International Journal of Computational Geometry & Applications

TETRIS IS HARD, EVEN TO APPROXIMATE*

RON BREUKELAAR
Leiden Institute of Advanced Computer Science
Universiteit Leiden
P.O. Box 9512
2300 RA Leiden, The Netherlands
rbreukel@liacs.nl

ERIK D. DEMAINE and SUSAN HOHENBERGER
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
32 Vassar Street
Cambridge, MA 02139, USA
{edemaine,srhohen}@theory.lcs.mit.edu

HENDRIK JAN HOOGEBOOM and WALTER A. KOSTERS
Leiden Institute of Advanced Computer Science
Universiteit Leiden
P.O. Box 9512
2300 RA Leiden, The Netherlands
{hoogeboo,kosters}@liacs.nl

DAVID LIBEN-NOWELL
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
32 Vassar Street
Cambridge, MA 02139, USA
dln@theory.lcs.mit.edu

Received 23 September 2003
Communicated by Binhai Zhu

In the popular computer game of *Tetris*, the player is given a sequence of tetromino pieces and must pack them into a rectangular gameboard initially occupied by a given configuration of filled squares; any completely filled row of the gameboard is cleared and all filled squares above it drop by one row. We prove that in the offline version of Tetris, it is NP-complete to maximize the number of cleared rows, maximize the number of tetrises (quadruples of rows simultaneously filled and cleared), minimize the maximum height of an occupied square, or maximize the number of pieces placed before

*This paper is the merged version of two previous papers: "Tetris is Hard, Even to Approximate" by Erik D. Demaine, Susan Hohenberger, and David Liben-Nowell, and "Tetris is Hard, Made Easy" by Ron Breukelaar, Hendrik Jan Hoogeboom, and Walter A. Kosters.[1,2]

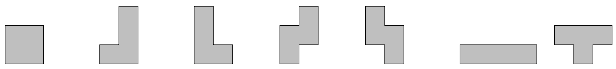


Fig. 1. The tetrominoes Sq ("square"), LG ("left gun"), RG ("right gun"), LS ("left snake"), RS ("right snake"), I ("I"), and T ("T").

the game ends. We furthermore show the extreme inapproximability of the first and last of these objectives to within a factor of $p^{1-\varepsilon}$, when given a sequence of p pieces, and the inapproximability of the third objective to within a factor of $2-\varepsilon$, for any $\varepsilon > 0$. Our results hold under several variations on the rules of Tetris, including different models of rotation, limitations on player agility, and restricted piece sets.

Keywords: Tetris; computational complexity; algorithmic combinatorial game theory.

1. Introduction

Tetris[3] is a popular computer game invented by mathematician Alexey Pazhitnov in the mid-1980s. Tetris is one of the most widespread computer games ever created. By 1988, just a few years after its invention, it was already the best-selling game in the United States and England. Over 50 million copies have been sold worldwide. (Incidentally, Sheff gives a fascinating account of the tangled legal debate over the profits, ownership, and licensing of Tetris.[4])

In this paper, we embark on the study of the computational complexity of playing Tetris. We consider the *offline* version of Tetris, in which the sequence of pieces that will be dropped is specified in advance. Our main result is a proof that optimally playing offline Tetris is NP-complete, and furthermore is highly inapproximable.

The game of Tetris. Concretely, the game of Tetris is as follows. (We give precise definitions in Section 2, and discuss some variants on these definitions in Section 6.) We are given an initial *gameboard*, which is a rectangular grid with some gridsquares filled and some empty. (In typical Tetris implementations, the gameboard is 20-by-10, and "easy" levels have an initially empty gameboard, while "hard" levels have non-empty initial gameboards, usually with the gridsquares below a certain row filled independently at random.)

A sequence of *tetrominoes*—see Figure 1—is generated, typically probabilistically; the next piece appears in the middle of the top row of the gameboard. The piece falls, and as it falls the player can rotate the piece and slide it horizontally. It stops falling when it lands on a filled gridsquare, though the player has a final opportunity to slide or rotate it before it stops moving permanently. If, when the piece comes to rest, all gridsquares in an entire row h of the game board are filled, row h is *cleared*. All rows above h fall one row lower; the top row of the gameboard is replaced by an entirely unfilled row. As soon as a piece is fixed in place, the next piece appears at the top of the gameboard. To assist the player, typically a one-piece *lookahead* is provided: when the ith piece begins falling, the identity of the $(i+1)$st piece is revealed.

A player *loses* when a new piece is blocked by filled gridsquares from entirely entering the gameboard. Normally, the player can never win a Tetris game, since pieces continue to be generated until the player loses. Thus the player's objective is to maximize his or her score, which increases as pieces are placed and as rows are cleared.

Our results. In this paper, we introduce the natural full-information (offline) version of Tetris: we have a *deterministic, finite* piece sequence, and the player knows the identity and order of all pieces that will be presented. (*Games* magazine has, incidentally, posed several Tetris puzzles based on the offline version of the game.[5]) We study the offline version because its hardness captures much of the difficulty of playing Tetris; intuitively, it is only easier to play Tetris with complete knowledge of the future, so the difficulty of playing the offline version suggests the difficulty of playing the online version. It also naturally generalizes the one-piece lookahead of implemented versions of Tetris.

It is natural to generalize the Tetris gameboard to m-by-n, for variable m and n, since a relatively simple dynamic program solves the $m \cdot n = O(1)$ case in time polynomial in the number of pieces. Furthermore, in an attempt to consider the inherent difficulty of the game—and not any accidental difficulty due to the limited reaction time of the player—we begin by allowing the player an arbitrary number of shifts and rotations before the current piece drops in height. (We will restrict these moves to realistic levels later.)

In this paper, we prove that it is NP-complete to optimize any of several natural objective functions for Tetris: (1) maximizing the number of rows cleared while playing the given piece sequence; (2) maximizing the number of pieces placed before a loss occurs; (3) maximizing the number of *tetrises*—the simultaneous clearing of four rows; and (4) minimizing the height of the highest filled gridsquare over the course of the sequence. We also prove the extreme inapproximability of the first two (and the most natural) of these objective functions: given an initial gameboard and a sequence of p pieces, for any constant $\varepsilon > 0$, it is NP-hard to approximate to within a factor of $p^{1-\varepsilon}$ the maximum number of pieces that can be placed without a loss, or the maximum number of rows that can be cleared. We also show that it is NP-hard to approximate the minimum height of a filled gridsquare to within a factor of $2 - \varepsilon$.

To prove these results, we first show that the cleared-row maximization problem is NP-hard, and then give extensions of our reduction for the remaining objectives. Our initial proof of hardness proceeds by a reduction from 3-Partition, in which we are given a set S of $3s$ integers and a bound T, and asked to partition S into s sets of three numbers each, so that the sum of the numbers in each set is exactly T. Intuitively, we define an initial gameboard that forces pieces to be placed into s piles, and give a sequence of pieces so that all of the pieces associated with each integer must be placed into the same pile. The player can clear all rows of the gameboard if and only if all s of these piles have the same height. A key difficulty in our reduction is that there are only a constant number of piece types, so any interesting component of a desired NP-hard problem instance must be encoded by a sequence of multiple pieces. The bulk of our proof of soundness is devoted to showing that, despite the decoupled nature of a sequence of Tetris pieces, the only way to possibly clear the entire gameboard is to place in a single pile all pieces associated with each integer.

Our reduction is robust to a wide variety of modifications to the rules of the game. In particular, we show that our results also hold in the following settings: (1) with restricted player agility—allowing only one rotation/translation move before each piece drops in height; (2) under a wide variety of different rotation models—including the somewhat non-intuitive model that we have observed in real Tetris implementations; and (3) without any losses—i.e., with an infinitely tall gameboard; and (4) when the piece set is restricted to $\{\mathsf{LG}, \mathsf{RG}, \mathsf{I}, \mathsf{Sq}\}$, plus at least one other piece. (A more complicated reduction using similar ideas to the one in this paper also establishes hardness when the piece set is restricted to $\{\mathsf{LG}, \mathsf{LS}, \mathsf{I}, \mathsf{Sq}\}$ or $\{\mathsf{RG}, \mathsf{RS}, \mathsf{I}, \mathsf{Sq}\}$, plus at least one other piece.[1])

Related work: Tetris. This paper is, to the best of our knowledge, the first consideration of the complexity of playing Tetris. Kostreva and Hartman consider Tetris from a control-theoretic perspective, using dynamic programming to choose the "optimal" move, using a heuristic measure of configuration quality.[6] Other previous work has concentrated on the possibility of a *forced eventual loss* (or a *perpetual loss-avoiding strategy*) in the online, infinite version of the game. In other words, under what circumstances can the player be forced to lose, and how quickly? Brzustowski has characterized all one-piece (and some two-piece) piecesets for which there are perpetual loss-avoiding strategies.[7] He has also shown that, if the machine can adversarially choose the next piece (following the lookahead piece) in reaction to the player's moves, then the machine can force an eventual loss using any pieceset containing $\{\mathsf{LS}, \mathsf{RS}\}$. Burgiel has strengthened this, showing that an alternating sequence of LS's and RS's will eventually cause a loss, in any gameboard of width $2n$ for odd n, regardless of the player's strategy.[8] This implies that, if pieces are chosen independently at random with a non-zero probability mass assigned to each of LS and RS, then there is a forced eventual loss with probability one for any such gameboards.

Related work: other games and puzzles. A number of other popular one-player computer games have been shown to be NP-hard, most notably Minesweeper—or, more precisely, the Minesweeper "consistency" problem.[9] See the survey of the second author for a summary of other games and puzzles that have been studied from the perspective of computational complexity.[10] These results form the emerging area of *algorithmic combinatorial game theory*, in which many new results have

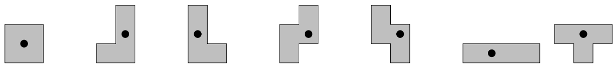


Fig. 2. Piece *centers*.

been established in the past few years, e.g., Zwick's positive results on optimal strategies for the two-player block-stacking game *Jenga*.[11]

2. Rules of Tetris

Here we rigorously define the game of Tetris, formalizing the intuition of the previous section. While tedious, we feel that such rigor is necessary so that the many subtle nuances of Tetris become transparent (following immediately from the rules). For concreteness, we give specific rules in this section, but in fact the remainder of this paper is robust to a variety of modifications to these rules; in Section 6, we discuss some variations on these rules for which our results still apply.

The gameboard. The *gameboard* is a grid of m rows and n columns, indexed from bottom-to-top and left-to-right. The $\langle i, j \rangle$th *gridsquare* is either *unfilled* (*open*, *unoccupied*) or *filled* (*occupied*). In a legal gameboard, no row is completely filled, and there are no completely empty rows that lie below any filled gridsquare. We consider all gridsquares outside the gameboard as always-occupied sentinels; this ensures that pieces can never move outside the boundaries of the gameboard.

Game pieces. The seven Tetris pieces, shown in Figure 1, are exactly those connected rectilinear polygons that can be created by assembling four 1-by-1 gridsquares, up to rotational symmetry. The *center* of each piece is shown in Figure 2. A *piece state* $P = \langle t, o, \langle i, j \rangle, f \rangle$ is a 4-tuple, consisting of:

(1) a *piece type* $t \in \{\mathsf{Sq}, \mathsf{LG}, \mathsf{RG}, \mathsf{LS}, \mathsf{RS}, \mathsf{I}, \mathsf{T}\}$.
(2) an *orientation* $o \in \{0°, 90°, 180°, 270°\}$, the number of degrees clockwise from the piece's *base orientation* (shown in Figure 1).
(3) a *position* $\langle i, j \rangle \in \{1, \ldots, m\} \times \{1, \ldots n\}$ of the piece's center on the gameboard. (The *position* of a Sq is the location of the upper-left gridsquare of the Sq, since its center falls on the boundary of four gridsquares rather than in the interior of one.)
(4) the value $f \in \{\textit{fixed}, \textit{unfixed}\}$, indicating whether the piece can continue to move.

In an *initial piece state*, the piece is in its base orientation, and the initial position places the highest gridsquares of the piece into row m, and the center into column $\lfloor n/2 \rfloor$, and the piece is unfixed.

Rotating pieces. A *rotation model* is a computable function $R : \langle P, \theta, B\rangle \mapsto P'$, where P and P' are piece states, $\theta \in \{-90°, 90°\}$ is the rotation angle, and B is a gameboard. We impose the following conditions on R:

(1) If $P = \langle t, o, \langle i, j\rangle, f\rangle$ and the rotation is *legal*, then $P' = \langle t, (o + \theta) \bmod 360°, \langle i', j'\rangle, f\rangle$ for some i' and j'. If the rotation is *illegal*, then $P' = P$.
(2) A rotation's legality depends only on the state of an $O(1)$-sized *neighborhood* of the piece P.
(3) If all the gridsquares in the neighborhood of P are unfilled, then the rotation is legal.
(4) If the rotation is legal, then P' does not occupy any gridsquare already filled in B.
(5) The function R must be *reasonable*:
    (a) A piece cannot "jump" from one region to another. Namely, the pieces P and P' must be connected by a path of unfilled gridsquares in B.
    (b) A piece cannot "squeeze past" an overly constricted portion of the gameboard. Consider a simple path of filled gridsquares that connect the (filled) gridsquare $\langle i, j-1\rangle$ to the (filled) gridsquare $\langle i, j+1\rangle$, where $\langle i, j\rangle$ is unfilled. Suppose that there are other unfilled gridsquares found inside the region R of the gameboard defined by this path plus the gridsquare $\langle i, j\rangle$. Thus R is an unfilled section of the gameboard which is almost disconnected—i.e., only the unfilled gridsquare $\langle i, j\rangle$ connects this region to the other unfilled gridsquares in the gameboard. The second requirement for reasonability is that no piece except an I can rotate "into" the region R—i.e., no piece other than an I can rotate to occupy any gridsquare of R which is not in column i.

For now, we will consider the *instantaneous rotation model*: fix the piece center (shown in Figure 2), and rotate the piece around that point. The position after rotation is unchanged—i.e., $\langle i', j'\rangle = \langle i, j\rangle$. A rotation is illegal only if it violates Condition 4. It is clear by inspection that this model satisfies the reasonability requirements. Note that our proof assumes an arbitrary reasonable rotation model; in Section 6, we discuss a number of important such models.

Playing the game. The following moves are legal for a piece $P = \langle t, o, \langle i, j\rangle, \textit{unfixed}\rangle$, with current gameboard B:

(1) A *rotation*. The new piece state is $R(P, \pm 90°, B)$.
(2) A *translation*. If the gridsquares adjacent to P are open in B, then we can translate P by one column. The new piece state is $\langle t, o, \langle i, j \pm 1\rangle, \textit{unfixed}\rangle$.
(3) A *drop* by one row, if all of the gridsquares beneath the piece are open in B. The new piece state is $\langle t, o, \langle i - 1, j\rangle, \textit{unfixed}\rangle$.
(4) A *fix*, if at least one gridsquare immediately below the piece is filled in B. The new piece state is $\langle t, o, \langle i, j\rangle, \textit{fixed}\rangle$.

No moves are legal for a piece $P = \langle t, o, \langle i, j \rangle, \mathit{fixed} \rangle$.

A *trajectory* σ of a piece P is a sequence of legal moves starting from an initial state and ending with a fix move. The result of a trajectory for a piece P on gameboard B is a new gameboard B', defined as follows:

(1) The new gameboard B' is initially B with the gridsquares of P filled.
(2) If the piece is fixed so that, for some row r, every gridsquare in row r of B' is full, then row r is *cleared*. For each $r' \geq r$, replace row r' of B' by row $r' + 1$ of B'. Row m of B' is an empty row. Multiple rows may be cleared by the fixing of a single piece.
(3) If the next piece's initial state is blocked in B', the game ends and the player *loses*.

For a *game* $\langle B_0, P_1, \ldots, P_p \rangle$, a *trajectory sequence* Σ is a sequence $B_0, \sigma_1, B_1, \ldots, \sigma_p, B_p$ so that, for each i, the trajectory σ_i for piece P_i on gameboard B_{i-1} results in gameboard B_i. However, if there is a losing move σ_q for some $q \leq p$ then the sequence Σ terminates at B_q instead of B_p.

The Tetris problem. We will consider a variety of different objectives for Tetris (e.g., maximizing the number of cleared rows, maximizing the number of pieces placed without a loss, etc.). For the decision version of a particular objective Φ, the TETRIS problem is formally as follows:

Given: A Tetris game $\mathcal{G} = \langle B, P_1, P_2, \ldots, P_p \rangle$.
Output: Does there exist a trajectory sequence Σ so that $\Phi(\mathcal{G}, \Sigma)$ holds?

We say that an objective function Φ is *acyclic* when, for all games $\mathcal{G}$, if there is a trajectory sequence Σ so that $\Phi(\mathcal{G}, \Sigma)$ holds, then there is a trajectory sequence Σ' so that $\Phi(\mathcal{G}, \Sigma')$ holds and there are no repeated piece states in any trajectory in Σ'. Most interesting Tetris objective functions are acyclic; in fact, many depend only on the final placement of each piece.

An objective function Φ is *checkable* when, given a game $\mathcal{G}$ and a trajectory sequence Σ, we can compute the truth value of $\Phi(\mathcal{G}, \Sigma)$ in time $\text{poly}(|\mathcal{G}|, |\Sigma|)$.

Lemma 1. *For any checkable acyclic objective* Φ*, we have* TETRIS $\in \mathsf{NP}$.

Proof. We are given a Tetris game $\mathcal{G} = \langle B, P_1, \ldots, P_p \rangle$. Here is an NP algorithm for TETRIS:

Guess an acyclic trajectory sequence Σ, and confirm that Σ is a legal, acyclic trajectory in time $\text{poly}(|\Sigma|)$. Confirming that all rotations in Σ are legal depends on the computability of the rotation function, and the fact that legality can only depend on the constant-sized neighborhood of the piece.

Since Σ is acyclic, each of its p trajectories can only contain at most $4 \cdot |B| + 1$ states—unfixed once in each position and each orientation, and one final fixed state. Thus $|\Sigma| = \text{poly}(|\mathcal{G}|)$. Since Φ is checkable, we can then in time $\text{poly}(|\mathcal{G}|, |\Sigma|) =$

poly($|\mathcal{G}|$) verify that $\Phi(\mathcal{G}, \Sigma)$ holds, and since Φ is acyclic, guessing an acyclic trajectory sequence Σ suffices. □

The $\Phi(\mathcal{G}, \Sigma)$ that we will initially concern ourselves with is the following: in the game $\mathcal{G}$, does the trajectory sequence Σ clear at least k rows without incurring a loss? (In Section 6, we will consider a variety of other objective functions.) This objective is acyclic because it only depends on the fixed piece state at the end of each trajectory, so the piece's path in the trajectory is irrelevant, and is checkable since a simple scan of the status of the gameboard after each trajectory allows one to count the number of rows cleared by Σ on $\mathcal{G}$.

3. NP-Completeness of Tetris

3.1. *The Reduction*

In this section, we define a mapping from instances of 3-Partition [12] to instances of Tetris. Recall the 3-Partition problem:

Given: A sequence $a_1, \ldots, a_{3s}$ of non-negative integers and a non-negative integer T, so that $T/4 < a_i < T/2$ for all $1 \leq i \leq 3s$ and so that $\sum_{i=1}^{3s} a_i = sT$.

Output: Can $\{1, \ldots, 3s\}$ be partitioned into s disjoint subsets $A_1, \ldots, A_s$ so that, for all $1 \leq j \leq s$, we have $\sum_{i \in A_j} a_i = T$?

We limit our attention to 3-Partition instances for which T is divisible by four, for technical reasons that will become apparent when we define the gameboard later in this section. We can map an arbitrary 3-Partition instance into one obeying this requirement by multiplying T and each a_i by 4. This mapping does not affect whether or not the instance has a valid 3-partition.

Note that, even after this multiplication, we still have that $T/4 < a_i < T/2$ for all $1 \leq i \leq 3s$ and so that $\sum_{i=1}^{3s} a_i = sT$. This guarantees that, for any set $S \subseteq \{1, \ldots, 3s\}$, if $\sum_{i \in S} a_i = T$ then $|S| = 3$.

We choose to reduce from this problem because it is NP-hard to solve 3-Partition even if the inputs a_i and T are provided in unary:

Theorem 1 (Garey and Johnson[13]). 3-Partition *is* NP*-complete in the strong sense.* □

Given a 3-Partition instance $\mathcal{P} = \langle a_1, \ldots, a_{3s}, T \rangle$ with T divisible by four, we will produce a Tetris game $\mathcal{G}(\mathcal{P})$ whose gameboard can be completely cleared precisely if $\mathcal{P}$ is a "yes" instance of 3-Partition.

The initial gameboard is shown in Figure 3. Intuitively, there are s *buckets* corresponding to the sets $A_1, \ldots, A_s$ for the 3-Partition problem. The piece sequence will consist of a number of tetrominoes corresponding to each a_i, chosen carefully so that all pieces corresponding to a_i must be placed into the same bucket. There is a legal 3-partition for $a_1, \ldots, a_{3s}$ exactly when the piles of pieces in each bucket have the same height. The last three columns of the gameboard form a *lock* which

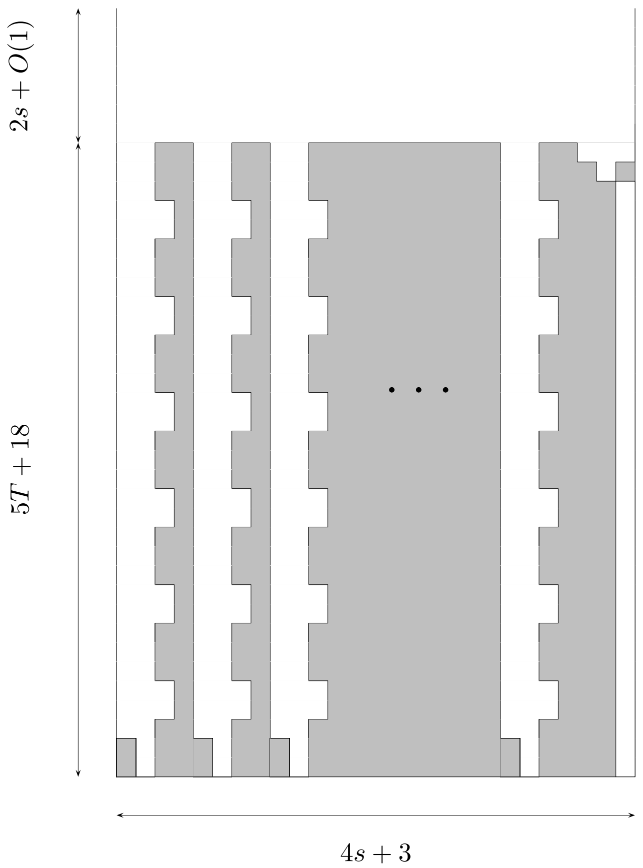


Fig. 3. The initial gameboard for a Tetris game mapped from an instance of 3-PARTITION.

prevents any rows from being cleared until the end of the piece sequence; if all buckets are filled exactly to the same height, then the entire board can be cleared using the last portion of our piece sequence. Formally, our game $\mathcal{G}$ consists of the following:

Initial board: Our gameboard will have $5T + 18 + 2s + O(1)$ rows and $4s + 3$ columns. Intuitively, the factor of five in the height is because each a_i will be represented by $a_i + 1$ blocks of five rows and three columns each; since the three elements from A_j sum to T, this is $5(T + 3) = 5T + 15$. In addition to these $5T + 15$ rows, there are three rows at the bottom ensuring that the initial blocks are placed correctly.

The top $2s + O(1)$ rows—the $O(1)$ depends linearly on the size of the $O(1)$-sized neighborhood in the rotation model—are initially empty, and are included solely as a staging area in which to rotate and translate pieces before they fall into the bottom $5T + 18$ rows. We will not mention them again in the construction (and, below, the *highest* row is the $(5T + 18)$th). Our choice of $2s + O(1)$ as the number of staging rows will be discussed in Section 6.

The remainder of the initial board can be thought of in $s+1$ logical pieces, the first s of which are four columns wide and the last of which is three columns wide. The first s logical pieces are *buckets*, arranged in the following four-column pattern:

- the first column is empty except that the two lowest rows are full;
- the second column is completely empty;
- the third column is full in each row $h \equiv \{1, 2, 3\} \pmod 5$ and empty in each $h \equiv \{0, 4\} \pmod 5$;
- the fourth column is completely full;

We refer to adjacent unfilled rows of column 3 as a *notch*. Each bucket contains $T + 3$ notches.

The last logical piece is a three-column *lock*, and consists of the following:

- the first column is full except that the highest row is empty;
- the second column is full except that the two highest rows are empty;
- the third column is empty except that the second-highest row is full.

Pieces: The sequence of pieces for our game consists of a sequence of pieces for each a_i, followed by a number of additional pieces after all the a_i's. For each integer $a_1, \ldots, a_{3s}$, we have the following pieces:

- the *initiator*, which consists of a RG;
- the *filler*, which consists of the sequence $\langle \mathsf{Sq}, \mathsf{LG}, \mathsf{Sq} \rangle$ repeated a_i times;
- the *terminator*, which consists of the sequence $\langle \mathsf{Sq}, \mathsf{I} \rangle$.

These pieces are given for a_1, a_2, etc., in exactly this order. After the pieces corresponding to a_{3s}, we have the following pieces:

- s successive RG's;
- one T;
- $(5T + 16)/4$ successive I's. (We required that T be divisible by four so that this would be an integral number of pieces. We can remove this technical requirement on T at the cost of a slightly more complicated initial gameboard: by adding, at the bottom of the gameboard, $(-T) \bmod 4$ rows filled in each column except the last, we guarantee that the height of the unfilled segment of the last column is divisible by four.)

Lemma 2. *The game $\mathcal{G}(\mathcal{P})$ is polynomial in the size of $\mathcal{P}$, and can be constructed from $\mathcal{P}$ in polynomial time.*

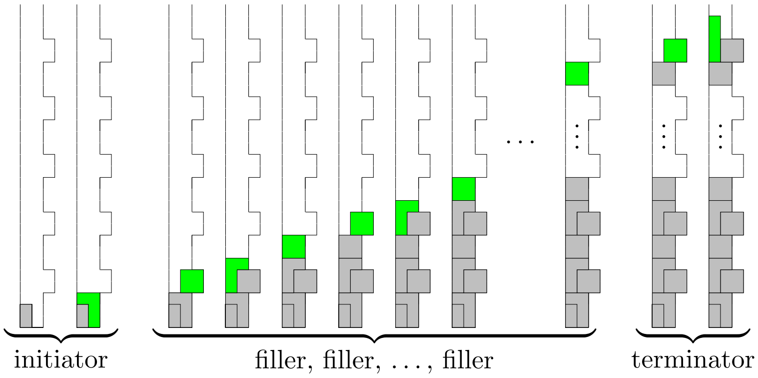


Fig. 4. A valid sequence of moves within a bucket.

Proof. The gameboard has size $5T+18+2s+O(1)$ by $4s+3$, and the total number of pieces is

$$\sum_{i=1}^{3s} [1 + 3a_i + 2] + s + 1 + \left(\frac{5T+16}{4}\right) \quad = \quad 10s + 3sT + \frac{5T}{4} + 5.$$

The a_i's and T are represented in unary, so the size of the game is polynomial. Polynomial time constructibility is obvious. □

3.2. *Completeness*

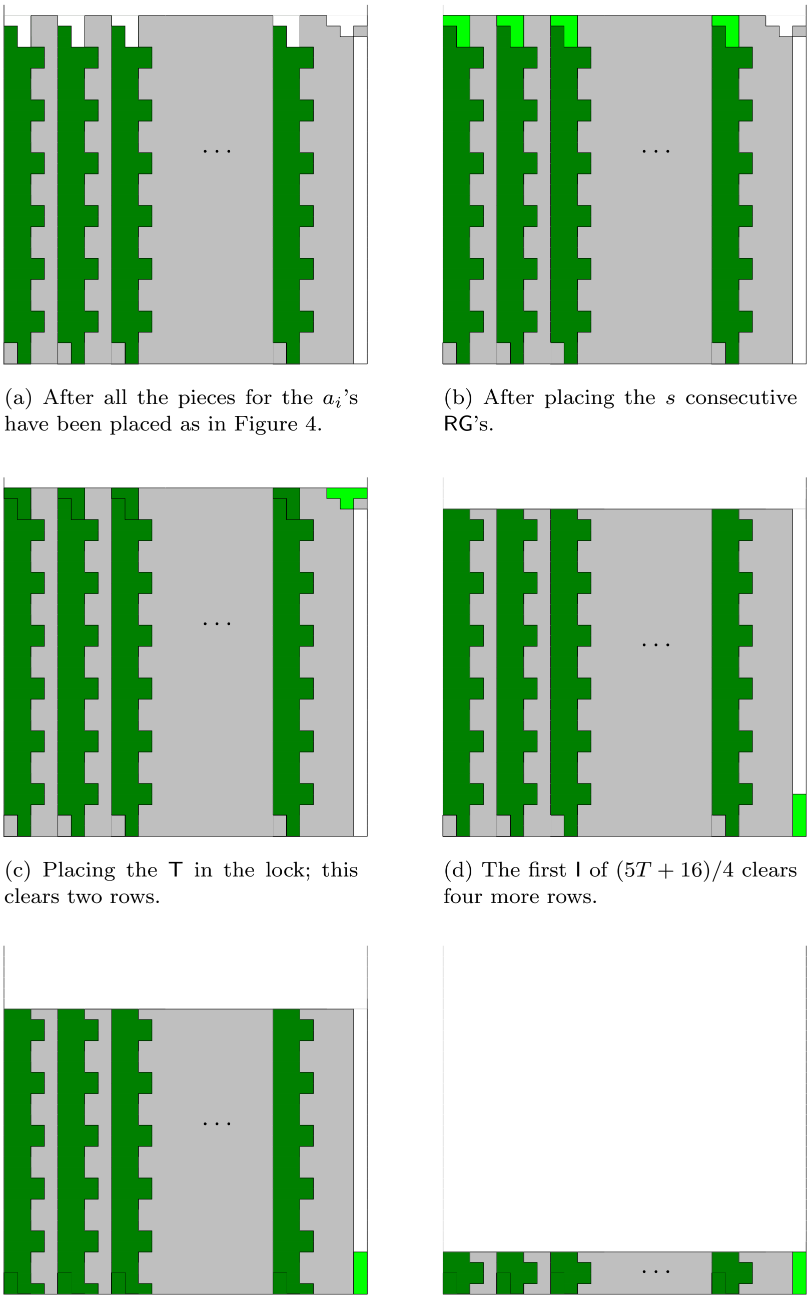
Here we show the easier direction of the correctness of our reduction: for a "yes" instance of 3-Partition, we can clear the entire gameboard.

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

Proof. In Figure 4, we show the move sequence to place all of the pieces associated with the number a_i (initiator, a_i fillers, and terminator) into a particular bucket.

Since $\mathcal{P}$ is a "yes" instance, there is a partitioning of $\{1, \ldots, 3s\}$ into sets $A_1, \ldots, A_s$ so that $\sum_{i \in A_j} a_i = T$. We have ensured that $|A_j| = 3$ for all j. Place all pieces associated with set $A_j = \{x, y, z\}$ into the jth bucket of the gameboard, as in Figure 4.

Figure 5(a) shows the configuration after all of the pieces associated with $a_1, \ldots, a_{3s}$ have been placed, as follows. After all pieces associated with the number a_x have been placed into bucket j, the first $3 + 5a_x + 2$ rows of bucket j are full, and the left-hand column of bucket j is filled two rows above that. The pieces associated with the number a_y fill the next $3+5a_y+2$ rows, and those with a_z the next



(a) After all the pieces for the a_i's have been placed as in Figure 4.

(b) After placing the s consecutive RG's.

(c) Placing the T in the lock; this clears two rows.

(d) The first I of $(5T + 16)/4$ clears four more rows.

(e) The second I, also clearing four rows.

(f) The last I; the board is now completely clear.

Fig. 5. Finishing a valid move sequence.

$3 + 5a_z + 2$ rows, leaving again one column with two additional rows filled. So the total number of rows filled after the "numbers" is $15 + 5(a_x + a_y + a_z) = 15 + 5T$. Doing this for each bucket j yields a configuration in which all buckets are totally filled up to row $15+5T$, with the first column of each bucket filled up to row $17+5T$.

We next get s successive RG's in the sequence. We produce the configuration in Figure 5(b) by dropping one of the RG's into each bucket, to fill rows $16 + 5T$ through $18 + 5T$. Now the configuration has the first $18 + 5T$ rows filled in all of the buckets, and the lock is untouched.

Next we get a T. Drop it into the slot in the lock, yielding the configuration of Figure 5(c); the first two rows are then cleared. In the resulting configuration, all columns are filled to the $(16 + 5T)$th row, except the last, which is completely empty.

Figures 5(d), 5(e), and 5(f) show the final stage of the sequence, as the $(5T + 16)/4$ successive I's arrive. Drop each into the last column of the lock. Each of the I's clears four rows; in total, this clears $4 \cdot (5T + 16)/4 = 5T + 16$ rows, clearing the entire the gameboard. The first, second, and last of these I's are illustrated, respectively, in Figures 5(d), 5(e), and 5(f). □

3.3. *Soundness*

Call *valid* any trajectory sequence that, using the pieces from the reduction, clears $5T + 18$ rows in the gameboard of $\mathcal{G}(\mathcal{P})$. We will refer to a move or trajectory as *valid* if it can appear in a valid trajectory sequence. In this section, we show that the existence of a valid strategy for the Tetris game $\mathcal{G}(\mathcal{P})$ implies that $\mathcal{P}$ is a "yes" instance of 3-Partition. We will often omit reference to $\mathcal{G}(\mathcal{P})$, and refer to its parts simply as the *gameboard* and the *piece sequence*.

Proposition 1. *In any valid trajectory sequence, no gridsquare above row $5T+18$ is ever filled.*

Proof. As a direct consequence of Lemma 3, we have that the number of gridsquares in the piece sequence is exactly the same as the number of unfilled gridsquares in the bottom $5T + 18$ rows of the gameboard. (This holds because $\sum_{i=1}^{3s} a_i = sT$, and does not depend on whether we are given a "yes" instance of 3-Partition.)

Thus the total number of gridsquares contained in the entire game (either initially filled in the gameboard or present in the piece sequence) is exactly the same as the area of the bottom $5T + 18$ rows of the gameboard. Since initially there are filled gridsquares in each of these rows, clearing the gameboard requires clearing all of these $5T + 18$ rows, which in turn requires placing all pieces of the sequence entirely in these rows. □

Call a bucket *unfillable* if it cannot be filled completely using arbitrarily many pieces from the set $\{\mathsf{LG}, \mathsf{RG}, \mathsf{Sq}, \mathsf{I}\}$. Note that unfillability is defined with respect to a bucket in isolation, and does not allow the clearing of rows. This definition is motivated by the following lemma:

Proposition 2. *In any valid trajectory sequence:*

(1) all pieces preceding the T *in the piece sequence are placed into buckets, completely filling them;*
(2) no rows are cleared before the T*;*
(3) no unfillable bucket arises.

Proof. For (1), we need only observe that if any piece other than T is the first piece placed in the lock columns, then it must fill some gridsquare above the $(5T+18)$th row. Thus any piece before the T that is not placed entirely within a bucket will violate Proposition 1.

Fact (2) follows straightforwardly from (1), since no rows can be cleared until at least one piece is placed into the lock columns.

For (3), note that Proposition 1 and (1) imply that there are the same number of unfilled bucket gridsquares as there are gridsquares contains in the pieces preceding the T in the given sequence. Therefore, if we do not entirely fill each bucket, then at least one of these gridsquares will not go into a bucket, violating (1). (Since no rows are cleared before the T, an unfillable bucket cannot be made fillable again, and therefore makes the trajectory sequence invalid.) □

There are two classes of unfillable buckets on which we will focus our attention:

Holes: A *hole* is an unfilled gridsquare in some bucket so that there is a contiguous series of filled gridsquares separating that gridsquare from the empty rows above the buckets.

Choke points and choked regions: A *choke point* is a row r of a bucket in which only one gridsquare is unfilled. Intuitively, a choke point prevents any piece other than I from "getting past" row r, implying that any unfilled gridsquares below r must be filled entirely using I's.

A *choked region* is a maximal contiguous vertical run of α unfilled gridsquares that (1) appears below a choke point in the bucket, (2) is in a different column than the unfilled gridsquare in the choke point, and (3) has $\alpha \not\equiv 0 \pmod 4$.

Proposition 3. *Any bucket containing a hole or a choked region is unfillable.*

Proof. It is immediate from our definition of a rotation model that a bucket with a hole is unfillable, and that only an I can "pass" a choke point—i.e., fill any gridsquares below the choke point that are in any column other than the one left open at the choke point. Since our buckets have width at most three, any I's must be placed vertically, and thus can only fill regions in which all maximal contiguous

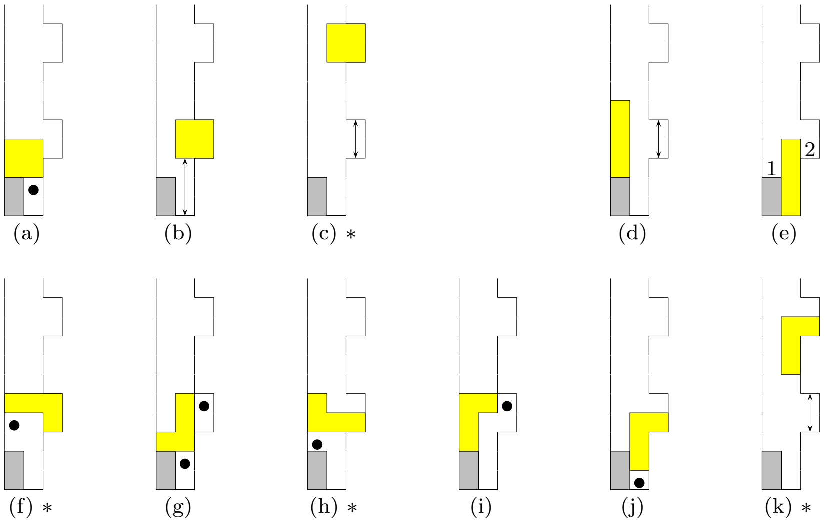


Fig. 6. All possible placements of {Sq, LG, I} in an unprepped bucket. A '•' denotes (the upper-most, left-most gridsquare in) a hole, and a vertical arrow indicates a choked region. In Figure 6(e), filling either numbered gridsquare makes the other numbered gridsquare into a hole or a choked region. Buckets labelled with an asterisk also denote the placement of the given piece in an arbitrary higher notch. See Proposition 4.

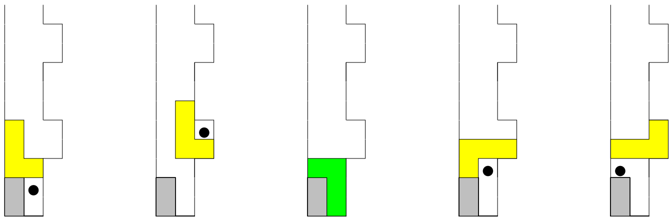
vertical runs of unfilled gridsquares have length divisible by four. Thus choked regions cannot be filled by I's, and therefore are unfillable. □

Call a bucket *unprepped* if it has exactly the form of the buckets in the initial gameboard, i.e., completely filled up to three rows below the bottom of a notch, and with the next two rows of the first column filled as well. Call the configuration of a gameboard unprepped if all of its buckets are unprepped. Call a bucket *prepped* if it is filled completely up to just below the bottom of a notch.

Proposition 4. *Placing any of* {Sq, LG, I} *in an unprepped bucket is invalid.*

Proof. All of the possible placements are shown in Figure 6. Note that Figure 6(e) shows the only placement of any of {Sq, LG, I} that does not immediately result in a hole or a choked region. In this case, observe that any Tetris piece placed to fill the gridsquare marked '1' makes the gridsquare marked '2' into a hole or a choked region, and vice versa. Thus this bucket is also unfillable. □

Proposition 5. *The only valid trajectory sequences must exactly follow the sequence in Theorem 3:*



(a) (b) $*$ (c) (d) $*$ (e) $*$

Fig. 7. All possible placements of RG in an unprepped bucket. See Figure 6 for explanation of notation. Figure 7(c) shows the only potentially valid move, which produces a prepped bucket. See Proposition 5.

(1) If a RG *is placed into an unprepped bucket, it must produce a prepped bucket;*
(2) If the sequence $\langle \mathsf{Sq}, \mathsf{LG}, \mathsf{Sq} \rangle$ *is placed into a prepped bucket, it must produce a prepped bucket;*
(3) If the sequence $\langle \mathsf{Sq}, \mathsf{I} \rangle$ *is placed into a prepped bucket, it must produce an unprepped bucket.*

Proof. All of the possible placements for these pieces are shown in Figures 7, 8, and 9, respectively. As in Proposition 4, the only way to place the given pieces without immediately producing a bucket with a hole or a choked region results in the stated resulting bucket. The only exception is Figure 9(d); there, the only possible way to fill the gridsquare marked '3' is by placing a I into the first column, but the resulting bucket is then exactly that of Figure 6(e), which was established as unfillable by Proposition 4. □

Proposition 6. *For any* $r \geq 0$*, in an unprepped gameboard, the only possibly valid strategy for the sequence* $\mathsf{RG}, r \times \langle \mathsf{Sq}, \mathsf{LG}, \mathsf{Sq} \rangle, \mathsf{Sq}, \mathsf{I}$ *is to place all of the pieces into a single bucket, yielding an unprepped configuration.*

Proof. Immediate by induction on r using Propositions 4 and 5. □

Lemma 4 (Soundness). *There is a valid trajectory sequence for* $\mathcal{G}(\mathcal{P})$ *only if* $\mathcal{P}$ *is a "yes" instance of* 3-Partition.

Proof. By Proposition 6, there exists a valid trajectory sequence only if there is a sequence that places, for each i, all of the pieces associated with a_i into the same bucket. For each bucket j, define $A_j := \{i :$ all of the pieces associated with a_i are placed in bucket $j\}$.

Note that placing all of the pieces of a_i into bucket j fills $a_i + 1$ notches in that bucket—one for each of the three-piece filler blocks, and one additional notch for

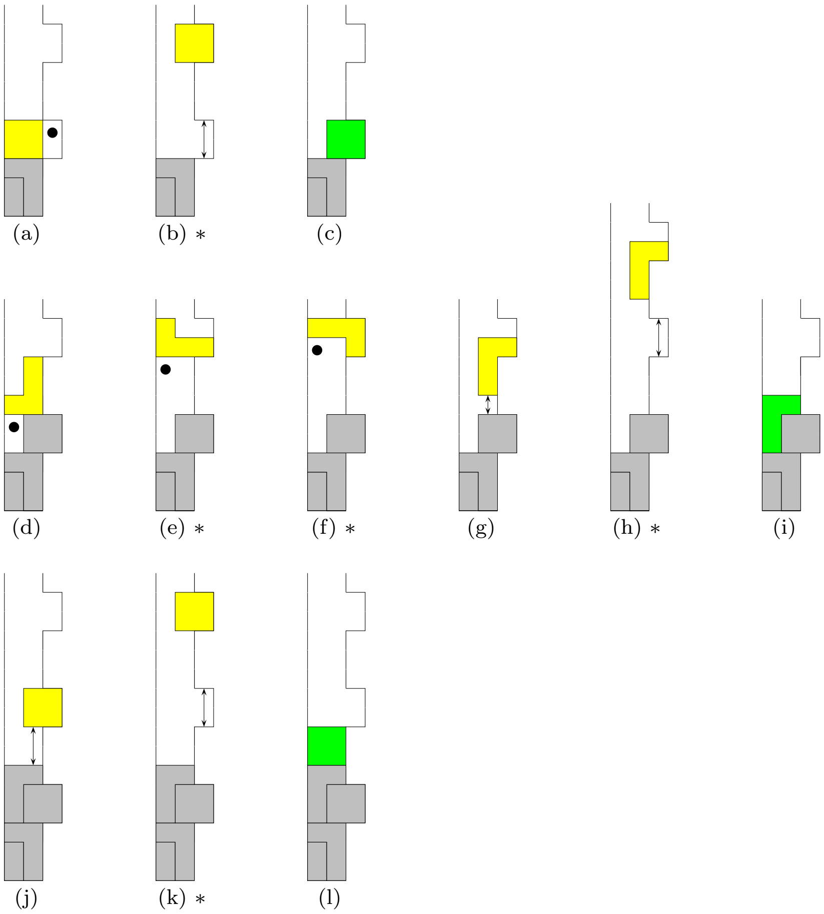


Fig. 8. All possible placements of $\langle \mathsf{Sq}, \mathsf{LG}, \mathsf{Sq} \rangle$ in a prepped bucket. See Figure 6 for explanation of notation. Figures 8(c), 8(i), and 8(l) show the only potentially valid moves, which produce a prepped bucket. See Proposition 5.

the terminator. Thus the number of notches filled in each bucket is $\sum_{i \in A_j}(a_i+1)$. In total, this is $\sum_j \sum_{i \in A_j}(a_i+1) = 3s + \sum_{1 \le i \le 3s} a_i = s(T+3)$. Since there are exactly $T+3$ notches in each of the s buckets, this implies that $\sum_{i \in A_j}(a_i+1) = T+3$ for every bucket j. Recall from Section 3.1 that our instance of 3-Partition guarantees that, for every i, we have $T/4 < a_i < T/2$. Thus $\sum_{i \in A_j}(a_i+1) = T+3$ implies that, for every bucket j, we have that $|A_j| = 3$ and $\sum_{i \in A_j} a_i = T$—i.e., P is a "yes" instance of 3-Partition. $\square$

Theorem 2. *It is* NP*-complete to maximize the number of rows cleared in a Tetris*

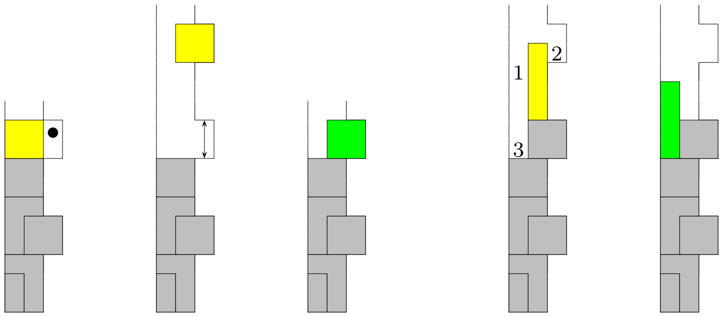


Fig. 9. All possible placements of $\langle \mathsf{Sq}, \mathsf{I} \rangle$ in a prepped bucket. See Figure 6 for explanation of notation. Figures 9(c) and 9(e) show the only potentially valid moves, which produce an unprepped bucket. See Proposition 5.

game.

Proof. Immediate from Theorem 1 and Lemmas 1, 2, 3, and 4. □

4. NP-Completeness for Other Objectives

In this section, we describe reductions extending that of Section 3.1 to establish the hardness of optimizing several other natural Tetris objectives.

4.1. *Maximizing Tetrises*

One natural objective for (the actual play of) Tetris is the maximization of *tetrises*—i.e., the number of times that four rows are cleared simultaneously, by the vertical placement of an I. Note that the decision version of this objective is acyclic and checkable—i.e., we can check if a trajectory sequence Σ achieves at least k tetrises on $\mathcal{G}$ in polynomial time. Thus this version of the Tetris problem is in NP.

Theorem 3. *Maximizing the number of tetrises in a Tetris game is* NP*-complete.*

Proof. We use a reduction very similar to that of Section 3.1, as shown in Figure 10. Given an instance $\mathcal{P}$ of 3-Partition, our game is as follows:

- The top $5T + 18 + 2s + O(1)$ rows of the gameboard are exactly the same as in our previous reduction. We add four rows below these, entirely full except in the fourth column.
- The piece sequence is exactly the same as in the previous reduction, with a single I appended.

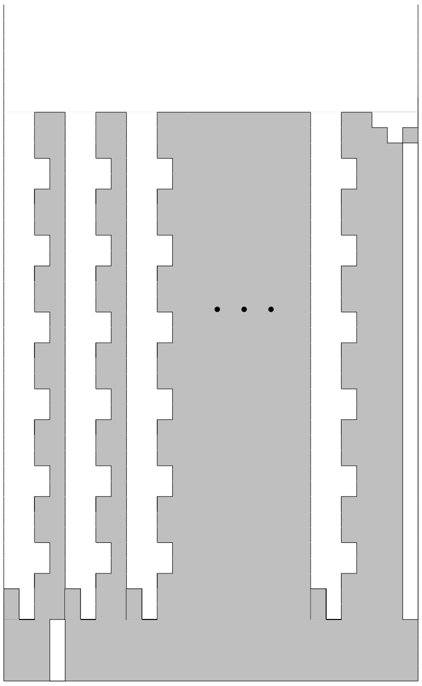


Fig. 10. Initial gameboard showing the hardness of maximizing tetrises.

We extend our previous reduction since it is not clear that $(5T + 16)/4$ tetrises—i.e., the number achieved in the "yes" case—cannot be achieved in the "no" case: perhaps by clearing only $5T + 17$ rows in total, it would still be possible to clear $5T + 16$ rows using tetrises. In our extension, however, the only way to achieve enough tetrises is by entirely clearing the original part of the gameboard.

For a "yes" instance of 3-Partition—namely, one in which we can clear the top $5T+18$ rows using the original part of the piece sequence—$(5T+16)/4+1 = 5T/4+5$ tetrises are achievable. (The first $5T/4 + 4$ occur when we place the I's vertically into the lock column; the last occurs when the appended I is placed into the new bottom rows.)

For a "no" instance, we cannot clear the top $5T+18$ rows using the original part of the piece sequence, as shown in the previous section. Since the fourth column is full in all of the original rows, we cannot clear the bottom four rows with the last I in the sequence. Thus we clear at most $5T + 18$ rows in total. This implies that there were at most $(5T + 18)/4 < 5T/4 + 5$ tetrises.

Therefore we can achieve $5T/4+5$ tetrises just in the case that the top $5T+18$ rows can be cleared by the first part of the sequence, which occurs exactly when the 3-Partition instance is a "yes" instance. □

4.2. *Maximizing Lifetime*

Another type of metric—considered by Brzustowski[7] and Burgiel[8], for example—is that of *survival.* How many pieces can be placed before a loss must occur? The decision problem—given a number p and a Tetris game $\mathcal{G}$, can the first p pieces of $\mathcal{G}$ be placed without incurring a loss?—is in NP, since the corresponding Φ is checkable and acyclic.

Our original reduction already yields some initial intuition on the hardness of maximizing lifetime. A closely related problem is the following: given a height h and a Tetris game $\mathcal{G}$, can all of the pieces of $\mathcal{G}$ be placed so that no gridsquare above height h is ever filled? (Membership in NP follows as above.) In the "yes" case of 3-Partition, there is a trajectory sequence that fills no gridsquares above the $(5T+18)$th row, while in the "no" case we must fill some gridsquare in the $(5T+19)$th row:

Theorem 4. *Minimizing the maximum height of a filled gridsquare is* NP*-complete.* □

However, this does not immediately imply the hardness of maximizing the number of pieces that the player can place without losing, because Theorem 4 only applies for certain heights—and, in particular, does not apply for height m (the top row of the gameboard), because our trajectory sequence from Section 3.2 requires some operating space above the $(5T+18)$th row for rotations and translations. To show the hardness of maximizing survival time, some additional work is required.

Theorem 5. *Maximizing the number of pieces placed without losing is* NP*-complete.*

Proof. We augment our previous reduction as shown in Figure 11. Intuitively, we have created a large area at the bottom of the gameboard that can admit a large number of Sq's, but we place a lock so that Sq's can reach this area only if the gameboard of the original reduction is cleared. Crucially, the entire gameboard has odd width, so after a large number of Sq's a loss must occur. Formally, our new gameboard consists of the following layers, for a value r to be determined below:

- The top $5T+18+2s+O(1)$ rows are exactly the same as in our previous reduction, with the addition of four completely filled columns on the right-hand side of the gameboard.
- The two next-highest rows form a second *lock*, preventing access to the rows beneath. This lock requires a T to be unlocked, just as the lock at the top of the previous reduction.
- The bottom r rows form a *reservoir*, and are empty in all columns but the first.

The gameboard has $5T+18+2s+r+O(1)$ rows and $4s+7$ columns. Let $A = O((T+s)s)$ be the total area in and above the (second) lock rows, and let $R = r(4s+6)$ be the total initially unfilled area in the reservoir.

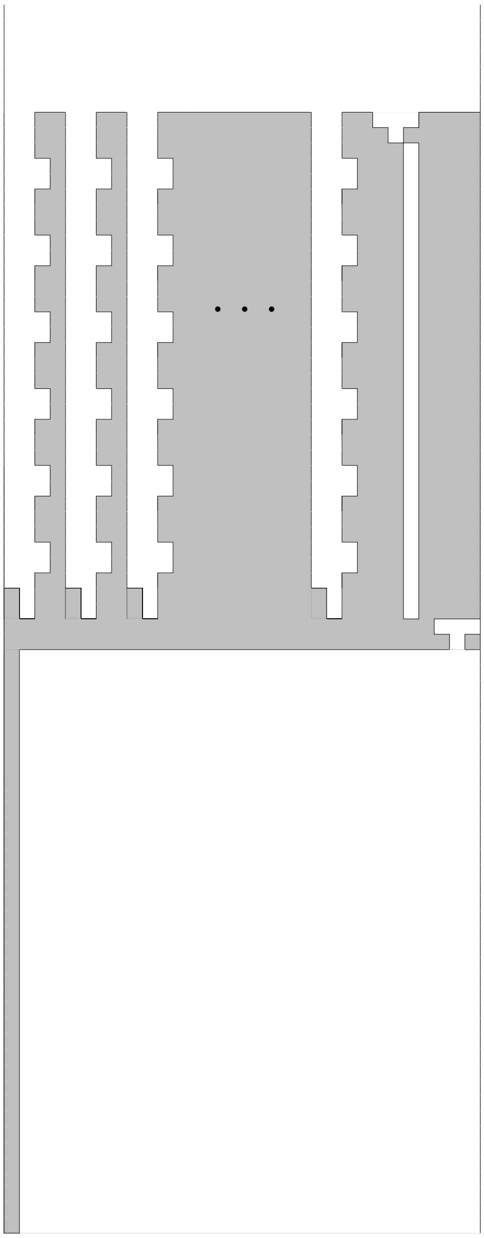


Fig. 11. Initial gameboard showing the hardness of maximizing survival time.

Our piece sequence is augmented as follows: first we have all pieces of our original reduction, then a single T, and finally $R/4$ successive Sq's.

For the moment, choose $r = \text{poly}(T, s)$ so that $R \geq 2A + 4$.

In the "yes" case of 3-PARTITION, the first part of the sequence can be used to entirely clear the $5T + 18$ rows of the original gameboard. The T clears the second lock, and the $R/4$ successive Sq's can then be packed into the reservoir to clear all of the reservoir rows. (By construction, there are an even number of unfilled gridsquares in each of the rows of the reservoir.)

In the "no" case of 3-Partition, the first part of the sequence cannot entirely clear the top $5T + 18$ rows of the gameboard. Since all rows above the second lock are filled, this means that the T cannot unlock the reservoir, and crucially the T is the last chance to do so—no number of Sq's can ever subsequently clear the lock rows. We claim that within $A/2 + 1$ Sq's (which cover $2A + 4$ gridsquares), a loss will occur. Since there are an odd number of columns in the entire gameboard, only rows that initially contain an odd number of filled gridsquares can be cleared by a sequence of Sq's; thus each row can be cleared at most once in the Sq sequence. In order to survive $2A + 4$ gridsquares from a Sq sequence, at least one row must be cleared more than once. Therefore after $A/2 + 1 \leq R/4$ successive Sq's, a loss must occur. □

5. Hardness of Approximation

In this section, we give results on the hardness of approximating the objectives discussed above. By modifying the reduction of Theorem 5, we can prove extreme inapproximability for either maximizing the number of rows cleared or maximizing the number of pieces placed without a loss, and a weaker inapproximability result for minimizing the maximum height of a filled gridsquare.

Theorem 6. *Given a sequence of p pieces, approximating the maximum number of pieces that can be placed without a loss to within a factor of $p^{1-\varepsilon}$ for any constant $\varepsilon > 0$ is* NP*-hard.*

Proof. Our construction is as in Figure 11, but with a larger reservoir: choose r so that the r-row reservoir's unfilled area R is larger than $(2A)^{1/\varepsilon}$, where A is the total area of the gameboard excluding the reservoir rows. As before, we append to the original piece sequence one T followed by exactly enough Sq's to completely fill the reservoir. As in Theorem 5, in the "yes" case of 3-Partition, we can place all of the pieces in the given sequence (which in total cover an area of at least R), while in the "no" case we can place pieces covering at most $2A$ area before a loss must occur. Thus it is NP-hard to distinguish the case in which we can survive all p pieces of the original sequence from the case in which we can survive at most $2A/4 < (R^{\varepsilon})/4 < p^{\varepsilon}$ pieces. □

Theorem 7. *Given a sequence of p pieces, approximating the maximum number of rows that can be cleared to within a factor of $p^{1-\varepsilon}$ for any constant $\varepsilon > 0$ is* NP*-hard.*

Proof. Our construction is again as in Figure 11, with $r > a^{2/\varepsilon}$ rows in the reservoir, where there are a total rows at or above the second lock. As above, in the "yes" case of 3-Partition, we can completely fill and clear the gameboard, and in the "no" case we can clear at most a rows. Thus it is NP-hard to distinguish the case in which at least r rows can be cleared from the case in which at most $a < r^{\varepsilon/2}$ rows can be cleared.

Note that the number of columns c in our gameboard is fixed and independent of r, and that the number of pieces in the sequence is constrained by $r < p < (r+a)c$. We also require that r be large enough that $p < (r+a)c < r^{2/(2-\varepsilon)}$. (Note that r, and thus our game, is still polynomial in the size of the 3-Partition instance.) Thus in the "yes" case we clear at least $r > p^{1-\varepsilon/2}$ rows, and in the "no" case we clear at most $a < r^{\varepsilon/2} < p^{\varepsilon/2}$. Thus it is NP-hard to approximate the number of cleared rows to within a factor of $(p^{1-\varepsilon/2})/(p^{\varepsilon/2}) = p^{1-\varepsilon}$. □

Theorem 8. *Given a sequence of p pieces, approximating the minimum height of the highest filled gridsquare to within a factor of $2-\varepsilon$ for any constant $\varepsilon > 0$ is* NP*-hard.*

Proof. Once again, our construction follows Figure 11. Let $F = O(Ts)$ be the total number of filled gridsquares at or above the rows of the lower lock, and let $P = O(Ts)$ be the total number of gridsquares in the piece sequence up to and including the second T. Choose $r = (F+P)/\delta$, where $\delta = \varepsilon/(3-\varepsilon)$.

As before, in the "yes" instance of 3-Partition, we can place the pieces of the given sequence so that the highest filled gridsquare is in the $(5T+18)$th row of the original gameboard, which is height $5T + 20 + r \leq r + P + F \leq r(1+\delta)$ in our gameboard.

In the "no" case, all of the Sq's appended to the piece sequence will have to be placed at or above the second lock, since we can never break into the reservoir. Note that, if rows are not cleared, we can never pack the appended Sq's into fewer than r rows. Thus the height of the highest filled gridsquare is at least $2r - \kappa$, where κ is the number of rows cleared during the sequence. As before, we can only clear rows that have an odd number of filled gridsquares in them before the Sq's in the sequence. Since there are only $F + P$ gridsquares in total in this part of the sequence, obviously $\kappa \leq F + P = r\delta$. Thus there is a filled gridsquare at height at least $r(2-\delta)$.

Therefore it is NP-hard to approximate the minimum height of the highest filled gridsquare to within a factor of $r(2-\delta)/r(1+\delta) = (2-\delta)/(1+\delta) = 2-\varepsilon$. □

6. Varying the Rules of Tetris

The completeness of our reduction does not depend on the full set of allowable moves in Tetris, and the soundness does not depend on all of its limitations. Thus our results continue to hold in some modified settings. In this section, we will prove the following theorem:

Theorem 9. *It remains* NP*-hard to optimize (or approximate) the maximum height* *of a filled gridsquare, the number of rows cleared, tetrises attained, or pieces placed without a loss when any of the following hold:*

(1) the player is restricted to one rotation/translation move before each piece drops in height.
(2) pieces are restricted to $\{\mathsf{LG}, \mathsf{RG}, \mathsf{I}, \mathsf{Sq}\}$, $\{\mathsf{LG}, \mathsf{LS}, \mathsf{I}, \mathsf{Sq}\}$, *or* $\{\mathsf{RG}, \mathsf{RS}, \mathsf{I}, \mathsf{Sq}\}$, *plus at least one other piece. (For the latter two cases, the player can only be restricted to* two *rotation/translation moves at each height.)*
(3) losses are not triggered until after *filled rows are cleared, or if losses never occur. (For the latter case, maximizing the number of pieces placed without a loss is meaningless, so the hardness results only apply for the other objectives.)*
(4) rotations follow any reasonable rotation model. □

6.1. *Limitations on Player Agility*

We have phrased the rules of Tetris so that the player can, in principle, make infinitely many translations or rotations before moving the piece down to the next-highest row. When actually playing Tetris, there is a fixed amount of time (varying with the difficulty level) in which to make manipulations at any particular height h; one cannot slide pieces arbitrarily far to the left or right before the piece falls.

Our reduction requires only that the player be able to make a single translation before the piece falls by another row (or is fixed), to slide a Sq into a notch. This is why we have chosen to have $2s + O(1)$ empty rows at the top of the game board: at most $2s$ translations and 2 rotations are required to place a piece into any desired column and orientation, since each piece starts in the middle of the top of the gameboard. Thus $2s + O(1)$ rows in the staging area affords us enough room to do any desired translation and rotation before the piece reaches the top of a bucket, while still only making one such move at any given height. (In the "no" case for loss-avoidance, this may cause the game to end more quickly, but the "yes" case remains feasible.)

Thus the problem remains NP-hard even when move sequences are restricted to at most one move between drops, for any of the objectives.

6.2. *Piece Set*

Our reduction uses only the pieces $\{\mathsf{LG}, \mathsf{RG}, \mathsf{I}, \mathsf{Sq}, \mathsf{T}\}$, so Tetris remains NP-complete when the pieceset is thus restricted.

Additionally, the reduction described in a previous paper [1] uses the pieceset $\{\mathsf{LG}, \mathsf{LS}, \mathsf{I}, \mathsf{Sq}, \mathsf{T}\}$, and, by taking the mirror image of that reduction, the reduction also holds for the pieceset $\{\mathsf{RG}, \mathsf{RS}, \mathsf{I}, \mathsf{Sq}, \mathsf{T}\}$. (That reduction uses similar ideas to the one in this paper, but is much more complicated. In particular, the notches in that reduction are of width two, and thus for completeness the player must be able to make *two* translation moves at a particular height.) Thus Tetris is still hard when restricted to any of these piecesets.

In fact, the use of the T in the lock was not required; we simply need some piece that does not appear elsewhere in the piece sequence. Thus Tetris remains NP-hard for any piece set consisting of $\{\mathsf{LG}, \mathsf{RG}, \mathsf{I}, \mathsf{Sq}\}$, $\{\mathsf{LG}, \mathsf{LS}, \mathsf{I}, \mathsf{Sq}\}$, or $\{\mathsf{RG}, \mathsf{RS}, \mathsf{I}, \mathsf{Sq}\}$, plus at least one other piece. When we use a snake as the lock piece, the bottom three gridsquares of a vertically oriented snake serves as the key, and we observe that no other piece—except T, which is thus not in our sequence—can be placed without filling at least *two* gridsquares above the $(5T + 18)$th row. This implies the same hardness results.

6.3. *Losses*

In Section 2, we defined a loss as the fixing of a piece so that it does not fit entirely within the gameboard; i.e., the piece fills some gridsquare in the would-be $(m+1)$st row of the m-row gameboard. Instead, we might define losses as occurring only *after* rows have been cleared—that is, a piece can be fixed so that it extends into the would-be $(m+1)$st row, so long as this is not the case once all filled rows are cleared. Since the completeness sequence (of Proposition 3) never fills gridsquares anywhere near the top of the gameboard, our results hold for this definition as well.

In fact, for our reduction, we do not depend on the definition of losses at all—the completeness trajectory sequence does not near the top of the gameboard, and the soundness proof does not rely on losses. Obviously the objective of Theorem 5 is nonsensical without a definition of losses, but all of the other hardness results still hold even if there are no losses whatsoever.

6.4. *Rotation Rules*

Above, we specified concrete rules for the rotation of pieces around a particular fixed point in each piece. However, our proof of correctness only relied on the general properties of reasonable rotation models, as defined in Section 2. In particular, there are two other reasonable rotation rules of interest: the *continuous* model and the *Tetris* model that we have observed in real implementations.

In the *continuous* (or *Euclidean*) rotation model, the rotation of a piece is around its center, as before, but we furthermore require that all gridsquares that the piece *passes through* must be unoccupied. Note that this rotation model is significantly more restrictive than a generic reasonable rotation model, and, in particular, the buckets shown in Figures 6(f), 6(h), 7(d), 7(e), 8(e), and 8(f) cannot be produced via translations and Euclidean rotations.

The *Tetris rotation model*, which we have observed to be the one used in a number of actual Tetris implementations, is illustrated in Figure 12. Intuitively, this model works as follows: for each piece type, choose the smallest k so that the piece fits within a k-by-k bounding box ($k = 2$ for Sq, $k = 4$ for I, and $k = 3$ otherwise). In a particular orientation, choose the smallest k_1 and k_2 so that the piece fits in k_1-by-k_2 bounding box. Place the piece so that the k_1-by-k_2 bounding box is exactly centered in the k-by-k box. This does not in general yield a position

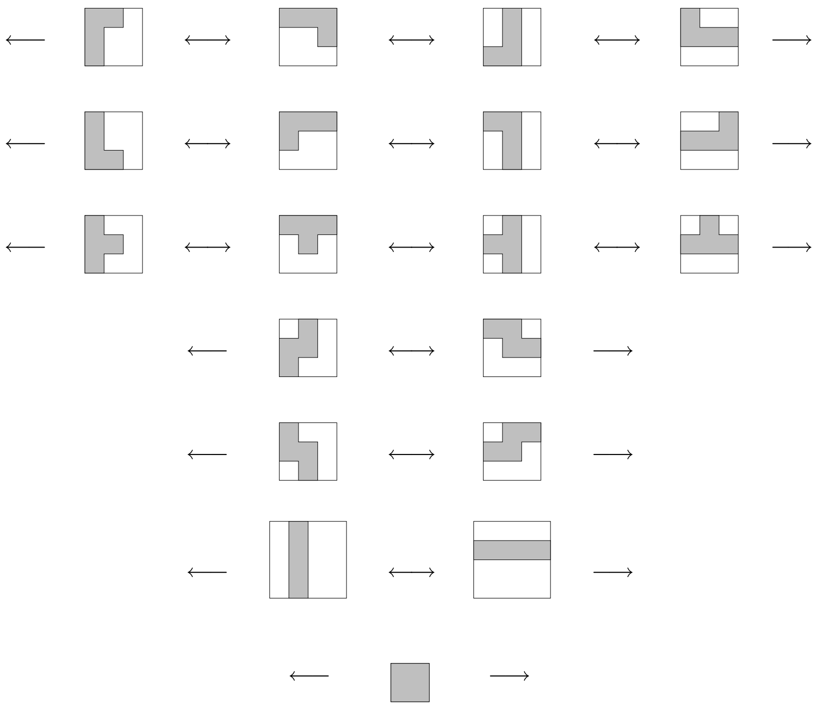


Fig. 12. The Tetris model of rotation. The pictured k-by-k bounding box is in the same position in each configuration; each piece can be rotated clockwise to yield the configuration on its right (wrapping to the leftmost column) or counterclockwise to yield the configuration on its left.

aligned on the grid, so shift the k_1-by-k_2 bounding box to the left and up, as necessary. (Incidentally, it took us some time to realize that the "real" rotation in Tetris did not follow the instantaneous model, which is intuitively the most natural one.)

One can quickly confirm that both the continuous and Tetris rotation models are reasonable, and thus that our results continue to hold under them.

7. Conclusion and Future Work

An essential part of our reduction is a complicated initial gameboard; it is a major open question whether Tetris can be played efficiently with an empty initial configuration. One possible approach to this question would be to give a sequence of pieces that forces the player to produce the initial gameboard in Figure 3. We note that, at least, there is a piece sequence that *can* be played to produce this configuration. This fact suggests an interesting combinatorial problem: given a gameboard B, does there exist a piece sequence that can be played to produce B? Recently, the fourth and fifth authors have proved that a gameboard B can be constructed if and only if a simple parity condition on the number of filled gridsquares of B is met, and have given an efficient algorithm to construct any such B.[14]

Our results are largely robust to variations on the rules (see Section 6), but our completeness result relies on the translation of pieces as they fall. At more difficult levels of the game, it may be very hard to make even one translation before the piece drops another row in height. Suppose the model for moves (following Brzustowski[7]) is the following: the piece can be translated and rotated as many times as the player pleases, and then falls into place. (That is, no translation or rotation is allowed after the piece takes its first downward step.) Is the game still hard?

It is also interesting to consider Tetris with gameboards of restricted size. What is the complexity of Tetris for a gameboard with a constant number of rows? A constant number of columns? Is Tetris fixed-parameter-tractable with respect to either dimension of the gameboard? (We have polynomial-time algorithms for the special cases in which the total number of gridsquares is logarithmic in the number of pieces in the sequence, or for the case of a gameboard with two columns.)

We have reduced the pieceset down to five of the seven pieces. For what piecesets is Tetris polynomial-time solvable? (For example, with the pieceset {I} the problem seems polynomially solvable, though non-trivial because of the initial partially filled gameboard.)

Finally, in this paper we have concentrated our efforts on the offline, adversarial version of Tetris. In a real Tetris game, the initial gameboard and piece sequence are generated probabilistically, and the pieces are presented in an online fashion. What can be said about the difficulty of playing online Tetris if pieces are generated independently at random according to the uniform distribution, and the initial gameboard is randomly generated? Some possible directions for this type of question have been considered by Papadimitriou.[15]

Acknowledgments

We would like to thank Amos Fiat, Siegfried Nijssen, and Ming-wei Wang for helpful discussions and comments. Thanks also to Josh Tauber for pointing out the puzzles in *Games Magazine*.[5]

This work was partially supported by an NDSEG Graduate Research Fellowship and an NSF Graduate Research Fellowship.

References

1. Erik D. Demaine, Susan Hohenberger, and David Liben-Nowell. Tetris is hard, even to approximate. In Tandy Warnow and Binhai Zhu, editors, *Proc. 9th Annual International Conference on Computing and Combinatorics (COCOON'03)*, volume 2697 of *Lecture Notes in Computer Science*, pages 351–363. Springer-Verlag, July 2003. Also available as Technical Report MIT-LCS-TR-865, Laboratory for Computer Science, Massachusetts Institute of Technology, September 2002, and as `cs.CC/0210020`.

2. Ron Breukelaar, Hendrik Jan Hoogeboom, and Walter A. Kosters. Tetris is hard, made easy. Technical Report 2003-9, Leiden Institute of Advanced Computer Science, Universiteit Leiden, 2003.
3. Tetris, Inc. `http://www.tetris.com`.
4. David Sheff. *Game Over: Nintendo's Battle to Dominate an Industry*. Hodder and Stoughton, London, 1993.
5. Scott Kim. Tetris unplugged. *Games Magazine*, pages 66–67, July 2002.
6. Michael M. Kostreva and Rebecca Hartman. Multiple objective solution for Tetris. Technical Report 670, Department of Mathematical Sciences, Clemson University, May 1999.
7. John Brzustowski. Can you win at Tetris? Master's thesis, University of British Columbia, 1992.
8. Heidi Burgiel. How to lose at Tetris. *Mathematical Gazette*, pages 194–200, July 1997.
9. Richard Kaye. Minesweeper is NP-Complete. *Mathematical Intelligencer*, 22(2):9–15, 2000.
10. Erik D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In Jiří Sgall, Aleš Pultr, and Petr Kolman, editors, *Proc. 26th Symposium on Mathematical Foundations in Computer Science*, volume 2136 of *Lecture Notes in Computer Science*, pages 18–32, August 2001. Full version available as `cs.CC/0106019`.
11. Uri Zwick. Jenga. In *Proc. 13th Symposium on Discrete Algorithms*, pages 243–246, 2002.
12. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.
13. Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4:397–411, 1975.
14. Hendrik Jan Hoogeboom and Walter A. Kosters. How to construct Tetris configurations. Technical Report 2003-8, Leiden Institute of Advanced Computer Science, Universiteit Leiden, 2003.
15. Christos Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.