Tetris Is Hard with Just One Piece Type

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

Josh Brunner
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

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

Della Hendrickson
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

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

Abstract

We analyze the computational complexity of Tetris clearing (determining whether the player can clear an initial board using a given sequence of pieces) and survival (determining whether the player can avoid losing before placing all the given pieces in an initial board) when restricted to a single polyomino piece type. We prove, for any tetromino piece type P except for  , the NP-hardness of Tetris clearing and survival under the standard Super Rotation System (SRS), even when the input sequence consists of only a specified number of P pieces. These surprising results disprove a 23-year-old conjecture on the computational complexity of Tetris with only  pieces (although our result is only for a specific rotation system). As a corollary, we prove the NP-hardness of Tetris clearing when the sequence of pieces has to be able to be generated from a $7k$-bag randomizer for any positive integer $k \geq 1$. On the positive side, we give polynomial-time algorithms for Tetris clearing and survival when the input sequence consists of only dominoes, assuming a particular rotation model, solving a version of a 9-year-old open problem. Along the way, we give polynomial-time algorithms for Tetris clearing and survival with $1 \times k$ pieces (for any fixed k), provided the top $k-1$ rows are initially empty, showing that our  NP-hardness result needs to have filled cells in the top three rows.

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.2026.32

Related Version *Full Version*: `https://arxiv.org/abs/2603.09958`

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 – specifically Eugeniya Artemova, Cecilia Chen, Lily Chung, Jenny Diomidova, Holden Hall, Jonathan Li, Jayson Lynch, Frederick Stock, and Ethan Zhou – for helpful discussions and providing an inspiring atmosphere.

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 16-year-old Michael Artiaga being the first person to go through all 255 levels of the NES version of Tetris [12].

[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.").

In the base version of Tetris, a player is given a rectangular grid of unit-square cells (***squares***). In each round, a tetromino piece (one of ) spawns at the top of the 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, one unit down, or rotate the piece by $\pm 90°$. When a piece attempts to move downwards while any part of the piece rests on top of a filled square, the piece "locks" in place, and stops moving. If a piece "locks" in place above a certain height or where the next piece would spawn, the player loses; otherwise, the next piece spawns at the top of the grid, and play continues. Completely filling a row causes the row to clear, with all squares above that row moving downward by one unit. For more detailed rules, see [20].

Tetris has many variations, both official and unofficial [17], that change the piece types that spawn for the player. For example, *ntris* [11] and *Combinos* [16] generalize the piece types to ***k-ominoes*** (connected shapes made from gluing k unit squares edge-to-edge).

To study Tetris from a computational complexity perspective, we generally assume that the player is given an initial board state with some filled cells and either a finite sequence of pieces or an infinite number of pieces of a given type, making it a perfect-information game (as introduced in [5]). The two main objectives are "clearing" and "survival" (as introduced in [7]). In ***Tetris clearing***, we want to determine whether we can clear the entire board, either after placing all the given pieces in the finite piece sequence or after placing some number of pieces of a given type. In ***Tetris survival***, we want to determine whether the player can avoid losing from the given initial board while placing all the pieces in the given piece sequence. Previous work shows that, if the player is given a finite sequence of pieces, these problems are NP-complete, even to approximate various metrics within $n^{1-\varepsilon}$ [5], or with only 8 columns or 4 rows [3], or with unrotatable dominoes or k-ominoes for $k \geq 3$ clearing or $k \geq 4$ survival [7], or when the pieces are limited to *any two* of the seven piece types (assuming a specific rotation system) [14], also when restricted to dropping pieces or very high "gravity" [14].

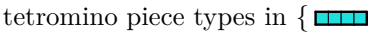
The *online* version of Tetris, which hides most of the sequence of pieces from the player (other than possibly giving the player a preview of some of the following pieces), has also been studied before, both within the Tetris community [8] and, most recently, in [9], which shows that no online algorithm can be competitive with any offline algorithm.

1.1 Our Results

In this paper, we analyze Tetris clearing and survival under the restriction that the input sequence contains exactly one polyomino piece type, resolving open problems posed in previous Tetris papers [5, 7, 14].

On the positive side, we give polynomial-time algorithms for Tetris clearing and survival when the input sequence consists of only dominoes. These results assume a particularly simple ***falling rotation model*** for dominoes, where pieces move monotonically downward; we leave open the complexity for other rotation models. The clearing result also only applies to infinite (or sufficiently long) sequences of dominoes. Other than these catches, our results solve two of the three remaining open problems posed in [7] regarding n-tris for various n. (The only remaining open problem is surviving a sequence of unrotatable dominoes.) Key subroutines we develop are strategies for both clearing and survival using only $1 \times k$ pieces, for any fixed k, provided the top $k-1$ rows of the board are empty. (Intuitively, it is easy to survive with stick-shaped pieces provided you are not too close to losing.)

On the negative side, we show that, for any tetromino piece type P except for  , Tetris clearing and survival are NP-hard even if the input sequence consists of only a specified number of P pieces. These results assume the ***Super Rotation System (SRS)*** [18], defined

by the Tetris Company's Tetris Guideline for how all modern (2001+) Tetris games should behave [19]. These results address most of the open problems posed in [14] regarding singular tetromino piece types in {  }, and disprove the 23-year-old conjecture that Tetris with only  pieces should be solvable in polynomial time [5], in notable contrast to our positive result when the top three rows are empty. However, our results rely crucially on the rotation system, in particular, the ability with SRS to climb up certain "staircases" using "kicks".

As a particular consequence, our NP-hardness results are the first to work in modern Tetris games with a ***holding*** feature, where the player can put one piece aside for later use. Because all the given pieces are of the same type, holding makes no difference on which pieces get placed in what order.

In addition, our NP-hardness result for Tetris clearing with only  pieces can be modified to prove NP-hardness for Tetris clearing if the sequence of pieces has to be able to be generated from a "7k-bag randomizer" for any positive integer $k \geq 1$. In a ***7-bag randomizer***, the sequence of pieces is divided into groups (or "bags") of 7 pieces, where each group contains exactly one of each tetromino in one of $7! = 5{,}040$ possible orderings. A ***7k-bag randomizer*** is a natural extension of such a randomizer, where the sequence is now divided into groups of $7k$ pieces, and each group contains exactly k of each tetromino in one of $\frac{(7k)!}{(k!)^7}$ possible orderings.

1.2 Outline

The structure of the rest of the paper is as follows. Section 2 details the rotation systems being considered in our paper, including the Super Rotation System (SRS) for tetrominoes and our falling rotation model for dominoes. Section 3 discusses our positive results regarding Tetris survival and clearing with $1 \times k$ pieces. Section 4 describes the 1-in-3SAT and Graph Orientation problems, which our hardness proofs reduce from. Section 5 describes our hardness result for Tetris clearing with SRS with only  pieces, along with how the hardness result can be modified to get NP-hardness for Tetris clearing if the sequence of pieces has to be able to be generated from a 7-bag randomizer or a 7k-bag randomizer for $k > 1$. Section 6 describes our hardness result for Tetris clearing with SRS with only  pieces or with only  pieces, Section 7 describes our hardness result for Tetris clearing with SRS with only  pieces, and Section 8 describes our hardness result for Tetris clearing with SRS with only  pieces or with only  pieces. Section 9 describes the modifications to our constructions that yield corresponding one-piece-type hardness results for Tetris survival with SRS.

2 Rotation Systems

2.1 Super Rotation System (SRS)

For our hardness results involving tetrominoes, we assume the standard ***Super Rotation System (SRS)*** [18].

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 1 and 2, and at [18]. 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.

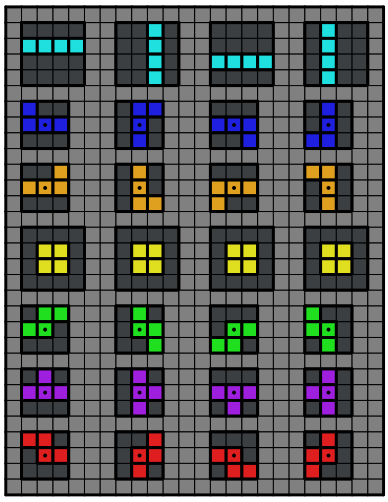


Figure 1 All tetromino pieces, in order from top to bottom: , , , , , , . The first column is the default orientation of a piece upon spawning in; each column to the right indicates a 90° rotation clockwise about the rotation center of the piece.

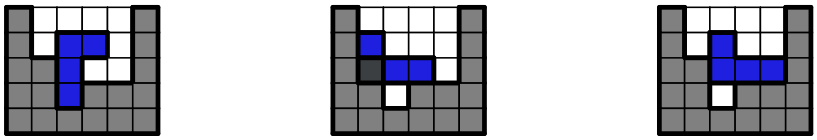


Figure 2 An example of the SRS kick system. Suppose the piece in (a) is being rotated 90° counterclockwise. Test 1 (which is $(0, 0)$) would fail, due to the dark gray square shown in (b). Test 2 (which is $(+1, 0)$) would succeed, as shown in (c), and so the piece would rotate to the position in (c).

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 our full paper.

2.2 Falling Rotation Model

For Tetris with dominoes, we assume a rotation model where pieces get monotonically lower when they rotate. Figure 3 defines this ***falling rotation model*** precisely. In general, whenever we rotate a domino, one of its lowest pixels remains in place, and the other pixel moves to a neighboring not-higher cell, assuming it is unoccupied. (If the cell is occupied, the rotation fails and the piece does not rotate.) There are two choices in this definition, which correspond to clockwise and counterclockwise rotations: when the domino was horizontal, we can choose either the left or right pixel to remain in place; when the domino was vertical, we can choose whether the neighboring cell is left or right.

Table 1 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 2 Kick data for  pieces, with same notation as Table 1.

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

3 Tetris with Dominoes and $1 \times k$

Demaine et al. [7] showed NP-hardness of Tetris clearing with unrotatable dominoes (where rotation is forbidden). They left open the complexity of survival with unrotatable dominoes, and both clearing and survival with rotatable dominoes. In this section, we give a polynomial-time algorithm for survival with rotatable dominoes, as well as a polynomial-time algorithm that determines whether a board is eventually clearable with infinitely/sufficiently many pieces.

We also give a strategy that always survives with only $1 \times k$ pieces (for any fixed k) if the $k-1$ top rows start empty, and with this strategy, we can decide in polynomial time whether it is possible to eventually fully clear the board given enough such pieces. We will use this result with $k = 2$ as a subroutine for our domino algorithms, but $k = 4$ is also an interesting contrast to our  hardness result.

3.1 Survival and Clearing with $1 \times k$ Pieces

▶ **Theorem 1.** *If the board has height at least $2k-1$, and the top $k-1$ rows are initially empty, then it is possible to survive an infinite sequence of $1 \times k$ pieces. Furthermore, there is a polynomial-time algorithm to determine whether the board can eventually be fully cleared.*

Proof. Our strategy proceeds in two phases. First, we eliminate all holes, where a ***hole*** is an empty cell that is below a filled cell. Second, we place horizontal pieces to make all columns have the same number of pieces modulo k.

For the first phase, we note that it is possible to eliminate the highest hole without creating new holes by repeatedly placing vertical pieces to clear rows containing squares above the highest hole. Then, by induction on the number of holes, we can remove all holes. Refer to Figure 4 for an example of this process.

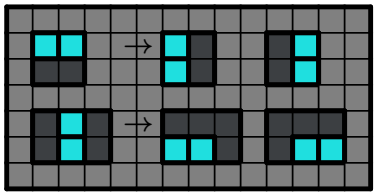


Figure 3 Falling rotation model for dominoes: when the domino on the left attempts to rotate clockwise or counterclockwise, it becomes the respective domino on the right, if that position is unobstructed.

Figure 4 Procedure for clearing holes for $k = 3$. The dots in (a) show which empty squares are holes.

Once there are no holes, it is easy to survive forever: there must be some empty cell on the bottom row (or it would have already cleared), and because it is not a hole, we can put a vertical piece in that column. By repeatedly placing pieces in the lowest available location, we can also make the height of the highest filled cell be at most $k - 1$, giving us at least k empty rows at the top because the total height of the board is at least $2k - 1$.

Now we give an overview of an algorithm that either clears the board from this position, or proves that the board was impossible to clear to begin with. The idea is to guess the number of horizontal pieces lying within the leftmost k columns to place, with our guess determining the number of horizontal pieces we must place in other columns modulo k. If the horizontal pieces create holes, we run the hole eliminating algorithm (phase one) again. Considering the effects of vertical pieces and line clears on the number of filled cells modulo k in each column, we only need to test k possible values for the number of horizontal pieces lying within the leftmost k columns to place, so we can try all possible values and determine whether clearing is possible in polynomial time. ◀

3.2 Algorithm for Clearing with Dominoes

In this subsection, we answer the open questions from [7] about surviving and clearing Tetris with only rotatable dominoes. In [7], they note that clearing a single line suffices to survive forever. We give an algorithm to decide whether it is possible to ever clear a line with an infinite sequence of domino pieces, thus deciding survivability. After clearing a line, using the algorithm from Section 3.1 gives a solution to whether fully clearing the board is possible as well.

We will use the ***falling rotation model*** from Section 2.2. Recall that the allowed rotations are any rotation that overlaps the previous position in exactly one of the two cells, and the other cell is in a lower row than it previously was.

We give an algorithm to determine whether a particular row r is ever clearable. To determine whether any row is clearable, we can simply run this algorithm separately for each row r.

We call a cell (r, c) ***reachable*** if it is possible from the initial configuration to move a domino to a position with one of the two cells of the domino in (r, c). Similarly, a pair of empty adjacent cells is a reachable domino location if it is possible from the initial configuration to

move a domino to that position. Note that we only require the domino to be able to pass through the location; it is still considered reachable even if the domino cannot stop there due to there being nothing to support it.

Call a configuration of dominoes ***supported*** if every piece has a filled cell or domino directly below it. Thus, in a supported configuration, for each domino, if it was the last one to be placed it would not fall. First, we show that as long as every domino's location is individually reachable (ignoring other dominoes in the configuration), it is possible to sequence the placements of dominoes such that we can build the given configuration. Call a domino location ***blocking*** another domino location if the second domino's position is not reachable after the first one is placed. Additionally, we say a domino location ***supports*** another domino location if the second one contains at least one cell directly above a cell in the first domino.

First we construct an ordering on all of the empty reachable cells, which we will call the ***path order***. Refer to Figure 5 for an example. We will label each cell with its position in this list. We proceed row by row, with the cells in higher rows always before lower ones. Within each row, we start with all of the cells which are directly below a reachable cell in the above row. (For the first row, this is all of the empty cells). Then we proceed outward along this row from those cells, appending each reachable to the path order cell the first time we come to it.

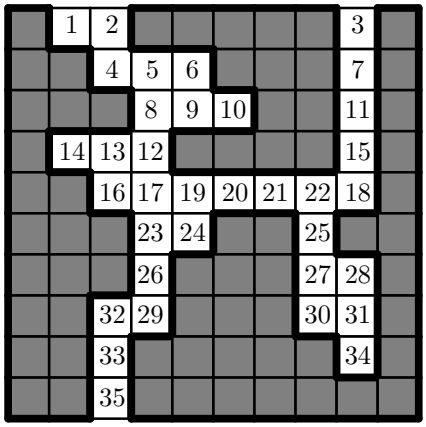


Figure 5 Path order of empty cells in an example board state. Here, a bigger number appears later in the path order.

Using this definition of path order, it is relatively straightforward to show the following:

▶ **Lemma 2.** *A domino can move along any path of empty cells that is monotonically decreasing in y value that starts from an empty top position.*

▶ **Lemma 3.** *For every reachable domino location, there is a path a domino can take to reach it that is monotonically increasing in path order.*

▶ **Lemma 4.** *Every supported configuration of dominoes is placeable.*

Now that we know that any configuration is placeable, we describe how to find a configuration that will result in a line clear if one exists. Call a column c ***good*** if the following are true:

- (r, c) is reachable; and
- either $(r + 1, c)$ is empty, reachable, and r is not the topmost row, or the nearest filled cell in column c below r is an even distance below r.

In other words, a good column is one where it is possible to stack vertical dominoes until (r, c) is occupied. A ***bad*** column is any column that is not good.

A horizontal domino ***fixes*** a bad column c if the following are true:

- the domino's location is reachable;
- it is an even distance below r; and
- there are no filled cells between it and r in column c.

In other words, a horizontal domino fixes a column c if we can now place a (possibly empty) stack of vertical dominoes on top of it that results in (r, c) being occupied. Note that it is often the case that a single horizontal domino will fix both columns it is in.

▶ **Lemma 5.** *r is clearable if and only if there is a non-overlapping set of horizontal dominoes that fix every bad column.*

Proof. In order to clear row r, every bad column must have some horizontal domino in it, since the definition of bad prevents filling it entirely with vertical dominoes. Furthermore, that domino must fix the column, because otherwise the empty space above the horizontal domino would still make the column bad. Thus, r is clearable only if there is a non-overlapping set of horizontal dominoes that fix every bad column.

Now we show that such a set of horizontal dominoes suffices. If any are not supported, add a stack of horizontal dominoes below them until they become supported. Now add a stack of vertical dominoes in every column that has r empty to fill r. By the definition of horizontal dominoes fixing columns, this will successfully fill every cell in row r in the previously bad columns, and by the definition of good this will fill every cell in row r in the previously good columns.

This is a supported configuration, so by Lemma 4, there is a sequence that they can be placed in. ◀

(a) **(b)**

Figure 6 A possible board state, along with a set of horizontal dominoes (cyan) which fixes all bad columns, additional horizontal dominoes (dark blue) so that the horizontal dominoes are supported, and vertical dominoes (red) that aid in clearing the top row.

Now, we give an algorithm to find the set of horizontal dominoes required by Lemma 5. Since each domino can only overlap two columns, we can use a dynamic program with a 2 column wide window from left to right, where we maintain all of the ways we can place horizontal dominoes fixing the last two columns as we sweep from left to right. Since we only need at most 1 horizontal domino to fix each column, there are at most $O(n^2)$ possible ways to fix the previous 2 columns, so our dynamic program takes at most $O(n^3)$ time.

Combining this algorithm with Lemma 4, Lemma 5, and Theorem 1, we get the following:

▶ **Theorem 6.** *There is a polynomial-time algorithm that determines whether a Tetris position is survivable and whether it is eventually clearable with an infinite sequence of dominoes.*

With slightly more work (particularly in the case where it is not possible to survive forever with dominoes), we can get the following corollary:

▶ **Corollary 7.** *There is a polynomial-time algorithm that determines whether a Tetris position is survivable with a sequence of k dominoes.*

4 SAT and Graph Orientation

Here, we discuss all of the SAT and Graph Orientation problems required for all of our reductions to Tetris clearing with one tetromino piece type.

Our reductions to -tris will be from or involve the following problems:

▶ **Problem 8** (1-in-3SAT). *Given a conjunctive normal form (CNF) formula ϕ consisting of m clauses with n variables such that each clause C_j contains three literals (i.e., variables or their negations), determine whether there exists a satisfying truth assignment to the variables such that each clause in ϕ has exactly one true literal.*

▶ **Problem 9** (Graph Orientation). *Given an undirected 3-regular graph $G = (V, E)$ such that V is partitioned into three (disjoint) node-subsets V_ℓ (*literal nodes*), V_C (*clause nodes*), and $V_{\overline{C}}$ (*negated-clause nodes*), determine whether there exists an orientation of G (i.e., an assignment of directions to all edges in G) such that:*

1. *every literal node in V_ℓ has in-degree either zero or three,*
2. *every clause node in V_C has in-degree one, and*
3. *every negated-clause node in $V_{\overline{C}}$ has in-degree two.*

Both problems are known to be NP-complete; 1-in-3SAT was shown to be NP-complete in [15], and Graph Orientation was shown to be NP-complete in [10].

Our reductions to -tris, -tris, and -tris will be from the following problem, where $k = 3$ for -tris and -tris and $k = 1$ for -tris:

▶ **Problem 10** (Planar Biconnected $\{\{0, 4\}$-in-4, k-in-4$\}$ Graph Orientation). *Given an undirected 4-regular planar* biconnected *graph $G = (V, E)$ such that V is partitioned into two (disjoint) node-subsets V_0 and V_k, determine whether there exists an orientation of G (i.e., an assignment of directions to all edges in G) such that every node $v \in V_0$ has indegree either 0 or 4 and every node $v \in V_k$ has indegree exactly k.*

The versions of these problems for $k = 1$ and $k = 3$ and without the biconnectivity condition were shown to be NP-complete in [13]. We note that adding the biconnectivity condition does not change the hardness of these problems:

▶ **Theorem 11.** *Planar Biconnected $\{\{0, 4\}$-in-4, 1-in-4$\}$ Graph Orientation and Planar Biconnected $\{\{0, 4\}$-in-4, 3-in-4$\}$ Graph Orientation are NP-hard.*

For -tris and -tris, we will use Planar Biconnected $\{\{0, 4\}$-in-4, 3-in-4$\}$ Graph Orientation, and for -tris, we will use Planar Biconnected $\{\{0, 4\}$-in-4, 1-in-4$\}$ Graph Orientation.

Lastly, our reductions to -tris and -tris will be from the following problem:

▶ **Problem 12** (Planar Monotone Rectilinear 3SAT). *Given a conjunctive normal form (CNF) formula ϕ consisting of m clauses with n variables that can be arranged as follows on the plane:*

1. *Every variable is a horizontal segment on the x-axis,*
2. *Every clause contains either all positive or all negative literals and is a horizontal segment off the x-axis with 3 vertical connections to the variables,*
3. *Clauses with all positive literals are above the x-axis,*
4. *Clauses with all negative literals are below the x-axis,*
5. *Segments do not cross besides at connections.*

Determine whether there exists a satisfying truth assignment to the variables such that each clause in ϕ has at least one true literal.

This problem was shown to be NP-complete in [6].

5 -tris Clearing

▶ **Theorem 13.** *Tetris clearing with SRS is NP-hard even if the type of pieces in the sequence given to the player is restricted to just pieces.*

Here, we will reduce from 1-in-3SAT, passing through Graph Orientation. Given a 1-in-3SAT instance with n variables and m clauses, we first construct a corresponding Graph Orientation instance on a grid using the reduction given by Horiyama et al. in [10, Theorem 2.3]; an example is given in Figure 7. Notice that the instance consists of two parts:

- **The upper part** – consisting of n variable cycles, vertical wires emanating from the literals, and crossovers created from the intersections of the variable cycles and vertical wires,
- **The lower part** – consisting of m copies of the clause/negated-clause construction in [10, Figure 5], each copy taking up 8 columns horizontally and no two copies using a common column.

In addition, we place a red dot at every lattice point that is on a vertex or edge of the graph. The red dots will indicate where we will put our main gadgets in the reduction to Tetris clearing.

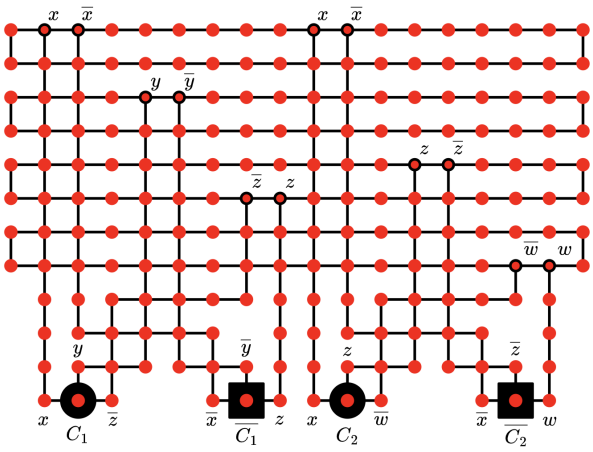


Figure 7 An example of the Graph Orientation instance, using the 1-in-3SAT instance $\phi = (x \vee y \vee \overline{z}) \wedge (x \vee z \vee \overline{w})$, including red dots at every lattice point on the graph.

5.1 General Structure

Now we construct our Tetris clearing instance, using the constructed Graph Orientation instance. We first describe the general structure of the construction, as shown in Figure 8. The main part of the construction, corresponding to the Graph Orientation instance, is indicated in green. There is a path of empty squares to the main part of the construction, which is located mostly within the leftmost 12 columns or the topmost 8 rows of the board, and which we will refer to as the "long tunnel". There are 4 additional empty squares on the second to rightmost column of the board, two at the two topmost rows of the board and two at the two bottom-most rows of the board. We give the player just enough pieces so that clearing the board requires use of all pieces; in other words, if there are M empty squares in the construction, we give the player $M/4$ pieces.

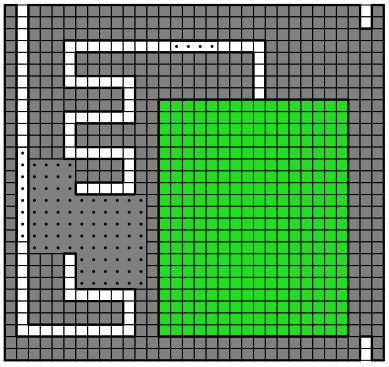


Figure 8 The general structure of the construction for Tetris clearing using only pieces. The main part of the construction is indicated in green.

We make the following observations:

- The 4 empty squares in the second-to-rightmost column must be the last squares to be filled in, after the middle rows are cleared.
- There is exactly one way to fill in/tile the long tunnel with pieces, due to the turns in the long tunnel, and it is possible (under SRS) to get pieces through the long hallway into the main part of the construction.
- If any piece locks in place in the long tunnel before the main part of the construction is completely filled, then the long tunnel becomes blocked, and we cannot clear the board anymore – no more pieces can be rotated into the main part of the construction even after some lines clear due to the two empty squares in the top-right corner and the piece blocking the long tunnel. Thus, no piece can be placed in the long tunnel before we are done filling in the main part of the construction.
- Before we place any pieces in the long tunnel, the long tunnel also prevents line clears in the rows corresponding to the main part of the construction.

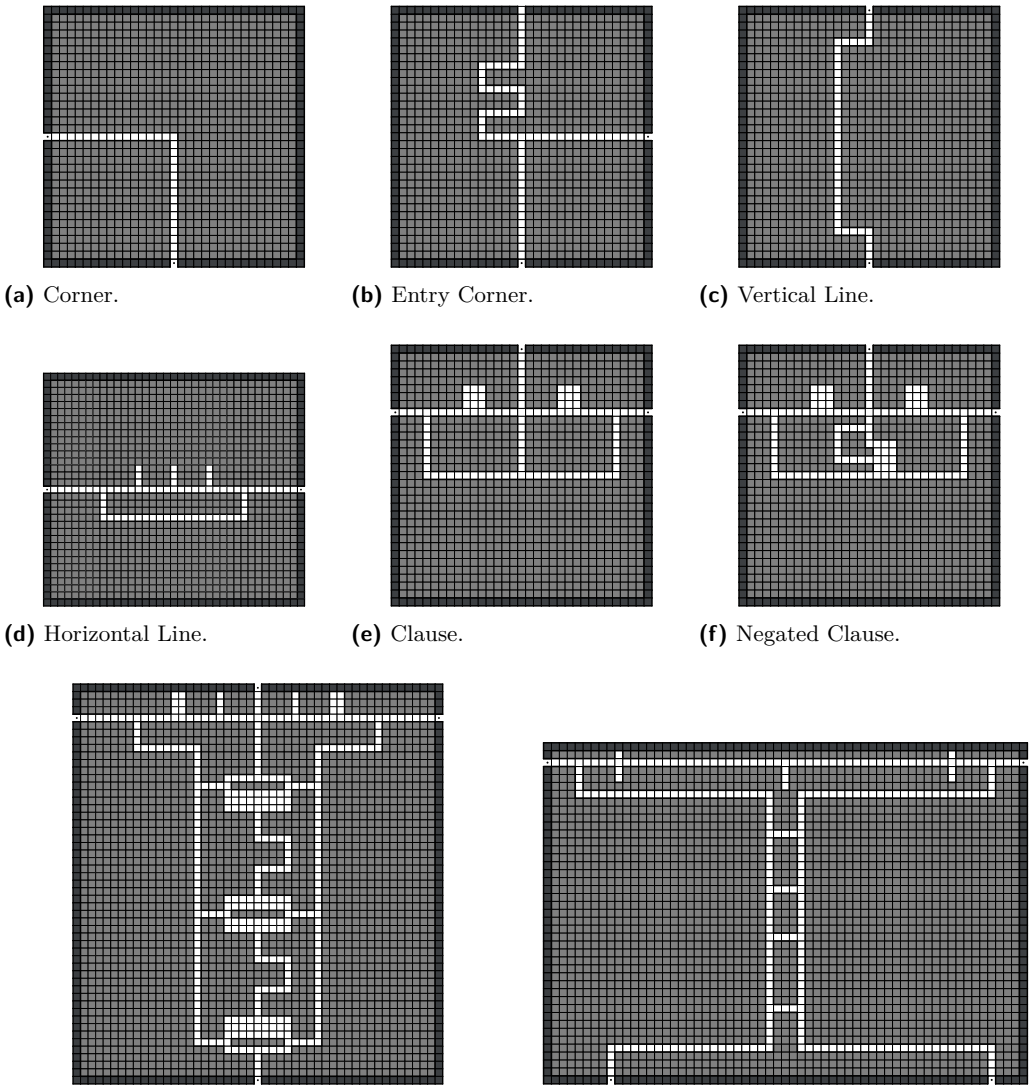
Thus, we need to be able to tile the main part of the construction with pieces exactly, with the additional restriction that we need to be able to maneuver all the pieces into place through Tetris and SRS rules.

5.2 Gadgets

Now, we introduce the gadgets used in the main part of the construction. Roughly speaking, the gadgets are similar to the gadgets used in the 2BAR-Tiling reduction in [10], with slight modifications to allow for the player to be able to maneuver pieces through the gadgets. This means that we have corner, vertical line, horizontal line, clause, negated-clause, crossover, and duplicator gadgets, along with a special entry corner gadget for pieces to enter from the long tunnel. The gadgets are shown in Figure 9.

Here, we will call the entrances into the gadgets **ports** (so for example, the horizontal line gadget has a port on the left and a port on the right); we will call the ports *up, left, right, down* (in the case of the duplicator, both of the lower ports are down ports). The ports will be indicated as an empty (white) square with a dot in the center. Note that these ports also serve as "extra" squares which help denote the "orientation" around the gadget.

We will also denote specific squares for each of the gadgets as "centers", to help with the overall description of the main part of the construction; each gadget will have one center, except for the duplicator gadget, which will have two. The centers are in the same row as any left and/or right ports and in the same column as any up and/or down port.



(a) Corner. **(b)** Entry Corner. **(c)** Vertical Line.

(d) Horizontal Line. **(e)** Clause. **(f)** Negated Clause.

(g) Crossover. **(h)** Duplicator.

Figure 9 All gadgets for -tris.

In general, the gadgets all have the following properties:

- Any traversal of the following form is possible, assuming both ports exist: up → {left, right, down}, left ↔ right (i.e., both left to right and right to left), {left, right} → down.
- The possible ways to tile the gadgets roughly match the possible ways to tile the corresponding gadgets used in the 2BAR-Tiling reduction in [10]; some gadgets have additional empty squares which are forced to be filled in a specific way across all possible tilings.
- For each possible tiling, each individual gadget can be filled with pieces under SRS, where if a gadget has an up port, the pieces come from the up port, and otherwise the pieces come from a left or right port. (In other words, the pieces are able to move to where they need to go in the tiling, particularly as the gadget is partially filled with pieces.)

5.3 Putting it all together

Now we describe the main part of the construction. Take the Graph Orientation instance and embed it on the Tetris board such that horizontally adjacent red dots are on the same row, vertically adjacent red dots are on the same column, and red dots are spaced appropriately. At each pair of horizontally adjacent red dots corresponding to a pair of literals, place a duplicator gadget such that the center squares of the duplicator gadget lie exactly on the red dots. At the topmost leftmost red dot, place an entry corner gadget such that the center square lies exactly on the red dot. Otherwise, at each red dot, place the corresponding gadget (corner, vertical line, horizontal line, clause, negated-clause, crossover) such that the center square lies exactly on the red dot.

In addition, use horizontal wires (i.e., a $1 \times n$ rectangle) or vertical wires (i.e., an $n \times 1$ rectangle) to hook up adjacent gadgets. Due to mod 4 constraints, all of the wires will have length equal to one less than a multiple of 4, so that for each wire, exactly one of the two ports at the ends of the wire must be covered by an I piece used in the tiling of the wire, which helps preserve/transmit information about the orientations of the gadgets.

5.4 Correctness

As we have now described all of the parts of the construction, we now give an overview of the correctness of the construction. From the discussion in Subsection 5.1, the whole construction can be cleared with I pieces under SRS if and only if the main part of the construction can be filled with I pieces under SRS. It is also clear that if the main part of the construction can be filled with I pieces under SRS, then we can extract a solution to the 1-in-3SAT instance in polynomial time.

Now, suppose we have a solution to the 1-in-3SAT instance. Using the solution, we can quickly (i.e., in polynomial time) get an orientation of the edges in the Graph Orientation instance, and thus the orientations of the gadgets and wires connecting the gadgets.

From here, we claim that we can maneuver and place the I pieces into their desired locations in the main part of the construction. The key idea is to fill in the gadgets from the bottom upwards in an order that prevents any remaining unfilled gadgets from being unreachable, similar to the reverse of the path order defined in Section 3.2. In particular, we fill the gadgets corresponding to the *lower part* of the construction first, which consists of m copies of the clause/negated-clause construction, before filling in the n variable cycles in the *upper part* of the construction. We omit further details due to space constraints.

Therefore, the main part of the construction can be filled with I pieces under SRS if and only if the 1-in-3SAT instance has a solution, meaning that our construction works as intended. Thus, we have shown that Tetris clearing with SRS is NP-hard even if the player is only given I pieces, as desired.

5.5 Aside: Bags

Now that we have NP-hardness for Tetris clearing even if the player is only given I pieces, we can actually prove NP-hardness for Tetris clearing under the 7-bag randomizer:

▶ **Theorem 14.** *Tetris clearing with SRS is NP-hard even if the piece sequence must be able to be generated from the 7-bag randomizer.*

We provide a high-level overview of our proof. We combine the reduction in Theorem 13 with the bottle blocking mechanisms in the reductions to two piece types in [14]. We modify the top two rows of the structure and add additional lines on top of the structure as shown in Figure 10 (the left tunnel, which is for O pieces, may have slightly different offsets depending on the parity of the number of I pieces needed in the reduction in Theorem 13).

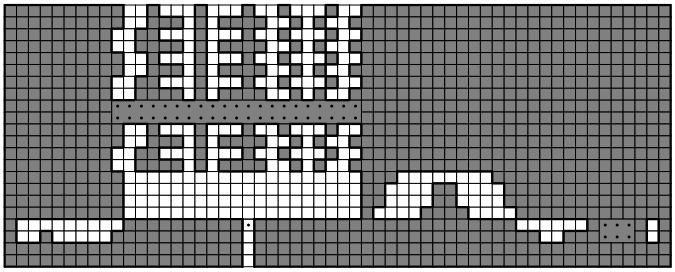


Figure 10 The additional structure for the reduction to Tetris clearing under the 7-bag randomizer.

Roughly speaking, the non- pieces must fill in their respective tunnels (in order from left to right: , , , , ,) in the topmost rows while pieces traverse through an open tunnel (there will always be at least one due to line-clearing mechanics) and into the main reduction structure. Some additional bags of 7 pieces are required at the end for cleanup after the main reduction structure gets filled and cleared. The additional structure thus does not significantly change the argument for correctness, so we get the desired NP-hardness result.

We also note that this reduction works if the player is allowed to *hold* pieces, or even *permute* the order of the pieces in the input sequence to their liking, even if the final ordering is not one that can be generated by the 7-bag randomizer. Furthermore, this structure can be slightly modified to prove NP-hardness for ***7k-bag randomizers*** for any positive integer $k \geq 1$.

6 -tris/ -tris Clearing

▶ **Theorem 15.** *Tetris clearing with SRS is NP-hard even if the type of pieces in the sequence given to the player is restricted to just pieces, or the type of pieces in the sequence given to the player is restricted to just pieces.*

We focus on pieces; the pieces case is symmetric.

Here, we will reduce from Planar Biconnected $\{\{0, 4\}$-in-4, 3-in-4$\}$ Graph Orientation. We first show the general structure of the construction, which is similar to the general structure for the -tris reduction and is shown in Figure 11. The main part of the construction, corresponding to the Planar Biconnected $\{\{0, 4\}$-in-4, 3-in-4$\}$ Graph Orientation instance, is indicated in green. Like with the -tris reduction, we have a "long tunnel" leading into the main part of the construction. There are 4 additional empty squares in the rightmost three columns of the board. We give the player just enough pieces so that clearing the board requires use of all pieces; in other words, if there are M empty squares in the construction, we give the player $M/4$ pieces.

Through similar reasoning used for the construction in the -tris reduction, we need to be able to tile the main part of the construction with pieces exactly, with the additional restriction that we need to be able to maneuver all the pieces into place through Tetris and SRS rules.

6.1 Gadgets

Now, we introduce the gadgets used in the main part of the construction. Like with the -tris gadgets, we will call the entrances **ports**, which will be indicated as empty squares with dots in their centers. We will need 0-or-4 and 3-in-4 gadgets (corresponding to the

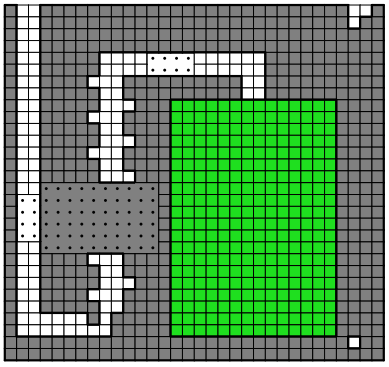
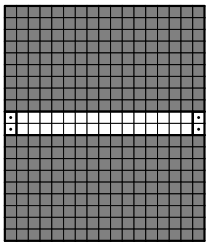
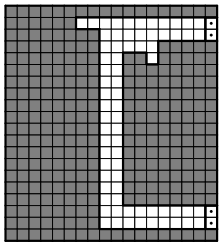


Figure 11 The general structure of the construction for Tetris clearing using only pieces. The main part of the construction is indicated in green.

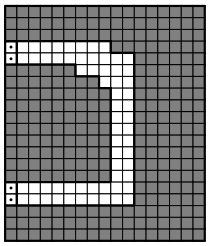
two different types of vertices), horizontal line and U-turn gadgets (corresponding to edges between vertices), and an entry corner gadget (for pieces to enter from the long tunnel), along with a special gadget called a parity fixer that helps "align" the coordinates of the ports between gadgets modulo 4 in case the coordinates of the ports modulo 4 are not the same. The gadgets are shown in Figure 12. Note that two different U-turn gadgets are needed due to asymmetries caused by piece movements and rotations.



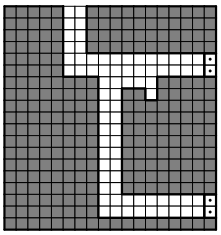
(a) Horizontal Line.



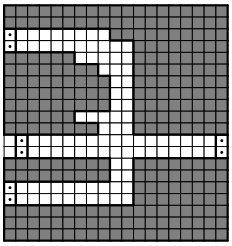
(b) U-Turn (left-down-right).



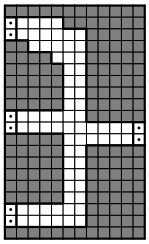
(c) U-Turn (right-down-left).



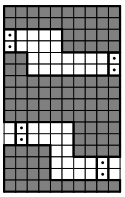
(d) Entry Corner.



(e) 0-or-4.



(f) 3-in-4.



(g) Parity fixers.

Figure 12 All gadgets for -tris.

The gadgets have the same properties regarding traversal, possible ways to be tiled, and ability to be filled given a possible tiling as the -tris gadgets, though the possible ways to be tiled now correspond to settings of edges around a vertex, in the case of the 0-or-4 gadgets and the 3-in-4 gadget, or the setting of an edge, in the case of the other gadgets.

6.2 Putting it all together

Now we describe the main part of the construction. Take the instance of Planar Biconnected $\{\{0,4\}$-in-4, 3-in-4$\}$ Graph Orientation and compute an embedding of the graph on a grid using the linear-time BICONN algorithm in [4]. The computed embedding also has some special properties, namely that if we add an artificial "root vertex" to the top-left "bend" of the topmost edge, then all vertices are on different y-coordinates, and every point (vertex or along an edge) can be reached via a "downward" (i.e., monotone decreasing y-coordinate) path from the root vertex. We will call the latter property *downward reachable*.

From here, we embed the grid graph drawing on the Tetris board with sufficient spacing around vertices, place an entry corner gadget at the root vertex, and place the 0-or-4 and 3-in-4 gadgets at their corresponding vertices (the former for vertices that must have indegree either 0 or 4, the latter at vertices that must have indegree 3). Lastly, use horizontal lines, both U-turns, and both parity fixers as necessary to connect the 0-or-4 and 3-in-4 gadgets (these play the role of edges and transmit information about the orientations of the gadgets).

6.3 Correctness

As we have now described all of the parts of the construction, we now give an overview of the correctness of the construction. From the discussion at the beginning of Section 6, the whole construction can be cleared with  pieces under SRS if and only if the main part of the construction can be filled with  pieces under SRS. It is also clear that, if the main part of the construction can be filled with  pieces under SRS, then we can extract a solution to the Planar Biconnected $\{\{0,4\}$-in-4, 3-in-4$\}$ Graph Orientation instance in polynomial time.

Now, suppose we have a solution to the Planar Biconnected $\{\{0,4\}$-in-4, 3-in-4$\}$ Graph Orientation instance. Using the solution, we can quickly (i.e., in polynomial time) get the orientations of all of the gadgets. From here, we want to show that we can maneuver and place the  pieces into their desired locations in the main part of the construction.

The key idea is to use the special properties of the graph embedding. Essentially, because the embedding is downward reachable, we can fill in all of the 0-or-4 and 3-in-4 gadgets in order based on height (from lowest to highest), filling in all other gadgets (horizontal line, U-turn, parity fixer gadgets) corresponding to edges to "lower" 0-or-4 and 3-in-4 gadgets before doing so, before filling in the entry corner itself and thus the entirety of the main part of the construction. We omit further details due to space constraints.

Therefore, the main part of the construction can be filled with  pieces under SRS if and only if the Planar Biconnected $\{\{0,4\}$-in-4, 3-in-4$\}$ Graph Orientation instance has a solution, meaning that our construction works as intended. Thus, we have shown that Tetris clearing with SRS is NP-hard even if the player is only given  pieces, as desired. We also get NP-hardness for  by symmetry.

7  -tris Clearing

▶ **Theorem 16.** *Tetris clearing with SRS is NP-hard even if the type of pieces in the sequence given to the player is restricted to just*  *pieces.*

Here, we will reduce from Planar Biconnected $\{\{0,4\}$-in-4, 1-in-4$\}$ Graph Orientation. The argument is almost the exact same as the proof for  -tris, with the only changes being the gadgets and the "long tunnel" used, so we show the long tunnel in the context of the general structure in Figure 13 and all other gadgets in Figure 14, and omit the rest of the proof due to space constraints.

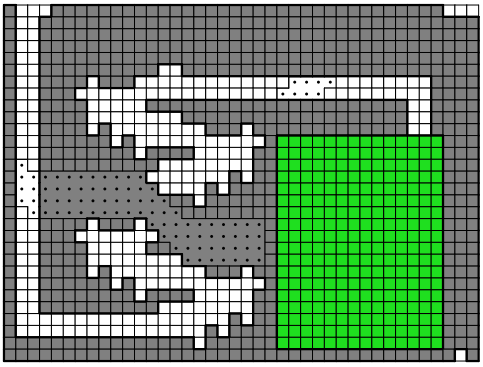
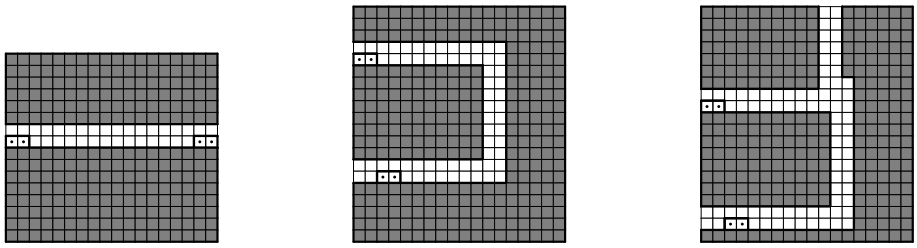
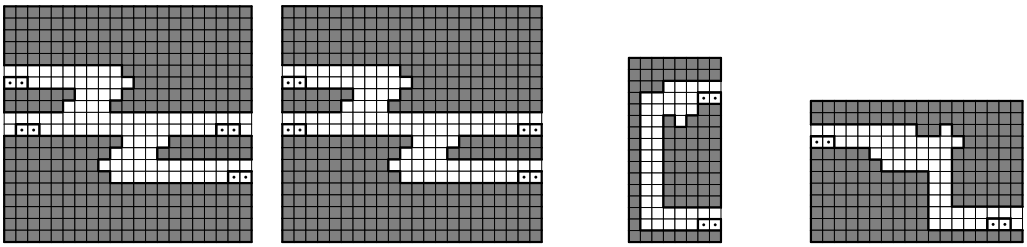


Figure 13 The general structure of the construction for Tetris clearing using only pieces. The main part of the construction is indicated in green.



(a) Horizontal Line. **(b)** U-Turn. **(c)** Entry Corner.



(d) 0-or-4. **(e)** 1-in-4. **(f)** Parity fixer 1. **(g)** Parity fixer 2.

Figure 14 All gadgets for -tris.

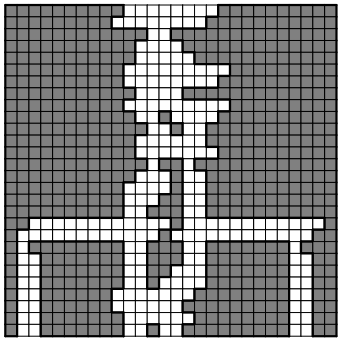
8 -tris/ -tris Clearing

▶ **Theorem 17.** *Tetris clearing with SRS is NP-hard even if the type of pieces in the sequence given to the player is restricted to just pieces, or the type of pieces in the sequence given to the player is restricted to just pieces.*

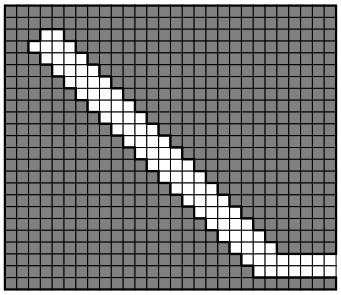
We focus on pieces; the pieces case is symmetric.

Here, we will reduce from Planar Monotone Rectilinear 3SAT. To do so, we first build variable and clause gadgets (along with wires to connect them) and a "climbable vertical structure". These gadgets are shown in Figure 15.

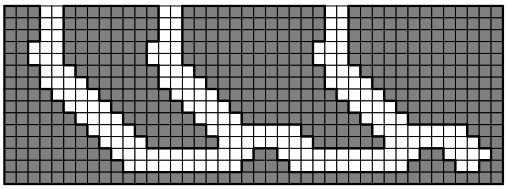
The idea behind the variable gadget is that there are two settings based on where an initial piece is placed, as shown in Figure 16, that allow pieces to traverse through one of the tunnels at the expense of the other tunnel being blocked. Before any line clears are possible, there are no possible piece placements that allow both tunnels to be traversable at the same time. Once lines are able to be cleared, the variable gadget can be collapsed into a form that allows for both tunnels to be traversed.



(a) Variable gadget with two wires (one on the "True" side, one on the "False" side).

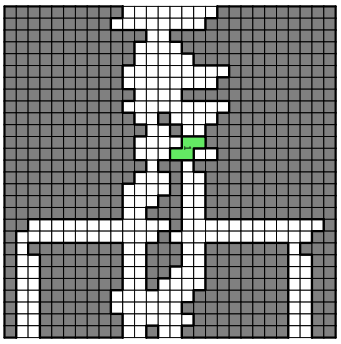


(b) Climbable vertical structure.

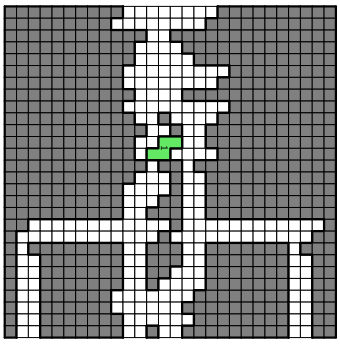


(c) Clause gadget.

Figure 15 Gadgets for -tris.



(a) "True" setting.



(b) "False" setting.

Figure 16 -tris variable settings.

For the clause gadget, there are three entrances corresponding to the three connections. Once one of the entrances is accessible, most of the clause gadget is fillable other than a constant number of squares in specific cases. The squares that might not get filled do not affect our later arguments and can be filled in at the very end if necessary. However, the squares that can get filled once at least one of the entrances is accessible contribute to some lines getting cleared that help make other parts of the structure become accessible.

The climbable vertical structure gives us a way to prevent line clears from happening before it is made accessible, and can be filled from the bottom upwards.

Now that we have our gadgets, we describe the general structure. Given an instance of Planar Monotone Rectilinear 3SAT, we can take the planar rectilinear arrangement, rotate the entire arrangement 90 degrees with the first variable segments rotating towards the top, rotate each individual clause segment 90 degrees, and rearrange the clause segments and wires such that the arrangement is still planar and rectilinear. Once we have this arrangement,

we can insert our variable and clause gadgets where there are variable and clause segments, connect the gadgets with wires, and add some broken-up wires below the last variable gadget that leads to the climbable vertical structure that extends upwards and to the left of the rest of the structure and stops two lines below the top line. We also add one additional copy of the top two lines of the variable gadget above the topmost variable gadget and some empty squares in the rightmost few columns to prevent the top two lines from being cleared prematurely. For the piece sequence, if there are M empty squares in the general structure, we give the player $M/4$  pieces so that no part of any  piece can be placed above the topmost row in the general structure if the player wants to clear the board.

8.1 Correctness

Now we prove correctness of the construction; in other words, the whole construction can be cleared with  pieces under SRS if and only if the Planar Monotone Rectilinear SAT instance has a solution.

First, if this construction can be cleared under SRS, then every clause must be accessible under some specific setting of the variable clauses. In particular, based on the way the structure is set up and the fact that no part of any  piece can be placed above the topmost row of the structure, the top two lines must be cleared last (as the empty squares in the top two lines and the rightmost few columns line up with the other empty squares in the rightmost few columns), meaning that the only way for the climbable vertical structure to be accessible is if we clear the lines corresponding to the lines the clause gadgets can clear. This means that there must be a specific way we can set the variable clauses that leads to every clause being accessible. In other words, there is an assignment of variables such that each clause has at least one true literal, meaning that the Planar Monotone Rectilinear SAT instance has a solution.

Now, suppose we have a solution to the Planar Monotone Rectilinear SAT instance. Using the solution, the general method to guarantee that we can clear the construction follows the general flow: set the variable gadget states (True/False), fill in the clause gadgets appropriately to allow for the climbable vertical structure to be accessible, then fill in the climbable vertical structure, then fill and clear out the variable gadgets, and finally perform any cleanup necessary to clear the board. We omit further details due to space constraints.

As this construction can be created in polynomial time given a Planar Monotone Rectilinear SAT instance, and the whole construction can be cleared with  pieces under SRS if and only if the Planar Monotone Rectilinear SAT instance has a solution, we have thus shown that Tetris clearing with SRS is NP-hard even if the player is only given  pieces, as desired. We also get NP-hardness for  by symmetry.

9 Survival

▶ **Theorem 18.** *For any piece type* $P \in \{$  ,  ,  ,  ,  ,  $\}$*, Tetris survival with SRS is NP-hard even if the type of pieces in the sequence given to the player is restricted to just* P*.*

We first discuss the  -tris case. Much of the proof remains the same as the proof for Tetris clearing with SRS and with just  pieces. However, the main modification we make is changing the rightmost two columns to have a checkerboard-like pattern in empty versus filled squares to prevent any  pieces from being placed in the rightmost two columns (otherwise the player will lose) and to prevent any lines from being cleared. In

addition, compared to the Tetris clearing construction, we give the player one less piece, so that the player can fill the entire board except for the empty squares in the rightmost two columns and can survive if and only if the corresponding 1-in-3SAT instance has a solution. With this modification, we get NP-hardness for Tetris survival with SRS even if the player is only given pieces.

A similar approach (modifying the rightmost two columns) yields NP-hardness for Tetris survival with SRS even if the player is only given pieces of type P for $P \in \{$, , $\}$.

For , in addition to modifying the rightmost two columns, we need to make additional modifications, as the general structure depends on line clears to access all parts of the structure, which we want to prevent in our reductions to Tetris survival. Here, we remove the climbable vertical structure, and modify each clause gadget as shown in Figure 17 such that each clause gadget is attached to a sufficiently large rectangle of A empty squares each, lengthening any "wires" as necessary (i.e., any hallways of length or width 2 not part of a variable or clause gadget or the climbable vertical structure) so that the rectangles do not intersect each other or any existing gadgets. For the piece sequence, we give the player $m(A - c\sqrt{A})$ pieces in the input sequence (where m is the number of clauses in the Planar Monotone Rectilinear SAT instance and c is some small constant). The general idea for correctness is that, due to the given number of pieces, being able to fit all given pieces is possible if and only if all clause gadgets are accessible, if and only if the Planar Monotone Rectilinear SAT instance has a solution.

Thus, we get NP-hardness for Tetris survival with SRS even if the player is only given pieces. We also get NP-hardness for by symmetry.

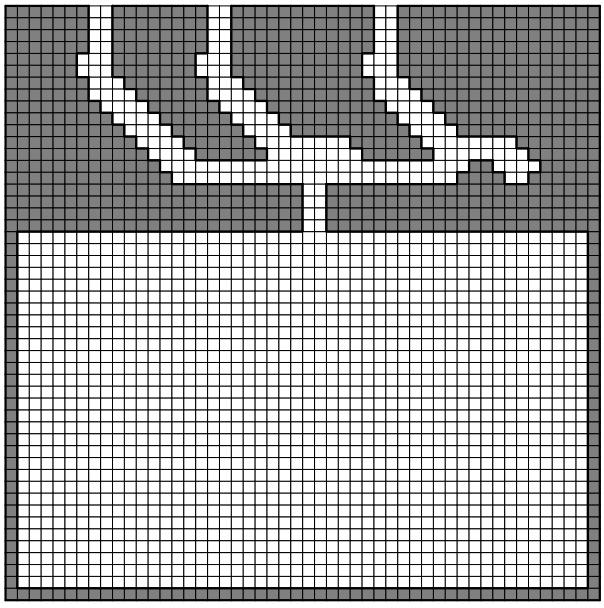


Figure 17 The (modified) clause gadget for Tetris survival with only pieces.

10 Open Problems

Some of the open problems in [7] and [14] still remain open. For rotatable dominoes, one problem that remains open is whether Tetris clearing and survival remain solvable in polynomial time if we assume a different rotation model, such as a version of SRS for

dominoes. In our polynomial-time algorithms for Tetris clearing and survival with rotatable dominoes, we assume that the rotation center only moves monotonically downwards. While the algorithm might be able to be adapted to different rotation systems that allow for the rotation center to move upwards, the details are much more tricky, and it is also possible that the problem is NP-hard.

For tetromino piece types, the main open problems of interest are the computational complexity of Tetris clearing and survival for  pieces. For clearing, while  pieces cannot rotate, thus simplifying the problem (as there are no spins involving  pieces) and leading us to believe that this is plausibly in P, it is still not straightforward to prove due to the row-clearing mechanic. For survival, we have evidence that suggests that the problem is NP-hard, as 2×2 square packing in orthogonally convex grid polygons is NP-hard [2], but there are still a few tricky details to work out, particularly in ensuring that all pieces can be successfully placed under Tetris rules.

Another open problem is whether or not Tetris clearing and survival with SRS and with just one piece type is #P-hard or ASP-complete.[2] In particular, it is unlikely that our construction is c-monious[3] for any c as each gadget type may not have the same number of ways to fill them depending on the tiling. Does there exist a different construction that is c-monious for some c?

Lastly, most of our results analyze Tetris clearing and survival with a finite input sequence. How does the complexity of Tetris clearing and survival change if the player is instead given an infinite number of pieces of a given type?

References

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

2 Mikkel Abrahamsen and Jack Stade. Hardness of packing, covering and partitioning simple polygons with unit squares. *SIAM Journal on Computing*, FOCS24:29–84, 2026. doi:10.1137/24M171824X.

3 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. doi:10.2197/ipsjjip.28.942.

4 Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry: Theory and Applications*, 9(3):159–180, 1998. doi:10.1016/S0925-7721(97)00026-6.

5 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. Conference version appeared at COCOON 2003. doi:10.1142/S0218195904001354.

6 Mark de Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In *Proceedings of the 16th Annual International Conference on Computing and Combinatorics (COCOON 2010)*, pages 216–225, Nha Trang, Vietnam, July 2010. Springer. doi:10.1007/978-3-642-14031-0_25.

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.

[2] Recall that an NP search problem is ***ASP-complete*** [21] if there is a parsimonious reduction from all NP search problems (including a polynomial-time bijection between solutions)

[3] A reduction is ***c-monious*** if it blows up the number of solutions by exactly a multiplicative factor of c.

8 Hard Drop Tetris Wiki. Playing forever. https://harddrop.com/wiki/Playing_forever. Accessed: 2026-01-08.

9 Matthias Gehnen and Luca Venier. Tetris is not competitive. In Andrei Z. Broder and Tami Tamir, editors, *Proceedings of the 12th International Conference on Fun with Algorithms (FUN 2024)*, volume 291 of *LIPIcs*, pages 16:1–16:16, La Maddalena, Italy, 2024. doi:10.4230/LIPIcs.FUN.2024.16.

10 Takashi Horiyama, Takehiro Ito, Keita Nakatsuka, Akira Suzuki, and Ryuhei Uehara. Complexity of tiling a polygon with trominoes or bars. *Discrete & Computational Geometry*, 58:686–704, 2017. doi:10.1007/s00454-017-9884-9.

11 Shaunak Kishore. ntris, 2011. URL: https://erikdemaine.org/ntris/.

12 Liv McMahon. Teenager claims first ever tetris 'rebirth'. *The British Broadcasting Corporation*, 8 october 2024. URL: https://www.bbc.com/news/articles/c70wnrg678lo.

13 MIT Hardness Group, Zachary Abel, Erik D. Demaine, Jenny Diomidova, Jeffery Li, and Zixiang Zhou. Planar graph orientation frameworks, applied to KPlumber and polyomino tiling. arXiv:SUBMITTED, 2026.

14 MIT Hardness Group, Erik D. Demaine, Holden Hall, and Jeffery Li. Tetris with few piece types. In Andrei Z. Broder and Tami Tamir, editors, *Proceedings of the 12th International Conference on Fun with Algorithms (FUN 2024)*, volume 291 of *LIPIcs*, pages 24:1–24:18, La Maddalena, Italy, 2024. doi:10.4230/LIPIcs.FUN.2024.24.

15 Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC 1978)*, pages 216–226, 1978. doi:10.1145/800133.804350.

16 TetrisWiki. Combinos. https://tetris.wiki/Combinos. Accessed: 2026-01-08.

17 TetrisWiki. List of fan games. https://tetris.wiki/List_of_fan_games. Accessed: 2026-01-08.

18 TetrisWiki. Super Rotation System. https://tetris.wiki/Super_Rotation_System. Accessed: 2026-01-08.

19 TetrisWiki. Tetris guideline. https://tetris.wiki/Tetris_Guideline. Accessed: 2026-01-08.

20 Wikipedia. Tetris. https://en.wikipedia.org/wiki/Tetris. Accessed: 2026-01-08.

21 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/.