Tetris with Few Piece Types

MIT Hardness Group[1]
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Erik D. Demaine
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Holden Hall
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Jeffery Li
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

We prove NP-hardness and #P-hardness of Tetris clearing (clearing an initial board using a given sequence of pieces) with the Super Rotation System (SRS), even when the pieces are limited to *any two* of the seven Tetris piece types. This result is the first advance on a question posed twenty years ago: which piece sets are easy vs. hard? All previous Tetris NP-hardness proofs used five of the seven piece types. We also prove ASP-completeness of Tetris clearing, using three piece types, as well as versions of 3-Partition and Numerical 3-Dimensional Matching where all input integers are distinct. Finally, we prove NP-hardness of Tetris survival and clearing under the "hard drops only" and "20G" modes, using two piece types, improving on a previous "hard drops only" result that used five piece types.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases complexity, hardness, video games, counting

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.24

Related Version *Full Version*: https://arxiv.org/abs/2404.10712

Acknowledgements This paper was initiated during open problem solving in the MIT class on Algorithmic Lower Bounds: Fun with Hardness Proofs (6.5440) taught by Erik Demaine in Fall 2023. We thank the other participants of that class for helpful discussions and providing an inspiring atmosphere. Figures drawn with SVG Tiler (https://github.com/edemaine/svgtiler).

1 Introduction

Tetris is one of the oldest and most popular puzzle video games, originally created by Alexey Pajitnov in 1984. Tetris has reached mainstream media many times, most recently in the biopic *Tetris* [1] and with the news of 13-year-old Willis Gibson being the first person to "beat" the NES version of Tetris by reaching a killscreen [6].

The rules of Tetris are simple. In each round, a tetromino piece (one of , , , , , ,) spawns at the top of a grid and periodically moves down one unit, assuming the squares below the piece are empty. The player can repeatedly move this piece one unit left, one unit right, or one unit down, or rotate the piece by ±90°. When any part of the piece rests on top of a filled square for long enough that it triggers an automatic downward move, the piece "locks" in place, and stops moving. If a piece stops above a certain height or where the next piece would spawn, the player loses; otherwise, the next

[1] 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 Hardness Group, Erik D. Demaine, Holden Hall, and Jeffery Li;
licensed under Creative Commons License CC-BY 4.0

12th International Conference on Fun with Algorithms (FUN 2024).
Editors: Andrei Z. Broder and Tami Tamir; Article No. 24; pp. 24:1–24:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 Our NP-hardness results for Tetris clearing assuming SRS. Each entry in a specific row and column corresponds to the proposition for the hardness of the two-element subset consisting of the row piece and column piece (for example, the entry "Prop. 6" in row ▬ and column 🟨 indicates that Proposition 6 proves hardness for the subset {▬, 🟨}). Letters in parentheses denote additional models ("H" for "hard drop only", "G" for "20G"); question mark indicates a conjecture for hardness under that additional model.

	▬	🟨	🟪	🟩	🟥	🟦	🟧
▬	–	Prop. 6 (H)	Prop. 6 (H)	Prop. 6	Prop. 6	Prop. 6 (H, G)	Prop. 6 (H, G)
🟨	–	–	Prop. 9	Prop. 8	Prop. 8	Prop. 7	Prop. 7
🟪	–	–	–	Prop. 10	Prop. 10	Prop. 12	Prop. 12
🟩	–	–	–	–	Prop. 11	Prop. 13	Prop. 12
🟥	–	–	–	–	–	Prop. 12	Prop. 13
🟦	–	–	–	–	–	–	Prop. 14 (G?)
🟧	–	–	–	–	–	–	–

piece spawns at the top of the grid, and play continues. Completely filling a row causes the row to clear, and all squares above that row move downward by one unit. For more detailed rules, see [16].

To study Tetris from a computational complexity perspective, we generally assume that the player is given a sequence of pieces and an initial board state of filled cells, making the game perfect information (as introduced in [4]). The two main objectives we consider here are "clearing" and "survival" (as introduced in [7]). In ***Tetris clearing***, we want to determine whether we can clear the entire board after placing all the given pieces. In ***Tetris survival***, we want to determine whether the player can avoid losing before placing all the given pieces. Previous work shows that these problems are NP-complete, even to approximate various metrics within $n^{1-\epsilon}$ [4], or with only 8 columns or 4 rows [2], or with additional constraints on drops [12], or with k-ominoes for $k \geq 3$ clearing or $k \geq 4$ survival [7].

1.1 Our Results

One of the open problems posed in the original paper proving Tetris NP-hard twenty years ago [4] is to determine which subsets of the seven Tetris piece types {▬, 🟨, 🟪, 🟩, 🟥, 🟦, 🟧} suffice for NP-hardness, and which admit a polynomial-time algorithm. All existing Tetris NP-hardness proofs [4, 2, 12] use at least five of the seven piece types. In particular, [4, Section 6.2] mentions various sets of five piece types that suffice. What about fewer piece types?

Our main results are the first to make progress on this question: for any *size-2* subset $A \subseteq$ {▬, 🟨, 🟪, 🟩, 🟥, 🟦, 🟧}, Tetris clearing is NP-complete with pieces restricted to A. Most pairs of piece types require different constructions for their reductions; refer to Table 1. Our results require us to specify more details of the piece rotation model, specifically what happens when the player rotates a piece in a way that collides with a filled square. We assume the ***Super Rotation System (SRS)*** [14], first introduced in the 2001 game *Tetris Worlds* and as part of the Tetris Company's Tetris Guideline for how all modern (2001+) Tetris games should behave [15].

For every size-2 subset A of piece types, we also establish #P-hardness for the corresponding problem of counting the number of ways to clear the board. Here we distinguish solutions by the final placement of each piece, not the sequence of moves to make those placements (as long as the placement is valid). This definition lets us ignore e.g. the null effect of moving a piece repeatedly left and right.

For certain size-3 subsets of piece types, we further establish ASP-completeness for Tetris clearing. Recall that an NP search problem is ***ASP-complete*** [17] if there is a parsimonious reduction from all NP search problems (including a polynomial-time bijection between solutions). In particular, ASP-completeness implies NP-hardness of finding another solution given k solutions, for any $k \geq 0$, as well as #P-completeness. These results hold for piece types {▬, 🟪, 🟧} and {▬, 🟪, 🟦}.

We also study Tetris under two more restrictive models on piece moves:

- ***Hard drops only***: In this model, pieces do not move downward on their own, and if the player moves a piece downward, the piece moves maximally downward before locking into place (a ***hard drop*** maneuver). The player is still free to rotate or move the piece left or right before hard-dropping the piece. This model is motivated by most Tetris games awarding higher scores for hard drops, and was posed in [4].
- ***20G***: In this model, instead of periodically moving down one unit, all pieces move maximally downward *instantly and on their own*, and the player is not allowed to control how fast a piece moves downward. The player is still free to rotate or move the piece left or right before the piece locks. This model is motivated by levels with the maximum possible gravity, as in Level 20+ of regular Tetris with 20 rows [13].

For certain size-2 subsets of piece types, we establish NP-hardness of both Tetris survival and clearing under either of these models. Table 1 labels which of our Tetris clearing results hold in which models.

Along the way, we prove new results about 3-Partition and Numerical 3-Dimensional Matching (3DM): both problems are strongly ASP-complete even when all integers are assumed distinct. These results are of independent interest for ASP-hardness reductions. Previously, these problems were known to be ASP-complete with multisets of integers [3], and strongly NP-complete with distinct integers [10].

1.2 Outline

The structure of the rest of the paper is as follows. Section 2 details the Super Rotation System (SRS), an important aspect of modern Tetris and used in our constructions. Section 3 proves ASP-completeness of 3-Partition with Distinct Integers and Numerical 3-Dimensional Matching with Distinct Integers, two problems we reduce from. Section 4 discusses our hardness results for Tetris clearing with SRS with only two piece types. Section 5 discusses some Tetris survival results under the "hard drops only" and "20G" models. Section 6 proves ASP-completeness of Tetris clearing with SRS.

2 Super Rotation System (SRS)

Most previous Tetris results are not sensitive to exactly how Tetris pieces rotation: most reasonable rotation models work [4, Section 6.4]. By contrast, many of the results in this paper focus specifically on (and require) the ***Super Rotation System (SRS)*** [14], defined as follows.

Each piece has a defined ***rotation center***, as indicated by dots in Figure 1, except for ▬ and 🟨, whose rotation centers are the centers of the 4×4 squares in Figure 1. When unobstructed, all non-🟨 tetrominoes will rotate purely about the rotation center (note that 🟨 pieces cannot rotate). The key feature about SRS is ***kicking***: if a tetromino is obstructed when a rotation is attempted, the game will attempt to "kick" the tetromino into one of four alternate positions, each tested sequentially; if all four positions do not work, then the rotation will fail. See Figure 2 for an example of this kicking process. The full data for wall kicks can be found in Tables 2 and 3, and at [14]. Of note is that SRS wall kicks are vertically symmetric for all pieces or pairs of pieces (i.e., ↔ and ↔) except for the piece, so all rotations can be mirrored.

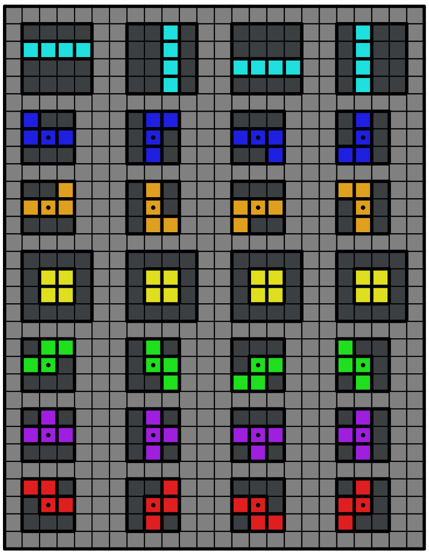
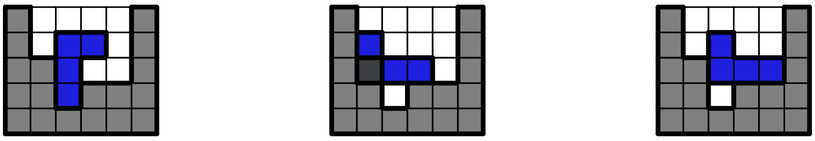


Figure 1 All tetromino pieces, in order from top to bottom: , , , , , , . The first column is the default orientation of a piece upon spawning in; each column to the right indicates a 90° rotation clockwise about the rotation center of the piece.



(a) (b) (c)

Figure 2 An example of the SRS kick system. Suppose the piece in (a) is being rotated 90° counter-clockwise. Test 1 (which is $(0,0)$) would fail, due to the dark gray square shown in (b). Test 2 (which is $(+1,0)$) would succeed, as shown in (c), and so the piece would rotate to the position in (c).

This system of kicking tetrominoes during rotations allows for moves which are often called ***twists*** or ***spins***. All the spins that we utilize are detailed in the appendix of the full version of our paper.

3 3-Partition and Numerical 3DM with Distinct Integers

Our reductions to Tetris are all from one of the following two problems, which are strengthenings of two standard strongly NP-complete problems:

▶ **Definition 1 (3-Partition with Distinct Integers).** *Given a **set** $A = \{a_1, a_2, \ldots, a_n\}$ of n **distinct** positive integers such that $\frac{t}{4} < a_i < \frac{t}{2}$ for each i, where $t = \frac{3}{n}\sum_{i=1}^{n} a_i$, determine whether there is a partition of A into $\frac{n}{3}$ groups $D_1, \ldots, D_{n/3}$ (each necessarily of size 3) having the same sum $\sum_{x \in D_j} x = t$.*

Table 2 Kick data for , , , , and pieces. 0 indicates the default orientation, and R, 2, and L indicate the orientation reached from a 90°, 180°, and 270° rotation clockwise (respectively) from the default orientation. An ordered pair (a, b) denotes a translation of the center by a units in the x direction and b units in the y direction. Positive x direction is rightwards, and positive y direction is upward.

	Test 1	Test 2	Test 3	Test 4	Test 5
$0 \to R$	$(0,0)$	$(-1,0)$	$(-1,+1)$	$(0,-2)$	$(-1,-2)$
$R \to 0$	$(0,0)$	$(+1,0)$	$(+1,-1)$	$(0,+2)$	$(+1,+2)$
$R \to 2$	$(0,0)$	$(+1,0)$	$(+1,-1)$	$(0,+2)$	$(+1,+2)$
$2 \to R$	$(0,0)$	$(-1,0)$	$(-1,+1)$	$(0,-2)$	$(-1,-2)$
$2 \to L$	$(0,0)$	$(+1,0)$	$(+1,+1)$	$(0,-2)$	$(+1,-2)$
$L \to 2$	$(0,0)$	$(-1,0)$	$(-1,-1)$	$(0,+2)$	$(-1,+2)$
$L \to 0$	$(0,0)$	$(-1,0)$	$(-1,-1)$	$(0,+2)$	$(-1,+2)$
$0 \to L$	$(0,0)$	$(+1,0)$	$(+1,+1)$	$(0,-2)$	$(+1,-2)$

Table 3 Kick data for pieces, with same notation as Table 2.

	Test 1	Test 2	Test 3	Test 4	Test 5
$0 \to R$	$(0,0)$	$(-2,0)$	$(+1,0)$	$(-2,-1)$	$(+1,+2)$
$R \to 0$	$(0,0)$	$(+2,0)$	$(-1,0)$	$(+2,+1)$	$(-1,-2)$
$R \to 2$	$(0,0)$	$(-1,0)$	$(+2,0)$	$(-1,+2)$	$(+2,-1)$
$2 \to R$	$(0,0)$	$(+1,0)$	$(-2,0)$	$(+1,-2)$	$(-2,+1)$
$2 \to L$	$(0,0)$	$(+2,0)$	$(-1,0)$	$(+2,+1)$	$(-1,-2)$
$L \to 2$	$(0,0)$	$(-2,0)$	$(+1,0)$	$(-2,-1)$	$(+1,+2)$
$L \to 0$	$(0,0)$	$(+1,0)$	$(-2,0)$	$(+1,-2)$	$(-2,+1)$
$0 \to L$	$(0,0)$	$(-1,0)$	$(+2,0)$	$(-1,+2)$	$(+2,-1)$

▶ **Problem 2 (Numerical 3-Dimensional Matching (3DM) with Distinct Integers).** *Given three* ***sets***

$$A = \{a_1, a_2, \dots, a_n\}, B = \{b_1, b_2, \dots, b_n\}, \text{ and } C = \{c_1, c_2, \dots, c_n\}$$

of n positive integers, where all $3n$ integers are ***distinct****, and a target sum $t = \frac{1}{n}\sum_{i=1}^{n}(a_i + b_i + c_i)$, determine whether there is a partition of $A \cup B \cup C$ into n groups $D_1, \dots, D_n$, each with exactly one element from each of A, B, and C, and $\sum_{x \in D_j} x = t$ for all j.*

Without the "distinct" and "set" conditions, both problems are well-known to be ***strongly NP-complete***, meaning that the problem is NP-hard even if the a_i integers are bounded by a polynomial in n. This property makes it feasible to represent each integer a_i (and t) in unary, which is the approach taken by all past Tetris NP-hardness proofs [4, 7, 2, 12], as then the total reduction size is still polynomial in n.

We want to ensure all integers are distinct in order to have more control over our reductions' blowup in the number of solutions, as needed for #P- and ASP-hardness. Bosboom et al. [3] proved that numerical 3DM is strongly ASP-complete when A is restricted to be a set, but allowed for B and C to be multisets as usual, and did not forbid repeated integers between A, B, C. Hulett, Will, and Woeginger [10] proved that both 3-Partition and Numerical 3DM remain strongly NP-hard with distinct integers. We extend their proof to obtain ASP-completeness:

▶ **Theorem 3.** *3-Partition with Distinct Integers, and Numerical 3-Dimensional Matching with Distinct Integers, are strongly ASP-complete.*

To prove this result, we use the following intermediate problems (which are thus also ASP-complete):

▶ **Problem 4** (**Positive 1-in-3SAT**). *Given a boolean formula in 3CNF (i.e., an* AND *of clauses consisting of 3 literals), where all literals are positive, does there exist an assignment of the variables to either true or false such that each clause has exactly one literal set to true?*

▶ **Problem 5** (**Tripartite Edge-Disjoint Triangle Partition**). *Given an undirected tripartite graph $G = (V, E)$, can we partition E into disjoint triangles?*

Proof of Theorem 3. We give a chain of parsimonious reductions from 3SAT, which is known to be ASP-complete [17]:

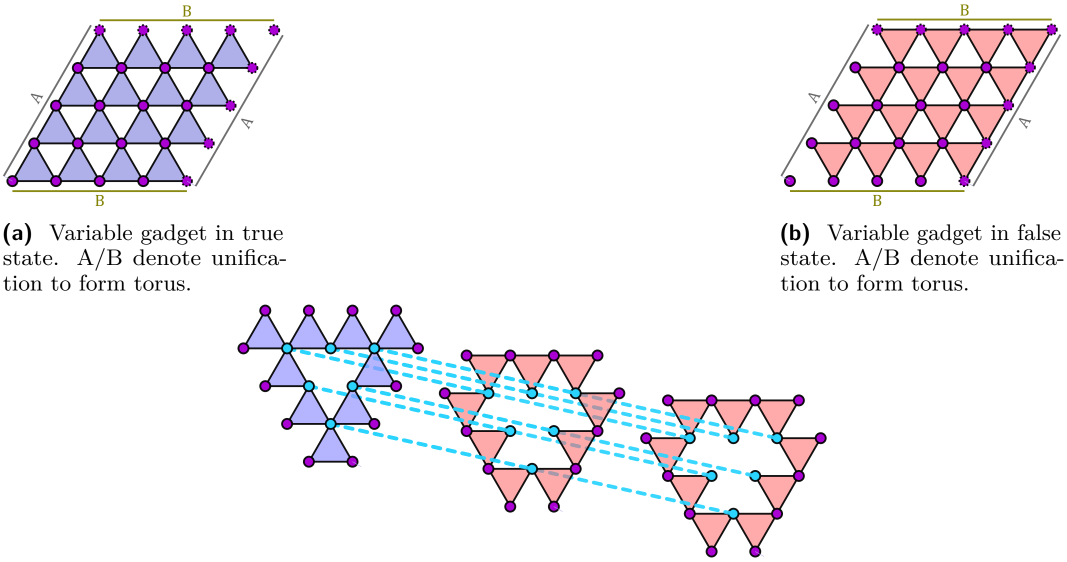
1. **3SAT → Positive 1-in-3SAT**: Hunt, Marathe, Radhakrishnan, and Stearns [11, Theorem 3.8] gave such a parsimonious reduction.[2] See also [3, Lemma 2.1].
2. **Positive 1-in-3SAT → Tripartite Edge-Disjoint Triangle Partition**: We follow a simplification of a reduction from FCP 1-in-3SAT to FCP Tripartite Edge-Disjoint Triangle Partition [8, Theorem 12], which in turn is based on a reduction from 3SAT to Tripartite Edge-Disjoint Triangle Partition [9]. By reducing from Positive 1-in-3SAT, we simplify the reduction of [8] by avoiding negative literals.

   We represent each variable by a sufficiently large triangular grid of vertices, with opposite sides of a parallelogram identified to form a flat torus, as shown in Figure 3a. This grid has exactly two solutions, corresponding to true (the triangles in Figure 3a) and false (the triangles in Figure 3b); note that the two solutions consist of exactly the same edges, and cover each exactly once. For each clause (x, y, z), we pick one triangle of positive orientation, remove its edges, and unify the corresponding vertices of these triangles and of the three neighboring triangles of negative orientation, as shown in Figure 3c. Exactly one variable must choose the true state so as to cover the edges surrounding the unified hole exactly once. By choosing the variable gadgets large enough, we can ensure that the clause gadgets are disjoint from each other. Each gadget has a unique way to implement a given assignment, so this reduction is parsimonious.
3. **Tripartite Edge-Disjoint Triangle Partition → Numerical 3DM with Distinct Integers**: We combine a chain of reductions, from Triangle Edge-Disjoint Triangle Partition to Latin Square Completion [5], and from Latin Square Completion to Numerical 3DM with Distinct Integers [10].

   If $U = \{u_1, u_2, \ldots\}, V = \{v_1, v_2, \ldots\}, W = \{w_1, w_2, \ldots\}$ is the vertex tripartition, then we do the following:
   - Let $q = 2\max\{|U|, |V|, |W|\}$, and let the target sum be $t = 19q^6$.
   - Map each edge (u_i, w_k) to $2q^6 + iq - k \in A$.
   - Map each edge (v_j, w_k) to $7q^6 + jq^2 + k \in B$.
   - Map each edge (u_i, v_j) to $t - (9q^6 + jq^2 + iq) = 10q^6 - jq^2 - iq \in C$.

   The lemmas in [10] show that all the integers in A, B, and C are distinct (i.e., we have a valid instance of Numerical 3DM with Distinct Integers); and that any triple summing to t consists of one element each from A, B, and C, with the elements corresponding to a triangle in the graph. Thus we obtain a bijection between triangle partitions and Numerical 3DM solutions, i.e., the reduction is parsimonious.
4. **Numerical 3DM with Distinct Integers → 3-Partition with Distinct Integers**: We use standard techniques to relate these problems. Convert each integer a_i, b_i, and c_i in Numerical 3DM to integers $8a_i + 1$, $8b_i + 2$, and $8c_i - 3$, respectively, in 3-Partition;

[2] Their problem "1-Ex3MonoSat" is Positive 1-in-3SAT with the additional constraint that every clause has exactly three literals. Their reduction is also planarity preserving, so chaining with their parsimonious reduction from 3SAT to Planar 3SAT, we obtain that Planar 1-in-3SAT is also ASP-complete.



(a) Variable gadget in true state. A/B denote unification to form torus.

(b) Variable gadget in false state. A/B denote unification to form torus.

(c) Clause gadget bringing together three variable gadgets. Vertices connected by dashes are unified.

Figure 3 Reduction from Positive 1-in-3SAT to Tripartite Edge-Disjoint Triangle Partition.

and convert t to $8t$. In particular, all integers are still distinct, because we scale up by a factor of 8 and then shift values by less than 4. Furthermore, working modulo 8, every triple of integers summing to t must take exactly one a_i, one b_j, and one c_k. Therefore we have a parsimonious reduction.

Composing these reductions, we obtain that 3-Partition with Distinct Integers, and Numerical 3DM with Distinct Integers, are ASP-hard. Both problems are NP search problems, so they are ASP-complete. ◀

4 Tetris with Two Piece Types

In this section, we will prove that for any size-2 subset $A \subseteq \{$ ▬, 🟨, 🟪, 🟩, 🟥, 🟦, 🟧 $\}$, Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the sequence of pieces given to the player only contains the piece types in A. We will also show that some of the reductions work under the "hard drop only" model and the "20G" model. Refer to Table 1 for a table of all of our results.

All of our reductions are from 3-Partition with Distinct Integers and are in the same flavor as the reduction for clearing 3-tris with rotation as given in the Total Tetris paper [7], which we will use some terminology from. In particular, the reductions will involve a starting board involving $\frac{n}{3}$ structures, which we will call "***bottles***", of equal height of $\Theta(t \cdot \text{poly}(n))$, spaced sufficiently far apart so that bottles do not interact with each other except for line clears, and possibly along with an additional structure, which we will call a "***finisher***", to the right of the rightmost bottle.

Each bottle consists of a neck portion with n constant-sized "***top segments***", a body portion with t poly(n)-sized "***units***", and possibly $O(n)$ extra lines either above the neck portion, between the "top segments", between the neck portion and the body portion, and/or below the body portion that get cleaned up after the rest of the lines. To simplify our arguments, we make the size of each unit larger than the size of the neck portion.

The finisher will be a structure that specifically prevents the rows in the body portion from clearing before all of the top segments are cleared, and is located in same rows as the body portion of the bottles when required. We will use three types of finishers, a 🟦 finisher, an 🟧 finisher (which is a vertically symmetric version of the 🟦 finisher), and a 🟪 finisher, shown in Figure 4. Note that the finishers can be adapted to any number of rows larger than 4, and there is exactly one way to clear each type of finisher.

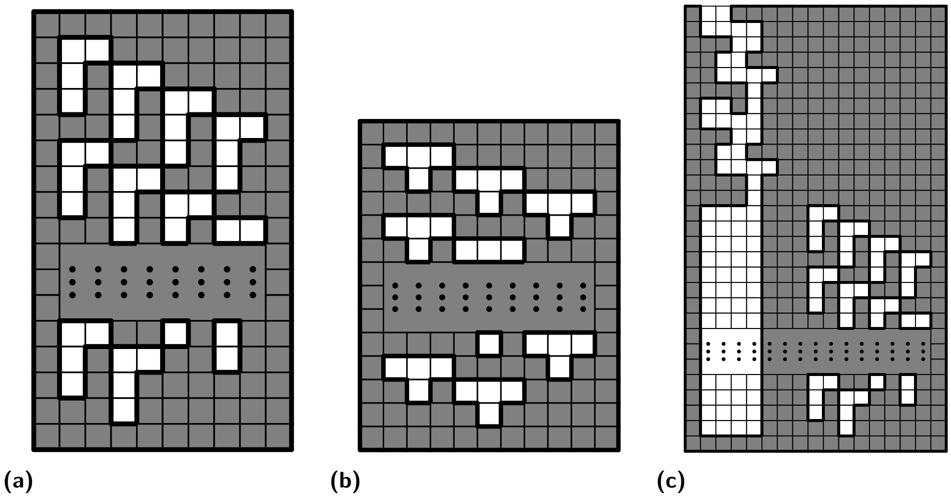


Figure 4 The 🟦 and 🟪 finishers (the 🟧 finisher can be obtained by reflecting the 🟦 finisher through a vertical line), and an example of the 🟦 finisher next to a bottle (in the 🟨, 🟦 setup).

For each element $a_i \in A$, we create a sequence of pieces S_i, which can be decomposed into three subsequences:

- ***Priming sequence***: A piece sequence that, if used correctly, properly blocks all bottles but one in the same "top segment", and if used incorrectly, either directly "overflows" the bottle (i.e., puts blocks above the line under which all of our pieces must go) or "clogs" the bottle (i.e., improperly blocks the bottle and prevents the player from being able to clear the lines necessary to re-open the bottle). For all of the bottle structures except for the one for {🟨, 🟪}, the pieces in the priming sequence cannot rotate or translate below the topmost "top segment" under SRS, and any piece placed into a "top segment" of a bottle prevents any piece in the filling sequence from reaching the body portion of that bottle.
- ***Filling sequence***: A piece sequence of length $\Theta(a_i)$ that "fills" a_i units in the body portion of the unblocked bottle. If there are not enough units left to fill, then the pieces corresponding to one of the units will cause an overflow due to there not being enough empty space in the neck portion for all of the pieces (using the fact that the size of each unit is larger than the size of the neck portion).
- ***Closing sequence***: A piece sequence that properly clears the lines corresponding to the "top segment" blocked by the priming sequence and resets the states of the neck portion of the bottles (albeit with one less "top segment").

We also have a ***finale sequence*** F, possibly the empty sequence, which helps clear any finishers on the board and the remaining lines on the board after the lines corresponding to the neck and body portions have been cleared.

In this section, when we write a sequence of pieces, we will use parentheses around sequences, commas between different piece types, and exponentiation to denote repeated pieces of the same piece type. For example, a sequence written as (🟪^{2}, 🟨^{3}, 🟩) consists of 2 🟪s, 3 🟨s, and an 🟩, in that order. The sequence of pieces given to the player will be of the form $(S_1, S_2, \ldots, S_n, F)$.

4.1 General Argument

We provide a very general argument for why these reductions work. If there exists a valid 3-partition $(D_1, \ldots, D_{n/3})$ for $\{a_1, a_2, \ldots, a_n\}$, then for each S_i, determine the corresponding j_i such that $a_i \in D_{j_i}$, then use the priming sequence to block all bottles properly except for the (j_i)th one, the filling sequence to fill a_i units in the body portion of the (j_i)th bottle, and the closing sequence to reset the states of the neck portion of the bottles. After all the S_i are used in this way, all lines corresponding to the "top segments" will be cleared as there are n such "top segments" with each S_i clearing exactly one of them, and each bottle will be filled to exactly t units. Thus, in the case where there are no pieces in the finale sequence, the lines corresponding to the body portions of the bottles will be cleared, meaning that no lines remain and we have cleared the board, and in the case where there are pieces in the finale sequence, the only lines that remain are those that can be cleared by the finale sequence. Thus, the sequence $(S_1, S_2, \ldots, S_n, F)$ can clear the board.

Conversely, if the sequence $(S_1, S_2, \ldots, S_n, F)$ can clear the board, then we claim that there is a corresponding 3-partition for $\{a_1, a_2, \ldots, a_n\}$. In particular, for each S_i, the priming sequence must properly block all but one bottle, say the (j_i)th bottle, forcing all the pieces in the filling sequence into the (j_i)th bottle. The filling sequence must then fill exactly a_i units in the (j_i)th bottle before the closing sequence, and it must do so without overfilling the body portion of the bottle, as otherwise there will be an overflow in that bottle. In particular, this means that, for each $1 \le j \le \frac{n}{3}$, the sum of the a_i corresponding to the S_j that filled some units in the jth bottle must be at most t. However, since $\sum a_i$ is exactly $t\left(\frac{n}{3}\right)$, the sum of the a_i corresponding to the S_j that filled some units in the jth bottle must actually be exactly t. In other words, there is a way to partition the a_i into $\frac{n}{3}$ subsets $D_1, \ldots, D_{n/3}$ such that the sum of the elements in each subset is t. Thus, there is a corresponding 3-partition for $\{a_1, a_2, \ldots, a_n\}$.

This general argument shows how YES instances of the two problems (3-Partition with Distinct Integers, Tetris clearing with SRS and restricted piece types) are equivalent, and hence that this reduction works. The rest of the subsections in this section show the bottle structures for each size-2 subset of piece types. Due to space constraints, we omit detailed construction-specific explanations; refer to the full version of our paper for more details.

4.2 Subsets with ▬ Pieces

First we show how the reduction in the Total Tetris paper [7] can be easily adapted to any subset of pieces with ▬ pieces plus an additional piece type:

▶ **Proposition 6.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if:*

- *The type of pieces in the sequence given to the player is restricted to any of* {▬, 🟦}, {▬, 🟧}, {▬, 🟨}, {▬, 🟩}, {▬, 🟪}, *or* {▬, 🟥}*,*
- *The model being considered is "hard drops only" and type of pieces in the sequence given to the player is restricted to any of* {▬, 🟦}, {▬, 🟧}, {▬, 🟨}, *or* {▬, 🟪}*, or*
- *The model being considered is "20G" and the type of pieces in the sequence given to the player is restricted to either* {▬, 🟦} *or* {▬, 🟧}*.*

Refer to Figure 5 for the bottle structures.

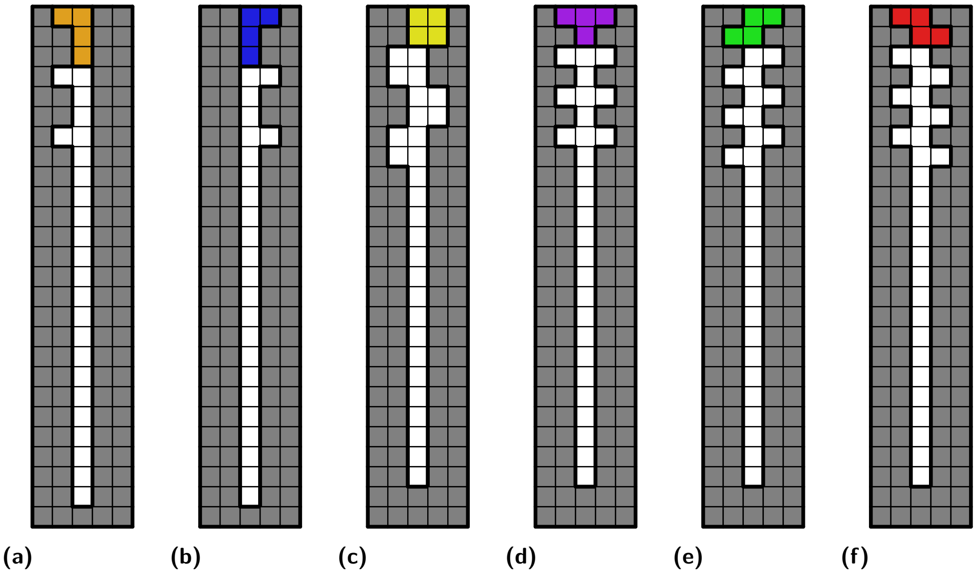


Figure 5 The bottle structures for the subsets containing ▬, including how the non-▬ piece must block a bottle during the priming and closing sequence.

4.3 Other Subsets with 🟨 Pieces

We now move on to all the remaining subsets which contain 🟨 pieces.

▶ **Proposition 7.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to either* {🟨, 🟦} *or* {🟨, 🟧}.

Refer to Figure 6a, which shows the bottle structure for {🟨, 🟦}. The bottle structure for {🟨, 🟧} can be obtained by reflecting the bottle structure for {🟨, 🟦} through a vertical line. We will also use a 🟦 (or 🟧) finisher in our setup to prevent rows in the body portion from clearing early.

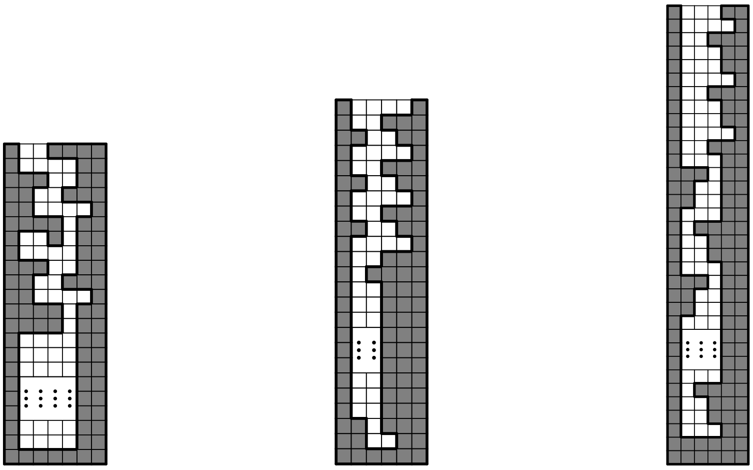
A demo of the {🟨, 🟦} bottle structure can be found at `https://jstris.jezevec10.com/map/80188`.

▶ **Proposition 8.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to either* {🟨, 🟩} *or* {🟨, 🟥}.

Refer to Figure 6b, which shows the bottle structure for {🟨, 🟩}. The bottle structure for {🟨, 🟥} can be obtained by reflecting the bottle structure for {🟨, 🟩} through a vertical line. We do not use a finisher in our setup.

A demo of the {🟨, 🟩} bottle structure can be found at `https://jstris.jezevec10.com/map/81818`.

▶ **Proposition 9.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to* {🟨, 🟪}.



(a) {🟨, 🟦}. **(b)** {🟨, 🟩}. **(c)** {🟨, 🟪}.

Figure 6 The bottle structures for the other subsets containing 🟨; the bottle structures for {🟨, 🟧} and {🟨, 🟥} can be obtained by reflecting the bottle structures for {🟨, 🟦} and {🟨, 🟩} through a vertical line.

Refer to Figure 6c for the bottle structure. We will also use a 🟪 finisher in our setup to prevent rows in the body portion from clearing early.

A demo of the {🟨, 🟪} bottle structure can be found at `https://jstris.jezevec10.com/map/80169`.

4.4 Two-Element Subsets of {🟩, 🟪, 🟥}

▶ **Proposition 10.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to either* {🟩, 🟪} *or* {🟪, 🟥}.

Refer to Figure 7a, which shows the bottle structure for {🟩, 🟪}. The bottle structure for {🟪, 🟥} can be obtained by reflecting the bottle structure for {🟩, 🟪} through a vertical line. We do not use a finisher in our setup.

A demo of the {🟩, 🟪} bottle structure can be found at `https://jstris.jezevec10.com/map/80184`.

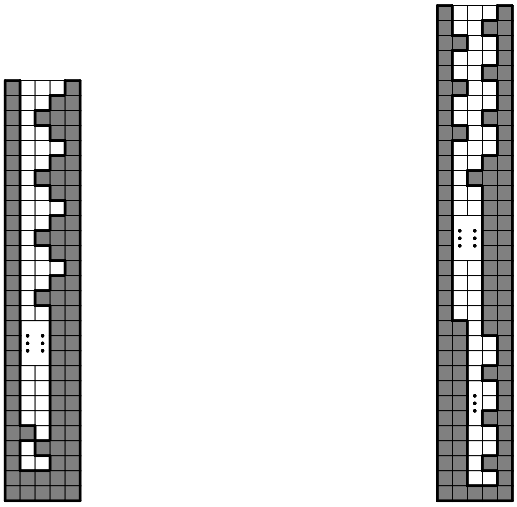
▶ **Proposition 11.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to* {🟩, 🟥}.

Refer to Figure 7b for the bottle structure. We do not use a finisher in our setup.

A demo of the {🟩, 🟥} bottle structure can be found at `https://jstris.jezevec10.com/map/80198`.

4.5 Remaining Subsets with More Complex Structures

▶ **Proposition 12.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to any of* {🟦, 🟪}, {🟦, 🟥}, {🟧, 🟩}, *or* {🟧, 🟪}.



(a) { 🟩, 🟪 }.

(b) { 🟩, 🟥 }.

Figure 7 The bottle structures for the two-element subsets of { 🟩, 🟪, 🟥 }; the bottle structure for { 🟪, 🟥 } can be obtained by reflecting the bottle structure for { 🟩, 🟪 } through a vertical line.

Refer to Figure 8a, which shows the bottle structure for { 🟦, 🟥 } and { 🟦, 🟪 }. The bottle structure for { 🟧, 🟩 } and { 🟧, 🟪 } can be obtained by reflecting the bottle structure for { 🟦, 🟥 } and { 🟦, 🟪 } through a vertical line. We will also use a 🟦 (or 🟧) finisher in our setup to prevent rows in the body portion from clearing early.

A demo of the { 🟦, 🟥 } bottle structure can be found at `https://jstris.jezevec10.com/map/80205`.

▶ **Proposition 13.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to either { 🟦, 🟩 } or { 🟧, 🟥 }.*

Refer to Figure 8b, which shows the bottle structure for { 🟦, 🟩 }. The bottle structure for { 🟧, 🟥 } can be obtained by reflecting the bottle structure for { 🟦, 🟩 } through a vertical line. We will also use a 🟦 (or 🟧) finisher in our setup to prevent rows in the body portion from clearing early.

A demo of the { 🟦, 🟩 } bottle structure can be found at `https://jstris.jezevec10.com/map/83069`.

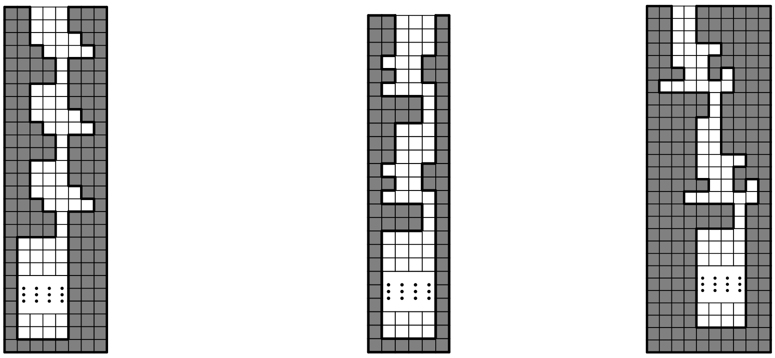
▶ **Proposition 14.** *Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to { 🟦, 🟧 }.*

Refer to Figure 8c(a) for the bottle structure for { 🟦, 🟧 }. We will also use a 🟦 finisher in our setup to prevent rows in the body portion from clearing early.

A demo of the { 🟦, 🟧 } bottle structure can be found at `https://jstris.jezevec10.com/map/80195`.

We also note the following:

▶ **Conjecture 15.** *The reduction in the proof of Proposition 14 works even if pieces experience 20G gravity.*



(a) {🟦, 🟥},{🟦, 🟪}. **(b)** {🟦, 🟩}. **(c)** {🟦, 🟧}.

Figure 8 The bottle structures for the other subsets containing 🟨; the bottle structure for {🟧, 🟩} and {🟧, 🟪} can be obtained by reflecting the bottle structure for {🟦, 🟥} and {🟦, 🟪} through a vertical line, and the bottle structure for {🟧, 🟥} can be obtained by reflecting the bottle structure for {🟦, 🟩} through a vertical line.

4.6 Putting It All Together

Combining all of these results, we get the following result:

▶ **Theorem 16.** *For any size-2 subset $A \subseteq$ {▭, 🟨, 🟪, 🟩, 🟥, 🟦, 🟧}, Tetris clearing with SRS is NP-hard, and the corresponding counting problem is #P-hard, even if the type of pieces in the sequence given to the player is restricted to the piece types in A.*

Proof. Propositions 6, 7, 8, 9, 10, 11, 12, 13, and 14 cover all size-2 subsets of piece types, as shown in Table 1; combining all of the reductions, we obtain the desired result. ◀

▶ Remark 17. All of our reductions involve a linear-factor blowup in the a_i for the filling sequences (i.e., we use $\Theta(na_i)$ pieces in the filling sequences); this makes it easier to argue about what happens when an overfill happens and makes the bottle analogy more fitting (since the neck portion is smaller while the body portion is much larger) but makes our reductions somewhat inefficient. Perhaps it is possible to reduce the blowup to a constant factor, though the argument may be a bit more complex.

5 Tetris Survival: Hard Drops Only and 20G

The previous section mentions NP-hardness of Tetris clearing under the "hard drops only" and "20G" Tetris models. Previous results about general Tetris [7, 2] have also proven NP-hardness of Tetris survival, so in this section, we prove that, in both of these variants, Tetris survival is NP-hard using {▭, 🟨} pieces. This improves upon a result by Temprano [12] which proves hardness for "hard drops only" mode using the piece subset {🟧, 🟦, ▭, 🟥, 🟪}.

▶ **Theorem 18.** *Tetris survival is NP-complete in the "hard drops only" and "20G" game modes, even if the type of pieces in the sequence given to the player is restricted to* {▭, 🟨}.

Proof. We reduce from 3-Partition with Distinct Integers using a similar bottle structure to other proofs in this paper. From a 3-Partition with Distinct Integers instance, we create a setup consisting of $\frac{n}{3}$ width 1 buckets, each of height $4t$, separated by width 1 columns. In addition, we create a bucket of height $t-1$ on the left which is blocked by a single square and has an open square on the upper left diagonal. We add one additional column on the left to obtain an even width board. We leave two rows at the top of the board empty. See Figure 9(a) for details. Each a_i is encoded by the sequence (⊞$^{n/3+1}$, ▭^{a_i}, ⊞).

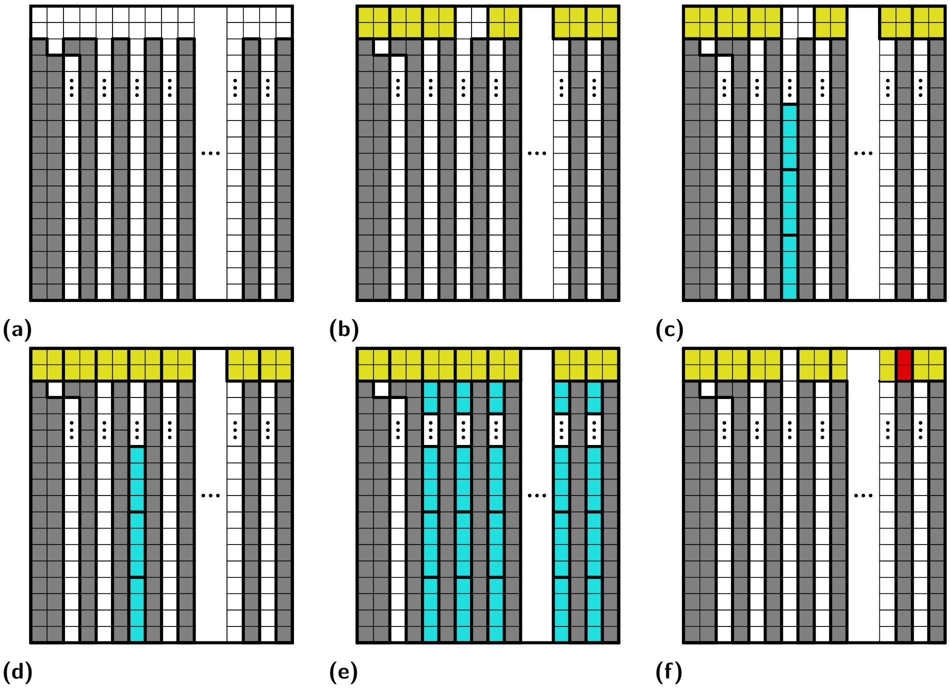


Figure 9 The bucket structure for "hard drops only" and "20G" Tetris modes. (b) shows how beginning sequence must block all but one bucket. (c) shows the result of a filling sequence. (d) shows how the closing sequence clears the top two rows. (e) shows a full board. (f) shows improper handling during the priming sequence.

The priming sequence is (⊞$^{n/3+1}$). Due to parity constraints, the player is forced to block all but one of the buckets using these pieces. They can choose to leave either one two-block wide gap or two one-block wide gaps in the top two rows. However, if they choose to leave one-block gaps, they will create spaces that can never be filled without overflowing the screen and causing a game loss (see the squares highlighted in red in Figure 9(f)), so we can assume that they will leave a 2×2 gap, as shown in Figure 9(b).

The filling sequence is (▭^{a_i}); the ▭ pieces must be placed in the pre-selected bucket, as shown in Figure 9(c). If pieces are not placed in accordance with a correct partition, then there will at some point be an extra ▭ piece which does not fit in the open bucket. This will cause the player to lose as an ▭ piece has length 4 and cannot fit in the 2×2 gap in the top 2 rows (note that no ▭ piece will stick partially out of a bucket since each bucket has a height that is a multiple of 4).

The closing sequence is (⊞). The ⊞ piece can only fit in the 2×2 square formed during the priming sequence (see Figure 9(e)), and it clears the top two rows, resetting the board to the initial state, except with a somewhat more filled bucket.

Once the player has received the entire sequence corresponding to each number in the 3-partition instance, including the final closing piece, the entire board will be full (see Figure 9(e)), with the exception of the extra inaccessible bucket. Because of this, the player must have filled each of the buckets to exactly the right height, solving the 3-partition instance, otherwise they would have lost.

For the entire duration of this sequence, every piece can be hard dropped into place as there are no overhangs in any of the buckets. Furthermore, each piece can be successfully maneuvered under 20G conditions. The ▦ pieces cannot fall down any buckets, so they can be safely slid to any location in the top 2 rows, and the ▬ pieces can move over the placed ▦ pieces to the only open bucket. ◀

▶ **Corollary 19.** *The above reduction can be extended to NP-hardness for board clearing under "hard drops only" or "20G" conditions if we also allow for* ┳ *pieces.*

▶ Remark 20. We have already proved that Tetris clearing under the "hard drops only" and "20G" models is hard with just two types of pieces. The primary reason for including hardness with a larger set of pieces is that this shows that, in both game modes, there is a board configuration and subset of pieces where the problems of surviving and clearing the board are both hard at the same time. In addition, our Tetris clearing results under the "20G" model does not include any proper subsets of {▬, ▦, ┳}, so this result is interesting in its own right.

Proof. We begin with the above proof that survivability with {▬, ▦} is hard. We extend the sequence with the finale sequence (┳, $▬^t$, $▦^{2n/3}$, $┳^3$). Figure 10 shows the clearing process. We begin by using the first ┳ piece to open the inaccessible bucket by clearing a row. We then use the ▬ pieces to fill the previously inaccessible bucket, clearing all but the final two rows in the process. The final ▬ piece protrudes one square from its bucket because of the row cleared by the first ┳. We use the ▦ pieces to fill the $\frac{2n}{3} \times 4$ space on the right of the board. Finally, we place the remaining three ┳ pieces to fill the remaining space and completely clear the board. All of these pieces can be placed in both special game modes. ◀

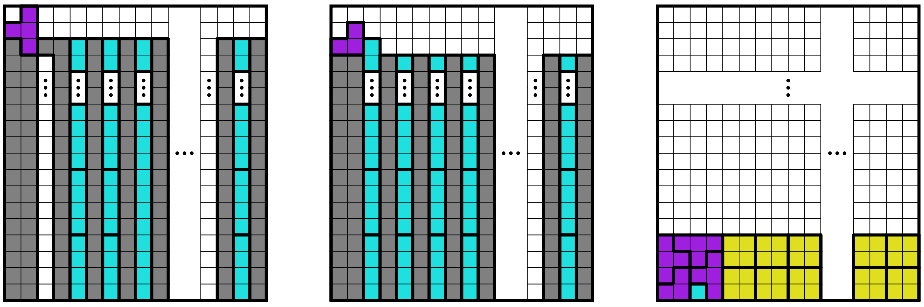

(a) (b) (c)

Figure 10 The clearing procedure for "hard drops only" and "20G" using {▬, ▦, ┳}.

6 ASP-Completeness of Tetris

Even though the reductions in Section 4 are sufficient to prove #P-hardness, the reductions are not parsimonious, so they cannot be used to prove ASP-completeness. Indeed, the "blocking" bottles paradigm likely cannot be used to show ASP-completeness as there are many ways to permute the pieces that block all but one bottle. Thus, for ASP-completeness, we turn back to the "priming" buckets paradigm in [4]:

▶ **Theorem 21.** *Tetris clearing with SRS is ASP-complete even if the type of pieces in the sequence given to the player is restricted to either* {▬, ┳, ┛} *or* {▬, ┳, ┗}.

Proof. First we discuss the {▬, ┳, ┘} case. We give a parsimonious reduction from Numerical 3-Dimensional Matching with Distinct Integers; refer to Figure 11(a), which shows the bucket structure plus rightmost columns for {▬, ┳, ┘}.

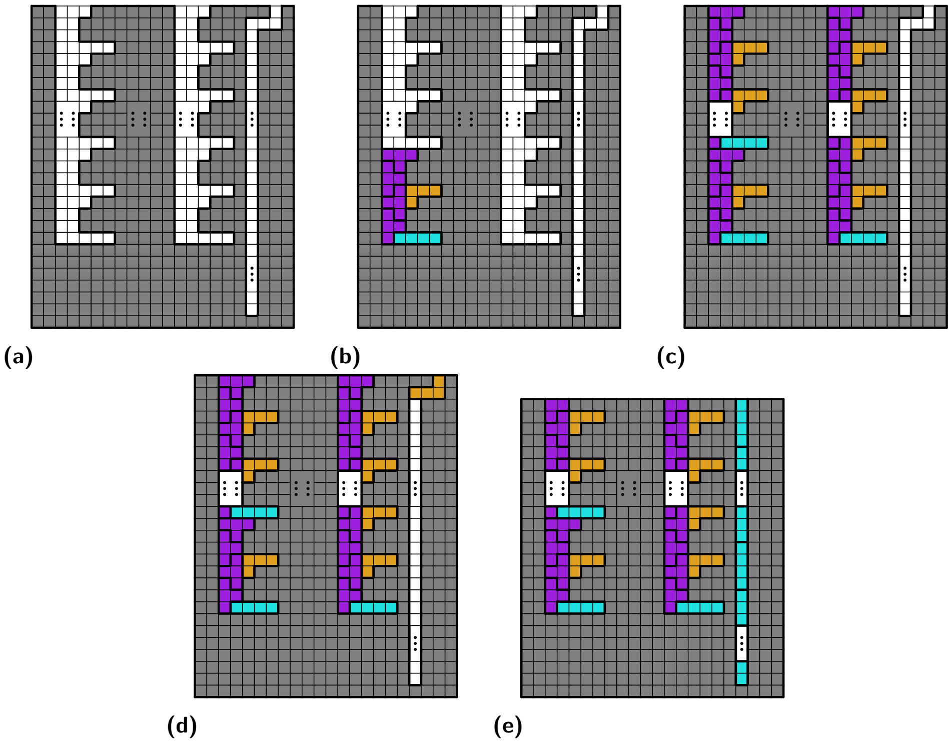


Figure 11 The bucket structure plus rightmost columns for {▬, ┳, ┘}. (b) shows how the ▬ piece must prime a bucket (requires an ▬ spin) and how the remainder of the pieces in a sequence must fit in the bucket ($m = 2$ here). (c) shows how the board looks like before the finale sequence. (d–e) show how the pieces in the finale sequence must be placed (requires an ┘ spin).

Here, a “unit” is the pattern that repeats every four rows. A positive integer m is encoded by the sequence of pieces (▬, (┳, ┘, ┳)$^{m-1}$, ┳, ┳). The finale sequence is (┘, ▬^{N}), where $N = \text{poly}(nt)$ is much larger than the height of the buckets, and is used to clear the rightmost columns after the buckets have been filled.

In this case, the ▬ piece serves as the “primer”, and must be placed as indicated in Figure 11(b) (the placement is possible due to ▬ spins). The ┳ pieces cannot be placed in a non-primed bucket without blocking off certain holes, particularly the squares in which an ▬ piece or an ┘ piece must be placed, and misplacing an ┘ piece (i.e., putting it in a different, non-primed bucket, putting it where the ┘ piece goes during the finale sequence, or putting it too high in the bucket) causes the next two ┳ pieces to block off squares in which an ▬ piece or an ┘ piece must be placed. Thus, once the ▬ piece is placed in a bucket, the placements of the rest of the pieces encoding the positive integer m are forced. Further discussion on and figures for improper piece placements can be found in the full version of our paper.

Lastly, to make the reduction parsimonious, from the instance

$$A = \{a_1, a_2, \ldots, a_n\}, B = \{b_1, b_2, \ldots, b_n\}, \text{ and } C = \{c_1, c_2, \ldots, c_n\}$$

of Numerical 3-Dimensional Matching with Distinct Integers, we scale the a_i, b_i, and c_i and add/subtract constants such that each group must consist of exactly one a_i, one b_j, and one c_k (i.e., no group with at least two elements from any of A, B, or C can sum to the target sum). In addition, we "pre-fill" all of the buckets so that the ith bucket from the left is already filled up to a_i units, and the sequence of pieces given to the player consist only of the pieces in the b_i sequences, the pieces in the c_i sequences, and the pieces in the finale sequence. This ensures that the subsets have a fixed ordering and must consist of exactly one a_i, one b_j, and one c_k. Combined with the fact that all the piece placements after each ▬ piece in a sequence encoding a positive integer are forced, this means that each solution to the Numerical 3-Dimensional Matching instance corresponds to exactly one way to clear the Tetris board, and vice versa.

Therefore, we have a parsimonious reduction from Numerical 3-Dimensional Matching with Distinct Integers to Tetris clearing with {▬, ┳, ┛}, meaning that Tetris clearing with SRS is ASP-complete even if the type of pieces in the sequence given to the player is restricted to {▬, ┳, ┛}.

We get a similar argument for {▬, ┳, ┗} by vertical symmetry; the ▬ spin required still works when mirrored even though the kick tests for ▬ are not vertically symmetric. ◀

Demos of the bucket structure can be found at `https://jstris.jezevec10.com/map/80170` for {▬, ┳, ┛} and at `https://jstris.jezevec10.com/map/83325` for {▬, ┳, ┗}.

7 Open Problems

One big open problem that still remains is the computational complexity of Tetris clearing with SRS if the player is only given one piece type (for example, if the sequence consists of entirely ┳ pieces). In this case, the "blocking" bottles paradigm no longer works, because the same piece type cannot be used both to block and to fill bottles without the reduction breaking, so a proof of hardness would involve an entirely different setup. It is also possible that Tetris clearing with SRS and with only one piece type is in P. For example, [4] conjectures polynomial time for the ▬ piece type.

Similarly, it is open whether or not Tetris clearing with SRS is ASP-complete for subsets of pieces that are not supersets of {▬, ┳, ┗} or {▬, ┳, ┛}. One could likely construct similar structures for other 3-piece subsets, but arguing whether Tetris clearing with SRS is ASP-complete for 1- or 2-piece subsets may require different ideas.

Some open questions arise regarding whether our results can be extended if we consider different objectives or add additional features. For example, can we establish results for 2-piece subsets, similar to those in Section 4, for Tetris survival? If we add a "holding" function, where the player can put one piece aside for later use, can get similar results?

Modern variants of Tetris also use different random generators to ensure that the player doesn't receive the same piece arbitrarily many times in a row. One of the simplest random generators is called a ***7-bag randomizer***. For this randomizer, the sequence of pieces is divided into groups (or "bags") of 7, each group containing one of each tetromino in one of $7! = 5{,}040$ possible orderings. Can we show NP-hardness even if the sequence of pieces has to be able to be generated from this randomizer?

References

1 Tetris, 2023. Apple TV+ movie directed by Jon S. Baird and written by Noah Pink.

2 Sualeh Asif, Michael Coulombe, Erik D. Demaine, Martin L. Demaine, Adam Hesterberg, Jayson Lynch, and Mihir Singhal. Tetris is NP-hard even with $O(1)$ rows or columns. *Journal of Information Processing*, 28:942–958, 2020.

3 Jeffrey Bosboom, Erik D. Demaine, Martin L. Demaine, Adam Hesterberg, Roderick Kimball, and Justin Kopinsky. Path puzzles: Discrete tomography with a path constraint is hard. *Graphs and Combinatorics*, 36:251–267, 2020.

4 Ron Breukelaar, Erik D. Demaine, Susan Hohenberger, Hendrik Jan Hoogeboom, Walter A. Kosters, and David Liben-Nowell. Tetris is hard, even to approximate. *International Journal of Computational Geometry and Applications*, 14(1–2):41–68, 2004. doi:10.1142/S0218195904001354.

5 Charles J. Colbourn. The complexity of completing partial Latin squares. *Discrete Applied Mathematics*, 8(1):25–30, 1984. doi:10.1016/0166-218X(84)90075-1.

6 Sopan Deb. Boy, 13, is believed to be the first to 'beat' tetris. *The New York Times*, 3 january 2024. URL: https://www.nytimes.com/2024/01/03/arts/tetris-beat-blue-scuti.html.

7 Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Adam Hesterberg, Andrea Lincoln, Jayson Lynch, and Y. William Yu. Total Tetris: Tetris with monominoes, dominoes, trominoes, pentominoes, ... *Journal of Information Processing*, 25:515–527, 2017. doi:10.2197/ipsjjip.25.515.

8 Erik D. Demaine, Fermi Ma, Erik Waingarten, Ariel Schvartzman, and Scott Aaronson. The fewest clues problem. *Theoretical Computer Science*, 748:28–39, November 2018. doi:10.1016/j.tcs.2018.01.020.

9 Ian Holyer. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981. doi:10.1137/0210054.

10 Heather Hulett, Todd G. Will, and Gerhard J. Woeginger. Multigraph realizations of degree sequences: Maximization is easy, minimization is hard. *Operations Research Letters*, 36(5):594–596, 2008. doi:10.1016/j.orl.2008.05.004.

11 Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, and Richard E. Stearns. The complexity of planar counting problems. *SIAM Journal on Computing*, 27(4):1142–1167, 1998. doi:10.1137/S0097539793304601.

12 Oscar Temprano. Complexity of a Tetris variant. arXiv:1506.07204, 2015. URL: https://arxiv.org/abs/1506.07204, arXiv:1506.07204.

13 TetrisWiki. 20G. https://tetris.wiki/20G. Accessed: 2023-09-23.

14 TetrisWiki. Super Rotation System. https://tetris.wiki/Super_Rotation_System. Accessed: 2023-09-23.

15 TetrisWiki. Tetris guideline. https://tetris.wiki/Tetris_Guideline. Accessed: 2023-09-23.

16 Wikipedia. Tetris. https://en.wikipedia.org/wiki/Tetris. Accessed: 2023-09-23.

17 Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, E86-A(5):1052–1060, 2003. Also IPSJ SIG Notes 2002-AL-87-2, 2002. URL: http://ci.nii.ac.jp/naid/110003221178/en/.