Bin Packing Extreme Edition

Jordan Crawford-O'Banner

March 2020

1 Introduction

Tetris is a computer game originally released in 1984 in which pieces of different geometric forms, called "tetriminos", descend from the top of the screen. While the pieces are falling, the player can move the pieces laterally and rotate them until they touch the bottom of the field or land on a piece that had been placed before it. The goal is to use the pieces to create as many horizontal lines of blocks as possible. When a line is completed, it disappears, and the blocks placed above fall one level. Completing lines scores points, and accumulating a certain number of points moves the player up a level, which increases the number of points granted per completed line. If the player cannot make the blocks disappear fast enough, the screen will start to fill, and when the pieces reach the top of the screen the game ends. The game never ends with the player's victory; the player can only complete as many lines as possible before an inevitable loss.

2 The Game

For the purposes of an NP-Complete proof of Tetris, I first define the game as defined in "Tetris is Hard, Even to Approximate" by Liben-Nowell, Demaine, and Hohenberger.

2.1 The Gameboard

The gameboard is defined as a mxn grid. Each space within the grid is either filled or unfilled.In a legal gameboard, no row is completely filled, and there are no completely empty rows below any filled spaces. When determining if a move is legal, we consider all space outside the gameboard as always-occupied sentinels.

2.2 Game pieces

There are 7 possible pieces to place on the board, which are each composed of four 1x1 blocks. They are all shown below in figure 1. Each piece is described by a 4-tuple consisting of.

Figure 1: All 7 Tetrominoes

- Piece Type: Sq, LG, RG, LS, RS, I, or T
- Orientation: 0, 90,180, or 270 degrees (the number of degrees clockwise from the piece's base.)
- Position: a position of the piece's center on the gameboard, chosen from the possible mxn spaces. The position of a Sq is the location of the upper-left block of the Sq;
- The value fixed or unfixed, indicating whether the piece can continue to move

Initially all pieces have a 0 degree orientation, are placed at position (0,n/2), and are unfixed. This can be described as (piece type(t), orientation(o),(position in the x direction(i), position in the y direction(j)), fixed or unfixed).

2.3 Rotating Pieces

There are models of Tetris that are NP-Complete that account for pieces being able to rotate, but the reductions for those models are far more complex than the reduction I am planning to do. Therefore I will say that for this model model all pieces are unable to be rotated.

2.4 Playing the Game

There are no legal moves for a fixed pieces on the gameboard. The pieces are at The legal moves for an unfixed piece on the gameboard are:

- A slide to the left. If the spaces to the left of the piece are open in the gameboard, we can translate the piece to the left by one column. The new piece state is (t,o,(i,j-1),unfixed).
- A slide to the right, similarly. The new piece state is (t, o,$(i, j + 1)$, unfixed).
- A drop by one row, if all of the spaces beneath the piece are open on the gameboard. The new piece state is (t, o,$(i - 1, j)$, unfixed).
- A fix, if at least one space below the piece is filled in the gameboard. The new piece state is (t, o,(i, j), fixed).

3 Reduction

3.1 The Problem

We will consider the variable $G = (B, P_1, P_2, ..., P_p)$, where B represents the initial gameboard and each P represents a piece that is legal to put on that gameboard. We will also consider the variable Σ that represents a sequence of pieces.

The Tetris problem will be defined as: given a game G, does there exist a sequence of piece Σ such that $\Phi(G, \Sigma)$ holds. In this case, $\Phi(G, \Sigma)$ represents the question: Can the Tetris pieces, supplied in the order given, be translated and rotated such that the game board will be cleared with these pieces, the last piece of the sequence filling the final gap.

In order to prove that this problem is NP-Complete, I will be reducing the 3-Partition problem to the Tetris problem.

3.2 3-Partition Problem

We can define the 3-Partition problem as the following:

Given: A sequence A of positive integer values $a_1, ..., a_{3s}$ and a positive integer value T such that $T/4 < a_i < T/2$ for all $1 \leq i \leq 3s$, and such that piece $\Sigma_{i=1}^{3s} ai = sT$.

Question: Can A be divided into s disjoint subsets (or rather subsequences) $B_1, ..., B_s$ such that: $\Sigma_{a_i \in B_j} a_i = T$ for all $1 \leq j \leq s$? (Call (A, T) a "yes" instance if this is the case and a "no" instance otherwise.)

Note that because $T/4 < a_i < T/2$ for all $1 \leq i \leq 3s$, in a "yes" instance $|B_j| = 3$ for all $1 \leq j \leq s$.

3.3 The Game

In order to reduce the Tetris problem to the 3-Partition problem, we must first define the exact gameboard and the sequence of pieces that appear on that gameboard, which will only be a 'yes' if the gameboard can be cleared.

3.3.1 The Initial Gameboard

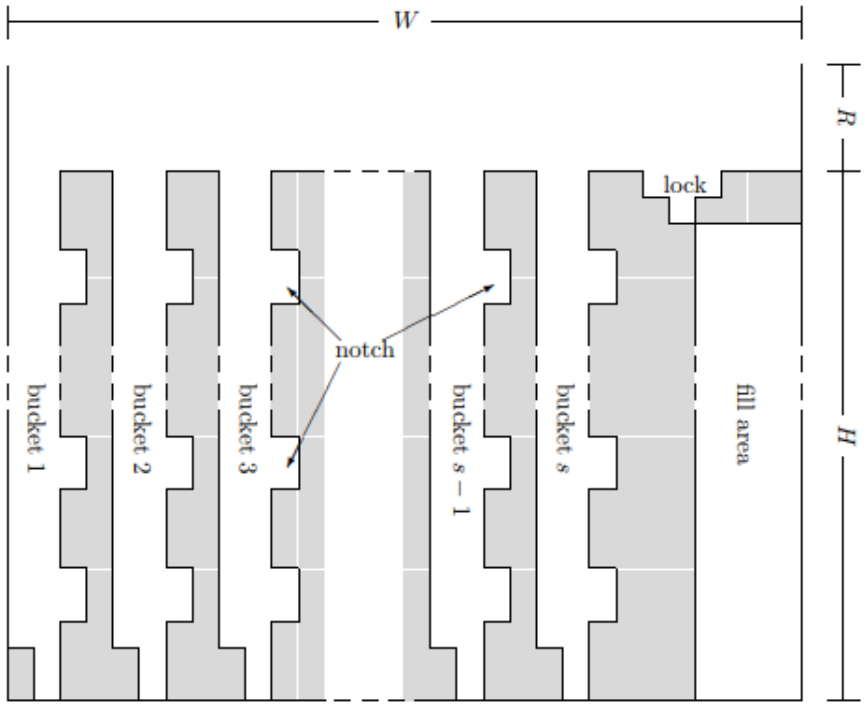


Figure 2: Initial Gameboard of the reduction

The figure above shows the initial gameboard we will use in this reduction. Similar to the 3-Partition problem, the Tetris board is filled in such that there are s empty columns that we will call buckets. The dimensions of the gameboard are:

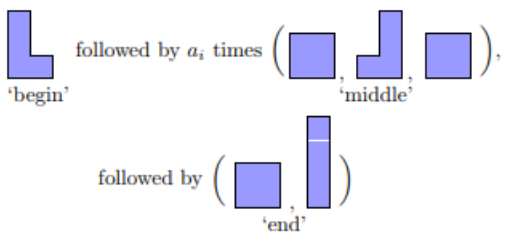
- R is the space needed to translate the pieces. We consider R to be big enough to translate Tetris pieces above the 'buckets' and therefore R can be ignored in the reduction.
- W is the width of the game board and is equal to $4s+6$. This is because there are s buckets that are each four blocks wide (if you count the single block between them), and the lock and fill are are six blocks wide together
- H is the height of the bottom part of the game board that needs to be cleared and is equal to 5T + 18. This measurement will be explained later.

Each of the columns in the gameboard have notches in them, which are there to ensure that there is only one way to fill each bucket, which correspond to one of the s subsets in the 3-Partition problem. Every buckets has $T+3$ notches in its right side. The fill area and the lock ensure that the gameboard cannot be cleared before the lock has been put in place. The fill area must always be of height $H-2$ in order to complete the rows the lock does not and has to be able to be filled in the reduction.

3.3.2 Sequence of Pieces

The sequence of pieces can be described as the following:

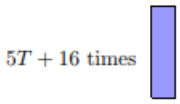
1. First for every $a_i \in A$ the sequence goes (in this order):



2. Then to fill the top of all the s buckets the subset fillers:

3. Then the T-shape to unlock the 'lock':

4. And to clear the whole board by filling the 'fill area':



3.4 Proof

Here I will prove that a "yes" instance of the 3-Partition problem will correspond to a Tetris game where the gameboard can be cleared.

The sequence of Tetris pieces before the 'subset fillers' represents the values $a_i \in A$. Since A is part of a "yes" instance it can be divided into s subsets, that all sum to T. To fill the game board you must put the 'values' in the right 'buckets'. IN order to put a 'value' into a bucket, you put the first sequence of pieces into the bucket as shown below (in this example for $a_i = 3$):

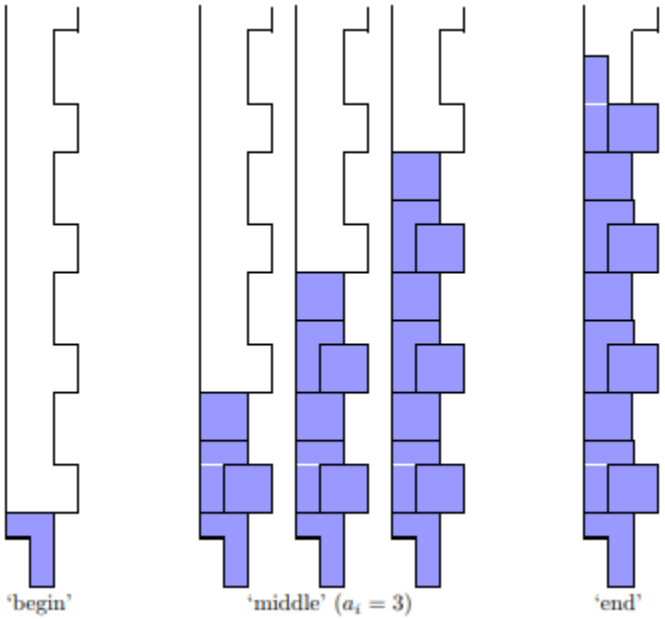


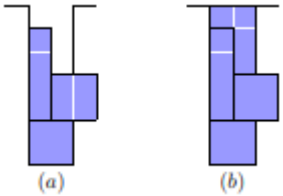
Figure 3: The first sequence of peices into the gameboard

This allows multiple values to be put into the same bucket because the shape left after putting 'value' in the bucket is the same as before the 'value' was put in. The reason there are $T + 3$ notches is because for each a_i, the beginning and end piece of the first sequence take up one more notch. Therefore the height of the filled column should be:

$$\Sigma_{a_i \in B_j} \text{ (height of 'begin')} + a_i\text{(height of 'middle')} + \text{(height of 'end')} = \Sigma_{a_i \in B_j}(3 + 5a_i + 2) = \Sigma_{a_i \in B_j}(5a_i + 5)$$

In the equation above, B_j stands for the bucket B for $1 \leq j \leq s$. Because $\Sigma_{a_i \in B_j} a_i = T$ and $|B_j| = 3$ in a "yes" instance, the height turns out to be $5T + 3\Delta 5 = 5T + 15$, and because the last piece sticks out two squares, the total height will be $5T + 17$, which is less then $H(= 5T + 18)$ and therefore the height of a 'bucket' is sufficient.

After filling all the 'buckets' as described above, there will be s buckets which look like (a). These buckets can all be filled to look like (b) using the s 'subset fillers'.



Now the whole board is filled except for the 'lock' and the 'fill area'. Next you put the 'lock'-piece in the 'lock'-space and thereby clear the top two lines of the board. The the 4 by $5T + 16$ rectangular 'fill area' is filled with the $5T + 16$ straight lined Tetris pieces. This is done by stacking the pieces horizontally, which clears the gameboard. Therefore, I have

4 Implementation

It would be difficult to create a SAT implementation of Tetris, so I decided to create a 3-partition solver and map it to the tetris example. The code for my 3-partition solver is shown below in the index of the report.

In simple terms, the solver creates an array with s elements that represent the s subsets of the problem. Each of the elements in the array are set the expected total for each subset. Then it checks if the given set matches the conditions required by the 3-Partition problem. After that the algorithm takes a rather brute force approach to solving the problem.

The algorithm goes through the given set of integers and puts them in a subset by subtracting the integer from the current element in the list. It continues to do this until it cannot fit an integer into that particular subset and puts it into the next subset. If it goes through the whole set of integers and not all of the values in the list are zero, then it returns to a previous iteration and moves an integer in the set from one subset to another. Once all elements in the list are zero, then the algorithm returns true. If there are no placements within the subsets for all of the integers then the algorithm returns false.

As an example I input the set $S = [23, 40, 27, 35, 24, 31]$ and the total $T = 90$ into the program. In this case there should be 2 subsets that both equal 90 because the sum of the set should be equal to sT and 23+40+27+35+24+31 = $180/90 = 2 = s$. The problem should evaluate to true with the sets [23,40,27] and [35,24,31], which both sum to 90. When this is input to the program, it evaluates to True as expected and prints out "Yes."

In terms of Tetris, this would be most similar to putting the sequences of blocks in to the buckets and taking them out until you have effectively filled every bucket to the top.

In the example of a no instance, I could simply change one of the numbers in the previous set so that it is now $S = [23, 32, 27, 35, 24, 31]$. When this is input into the program it evaluates to false and outputs "No." This could be mapped to Tetris having one of the buckets not filled completely so the entire gameboard is not cleared.

4.1 Index

```
import copy
# Helper function for solving 3 partition problem
# It return True if there exists three subsets with given sum
def subsetSum1(S, n,T, totals, bools):
        # return True if subset is found
        if (sum(totals)==0):
                return True
        # base case: no items left
        if (n < 0):
                n=len(S)
        #print(S)
        #print(totals)
```

```
        #print(bools)
        for i in range(len(totals)):
                mid = True
                for j in range(i):
                        if (bools[j]==False):
                                mid =True
                        else:
                                mid=False
                if((mid) and totals[i]-S[n]>=0):
                        totals[i] = totals[i]-S[n]
                        bools[i]=subsetSum1(S, n - 1, T, copy.deepcopy(totals),copy.deepcopy(
                        totals[i] = totals[i]+S[n]
        # Case 1. current item becomes part of first subset
        ren = False
        for i in range(len(bools)):
                ren = ren or bools[i]
        return ren


# Function for solving 3-partition problem. It return True if given
# set S[0..n-1] can be divided into three subsets with equal sum
def partition(S, n,T):
        if (n < 3):
                return False

        # get sum of all elements in the set
        total = sum(S)
        if (total!=(n/3)*T):
                return False
        totals=[]
        bools=[]
        for i in range(int(n/3)):
                totals.append(T)
                bools.append(False)
        # return True if sum is divisible by 3 and the set S can
        # be divided into three subsets with equal sum
        return ((total % 3)==0) and subsetSum1(S, n - 1,T, totals,bools)

# main function for 3-partition problem
def main():
        # Input: set of integers
        S = [23,40,27,35,24,31]
        T=90
        # number of items in S
        n = len(S)

        if (partition(S, n,T)):
                print("Yes")
        else:
                print("No")

        return 0

if __name__ == '__main__':
    main()
```

References

[1] David Liben-Nowell Erik D. Demaine, Susan Hohenberger. Tetris is hard, even to approximate. 2008.

[2] Hendrik Jan Hoogeboom Ron Breukelaar and Walter A. Kosters. Tetris is hard, made easy.
[1] [2]