The Theory of Tetris

Hendrik Jan Hoogeboom and Walter A. Kosters

Leiden Institute of Advanced Computer Science
Universiteit Leiden, The Netherlands
{hoogeboo,kosters}@liacs.nl

1 Introduction

Any algorithm requires a theoretical analysis. Such an analysis may address issues like complexity (e.g., NP-completeness [9]), decidability and practical properties concerning special cases. In this paper we would like to discuss the Tetris game in this light. We will first describe the game and some of its variants, show NP-completeness of a certain decision problem naturally attached to the game and then prove (un)decidability in some other cases. We conclude with some practical topics that arise from the NP-completeness proof.

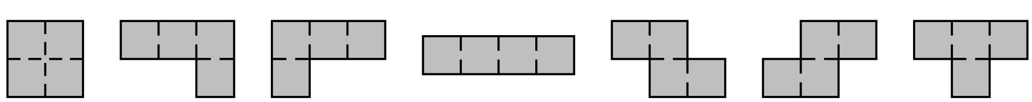
This paper is based on a series of articles that begins with the original NP-completeness proof of Demaine, Hohenberger and Liben-Nowell from MIT [7], that was well-noticed in the popular press. The proof was simplified in [3], leading to a joint journal paper [2]. In [13] and [14] the other issues mentioned above were dealt with. For full proofs we refer to these papers.

Tetris is a one-person game where random pieces consisting of four blocks fall down, one at a time, in an originally empty rectangular game board. The player is allowed to rotate and horizontally move the falling piece. If an entire row of the board is filled with blocks, it is removed ("cleared"). The main purpose of the game is to keep on playing as long as possible.

The decision problem under consideration is essentially the following. Given a partially filled game board (referred to as a Tetris configuration), and an ordered finite known series of pieces, is it possible to play in such a way that the whole board is cleared? The NP-completeness proof is by reduction. It is shown that instances of the so-called 3-Partition problem can be translated into rather involved Tetris configurations, where solutions correspond with boards that can be cleared. The configurations used suggest the question of constructibility: which configurations can be reached during game play? The rather surprising answer appears to be that almost any configuration can be reached, given suitable pieces. Another issue has to do with decidability: if the user interaction is somewhat bounded, is it then decidable whether certain natural sets of piece sequences contain "clearing" sequences? All these topics will be addressed in the sequel.

2 Rules

The game of Tetris is played on a rectangular board consisting of square (initially empty) cells. The board is of fixed width $w \geq 4$ and, for our purposes, of unbounded height. Seven game pieces can be used, each covering four board cells — from now on usually called blocks; they are depicted below. These pieces are known as (from left to right) `O` or *square*, `J` or *leftgun*, `L` or *rightgun*, `I` or *dash*, `Z` or *leftsnake*, `S` or *rightsnake*, and `T` or *tee*:



The "computer" generates a (usually random) sequence of pieces that drop down from the top of the board until they rest on top of (parts of) previously dropped pieces or on the bottom of the game board. The user/player can determine the position and orientation of the pieces by rotating and moving them horizontally while they fall. Whenever all the cells of a row (also called line) of the game board are occupied, the line is cleared; all blocks above it are lowered by one row (and no more). This row clearing can happen for several lines simultaneously.

In Tetris the purpose usually is to clear as many rows as possible given the generated sequence of pieces, while avoiding to run out of space vertically. As the game of Tetris itself is finite state (and hence decidable) when played on a board of given width and height, here we assume the board is of unbounded height.

Tetris has some peculiar implementation details. Let us mention a few examples.

1. In the NP-completeness proof below it is essential that Tetris pieces that touch the bottom of the game board or blocks from other pieces can still slide horizontally before they are "fixed". In some implementations this is however not possible, and pieces are then fixed immediately after touchdown. The NP-completeness proof might still hold for this other version, but a new construction is necessary, since the current one relies on filling overhangs with horizontally sliding squares.

2. In [7, 2] some attention is given to *rotation models*. It is indeed a problem, or rather a convention, to describe which "holes" allow Tetris pieces to pass through, perhaps involving meticulous intermediate rotations. And when are pieces still allowed to rotate? In the sequel we do not refer to this issue anymore, since the constructions involved do not give rise to problems of this kind.

3. Some people are surprised by seeing floating blocks in Tetris configurations. As will be shown later, (nearly) every configuration is constructible, i.e., can occur during regular game play. This includes situations where blocks do not rest on other blocks, but just remain floating — on air, so to speak. This is a consequence of the strictly applied rule that as one or more lines are cleared, they are removed from the game board; blocks above these disappearing lines precisely fall down as many lines as were cleared. This issue will only be of importance in the section on the construction of configurations.

3 NP-completeness

As mentioned in the introduction, we shall analyze the complexity of some decision problems related to Tetris. We shall also loosely describe the proof of the main NP-completeness result. Precise definitions, theorems, proofs and related results can be found in [7, 3, 2].

In this section we consider the following decision problem, called Tetris Clearing:

Instance. A Tetris game board partially filled with blocks, and an ordered sequence of Tetris pieces.

Question. Is it possible to play in such a way that the game board is left empty in the end?

It is not difficult to see that this problem is in the class NP. We now prove NP-hardness. As mentioned before, the proof is by reduction. We use the 3-Partition problem:

Instance. An sequence $a_1, a_2, \ldots, a_{3s}$ of positive integers and a positive integer T, so that $T/4 < a_i < T/2$ for all i with $1 \leq i \leq 3s$ and so that $\sum_{i=1}^{3s} a_i = s\,T$.

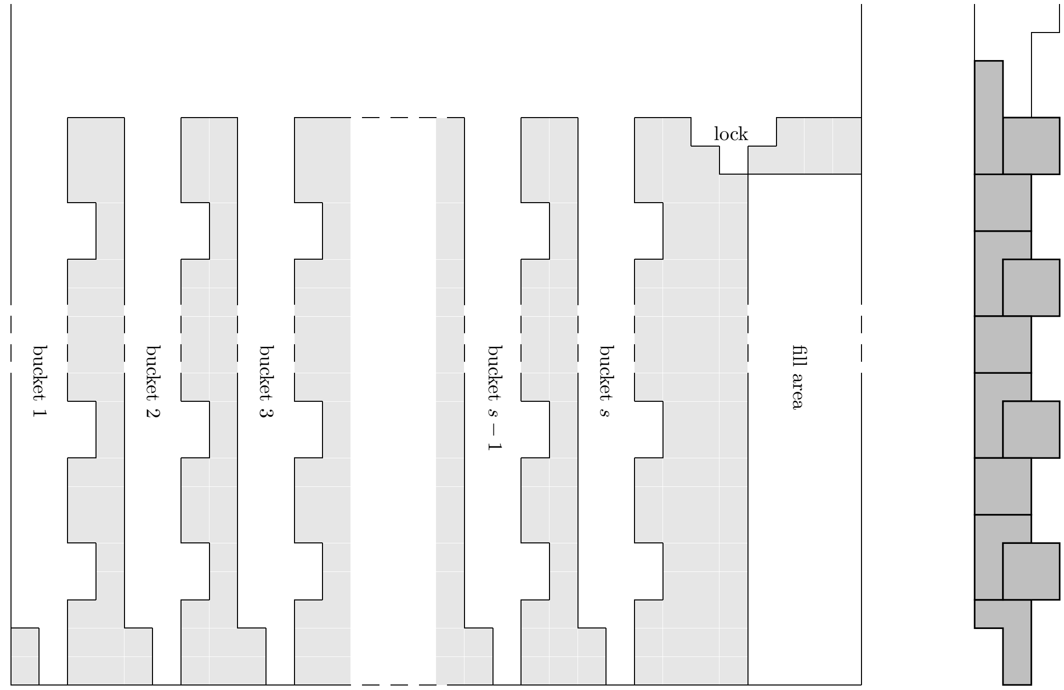
Question. Can $\{1, 2, \ldots, 3s\}$ be partitioned into s disjoint subsets $A_1, A_2, \ldots, A_s$ so that for all j with $1 \leq j \leq s$, we have $\sum_{i \in A_j} a_i = T$?

In [9] it is shown that this problem is NP-complete *in the strong sense*, which means that it is NP-hard even if the inputs a_i and T are provided in unary.

Now the main result is:

Theorem. Tetris Clearing is NP-complete.

We give a brief sketch of the proof. We start from an instance of the 3-Partition problem. We then construct the following initial Tetris game board (see picture below, left). The height of



the top row is $5T + 18$. We call the empty columns *buckets* and the big rectangular space on the right *fill area*; the T-shaped area above right is called the *lock*. Every bucket represents a subset as in 3-Partition. There are s buckets just like there are s subsets in 3-Partition.

The sequence of pieces for our game consists of a series of pieces for each a_i, followed by a number of pieces after all the a_i's. Each integer a_i is "coded" by one `L`, a_i times the triple `O`–`J`–`O`, and one pair `O`–`I` (see right part of the figure above for $a_i = 3$). The final pieces are: s successive `L`'s to fill the buckets, one `T` to open the lock and exactly enough (i.e., $5T + 16$) successive `I`'s to cover the fill area.

It is not much of a problem to show that the given sequence of pieces can clear the initial game board, in the case of a "yes" instance of 3-Partition. It is harder to prove that if the sequence cannot clear the board, the original instance could not fulfill the properties of a "no" instance. We just mention a few interesting details.

We suppose that we are looking at a sequence that can clear the original Tetris game board. The volume of the pieces is precisely what is needed to fill the empty cells in this board. This implies — among other things — that no pieces are allowed to stick out above the original highest row. The fill area and the lock ensure that there will be no line clearings before all the buckets to the left are filled. Then comes the unique `T` piece that opens the lock in the upper right, after which a series of `I`'s does the clearing. The main body of this part of the proof is devoted to showing that the series of pieces that "code" a number are precisely in this order required to fill the bucket, and cannot be spread over several buckets. Also notice the use of sliding `O`'s to fill the buckets. □

4 Decidability

In this section we discuss (un)decidability issues related to the game of Tetris, as reported in [13] (where details of the proofs can be found).

We consider different *models of user intervention*. On the one hand we have the normal Tetris rules, as described above, where the user has many possibilities to influence the result. At the other extreme we have the model where the user is not allowed any intervention: once the "computer" fixes the piece, its orientation and its horizontal position, the piece drops down in the specified orientation, and in the specified position.

As for a given game board the number of initial possibilities of each piece — its orientation and the columns occupied — is bounded, the sequence of pieces dropped can be described by a string over a finite alphabet. This suggests the following decision problem, TETRIS *with Intervention Model M*:

Instance. A regular language L describing sequences of TETRIS pieces (with their initial orientation and horizontal position) for a given width game board.

Question. Is there a sequence in L that leaves the game board empty after dropping all the pieces into an initially empty game board, according to the "model" M? I.e., does the user have a way to clear the entire sequence, while adhering to the rules in M?

Note that if the user is not allowed any intervention (we call this the *Null Intervention Model*, and refer to the corresponding decision problem as TETRIS *without intervention*), there are no choices to be made. For more complicated models, we are looking for "optimal" user actions that lead to total clearings.

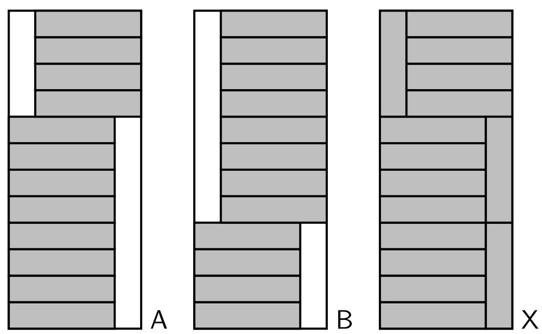
4.1 An undecidability result

We now have the following undecidability result:

Theorem. TETRIS without intervention, for sequences consisting only of I's on a board of width 10, is undecidable.

The proof is based on a reduction from the Post Correspondence Problem [15]. The basic idea is the following.

Given an instance of the PCP — two sequences $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_n)$ of strings over a two-letter alphabet $\{a, b\}$ — we construct an instance L of TETRIS without intervention, on a board of width 10. The left and right halves of the board (each a board of 5 cells wide) will act as stacks holding proposed solutions to the PCP, i.e., words of the form $u_{i_1} u_{i_2} \ldots u_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}$ for some $k \geq 1$, and $1 \leq i_1, i_2, \ldots, i_k \leq n$. To build the contents of the stacks we need three basic piles of I's, that we call A, B and X. The first two of these represent the two symbols a and b of the alphabet of the PCP; the last one is used for padding the two copies of the solutions.



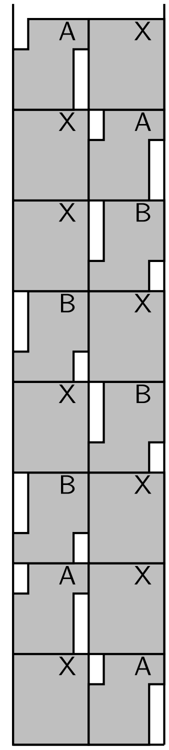
Note that the A and B piles can be removed (popped from the stack) following the rules of TETRIS, by dropping three vertical I's in the proper columns, provided the piles are next to an X pile on the other stack. The piles are designed in such a way that the vertical I's used to remove the blocks do not fall through to the next pile. Pieces dropped in the first column are blocked by the bottom rows of the pile, pieces falling in the fifth column are blocked by the topmost row of the pile below (or by the "floor").

Now first, the language L (or the corresponding finite automaton) prepares nondeterministically a sequence of piles, pushing onto the two stacks the same (nonempty) sequence of A's and B's, but randomly interleaved with X's. This part is independent of the particular instance of the PCP.

Then, in a second phase, L tries to clear the board, guessing a solution of the PCP, by repeatedly picking an index i ($1 \leq i \leq n$) and trying to pop the left stack according to the string u_i and the right stack according to the string v_i.

The rest of the proof (see [13]) shows that the original PCP has a solution if and only if the language L has a way to leave the empty game board.

As an example, a configuration left after the first phase of our construction is depicted to the right: in the left stack we can read (top-to-bottom) a, b, ba while we encounter ab, b, a in the right stack. This corresponds with a solvable PCP. □



4.2 Some decidability results

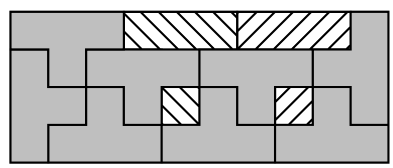
Quite amazingly, the undecidability result uses only a single type of piece. Let us now look at other ones. A simple argument shows that a nonempty sequence of either `S` or `Z` pieces cannot clear the board (cf. [5]), so the problem restricted to those pieces becomes trivially decidable. For the pieces `T`, `L` and `J` we can conceive a configuration that can be used to construct stacks, and similar arguments as for `I` hold (albeit on a board of width 16).

Finally, for `O` only very regular patterns are possible that leave an empty board. This is the basis for the following result:

Theorem. TETRIS without intervention, for sequences consisting only of `O`'s on a board of width 10, is decidable. □

We reconsider the decision result above, now allowing user translation and rotation of the pieces that are specified by the sequences in the given regular language L. The intervention is just as in the standard TETRIS game. We refer to the corresponding decision problem as TETRIS *with normal intervention*.

The general question is related to the many tiling problems for polyominoes (see, e.g., [11, 8]), as a tiling of a rectangle by TETRIS tetrominoes implies a possible clearing of the board using the TETRIS pieces in some suitable order. However, apart from the fact that the TETRIS problem also deals with the *order* in which the pieces are offered, classical tiling is more restricted: it does not allow intermediate clearing of rows. As an example, ten `T`'s can clear the TETRIS game board (of width 10, as below) whereas there is no tiling of the 10 by 4 rectangle using `T`'s [16].



The sequences of the rectangular TETRIS pieces `O` and `I` that can be used to leave an empty game board have a simple characterization. Our result is valid for standard width 10, but can be stated slightly more generally. We need the following Lemma:

Lemma. A sequence of `I`'s and `O`'s can be dropped into an initially empty game board of width $4k + 2$ ($k \geq 1$) leaving the empty board if and only if the number of pieces is a multiple of $2k + 1$, and the number of `I`'s is even. □

We have an immediate corollary:

Theorem. TETRIS with normal intervention, for sequences consisting only of `I`'s and `O`'s on a board of width 10, is decidable. □

There are connections with the strategies developed for winning two piece TETRIS games presented in [4].

Let us conclude this section with a slightly unexpected result. Restricted to a single piece (which can be other than the seven tetrominoes in standard TETRIS) TETRIS with normal intervention is decidable, even though we do not (need to) explicitly know the decision algorithm in each particular case:

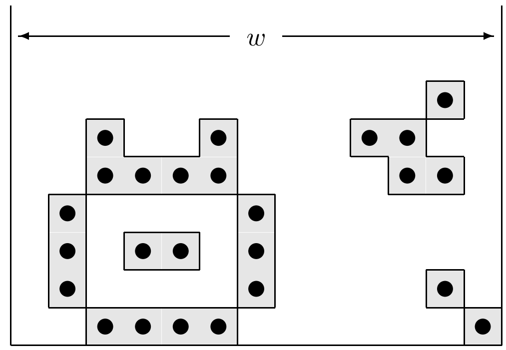
Theorem. TETRIS with normal intervention, for sequences consisting of copies a single fixed piece, on a board of fixed width, is decidable.

The proof relies on the fact that only the number of pieces matters, and that these numbers form a so-called semi-linear set [10]. □

5 Constructibility

The NP-completeness proof requires a rather intricate TETRIS configuration to start with. It seems a natural question to ask whether or not this configuration can occur during normal game play, and more general: what are the configurations that can occur? The answer is somewhat surprising: (almost) any configuration can show up!

A TETRIS *configuration* is a game board, where some of the cells are already occupied. A configuration is called *constructible* if it is possible to reach it, from an initially empty board, with a suitable series of pieces using appropriate translations and rotations. In this section we shall prove that essentially every reasonable configuration is constructible. The one non-trivial exception deals with boards of even width, where some simple parity condition should be fulfilled. The example configuration to the right, on a board of width $w = 13$, is constructible. Our construction requires 276 TETRIS pieces, clearing $(4 \cdot 276 - 25)/13 = 83$ intermediate rows.



Let us first remark that if the width w of the board is a multiple of 4, at any time the number of blocks in the current configuration is a multiple of 4: each new piece adds 4 blocks, whereas a line clearing removes w blocks. So clearly, the number of blocks in any constructible configuration should then be a multiple of 4. Similarly, if $w + 2$ is a multiple of 4, the number of blocks should be a multiple of 2. These two simple restrictions appear to be the only ones:

Theorem. A configuration of p blocks is constructible using suitable TETRIS pieces starting from the empty board of width w if and only if

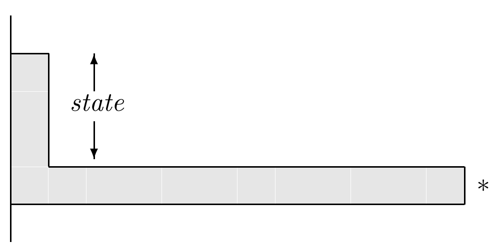
1. no row is completely full, and
2. no row below the highest one containing blocks is completely empty, and
3. if w is a multiple of 4, then so is p; if $w + 2$ is a multiple of 4, then p is even.

The next section is dedicated to the proof of the theorem, giving an explicit construction (see [12] for an implementation in the form of a Java-applet). In the sequel we shall assume that all three conditions mentioned in the theorem are met.

5.1 The construction

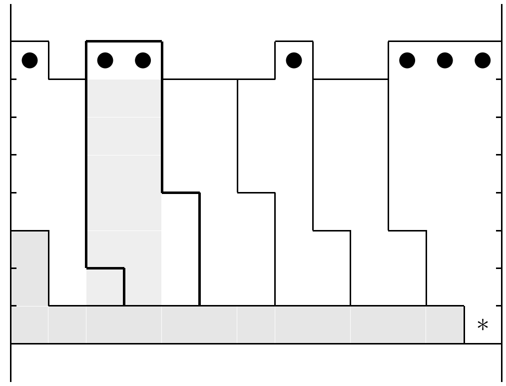
The configuration on the board is constructed row-by-row in a modular fashion. In [14] all details are given. For each row the construction consists of two phases.

First we build a *platform* that serves as a scaffolding for the construction work (it prevents TETRIS pieces from falling down to lower rows). In general the platform looks as follows, see the picture to the right. The $*$ denotes the bottom right empty square of the platform; once it is filled, its whole row will be cleared. The platform construction requires 3 or 7 intermediate rows that are cleared.

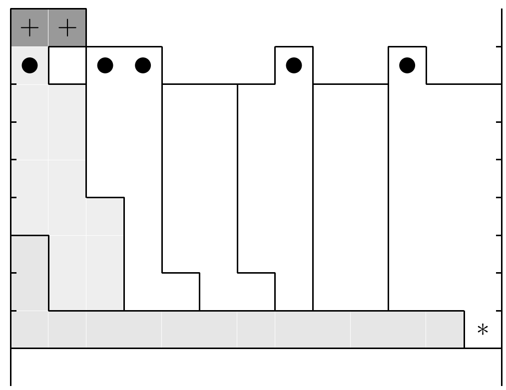


The number of squares sticking out vertically above the platform at the left end may vary between 0 and 3, and is referred to as the *state of the platform*. We need such a state as the total number of cells presently occupied or cleared in the past must be a multiple of 4.

In the second phase, the row construction phase, we build the blocks ● of the next row of the configuration on top of the platform, using six additional rows. Basically we construct consecutive "rectangles" two columns wide and six rows high with the necessary blocks on top, proceeding from left to right. Again, however, we have the multiple of 4 restriction, and we carry a surplus of blocks as *state* of the rectangles. This state is visible as indent of up to three blocks at the bottom in the left column of the rectangle. The last rectangle is designed to fill both the final block of the platform and the six rows of the rectangles, clearing all additional blocks that are not part of the final configuration.



As always, the number of blocks occupied in the construction need not be a multiple of 4, and we have to take this into account. We solve this by allowing a group of up to three additional blocks placed on top of one of the blocks. This *overflow* is indicated by + in the figure to the right. The overflow is used as a starting point in the construction of the platform for the next row of blocks. If there is no overflow then we start the construction by putting a horizontal `I` on top of one of the blocks of the last row (artificially introducing an overflow of four). In the case of odd width, the overflow can be removed each time.



The precise form of the rectangles (for each state, number of blocks and overflow) is rather tedious, see [14]. Particular care has to be taken for the last rectangle which has to clear the intermediate rows.

The whole construction starts with a horizontal `I`. It is extended to a platform with state 1. In order to remove the last overflow — if any —, the construction ends with some simple final details.

Note that in many cases there exist simpler constructions (for instance for boards of odd width), but the uniform approach has its own merits. Indeed, some configurations are even extremely simple to reach (e.g., a single vertical `I`), whereas our construction uses an abundance of pieces, clearing many rows on the way.

6 Conclusion

So far we have discussed several issues that are somehow attached to the game of Tetris. The fact that a well-known and easy to understand game as Tetris possesses such a rich structure is really surprising. There clearly is a connection between deep mathematical ideas such as NP-completeness and every day life. This fits in the larger research picture, where for many games these sorts of problems are addressed, cf. [1, 6].

Many problems remain open. Among others, one can think of variants of the game rules, but also of more general topics as the characterization of the clearing sequences in decidable cases.

References

[1] E.R. Berlekamp, J.H. Conway, and R.K. Guy. *Winning Ways for Your Mathematical Plays*. Volume 1 and 2, Academic Press, New York, 1982.

[2] R. Breukelaar, E.D. Demaine, S. Hohenberger, H.J. Hoogeboom, W.A. Kosters, and D. Liben-Nowell. Tetris is Hard, Even to Approximate. International Journal of Computational Geometry and Applications (IJCGA) 14 (2004) 41–68.

[3] R. Breukelaar, H.J. Hoogeboom and W.A. Kosters. Tetris is Hard, Made Easy. Technical Report 2003–9, Leiden Institute of Advanced Computer Science, 2003.

[4] J. Brzustowski. *Can You Win at Tetris?* Master's Thesis, University of British Columbia, 1992.

[5] H. Burgiel. How to Lose at Tetris. Mathematical Gazette 81 (1997) 194–200.

[6] E.D. Demaine. Playing Games with Algorithms: Algorithmic Combinatorial Game Theory. Proceedings 26th International Symposium on Mathematical Foundations of Computer Science MFCS2001, J. Sgall, A. Pultr and P. Kolman (editors), Lecture Notes in Computer Science 2136, pages 18–32, 2001.

[7] E.D. Demaine, S. Hohenberger and D. Liben-Nowell. Tetris is Hard, Even to Approximate. Computing and Combinatorics, 9th Annual International Conference, T. Warnow and B. Zhu (editors), Lecture Notes in Computer Science 2697, pages 351–363, 2003.

[8] B. Durand and L. Vuillon (editors). *Tilings of the Plane*. Special issue of Theoretical Computer Science, Volume 303, Numbers 2–3, 2003.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, New York, 1979.

[10] S. Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, New York, 1966.

[11] S.W. Golomb. *Polyominoes: Puzzles, Patterns, Problems, and Packings*. 2nd ed., Princeton University Press, Princeton, NJ, 1995.

[12] H.J. Hoogeboom and W.A. Kosters. Website — How to Construct Tetris Configurations. `http://www.liacs.nl/home/kosters/tetris/`.

[13] H.J. Hoogeboom and W.A. Kosters. Tetris and Decidability. Information Processing Letters 89 (2004) 267–272.

[14] H.J. Hoogeboom and W.A. Kosters. How to Construct Tetris Configurations. International Journal of Intelligent Games & Simulation (IJIGS) 3 (2004) 94–102.

[15] E.L. Post. A Variant of a Recursively Unsolvable Problem. Bulletin of the American Mathematical Society 52 (1946) 264–268.

[16] D.W. Walkup. Covering a Rectangle with T-Tetrominoes. American Mathematical Monthly 72 (1965) 986–988.