristwatch ($c = 6$) [6]. As an extra feature, this result immediately implies NP-hardness of Tetris where the player can make only a bounded number of moves between each unit piece fall ("bounded reaction speed").

Complementarily, in Section 3, we prove that c-column Tetris can be solved in polynomial time for $c \leq 2$. The case $c = 2$ was claimed without proof in the conclusion of [2]; we provide the first written proof, and generalize to 2-column $O(n)$-tris, by reducing to non-deterministic pushdown automata. The critical hardness threshold for c is thus between 3 and 8.

(2) r-row Tetris, where the playfield has exactly r rows (and an arbitrary number of columns). The original Tetris paper [2] also asked about the complexity of r-row Tetris for $r = O(1)$.

In Section 5, we prove that it is NP-complete to sur-

[1]   MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA
[a)]   sualeh@mit.edu
[b)]   coulombe@mit.edu
[c)]   edemaine@mit.edu
[d)]   mdemaine@mit.edu
[e)]   achesterberg@gmail.com
[f)]   jaysonl@mit.edu
[g)]   mihirs@mit.edu

vive or clear the board in r-ROW TETRIS for any $r \geq 4$.

Complementarily, we observe the trivial result that r-ROW TETRIS can be solved in polynomial time for $r = 1$. The critical hardness threshold for r is thus between 2 and 4.

Both the $O(1)$-row and $O(1)$-column NP-hardness results are based on more efficient packings of the "buckets" in the original reduction from 3-PARTITION [2]. While they share the main idea with the original proof, they require substantial care in how they provide a corridor that can reach all of the buckets without allowing unintended solutions. In particular, we prove NP-hardness of Tetris survival for the first time with even-width boards (e.g., $c = 8$ columns); the previous "reservoir" approach [2], Section 4.2 required an odd number of columns.

(3) EMPTY TETRIS, where the playfield starts empty instead of having a specified configuration. The original Tetris paper [2] highlighted the complexity of this variant as a "major open question", as all existing Tetris hardness proofs (including those in this paper) rely on a high-complexity initial configuration.

In Section 6, we solve this problem for the generalization of TETRIS to k-omino pieces, denoted k-TRIS, as implemented in the video games *Pentris* and *ntris*, and previously analyzed from a complexity perspective [3]. Specifically, we prove the following results:

(a) 8-COLUMN EMPTY (≤ 65)-TRIS is NP-hard.

(b) 3-COLUMN EMPTY $O(n)$-TRIS is NP-hard. This result is tight against our polynomial-time algorithm for 2-COLUMN $O(n)$-TRIS mentioned above.

(c) 2-ROW EMPTY $O(n)$-TRIS is NP-hard.

In Section 7, we present a Tetris font where each letter is made from exactly one copy of each tetromino piece. The font has several variants, including a puzzle font and an animated font, demonstrated in a companion web app.[*1]

2.   Rules of Tetris

We give a brief summary of the rules of Tetris and its generalization k-TRIS, referring the reader to [2], Section 2 and [3], Section 2 respectively for a complete description of the rules. There are in fact many real-world variations of the rules, eventually formalized by The Tetris Company into a modern rule set [1]. The rules we give here are consistent with some, but not all, implementations of Tetris, as detailed below.

Tetris consists of a rectangular *board* or *playfield* [6], which is rectangular in shape. Each cell is *filled* or *unfilled*. In the initial state and after each move, no row is completely filled.

In TETRIS, there are seven tetromino piece types (distinguished by reflection), labeled with letters that resemble their shape: (O), (J), (L), (S), (Z), (T), (I). In k-TRIS, the piece types consist of all k-ominoes, i.e., all connected shapes made by k unit squares

joined along edges. We also define $\leq k$-TRIS, where the piece types are all polyominoes made from $\leq k$ unit squares. In a game instance, n pieces arrive in a fixed order $p_1, p_2, \ldots, p_n$. Each piece p_i falls, starting above the top row of the board, and the player can rotate by $\pm 90°$, and translate left and/or right, as the piece drops down one unit at a time. When the piece tries to drop but would collide with another piece, then it stops moving ("locks down"). If a locked-down piece extends above the top row of the board, then the player immediately loses the game. This rule is called *partial lock out* [8], and applies to older versions of Tetris (e.g., Atari and NES), though modern Tetris rules [1] end the game only when a locked-down piece is entirely above the board; see Section 8. Finally, if any row is now entirely filled, then that row gets removed, and all rows above shift down, creating one new empty row at the top.

Like [2], Section 2, we allow any model of piece rotation that satisfies two "reasonable" restrictions. First, a piece cannot "jump" from one connected component of the unfilled space to another. Second, any piece that is not $1 \times k$ (in particular, not) cannot "squeeze through" a single-cell choke point. Precisely, if cell $\langle i, j \rangle$ is unfilled and either $\langle i \pm 1, j \rangle$ are both filled or $\langle i, j \pm 1 \rangle$ are both filled, and the filling of cell $\langle i, j \rangle$ would partition the unfilled space into two connected components, then a piece that is not $1 \times k$ cannot jump between these two connected components. This model is a slight strengthening of the one in [2], which only considered the $\langle i, j \pm 1 \rangle$ case because it only had to consider vertical passage; in our $O(1)$-row proof, we need to also consider horizontal passage. Notably, this rotation model includes the Classic Rotation System where each piece rotates about a center interior to the piece, and the operation fails if that rotation would overlap a filled square [1]; and it includes the more-complex Super Rotation System (SRS) that is now standard to Tetris [1], [7], which adds a series of possible translation checks that attempt to avoid collisions via "wall kicks".

The $(\leq)k$-TRIS problem is the following decision problem: given a starting configuration of filled cells, and the sequence $p_1, p_2, \ldots, p_n$ of $(\leq)k$-omino pieces that will arrive, can the player maneuver the pieces to avoid pieces freezing above the top row, and optionally, reach a state where the entire board is unfilled? TETRIS is the special case 4-TRIS. These problems are trivially in NP: a certificate is the sequence of player moves (a linear number of translations and/or rotations) between each unit piece drop. As a result, so are the following special cases considered in this paper:

(1) c-COLUMN $(\leq)k$-TRIS, where the board has c columns.

(2) r-ROW $(\leq)k$-TRIS, where the board has r rows.

(3) EMPTY $(\leq)k$-TRIS, where the board's initial configuration is entirely unfilled.

3.   2-column Tetris is polynomial

For completeness, we start with an easy result about one

Rows	Columns	Empty?	Piece Sizes	Complexity	Reference
1	$O(n)$	no	$O(n)$	strongly NP-hard	Proposition 5.13
1	$O(n)$	yes	$O(n)$	linear	Proposition 6.4
1	$O(n)$	no	k	linear	Proposition 3.1
2	$O(n)$	yes	$O(n)$	strongly NP-hard	Theorem 6.3
3	$O(n)$	no	4	OPEN	
4	$O(n)$	no	4	strongly NP-hard	Theorem 5.1
$O(n)$	1	no	$O(n)$	linear	Proposition 3.1
$O(n)$	2	no	$O(n)$	polynomial	Theorem 3.2
$O(n)$	3	yes	$O(n)$	strongly NP-hard	Theorem 6.2
$O(n)$	3–7	no	4	OPEN	
$O(n)$	8+	no	4	strongly NP-hard	Theorem 4.1
$O(n)$	8	yes	≤ 65	strongly NP-hard	Theorem 6.1

Table 1: Summary of our results. Each row of the table specifies the supported board size (numbers of rows and columns), whether the board starts empty or from an adversarial position (empty is stronger for hardness, while nonempty is stronger for algorithms), the allowed piece polyomino sizes (4 for Tetris), the complexity result for this case (red for hardness, blue for algorithm, and yellow for open), and where we prove the result.

row or column:

Proposition 3.1. 1-COLUMN $\leq k$-TRIS and 1-ROW k-TRIS are solvable in linear time.

Proof. In 1-COLUMN $\leq k$-TRIS, only $1 \times j$ pieces (e.g., ▦▦▦▦) are valid. (Any other piece does not fit in the board.) Every such piece immediately fills any rows it occupies, so immediately disappears. Thus every sequence of pieces that fit in the board is a trivial win for the player.

In 1-ROW k-TRIS, any piece that is not $1 \times k$ results in an immediate loss for the player. If there are only $1 \times k$ pieces in the sequence, then we can follow a greedy strategy: place each piece in the leftmost position where it fits. If there is any way to clear the initial row configuration, then this algorithm will produce one. If there is any way to fill a then-empty row (i.e., k divides the board width), then we claim that this algorithm will produce one. Furthermore, if the row cannot be cleared, then we claim that this algorithm will make the most moves possible before getting stuck. These claims follow from a simple greedy argument: take any strategy, sort its piece placements between line clears from left to right, and if a piece placement is not maximally left, shift it so. Thus, this algorithm will play for as long as is possible. □

Theorem 3.2. 2-COLUMN $O(n)$-TRIS is solvable in polynomial time.

Proof. We reduce to A_{PDA}, the acceptance problem for nondeterministic pushdown automata, which is known to be in P by reduction to the acceptance problem for CNF Context-Free Grammars [4] and the CYK algorithm that solves it [11]. Given a set Σ of $\leq k$-omino pieces, a piece sequence p of length n over Σ, the initial board configuration B, and board height h, we output (M, p), where M is a PDA that recognizes piece sequences from Σ that permit staying under h rows starting from board B, and optionally clearing the board.

The constructed pushdown automaton M represents the board state in its stack, with the topmost occupied row at the top. Its stack alphabet $\Gamma = \{$■□, □■$\}$ represents the two possible configurations of a 2-column row (as ■■ is invalid and □□ is invalid below the topmost occupied row). Each piece $p_i \in \Sigma$ is described by a string over $\{$■□, □■, ■■$\}$, as wider pieces cannot fit within two columns.

When a piece p_i comes in, it would suffice for M to pop and observe the top $|p_i| \leq k$ rows to determine how each placement choice changes the board state. Because a move can either delete or add up to k occupied rows at the top of the board, at most $2k$ new rows need to be pushed to realize the outcome of a chosen placement. This locality of updates may not seem true for $1 \times j$ pieces with pattern (□■)* like the ▦▦▦▦ tetromino, because they can descend arbitrarily far below the topmost row; but observe that each row it passes through and clears must be identical, either all ■□ or all □■, so the resulting board state would be the same as if the top j rows were cleared instead.

However, while this implementation would be sufficient, it will not be efficient for large k: there are 2^k possible sequences of k rows, so if M always popped k rows into its finite state space, we would have exponential blowup. We can fix this issue by noticing that there are only two possible row patterns that M needs to handle, (■□)*(□■)? and its mirror image (□■)*(■□)?, as (by the reasonable rotation assumption) no piece can pass through two unequal rows or affect any rows below them. This reduces the number of possibilities to $O(k)$, keeping M's state space small. Figs. 1, 2, and 3 show how to handle the standard tetromino pieces.

The execution of M starts by pushing B onto the stack before reading any input, initializing the board. For each piece p_i, M nondeterministically chooses which orientation to place p_i on the board. As it runs, M enforces the maximum row constraint by keeping a stack height counter in its state, which is incremented and decremented appropri-
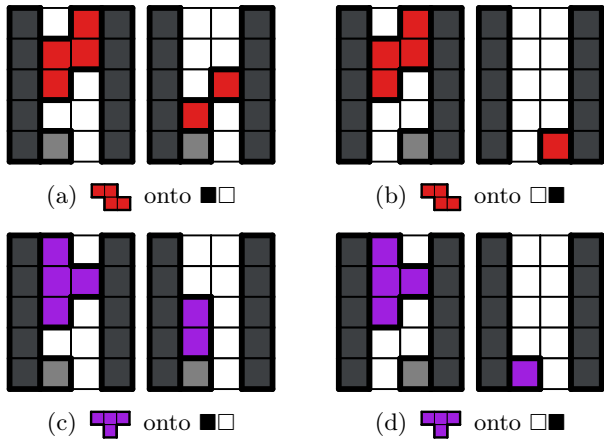
Fig. 1: Possible 2-column outcomes for placing 🟥 or 🟥 pieces. 🟩 and flipped 🟪 cases are symmetric.

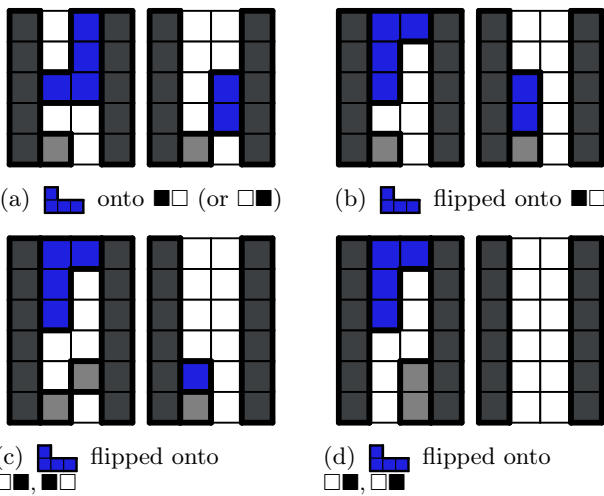Fig. 2: Possible 2-column outcomes for placing 🟦 pieces. 🟧 cases are symmetric.

Fig. 3: Possible 2-column outcomes for placing 🟨 and 🟦 pieces. Mirrored cases are symmetric.

ately as M pushes and pops rows. If M places a piece over the height limit h, then M enters an inescapable rejecting state on that branch, whereas all other states (or optionally only those with an empty stack) are accepting states. With these finite-state implementable rules, M performs a correct simulation which recognizes winning games, therefore (Σ, p, B, h) is in 2-column $\leq k$-tris if and only if (M, p) is in A_{PDA}.

To bound the running time, note that we may assume $h \leq |B| + kn$, as otherwise it would be impossible to reach the top row and lose. Because producing M is dominated by the computation of its transition function — enumerating and simulating the outcomes of the $O(|\Sigma| \times h \times k)$ scenarios of placing every piece in each orientation at all legal heights with every pattern of the top k rows — the size of M and the time required to produce it is polynomial in the input size. □

4. 8-column Tetris is NP-hard

In this section, we prove the following theorem:

Theorem 4.1. *It is NP-complete to survive or clear the*

board in c-column Tetris *for any $c \geq 8$.*

Like [2], we reduce from the strongly NP-hard 3-Partition problem.

Definition 4.2. The **3-Partition** problem is defined as follows:

Input: A set of nonnegative integers $\{a_1, a_2, \ldots, a_{3s}\}$ and a nonnegative integer T satisfying the constraints $\sum_{i=1}^{3s} a_i = sT$ and $\frac{T}{4} < a_i < \frac{T}{2}$ for all $1 \leq i \leq 3s$.

Output: Whether $\{a_1, a_2, \ldots, a_{3s}\}$ can be partitioned into s (disjoint) sets of size 3, each of which sum to exactly T.

For the reduction, we exhibit a mapping from 3-Partition instances to Tetris instances so that the following is satisfied:

Lemma 4.3 (c-column Tetris $\iff$ 3-Partition). *For a "yes" instance of 3-Partition, there is a way to drop the pieces that clears the entire board without triggering a loss. Conversely, if the board can be cleared, then the 3-Partition instance has a solution.*

Proof sketch. The initial board, illustrated in Fig. 4a (where filled cells are grey and the rest of the cells are un-

filled), has $12sT+48s+17$ rows. The reduction is polynomial size.

The piece sequence is as follows. First, for each a_i, we send the following a_i sequence (see Figs. 4(i–m)):

$$\langle \; \blacksquare, \langle \; \blacksquare, \; \blacksquare, \; \blacksquare \rangle^{a_i}, \; \blacksquare, \; \blacksquare \rangle.$$

After all these pieces, we send the following *clearing sequence* (see Figs. 4(n) and (b–h)):

$$\langle\langle \; \blacksquare, \; \blacksquare, \; \blacksquare \rangle^{s}, \; \blacksquare, \langle \; \blacksquare \rangle^{6sT+24s+6},$$
$$\blacksquare, \; \blacksquare, \langle \; \blacksquare \rangle^{3sT+12s+4}\rangle.$$

Figs. 4(b–n) illustrate that a solution to 3-Partition clears the Tetris board. To show the other direction, we progressively constrain any Tetris solution to a form that directly encodes a 3-Partition solution. Because the area of the pieces sent is exactly equal to $4(12sT + 48s + 13)$, no cell can be left empty. We enumerate all possible cases to show that this goal is impossible to meet (some cell must be left empty) if there is no 3-Partition solution. Figs. 4(o–w) show some of the cases. □

4.1 Reduction

In this section, we detail our polynomial reduction from an instance $\mathcal{P} = \langle \{a_1, a_2, \ldots, a_{3s}\}, T\rangle$ of 3-Partition to an instance $\mathcal{G} = \mathcal{G}(\mathcal{P})$ of Tetris. In later sections, we prove that $\mathcal{P}$ and $\mathcal{G}$ have the same answer, i.e., there exists a valid 3-partition if and only if there is a sequence of moves that survives or that clears the Tetris board.

4.1.1 Initial board

The initial board, illustrated in Fig. 4(a) (where filled cells are grey and the rest of the cells are unfilled), has 8 columns and $12sT+48s+17$ rows. The columns are numbered 1 to 8 from left to right. (To prove hardness for $c > 8$ columns, we simply fill all columns beyond the 8th.) The unfilled cells consist of five main parts:

- The *corridor*, which consists of a $2 \times (12sT + 48s + 12)$ rectangle, as well as a 3-square cut out on the bottom left.

- The *buckets* (of which there are s), which branch off the corridor to its left. These are similar in shape to the buckets used in [2]: except for the first few and last few rows, their shape is periodic with a period of 5 rows. Each bucket has a total height of $5T + 20$, and contains $T+3$ *notches* (the pairs of adjacent empty cells in column 5, not including the 3 empty cells at the top).

- The *T-lock*, which is in the shape of a $\blacksquare$ piece. The buckets are each separated by two rows.

- The *alley*, which is a $1 \times (12sT + 48s + 16)$ rectangle which is "unlocked" by the T-lock.

We also define the *horizon*, which is the horizontal line separating the empty rows at the top of the board from the topmost filled cell. When a row is cleared, the horizon

moves down by one row per row that is cleared. We also define the *bucket line* as the horizontal line immediately below the bottommost cell of the bottommost bucket.

4.1.2 Piece sequence

The pieces arrive in the following order.

First, for each a_i, the following pieces arrive in the following order, called the a_i-*sequence*. This part is identical to the reduction in [2].

- The *initiator*, which is a single $\blacksquare$ piece.

- The *filler*, which consists of the pieces $\langle \; \blacksquare, \; \blacksquare, \; \blacksquare \rangle^{a_i}$ (which we take to mean the sequence $\langle \; \blacksquare, \; \blacksquare, \; \blacksquare \rangle$ repeated a_i times).

- The *terminator*, which consists of the pieces $\langle \; \blacksquare, \; \blacksquare \rangle$.

After all the a_i-sequences, we then have the following pieces in the following order, called the *closing sequence*:

- The *bucket closer*, which consists of the pieces $\langle \; \blacksquare, \; \blacksquare, \; \blacksquare \rangle^{s}$.

- A single $\blacksquare$ (note this is the first $\blacksquare$ to arrive).

- The *corridor closer*, which consists of the pieces $\langle\langle \; \blacksquare \rangle^{6sT+24s+6}, \; \blacksquare \rangle$.

- A single $\blacksquare$ (note this is the first $\blacksquare$ to arrive).

- The *clearer*, which consists of the pieces $\langle \; \blacksquare \rangle^{3sT+12s+4}$.

The total size of the board is $8(12sT + 48s + 17)$ and the total number of pieces is

$$\sum_{i=1}^{3s}(3 + 3a_i) + 3s + 1 + (6sT + 24s + 6)$$
$$+ 1 + 1 + (3sT + 12s + 4) = 12sT + 48s + 13,$$

which are both polynomial in the size of the 3-Partition instance.

4.2 3-Partition solvable ⇒ Tetris solvable

In this section, we show one side of the bijection: for a "yes" instance of 3-Partition, we can clear the game board.

Theorem 4.4. *For a "yes" instance of* 3-Partition, *there is a trajectory sequence* Σ *that clears the entire gameboard of* $\mathcal{G}(\mathcal{P})$ *without triggering a loss.*

Proof. Since $\mathcal{P}$ is a "yes" instance, there is a partitioning of $\{1, 2, \ldots, 3s\}$ into sets $A_1, A_2, \ldots, A_s$ so that $\sum_{i \in A_j} a_i = T$. We have ensured that $|A_j| = 3$ for all j. All pieces associated with set $A_j = \{x, y, z\}$ should be placed into the jth bucket of the gameboard.

We place the a_x-sequence into bucket j as in Figs. 4(i–l). After all pieces associated with the number a_x have been placed into bucket j, the bucket has a_x+1 fewer notches, but otherwise still has the shape of a bucket. Similarly, a_y, a_z are
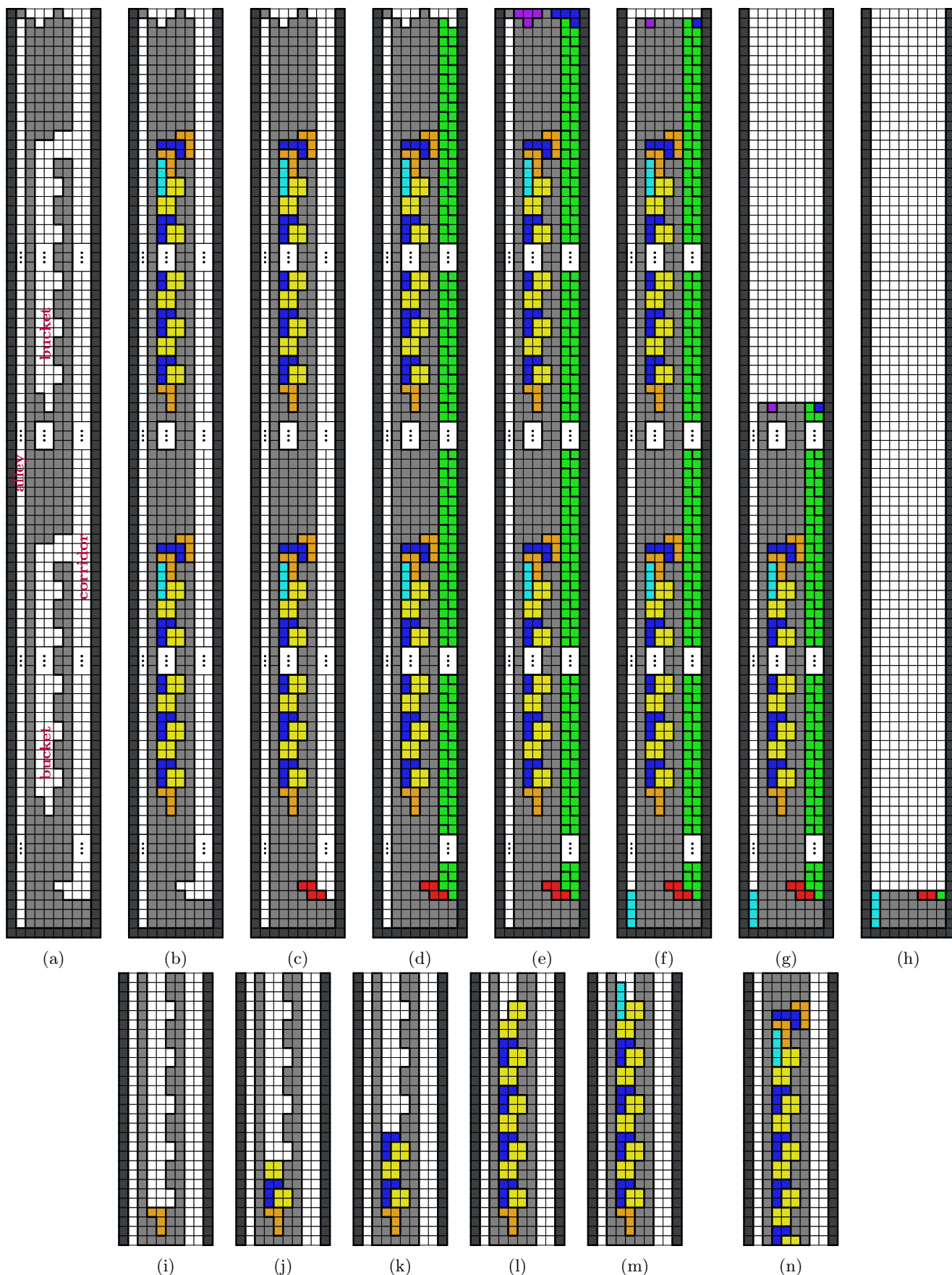
Fig. 4: (a) shows the initial board. (b–h) demonstrate filling and clearing the board in the final clearing sequence. (i–m) show a valid sequence of moves for $a_i = 5$. (n) shows our bucket terminator.

placed in bucket j, for a total of $(a_x+1)+(a_y+1)+(a_z+1) = T + 3$ notches being filled, so each bucket has 0 notches left and may then be filled by the bucket closer, as in Fig. 4(n). After all the a_i-sequences arrive, then, we fill all the buckets

with the bucket closer.

Next, now that the buckets are all filled, the remaining moves are straightforward. We drop a ⬛ into the corridor to fill the bottom 4 cells of the corridor. Now we use the $6sT + 24s + 6$ ⬛s in the corridor closer to fill the corridor, as depicted in Fig. 4d. We then drop the ⬛ into the T-lock, which fills the top row immediately below the horizon and clears it, opening the alley. Now we drop a sequence of $3sT + 12s + 4$ ⬛s which clear all the rows of the board since they clear $4(3sT + 12s + 4) = 12sT + 48s + 16$ rows, as shown in Figs. 4(f–h). This clears the whole board, as desired. □

4.3   Tetris solvable ⇒ 3-Partition solvable

Here we show that if $\mathcal{G}(\mathcal{P})$ has a sequence of moves that survive, then the 3-Partition instance $\mathcal{P}$ must also have a solution (i.e., a valid partition). Suppose there is such a surviving sequence of moves. By a sequence of claims, we progressively constrain this survival strategy into a form that directly encodes a 3-Partition solution.

Claim 4.5. The top row must be the first row to be cleared.

Proof. Every row except the top has an empty square in the alley. The alley is completely surrounded by pieces, and thus no part of the alley can be filled until a row is filled. □

Claim 4.6. Only a ⬛ can go in the T-lock.

Proof. By the previous claim, no rows can be cleared before the top row, and thus the T-lock will remain at the top of the board until it is filled. There are only four empty cells in the connected component of the T-lock within the board, so any piece placed other than a ⬛ will have at least one block above the board, causing a loss by partial lock out. □

Claim 4.7. No rows can be cleared before the ⬛ is given.

Proof. Follows from the prior two claims. □

Claim 4.8. All squares not in the alley and T-lock must be completely filled before the ⬛ arrives.

Proof. Cells in the alley and cells above the horizon cannot be filled before the ⬛ arrives by Claim 4.7. Cells in the T-lock cannot be filled before the ⬛ arrives by Claim 4.6. The total number of empty cells outside those two areas is $48sT + 192s + 52 = 4(12sT + 48s + 13)$, which is exactly four times the number of pieces that arrive before the ⬛. Each piece fills four cells, so every cell not in the alley and T-lock must be filled; otherwise, some piece will extend above the board, causing a loss by partial lock out. □

By Claim 4.8, the surviving trajectory sequence cannot leave any unfillable holes behind when placing pieces. Henceforth, we focus on the prefix of the surviving trajectory sequence that places the a_i-sequences without such holes; we only need the surviving trajectory sequence for the closing sequence in order to guarantee Claim 4.8.

Claim 4.9. During the a_i-sequences, no piece other than an ⬛ can be placed first in the corridor.

Proof. The casework in Fig. 5 shows that any ⬛, ⬛, ⬛, or ⬛ placed in the bottom of the corridor will leave empty squares, which is not allowed by Claim 4.8. No ⬛, ⬛, or ⬛ arrives during the a_i-sequences, leaving just ⬛.
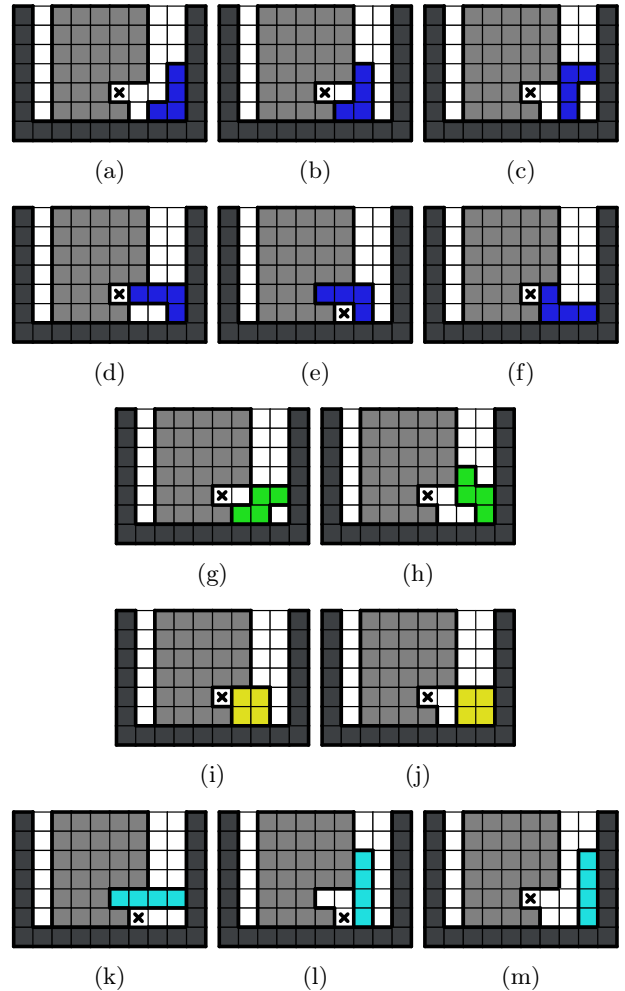
Fig. 5: Possibilities for placing various pieces other than ⬛ and ⬛ into the corridor, all of which make the puzzle unsolvable.

Any piece placed higher up in the corridor creates a choke point of width 1 through which no pieces but ⬛ can pass. Since an ⬛ placed at the bottom of the corridor leaves empty squares, this makes the bottom of the corridor unfillable. □

Next we show that the buckets must be filled in the manner given by Section 4.2. We define prepped and unprepped buckets as in [2]. An *unprepped* bucket is one that takes the form of a bucket in the initial board, but possibly with fewer notches, as shown in Fig. 6(a). The *height* of an unprepped bucket is the number of notches in the bucket; the buckets all initially have height $T + 3$. A special case is an unprepped bucket of height 0; this is also shown in Fig. 6(b).

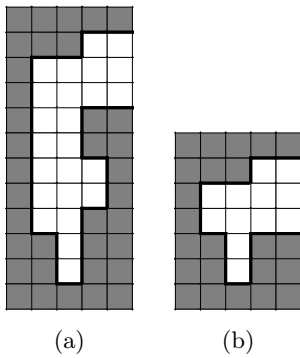We also define a *prepped* bucket as one in which all cells

(a)   (b)

Fig. 6: Unprepped buckets of heights 1 and 0, respectively.
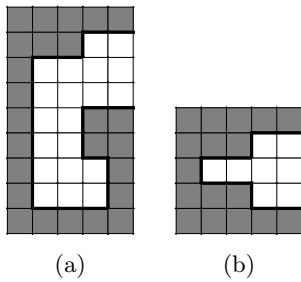

(a)   (b)

Fig. 7: Prepped buckets of heights 1 and 0, respectively.

below some notch are filled, as in Fig. 7(a). The *height* of a prepped bucket is again its number of notches. Again, there is the special case of a prepped bucket of height 0, also shown in Fig. 7(b).

Claim 4.10. None of ▦, ▦▦, ▦▦▦▦ may be placed in an unprepped bucket.

Proof.   Figs. 8 and 9 show all possible placements, and the crosses show cells that cannot be filled. In Fig. 9(q), there are two cells with crosses; these two cells cannot both be filled (noting that there is no ▦▦ in the piece sequence). Thus we have a contradiction by Claim 4.8, since all the crossed cells must be filled eventually.   □

Claim 4.11. If an ▦▦ is placed in an unprepped bucket, it must form a prepped bucket of the same height.

Proof.   We do casework on the possible placements of the ▦▦, showing that in each other case, there is a cell that can never be filled before any row is cleared. This would contradict Claim 4.8.

All possible placements are shown in Figs. 10 and 11, where we have split into cases based on whether or not the bucket has height 0. (The cases where the bucket does not have height 0 are essentially the same even as the height varies). Most cases are marked with a cross, which indicates a cell which can never be filled, making that placement invalid. In the only valid cases, Figs. 10(d) and 11(d), an unprepped bucket of the same height results.   □

Claim 4.12. An ▦ cannot be placed in a prepped bucket of height 0.

Proof.   There is only one possible placement of an ▦ in a prepped bucket of height 0, shown in Fig. 12. The cross

(a)   (b)   (c)

(d)   (e)   (f)   (g)

(h)   (i)   (j)   (k)

(l)   (m)
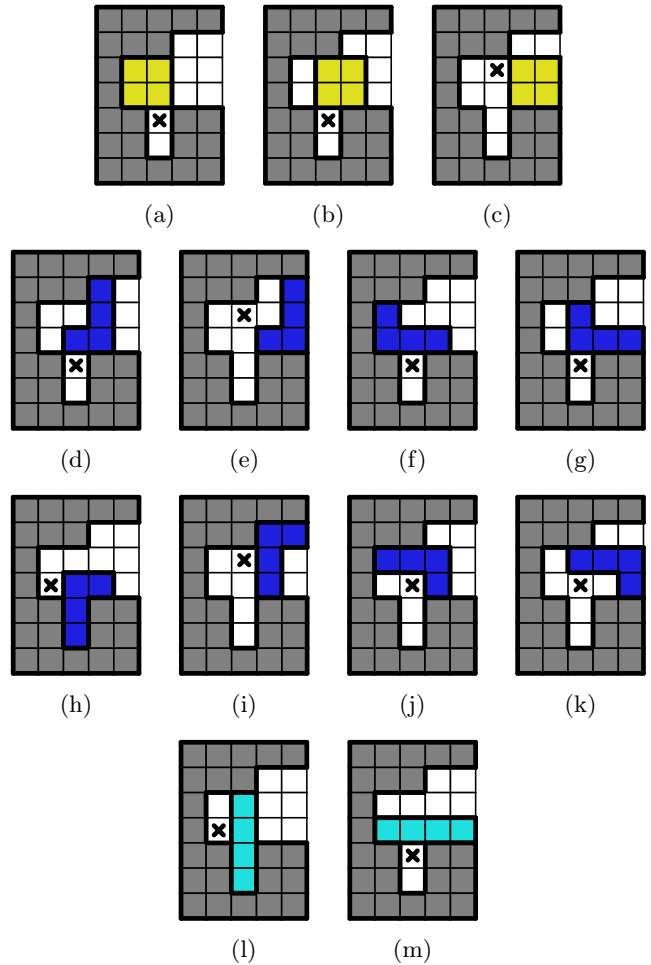
Fig. 8: Possibilities for placing an ▦, ▦▦, or ▦▦▦▦ into an unprepped bucket of height 0. All leave the puzzle unsolvable.

marks an unfillable cell; hence, this is not permissible.   □

Claim 4.13. When the sequence ⟨ ▦, ▦▦, ▦ ⟩ is placed in a prepped bucket of height h, the bucket must end up as an prepped bucket of height $h - 1$. (We know $h \geq 1$ by Claim 4.12.)

*Proof.*   Here we show that each of the parts of the sequence must be placed in a specific way. First, Fig. 13 shows all the ways an ▦ can be placed. Only Fig. 13(b) shows a valid placement.

After the ▦ has been placed as in Fig. 13(b), Fig. 14 shows all the ways the ▦▦ can be placed afterward. Again, only Fig. 14(c) shows a valid placement.

Finally, after the ▦▦ has been placed as in 14(c), Fig. 15 shows all the ways the second ▦ can be placed. In the only valid case, Fig. 15(a), what remains is a bucket of height $h - 1$, as desired.   □

Claim 4.14. When the sequence ⟨ ▦, ▦▦▦▦ ⟩ is placed in a prepped bucket of height h, the bucket must end as an unprepped bucket of height $h - 1$. (We know $h \geq 1$ by Claim 4.12.)
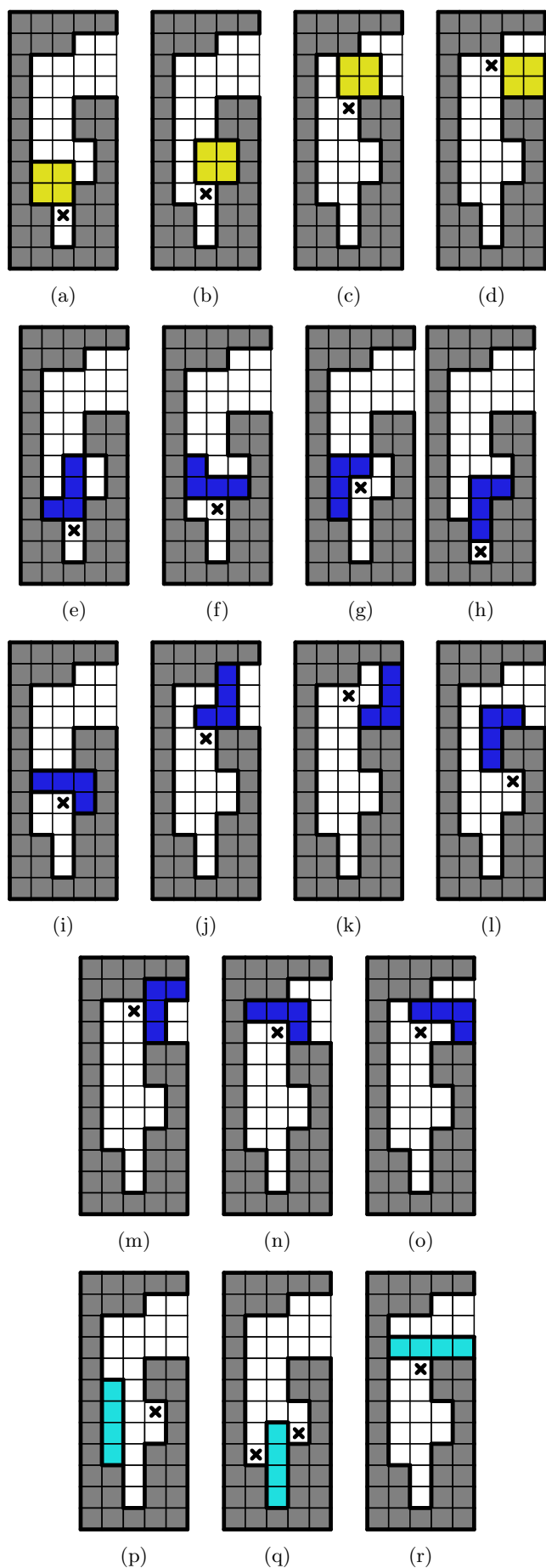
Fig. 9: Possibilities for placing an ▨ , ◼ , or ▭ in an unprepped bucket of positive height. All leave the puzzle unsolvable.
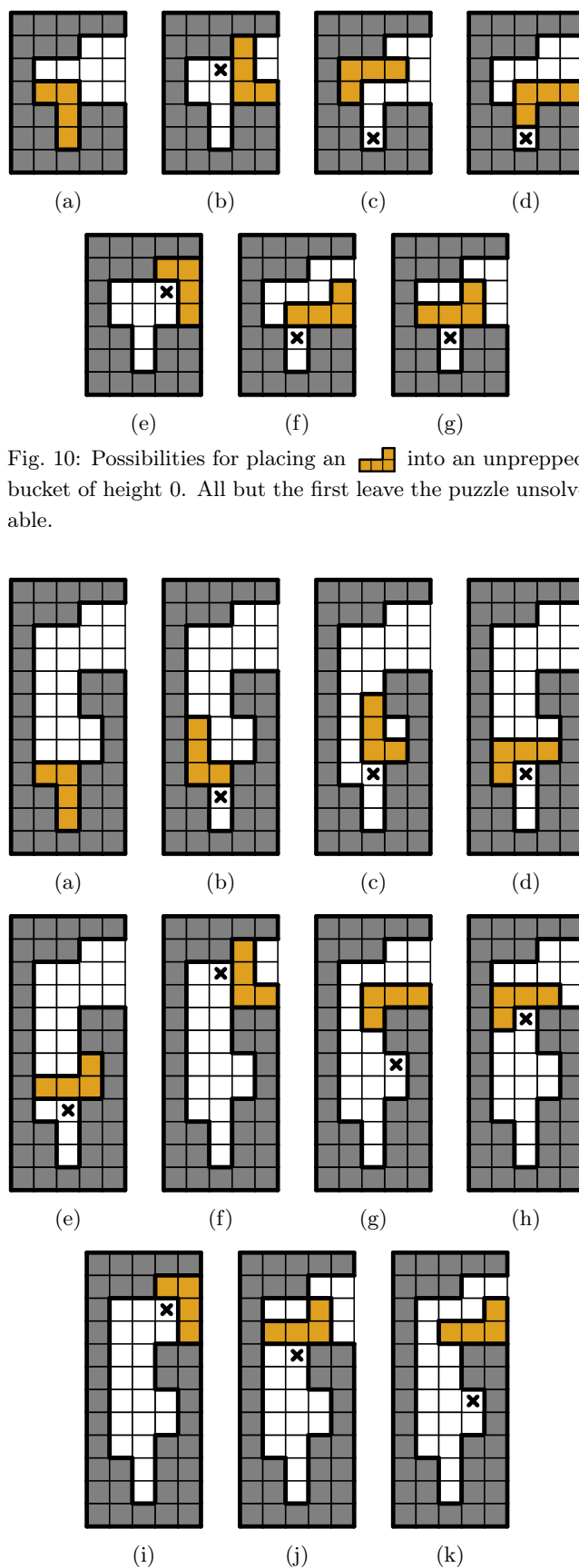

Fig. 10: Possibilities for placing an ▭ into an unprepped bucket of height 0. All but the first leave the puzzle unsolvable.
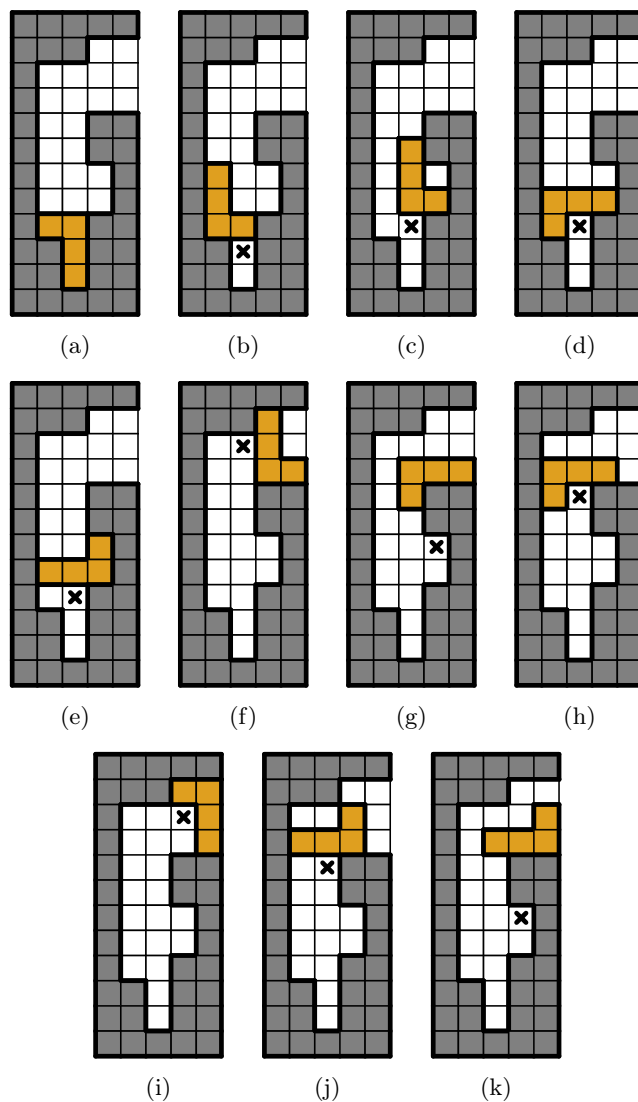

Fig. 11: Possibilities for placing an ▭ into an unprepped bucket of positive height. All but the first leave the puzzle unsolvable.
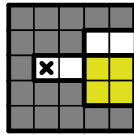
Fig. 12: The only way to place an ⊞ in a prepped bucket of height 0. This leaves the puzzle unsolvable.

(a) (b) (c)

(d) (e)
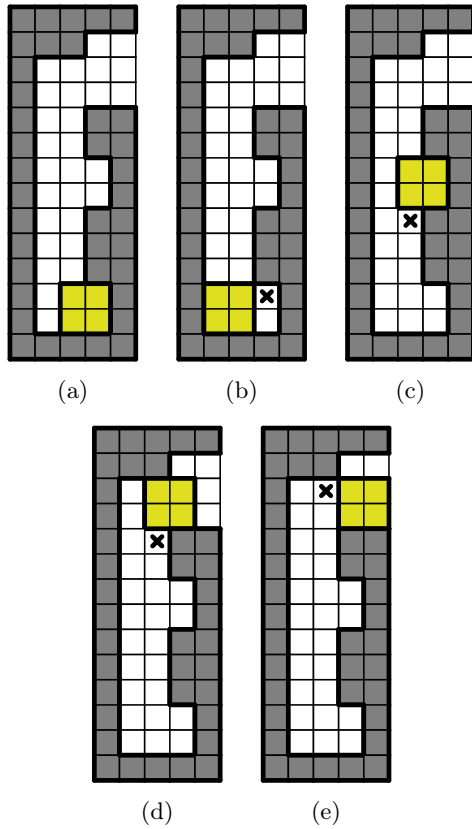
Fig. 13:   Possibilities for placing the first ⊞ in ⟨ ⊞ , ▪▪ , ⊞ ⟩. All but the first leave the puzzle unsolvable.

Proof. By the exact same cases as in Claim 4.13, the ⊞ must be placed as in 13(b).

Now, Fig. 16 shows all possible placements of the ▬▬▬ afterward, where again crosses show unfillable cells. (In this case, it is necessary to split between the $h = 1$ and $h > 1$ cases.) Again, in Fig. 16(e), there are two cells with crosses, which cannot both be filled since there is no ▪▪ in the sequence. In the only valid placements, Figs. 16(a) and 16(d), the result is a prepped bucket of height $h - 1$.   □

The following corollary follows directly from Claims 4.11, 4.10, 4.12, 4.13, and 4.14:

Corollary 4.15. Suppose that before the a_i-sequence arrives, all buckets are unprepped. If the ▪▪ starting the a_i-sequence is placed into a bucket, then the entire a_i-sequence must be placed into that bucket. Furthermore, the height of that bucket decreases by $a_i + 1$; or if the height were to become negative, then these placements are impossible.   □

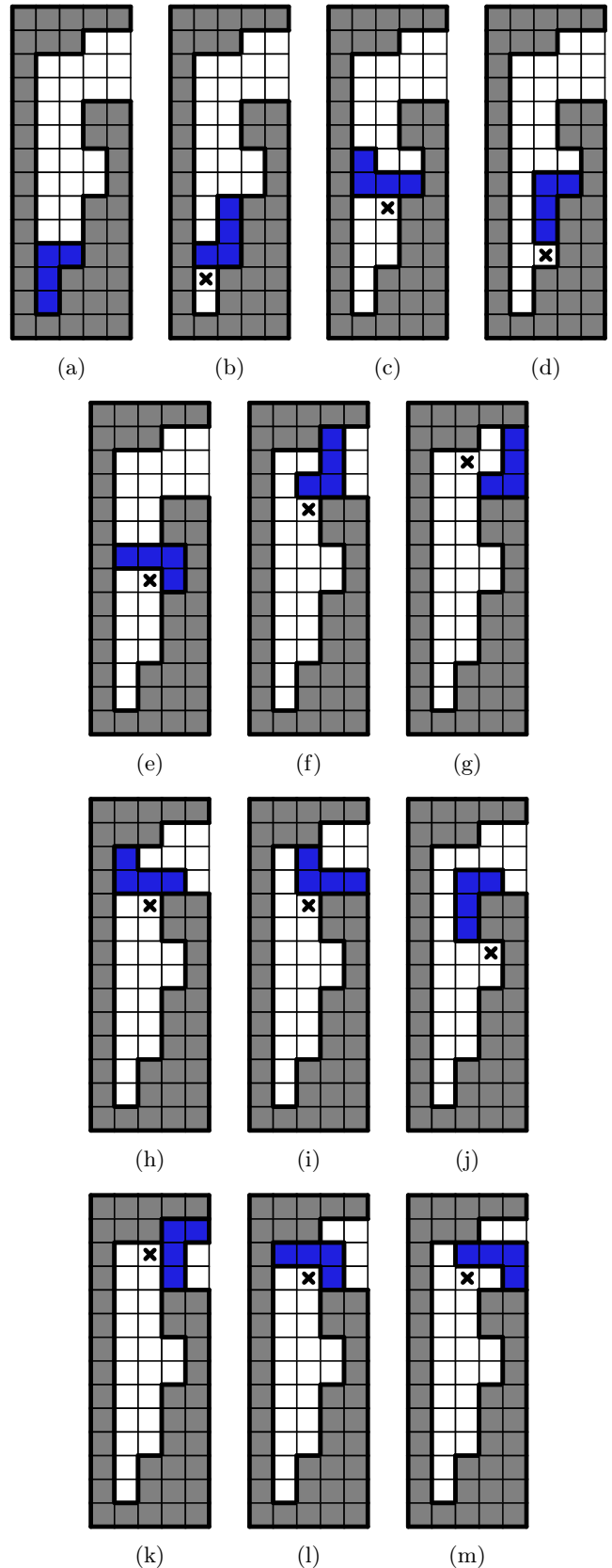We can now remove the assumption that the ▪▪ starting the a_i-sequence is placed into a bucket:

(a) (b) (c) (d)

(e) (f) (g)

(h) (i) (j)

(k) (l) (m)

Fig. 14: Possibilities for placing the ▪▪ in ⟨ ⊞ , ▪▪ , ⊞ ⟩. All but the first leave the puzzle unsolvable.

Claim 4.16. The ▪▪ starting each a_i-sequence must be placed into a bucket.

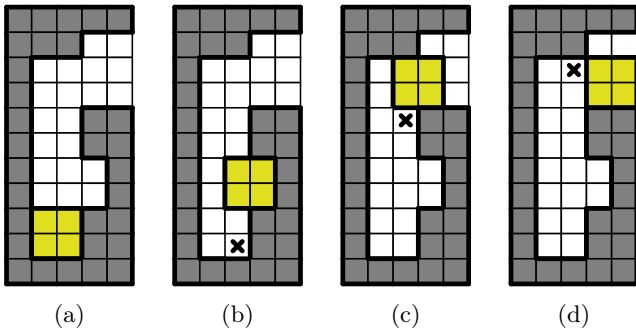*Proof.* By the definition of the a_i-sequence, the ▪▪ is im-

Fig. 15: Possibilities for placing the second ⬛ in ⟨⬛, ⬛, ⬛⟩. All but the first leave the puzzle unsolvable.

Fig. 16: Possibilities for placing the ▭▭. All but the first leave the puzzle unsolvable.

mediately followed by an ⬛. Consider for a contradiction the first ⬛ from an a_i-sequence that is placed into the corridor. We claim that that the following ⬛ cannot be placed. By Claim 4.9 and by induction, no piece other than the just-placed ⬛ has been placed in the corridor. If the ⬛ piece is placed in the corridor, the casework in Fig. 17 shows that it would create an unfillable hole. However, by Corollary 4.17, before each a_i-sequence, all buckets are un-prepped, so by Claim 4.10, the ⬛ also cannot be placed in a bucket, a contradiction.   □

Combining Corollary 4.15 with Claim 4.16, we obtain the following:

Corollary 4.17. Suppose that before the a_i-sequence arrives, all buckets are unprepped. Then the entire a_i-sequence must be placed into that bucket. Furthermore, the height of that bucket decreases by a_i+1; or if the height were to become negative, then these placements are impossible.
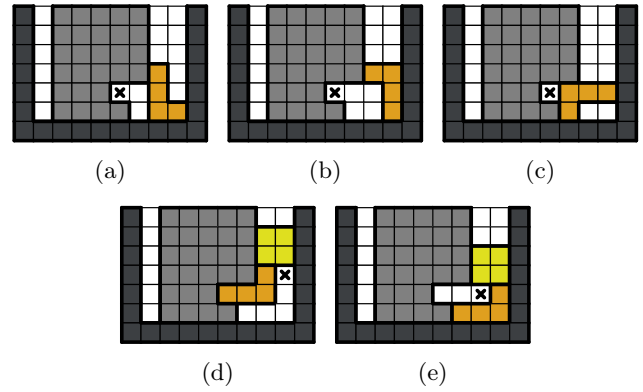

Fig. 17: Possibilities for placing ⬛ into the corridor and then ⬛ into the corridor.

□

We are finally ready to prove the other direction of the bijection.

Theorem 4.18. *If $\mathcal{G}(\mathcal{P})$ has a surviving trajectory sequence, then the* 3-PARTITION *instance $\mathcal{P}$ has a solution.*

Proof. Numbering the buckets $1, 2, \ldots, s$, let S_b be the set of i such that the a_i-sequence is placed in bucket b. By the first half of Corollary 4.17, the S_b's form a partition of $\{a_1, a_2, \ldots, a_{3s}\}$. By the second half of Corollary 4.17, the sum $\sum_{i \in S_b}(a_i + 1)$ is at most the original height of each bucket, which is $T + 3$. However, $\sum_{i=1}^{3s}(a_i + 1) = s(T + 3)$, so equality must hold.

Thus, we have $\sum_{i \in S_b}(a_i + 1) = T + 3$ for each bucket b. But the condition $T/4 < a_i < T/2$ means that this sum cannot have at most 2 or at least 4 terms, so it must have 3 terms, and thus $|S_b| = 3$. Then the condition simplifies to $\sum_{i \in S_b} a_i = T$, and thus the S_b represent a valid 3-partition.   □

5.   4-row Tetris is NP-hard

Theorem 5.1. *It is NP-complete to survive or clear the board in r-ROW TETRIS for any $r \geq 4$.*

The proof by reduction from 3-PARTITION. Given an instance $\mathcal{P}$ of 3-PARTITION with elements $\{a_1, a_2, \ldots, a_{3s}\}$ and target T, we create an instance $\mathcal{G} = \mathcal{G}(\mathcal{P})$ of r-ROW TETRIS which has a valid 3-partition if and only if there is a sequence of moves to survive, and an extension of such a surviving sequence to leave the Tetris board empty.

The initial board, illustrated in Fig. 18 for $r = 6$ (where filled cells are grey and the rest of the cells are unfilled), has $15sT + 8s + 8 + 4r$ columns and r rows, containing the following unfilled cells:

- A number s of *buckets*, which branch off the corridor to its right. These are similar in shape to the buckets used for the proof of c-COLUMN TETRIS except for the first few and last few columns. Each bucket has a total width of $15T + 6$, and contains $3T$ *notches* (the pairs of adjacent empty cells in row 5, counting rows from the top). *Buckets* are separated by 2 columns.
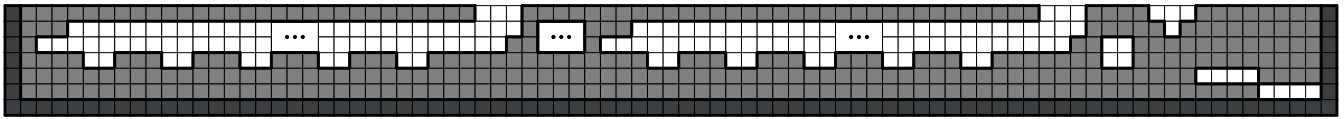
Fig. 18: The initial board for $r = 6$.

(a)

(b)
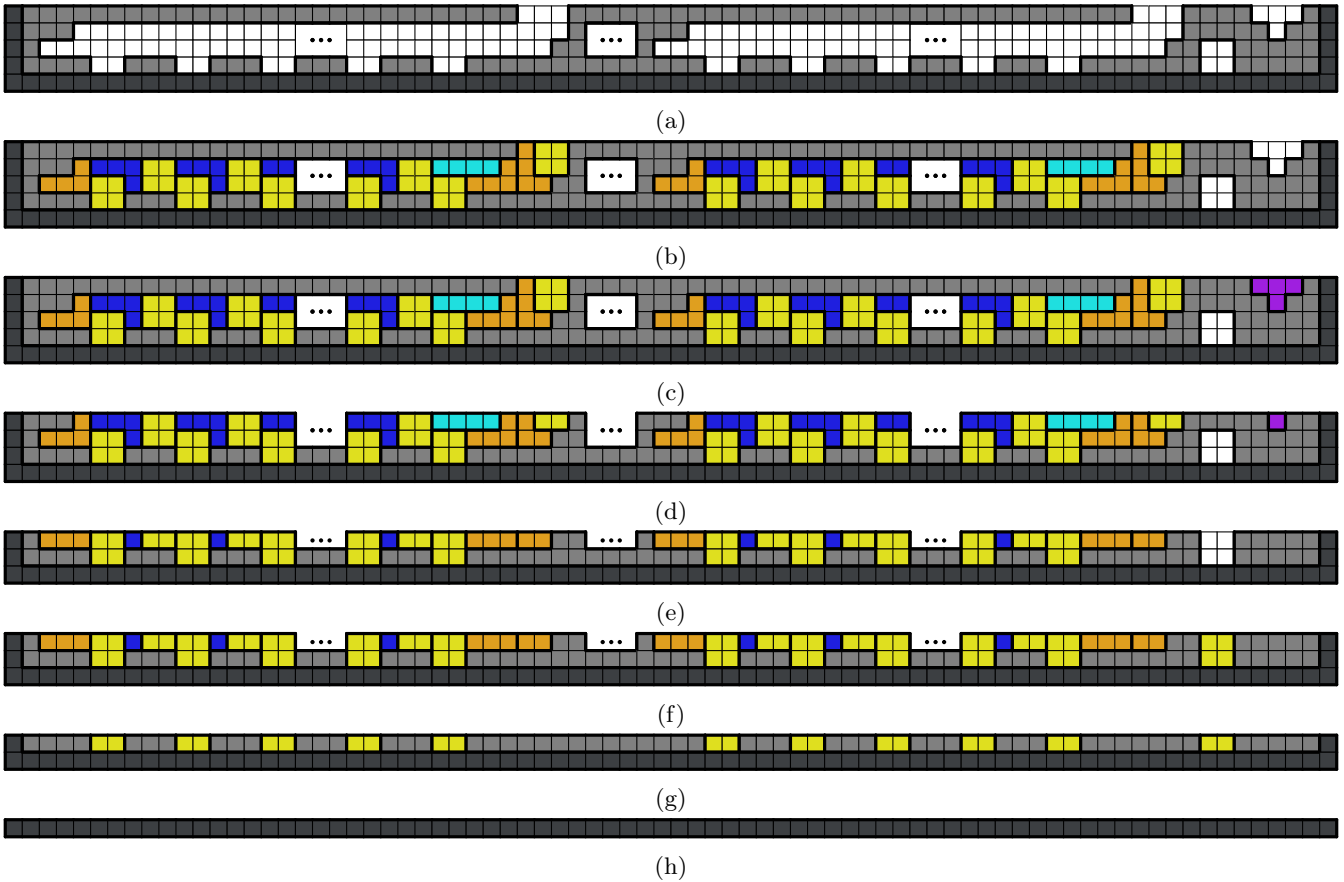
(c)

(d)

(e)

(f)

(g)

(h)

Fig. 19: (a) shows the initial board for $r = 4$. (b) shows a correctly filled board. After (c), there are many ways to survive; (c–h) show the clearing sequence.

- A *T-lock* in the shape of a piece in the top two rows.

- An *O-lock* in the shape of a piece in the next two rows.

- Right filler: in each row below the top four, there are exactly four empty spaces, to the right of any columns empty in any higher row.

The piece sequence is as follows. First, for each a_i, we send the following a_i sequence (see Figs. 4(i–m)): $\langle$, $\langle$, , $\rangle^{3a_i-1}$, , $\rangle$. After all these pieces, we send the following *clearing sequence* (see Figs. 4(n) and (b–h)): $\langle\langle$, , , , , $\rangle^s$, , $\rangle$. Finally, if $r > 4$, we send $\langle$ $\rangle^{r-4}$.

The total size of the board is $r(15sT + 8s + 8 + 4r)$ and the total number of pieces is

$$\sum_{i=1}^{3s} (3 + 3(3a_i - 1)) + 2 + 6s$$

which are both polynomial in the size of the 3-PARTITION instance (recalling that 3-PARTITION is strongly NP-hard).

Lemma 5.2 (r-ROW TETRIS $\iff$ 3-PARTITION). *For a "yes" instance of* 3-PARTITION, *there is a way to drop the pieces that survives and clears the entire board. Conversely, if the piece sequence can be survived, then the* 3-PARTITION *instance has a solution.*

Fig. 19 illustrates that a solution to 3-PARTITION survives and clears the Tetris board.

To show that if $\mathcal{G}(\mathcal{P})$ has a sequence of moves that survives, then the 3-PARTITION instance $\mathcal{P}$ must also have a solution (i.e., a valid partition), we progressively constrain any TETRIS solution to a form that directly encodes a 3-PARTITION solution.

Claim 5.3. Nothing may be placed in the T-lock except a .

Proof. No row can be cleared until some cell of the T-lock is filled and we also note that there is only one in the complete piece sequence. However, the only piece that can fill any cell of the T-lock without filling a cell above the

horizon (causing a loss by partial lock out) is a ⬛, so the claim follows from Claim 5.6. □

Corollary 5.4. Nothing may be placed in the O-lock except an ⬛. □

Proof. The first two rows cannot be cleared until T-lock is filled with an ⬛ by Claim 5.3. This means that no piece may reach the O-lock since it is covered by the first two rows. The only piece that follows the ⬛ is an ⬛ which must go into the O-lock as desired. □

So we have the following corollary which follows directly from Claim 5.4:

Corollary 5.5. No row may be cleared until the first ⬛ has arrived. □

We implicitly use Corollary 5.5 throughout this paper, since it implies that the buckets must maintain their shape until after the ⬛ arrives.

**Claim 5.6.** No cell above the horizon may ever be filled.

Proof. The area of the pieces sent up to the first ⬛ is exactly equal to

$$4(9sT + 6s + 1),$$

the area of the unfilled cells in the first four rows outside the ⬛ lock is

$$s(36T + 24) + 4 = 4(9sT + 6s + 1),$$

so when the first ⬛ arrives, every cell in the first four rows outside the ⬛ lock must be full for survival, and no cell can ever be placed in an empty row. □

If we *survive* until the first ⬛ by filling all cells in the first four rows except the ⬛ lock, then the remaining ⬛ and ⬛ pieces *clear* the board. Hence it suffices to show that for $r = 4$, the given Tetris game is clearable if and only if the instance of 3-Partition is solvable. Henceforth we assume $r = 4$.

Now, to complete the proof, we show that the buckets must be filled in the manner shown in Fig. 19, so some cell must be left empty if there is no 3-Partition solution. Define *prepped* (Fig. 21) and *unprepped* (Fig. 20) buckets as in Section 4.3. We define the height in the same manner by the number of notches.
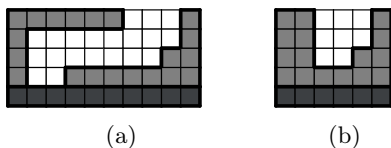

(a)   (b)

Fig. 20: Prepped buckets of heights 1 and 0, respectively.

We can now prove an analogue of Claim 4.11:

Claim 5.7. If an ⬛ is placed in an unprepped bucket of height at least 1, it must form a prepped bucket of the same height.

Proof. In Fig. 22, we show all cases that do not leave the leftmost cell disconnected from the outside. The first one

(a)   (b)
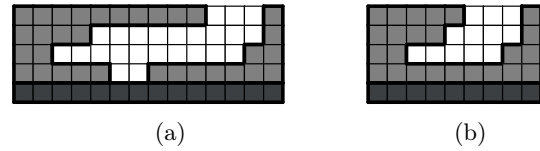
Fig. 21: Unprepped buckets of heights 1 and 0, respectively.

works; in the others, we attempt to place an ⬛ incorrectly, and mark some cell that can never thereafter be filled. □

(a)   (b)

(c)   (d)
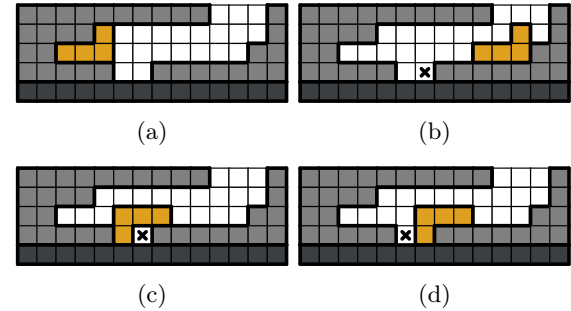
Fig. 22: Ways to place an ⬛ into an unprepped bucket of positive height. Cases in which the leftmost empty cell is neither filled nor connected by a path of empty cells to the outside are not shown.

Claim 5.8. None of ⬛, ⬛, ⬛ may be placed in an unprepped bucket of height at least 1.

Proof. In Fig. 23, we show all possible cases. In every one, we attempt to place an ⬛ incorrectly, and mark some set of cells (usually exactly one) that can never thereafter be simultaneously filled. For instance, in Fig. 23(p), the only piece (of the ones we ever use) that can fill the bottom marked cell is ⬛, but once a ⬛ is placed, no piece can fill the top marked cell. □

Proof. There are only eight empty cells in an unprepped bucket of height 0, less than the 12 needed. □

Claim 5.9. When the sequence ⟨⬛, ⬛, ⬛⟩ is placed in a prepped bucket of height $h \geq 2$, the bucket must end up as a prepped bucket of height $h - 1$.

Proof. In Fig. 24, we show all possible cases. In every one, we attempt to place a ⬛, ⬛, ⬛. The first one works. For every other one, we show some cell that cannot be filled by the next piece. □

Claim 5.10. When the sequence ⟨⬛, ⬛⟩ is placed in a prepped bucket of height $h \geq 2$, the bucket must end as an unprepped bucket of height $h - 1$.

Proof. In Fig. 25, we show all possible cases. In every one, we attempt to place a ⬛, ⬛. The first one works. For every other one, we show some cell that cannot be filled by the next piece. □

The following corollary follows from Claims 5.7, 5.8, 5.9, and 5.10:
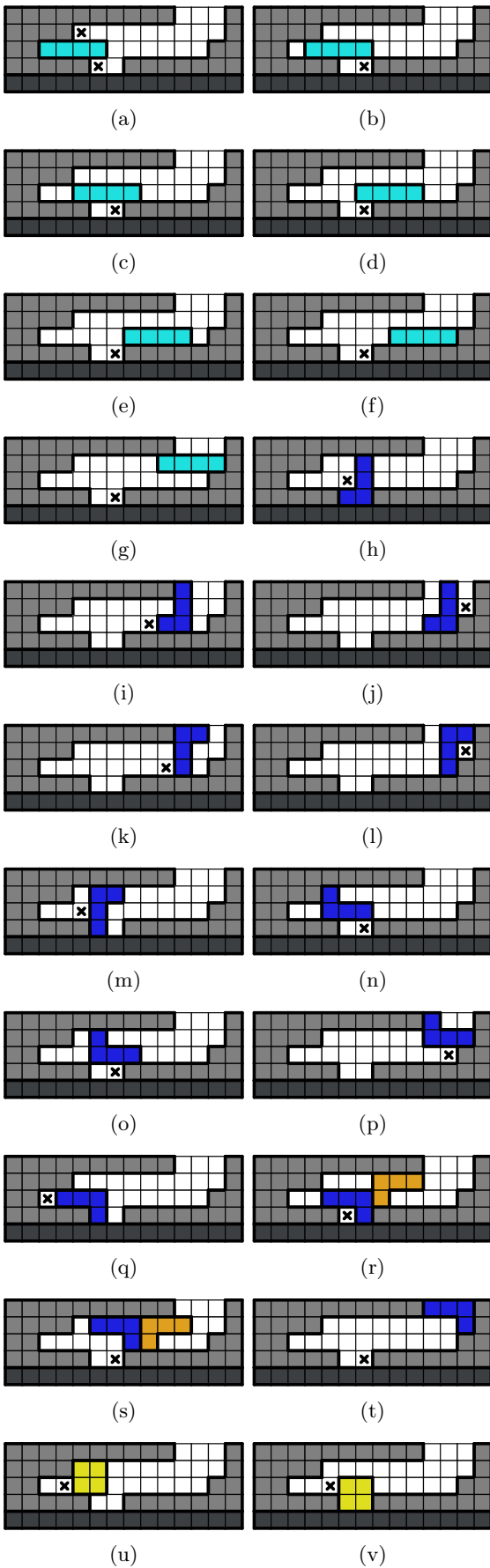
Fig. 23: Ways to place an , , or into an unprepped bucket.
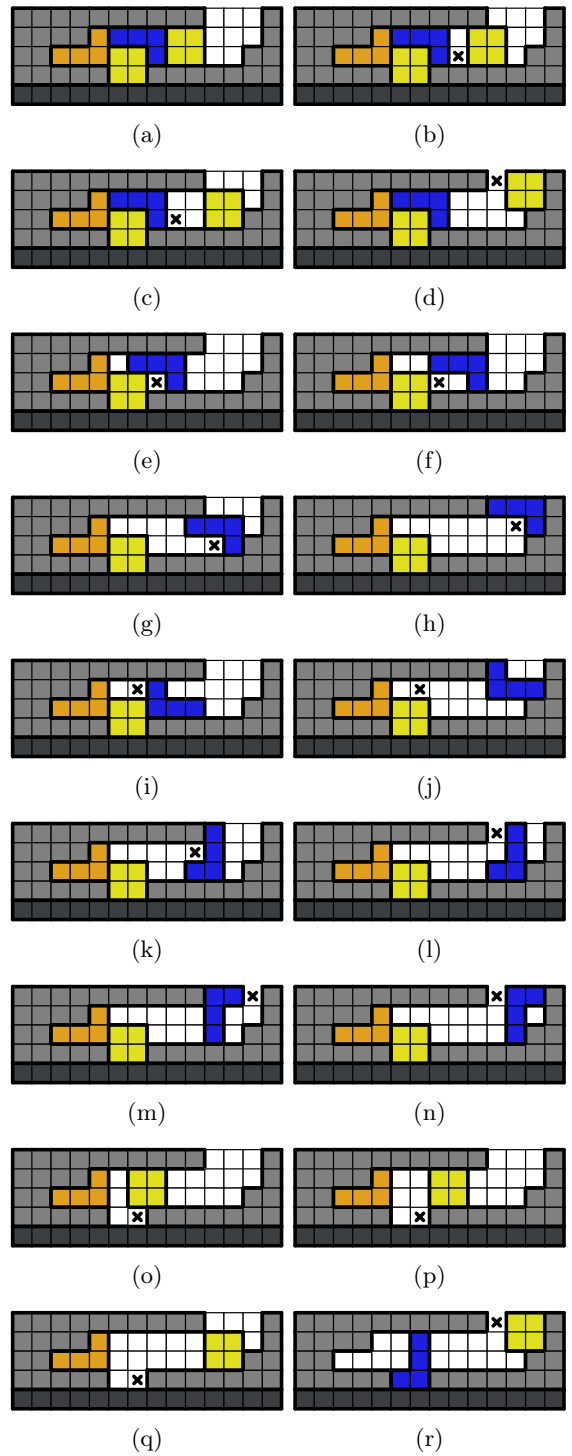
Fig. 24: Ways to place an , then a , then an into a prepped bucket, where the following piece is an .

Corollary 5.11. Suppose that before the a_i-sequence arrives, all buckets are unprepped and have height 1 (mod 3). Then, the entire a_i-sequence must be placed in one bucket, and the height of that bucket decreases by $3a_i$ (the height cannot go below 0) and is unprepped at the end.

Proof. The initial of the a_i-sequence must go in some bucket; say it has height $3h + 1$. By Claim 5.7, the bucket is now prepped with height $3h + 1$. By Claim 5.8, all pieces of the a_i-sequence must go into this bucket. Now, we have
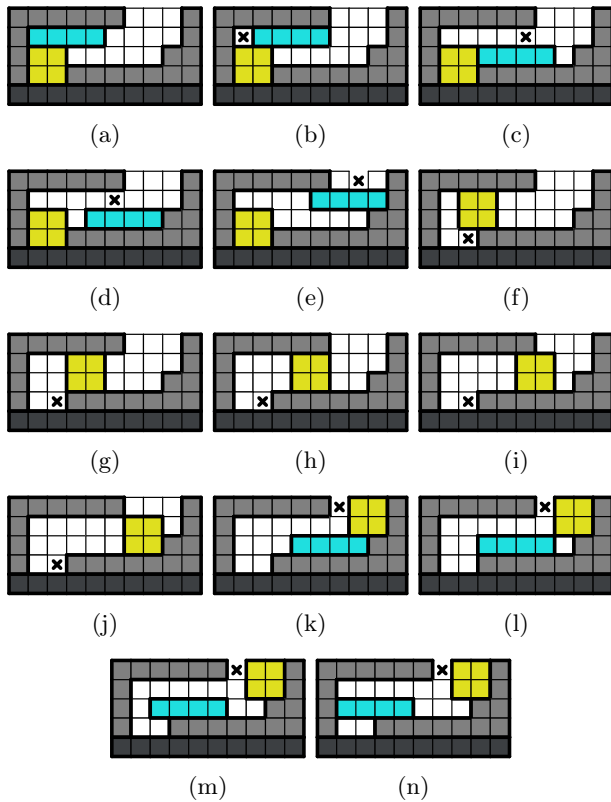
Fig. 25: Ways to place an ⬜ and then an ▭▭ into a prepped bucket, where the following piece is an ⬜.

that the total area of pieces remaining in the a_i-sequence is $36a_i - 4$, while the total area remaining in the bucket is $36h + 20$. Thus, since the total area of the pieces cannot exceed the area of the bucket, we must have $36a_i - 4 \leq 36h + 20$, and therefore (since a_i, h are integers), $a_i \leq h$. Now, by Claim 5.9, each $\langle$ ⬜, ◰, ⬜ $\rangle$ sequence must decrease the height of the bucket by 1, so after all of these the bucket is now prepped and has height $3h - 3a_i + 2$ (note that at each step the bucket had height at least 2). Now, the height of the bucket is still at least 2, so we can apply Claim 5.10. Thus, after the final $\langle$ ⬜, ▭▭ $\rangle$, the bucket must become an unprepped bucket of height $3h - 3a_i + 1$, as desired. □

Theorem 5.12. *If $\mathcal{G}(\mathcal{P})$ has a clearing trajectory sequence, then the* 3-Partition *instance $\mathcal{P}$ has a solution.*

Proof. Numbering the buckets $1, 2, \ldots, s$, let S_b be the set of i such that the a_i-sequence is placed in bucket b, so the S_b form a partition of $\{a_1, a_2, \ldots, a_{3s}\}$. By Corollary 5.11, the sum $\sum_{i \in S_b} 3a_i$ is at most the original width of each bucket, which is $3T + 1$. But note that the sum is a multiple of 3, so it must in fact be at most $3T$. However, $\sum_{i=1}^{3s} 3a_i = 3sT$, so equality must hold for each individual sum.

Thus, we have $\sum_{i \in S_b} 3a_i = 3T$ for each b. Dividing out by 3, $\sum_{i \in S_b} a_i = T$, and thus the S_b represent a valid 3-partition.

□

5.1   Linear-size pieces, 1 row

As a transition to the next topic, which considers $\leq k$-

tris, we point out that even just one row is trivially NP-hard when we allow $k \gg 4$:

Proposition 5.13. 1-row $\leq k$-tris *is strongly NP-hard.*

Proof. We reduce from 3-Partition. The board is $s(T+1)$ units wide, with an initially filled square every $T+1$ spaces, leaving s gaps of length exactly T. The first $3s$ pieces are $1 \times a_i$ for $i = 1, 2, \ldots, 3s$. The line can clear if and only if 3-Partition has a solution. A final $1 \times s(T+1)$ piece forces a loss otherwise. □

6.   Starting from an empty board is NP-hard

6.1 $O(1)$-size pieces, 8 columns

Theorem 6.1. c-column Empty $(\leq c^2 + 1)$-tris *is NP-complete for any $c \geq 8$. In particular,* 8-column Empty (≤ 65)-tris *is NP-complete.*

*Proof.* We force the player to build the board initial configuration B from Theorem 4.1's proof, starting from an empty board, using pieces of size at most $c^2 + 1$. We build B from the bottom up using pieces of height $c + 1$ (so they cannot be rotated) and width c (so they also cannot be translated). Specifically, if we want to add a cell in column i of the top row that already has a cell, we send a $c \times c$ square with an extra cell in column i below it (in the bottom $(c+1)$st row); the $c \times c$ square lands above the top existing row and clears, leaving just a cell in column i in the previous top row. To start a new row with a cell in column i, we send a $c \times c$ square with an extra cell in column i above it (in the top $(c+1)$st row); the $c \times c$ square lands above the top existing row and clears, leaving just a cell in column i in its own row.

These two operations suffice to create any legal board configuration, until we get near the top of the board in which case the partial lock out would cause the player to lose. When we fill the last pixel in the $(c+1)$st row of the board, we send a different piece: instead of putting a $c \times c$ square above that pixel, we put the desired board configuration for the top c rows. By modifying the construction of Theorem 4.1 to have a lot of rows after the first two where just the first, seventh, and eighth columns are empty, we can guarantee the additional property that this piece shape is a connected polyomino.

Then we proceed as in the reduction of Theorem 4.1. □

6.2   Linear-size pieces, 3 columns

Theorem 6.2. 3-column Empty $O(n)$-tris *is NP-complete.*

Proof. The problem is in NP because checking whether a sequence of Tetris piece placements clears the board can be done in polynomial time, so it suffices to prove that 3-column Tetris with polynomially sized pieces is NP-hard.

As in the previous section, we reduce from 3-Partition to perfect-information Tetris.

Given an instance of 3-Partition with target sum T and integers $\{a_1, \ldots, a_{3s}\}$, we construct an instance of 3-column

Tetris as follows, and as pictured in Fig. 26:

(1)   The board is 3 columns wide and $(3t + 1)s$ rows tall.

(2)   In the left column, the cells whose row is a multiple of $3t + 1$ are filled (starting with the bottom one, row 0).

(3)   The middle column is empty.

(4)   In the right column, the cells whose row is not a multiple of $3t + 1$ are filled.

(5)   The sequence of pieces is an a_i *sequence*, a *right filling sequence*, and a *clearing sequence*:

(a)   The a_i sequence is, for each a_i, a $3a_i \times 1$ rectangle.
(b)   The right filling sequence is s 1×1 squares.
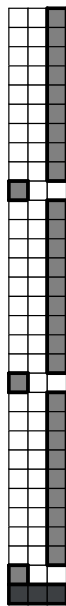(c)   The clearing sequence is a single $(3T+1)s \times 1$ rectangle.

Fig. 26: The initial board when $T = 3$ and $s = 3$.

This initial position can be reached by normal Tetris play, by placing ▨ pieces, ▨ pieces, and one 1×1 piece at the top. If, as in Section 6, all lines clear (and lines above descend), before the game checks whether the player lost by breaching the ceiling, then we can force this position from an empty board, by sending pieces consisting of a 3×3 square and one extra cell, which cannot rotate and therefore add a pixel.

If the 3-PARTITION instance has a solution, then the constructed Tetris problem has a solution: for each triple $\{a_i, a_j, a_k\}$ with sum T in the 3-PARTITION solution, we will fill one of the empty blocks of size $3t$ in the left column by moving the rectangles of size $3a_i \times 1$, $3a_j \times 1$, and $3a_k \times 1$ down the empty middle column and then left. Then use the s squares of the right filling sequence to fill the s empty cells in the right column, which are again accessible by the middle column. Finally, place the $(3T+1)s \times 1$ rectangle in the middle column, which clears the puzzle.

If the Tetris problem has a solution, we can construct a solution to the 3-PARTITION instance.

First, there are $2(3T + 1)s$ empty cells in the starting board, so at least $2(3T + 1)s$ cells from the given pieces must fill those empty cells. The total number of cells in the given sequence of pieces is $2(3T+1)s$: $(3T)s$ from the a_i sequence, s from the right filling sequence, and $(3T+1)s$ from the clearing sequence, so every cell from the given pieces must fill one of the empty cells in the $(3T + 1)s$ rows that are initially nonempty. The final $(3T + 1)s \times 1$ rectangle puts pieces in $(3T + 1)s$ rows, so those must be exactly the initially nonempty rows; that is, no rows can be cleared before the final piece, and no pieces can be placed in the center column before the final piece.

The T empty cells in the right column can be filled only by 1×1 rectangles, so the T pieces of the right filling sequence must be placed there.

Finally, the pieces of the a_i sequence can only go in the left column and fill the s empty spaces of size $1 \times 3T$, so the assignment of a_i blocks to those spaces gives a solution to the 3-PARTITION problem.   □

If we relax the constraint that the initial position can be reached by normal Tetris play, then essentially the same proof shows that Tetris is hard even with only 2 columns: just delete the right column and the right filling sequence. More interestingly, we can apply this idea to the regular game with two rows:

Theorem 6.3.   2-ROW EMPTY $O(n)$-TRIS *is NP-complete.*

Proof.   We reduce from 3-PARTITION. If the instance of 3-PARTITION has s triples and target sum t, the Tetris board has 2 rows and $s(4t+1) + 2$ columns. We first send a piece of width $s(4t + 1) + 1$ columns; on the bottom row, with every cell present on the bottom row and only the multiples of $4t+1$ (starting with 0, for a total of $s+1$ of them) present on the top row. This piece can only be placed in two positions, and either of them leaves s buckets of size $4t$ on the top row and a single 1×2 hole at the side. We then send, for each a_i, a $1 \times 4a_i$ rectangle; these cannot fill the 1×2 hole without losing, so they must be placed in the buckets on the top row. If there is a solution to the 3-PARTITION instance, the buckets can be filled by the $1 \times 4a_i$ rectangles exactly by filling the buckets according to such a solution; if there is no solution, the Tetris game is lost. Finally, we send a 1×2 piece, which can go in the hole at the side to clear the Tetris board and survive.   □

This result is tight:

Proposition 6.4.   1-ROW EMPTY $O(n)$-TRIS *can be solved in linear time.*

Proof.   In 1-ROW EMPTY $\leq k$-TRIS, any piece that is not $1 \times k$ results in an immediate loss for the player (as in the proof of Proposition 3.1). If there are only $1 \times k$ pieces in the sequence, we proceed in rounds between line clears. For each round except the last, we compute the number m of pieces before the next round by finding the prefix of remaining pieces whose total area is exactly the board width. If

such an m exists, we can place those m pieces greedily from left to right in the initially empty row, and the row clears. If no such m exists, then we attempt to place the remaining pieces greedily from left to right; if we run out of space, then no strategy could have survived.   □

7.   Font

To demonstrate the versatility of Tetris constructions, we designed an 8-row font where each letter of the alphabet is constructed as a stacking of exactly one copy of each tetromino (treating reflections as distinct, as in Tetris). Fig. 27 shows the fully assembled font. Crucially, these letters can actually be constructed in Tetris by stacking the pieces one at a time in some order (dependent on the letter), while being supported by the previously stacked pieces according to Tetris physics. Fig. 28 illustrates the stacking order in a puzzle version of the font, where the pieces are spread out vertically according to their fall order, but placed correctly horizontally; letting the pieces fall straight down reveals the letters in Fig. 27. A companion web app[*2] allows you to type a custom message, and animate the stacking. Fig. 29 shows a sample animation.
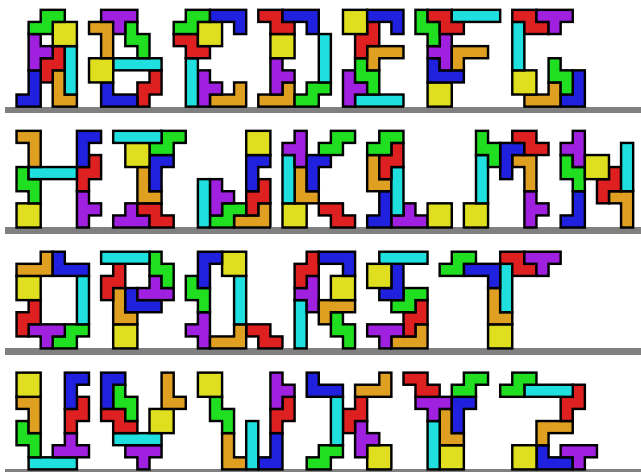

Fig. 27: Tetris font: each 8-row letter can be made by stacking each of the seven tetrominoes exactly once in some order.

8.   Open Problems

The main open problems are to determine the critical threshold for the minimum number c^* of columns and the minimum number r^* of rows for which c^*-COLUMN TETRIS and c^*-ROW TETRIS are NP-complete, respectively. We proved here that $c^* \in [3, 8]$ and $r^* \in [2, 4]$. We conjecture that $r^* = 2$, i.e., that 2-ROW TETRIS is NP-complete. These problems are open even for c^*-COLUMN $O(1)$-TRIS and r^*-ROW $O(1)$-TRIS, i.e., allowing constant-size pieces.

Our hardness proof for c-COLUMN TETRIS survival relies on the partial lock out rule, which has been changed in modern versions of Tetris [1]. We can avoid this assumption, and also allow constant reaction times for the player, by adding
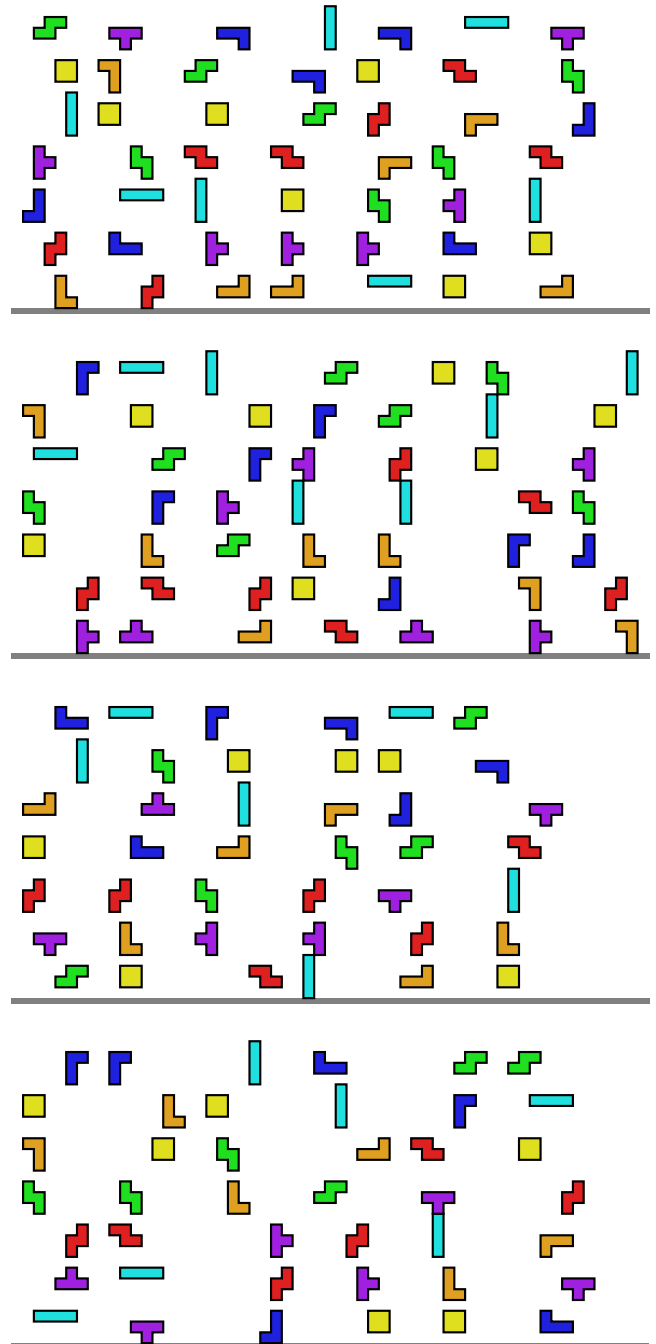
Fig. 28: Tetris puzzle font: if each piece falls vertically, the result is Fig. 27.

many rows on top and using the reservoir trick from [2], Section 4.2. However, this approach works only for c odd. Is 8-COLUMN TETRIS survival NP-hard without the partial lock out rule?

Modern versions of Tetris also have a "holding" function, where the player can put one piece aside for later use. Can existing results be re-established, or existing open problems be solved, with the addition of this feature?

Many other questions posed in prior papers on Tetris still remain open. For example, does Tetris remain hard from an empty board? What is the complexity of Tetris with imperfect information or randomness? Are there guaranteed loss sequences in n-tris for all n?

Fig. 29: Tetris animated font. See http://erikdemaine.org/ fonts/tetris/?text=tetris for animation with falling, sliding, and rotation.

References

[1] Blue Planet Software: *2009 Tetris Design Guideline* (2009). https://tetris.fandom.com/wiki/Tetris_Guideline.
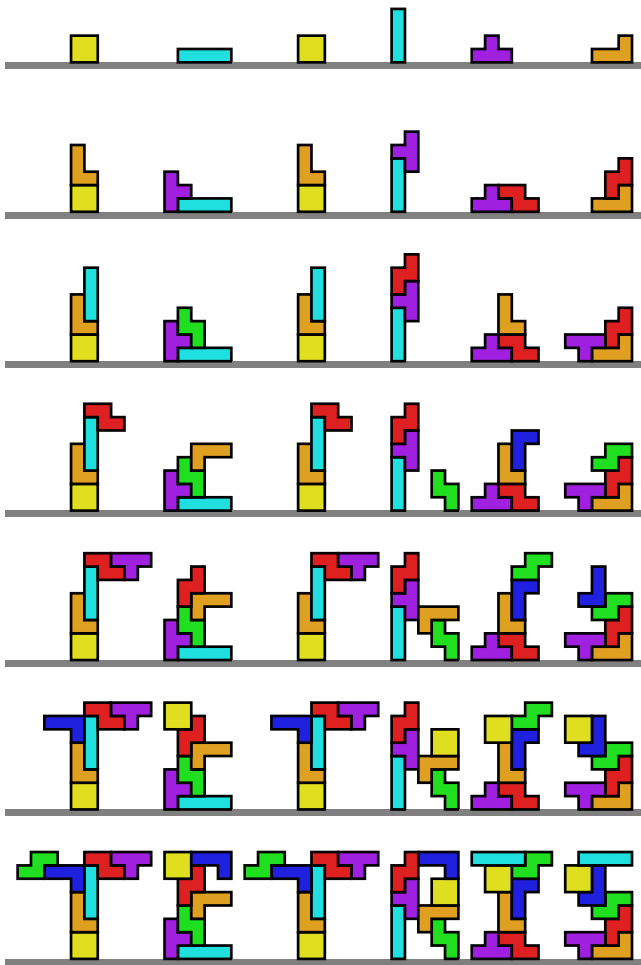
[2] Breukelaar, R., Demaine, E. D., Hohenberger, S., Hoogeboom, H. J., Kosters, W. A. and Liben-Nowell, D.: Tetris is Hard, Even to Approximate, *International Journal of Computational Geometry and Applications*, Vol. 14, No. 1–2, pp. 41–68 (2004).

[3] Demaine, E. D., Demaine, M. L., Eisenstat, S., Hesterberg, A., Lincoln, A., Lynch, J. and Yu, Y. W.: Total Tetris: Tetris with Monominoes, Dominoes, Trominoes, Pentominoes, . . . , *Journal of Information Processing*, Vol. 25, pp. 515–527 (2017).

[4] Sipser, M.: *Introduction to the Theory of Computation*, Vol. 2, Cengage learning (2006).

[5] Sparkes, M.: Tetris at 30: a history of the world's most successful game, *The Telegraph* (2014). https://www.telegraph.co.uk/technology/video-games/10877456/Tetris-at-30-a-history-of-the-worlds-most-successful-game.
html.

[6] Tetris Wiki: Playfield, https://tetris.fandom.com/wiki/Playfield (2019).

[7] Tetris Wiki: SRS, https://tetris.fandom.com/wiki/SRS (2019).

[8] Tetris Wiki: Top out, https://tetris.fandom.com/wiki/Top_out (2019).

[9] Wikipedia: List of best-selling video games, https://en.wikipedia.org/wiki/List_of_best-selling_video_games (2019).

[10] Wikipedia: Tetris, https://en.wikipedia.org/wiki/Tetris (2019).

[11] Younger, D. H.: Recognition and parsing of context-free languages in time n^3, *Information and control*, Vol. 10, No. 2, pp. 189–208 (1967).

Sualeh Asif is an undergraduate student at MIT expecting to graduate in 2022. His research interests include computational complexity, efficient computing and topics at the intersection of computing and number theory.

Michael Coulombe received his B.S. degree from the University of California at Davis in 2013 and his M.S. degree from the Massachusetts Institute of Technology in 2015. He is currently working towards a Ph.D. under Erik Demaine.

Erik D. Demaine received a B.Sc. degree from Dalhousie University in 1995, and M.Math. and Ph.D. degrees from the University of Waterloo in 1996 and 2001, respectively. Since 2001, he has been a professor in computer science at the Massachusetts Institute of Technology. His research interests range throughout algorithms, from data structures for improving web searches to the geometry of understanding how proteins fold to the computational difficulty of playing games. In 2003, he received a MacArthur Fellowship as a "computational geometer tackling and solving difficult problems related to folding and bending— moving readily between the theoretical and the playful, with a keen eye to revealing the former in the latter". He cowrote a book about the theory of folding, together with Joseph O'Rourke (*Geometric Folding Algorithms*, 2007), and a book about the computational complexity of games, together with Robert Hearn (*Games, Puzzles, and Computation*, 2009). With his father Martin, his interests span the connections between mathematics and art.

Martin L. Demaine is an artist and computer scientist. He started the first private hot glass studio in Canada and has been called the father of Canadian glass. Since 2005, he has been the Angelika and Barton Weller Artist-in-Residence at the Massachusetts Institute of Technology. Martin works together with his son Erik in paper, glass, and other material. Their artistic work includes curved origami sculptures in the permanent collections of the Museum of Modern Art in New York, and the Renwick Gallery in the Smithsonian. Their scientific work includes over 140 published joint papers, including several about combining mathematics and art.

Adam Hesterberg received an A.B. degree summa cum laude from Princeton University in 2011 and a Ph.D. degree from Massachusetts Institute of Technology in 2018. He is now Assistant Director of Undergraduate Studies in Computer Science at Harvard University.

Jayson Lynch received a Ph.D. from MIT 2020 for work on the computational complexity of motion planning problems under Erik Demaine. Jayson is now a Postdoctoral Researcher at the University of Waterloo continuing to do work on computational geometry and origami, graph algorithms, resource efficient computing, and the computational complexity of games and puzzles.

Mihir Singhal is an undergraduate studying mathematics at MIT. His research interests include combinatorics, algorithms, and complexity theory.