Tiling Problems

Caleb Alexander

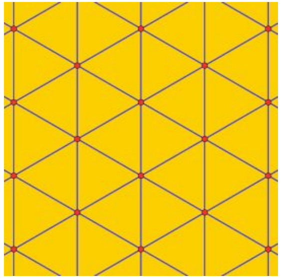
Background

Tiling problems began with so-called “regular” tilings.

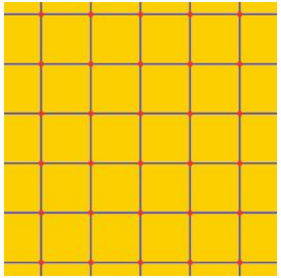
- Tilings of the plane with which we allow only a single type of regular polygon.

As it turns out, there exist only **3** regular tilings of the plane. Those being for the:

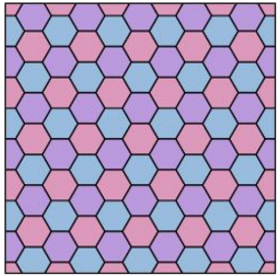
Triangle



Square



Hexagon



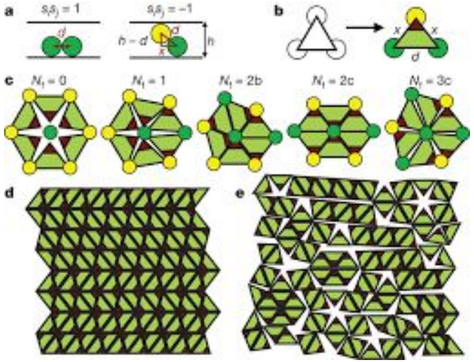
Images from wikipedia

Background (cont)

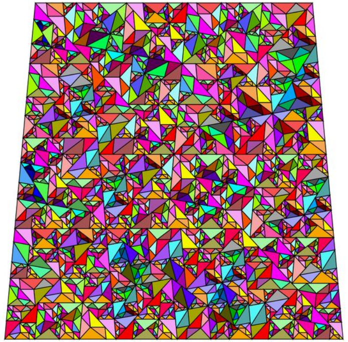
A natural next question which arose, was “if we loosen our restrictions, what tilings can we find”.

As it turns out, there are many interesting results, for example:

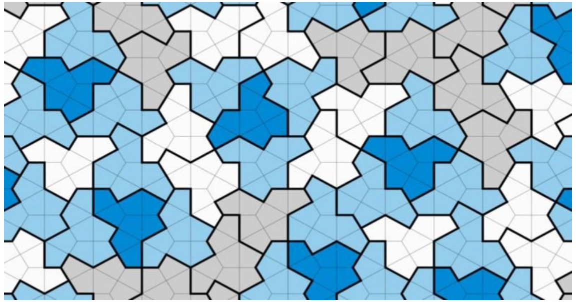
Any triangle can tile the plane.



We can find tilings which use multiple shapes



Tilings can be aperiodic (more to come)

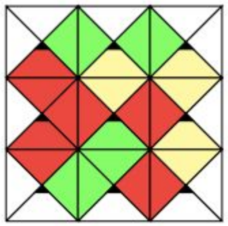


Practical Importance and Equivalences

- **Periodicity** of a tiling, has to do with periodic sets (those who know group theory should think of symmetry groups)
- Tiling can also be useful for **storage efficiency**. We can think of this as a question of which shapes allow us to effectively utilize all the space available to us.
- There are also some light equivalences between certain tiling problems and **graph coloring problems**, if we consider infinite graphs.

NP-Hardness (Generality)

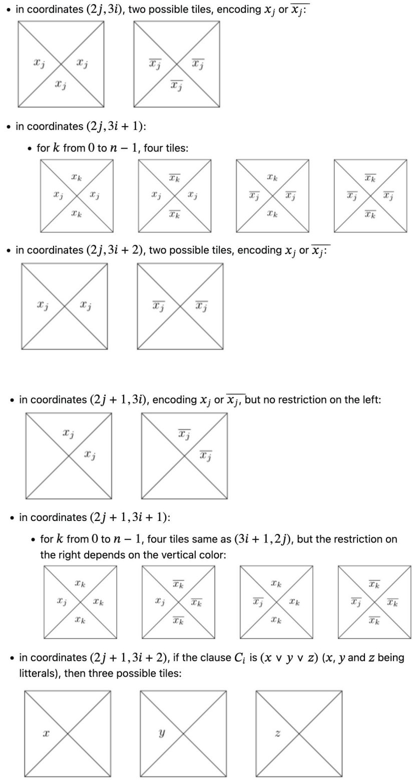
In general, tiling problems for a finite board of square sections where each square is split into 4 colors are **np-complete** and thus also **np-hard**.



This is because, all tiling problems reduce to the rectangle-tiling problem (the case where are board is rectangular) and 3-SAT can be reduced to this problem.

Consider a CNF φ such that each clause is in a set of "colors" $\{x_0, \dots, x_{n-1}\}$. Now we construct a tiling (T, 2n, 3m), such that T can tile a grid of sixe 2n x 3m iff φ is satisfiable.

We do this simply by following a procedure: for i in {0, …, m-1} and j in {0, …, n-1} we can choose colors as shown on the side. A rote checking shows that we have not broken any tiling rules, and the first column will give the satisfying TA for φ.



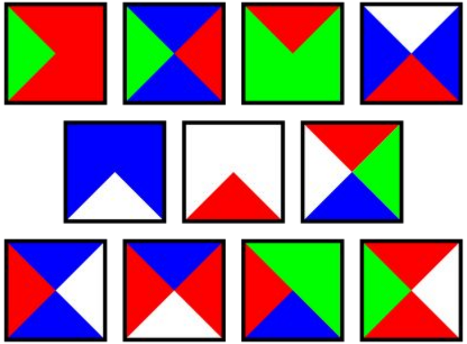
- in coordinates $(2j, 3i)$, two possible tiles, encoding x_j or $\overline{x_j}$:
- in coordinates $(2j, 3i+1)$:
  - for k from 0 to $n-1$, four tiles:
- in coordinates $(2j, 3i+2)$, two possible tiles, encoding x_j or $\overline{x_j}$:
- in coordinates $(2j+1, 3i)$, encoding x_j or $\overline{x_j}$, but no restriction on the left:
- in coordinates $(2j+1, 3i+1)$:
  - for k from 0 to $n-1$, four tiles same as $(3i+1, 2j)$, but the restriction on the right depends on the vertical color:
- in coordinates $(2j+1, 3i+2)$, if the clause C_i is $(x \vee y \vee z)$ (x, y and z being litterals), then three possible tiles:

Special Undecidable Case

Domino Problems:

- A class of tiling problems which use some arbitrary set of “dominoes” (essentially a set of valid squares)

It has been shown that **any** turing machine can be translated into a domino set, and thus there must be a recreation of the halting problem with a domino set. Thus in general, deciding whether an arbitrary set of dominos can tile the plane is undecidable.



Wang’s Dominos. A set of Dominoes which were shown to tile the plane **only** aperiodically.

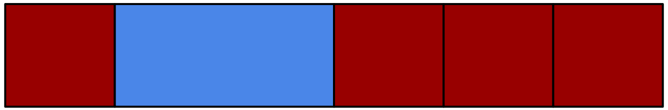
Specific Example With a DP Solution

Given a 1-dimensional board (an array) of length n, how many ways are there to tile the board with tiles which are either a square (of length 1), or a rectangle (of length 2).

We can use dynamic programming to solve this. The DP Approach is defined by the following recurrence relation. Notice that if f(n) is the count of ways to tile a board of length n>=2, then:

$$f(n) = f(n-1)+f(n-2)$$

This is because from a local perspective (consider the start of our array), either it starts with a red tile or a blue tile. Thus using recursion is natural for our DP Solution.



Example Tiling

```
global stored_vals = {}
stored_vals[f(0)] = 1
stored_vals[f(1)] = 1

find_tilings(n):
    board = [n]
    find_tilings_rec(0, board)

find_tilings_rec(n, board):
    if f(board.len - n) in stored_vals.keys:
        return f(board.len - n)
    else n is not end:
        board[n] = red
        red_tilings = find_tilings(n+1, board)
        board[n:n+1] = blue
        blue_tilings = find_tilings(n+2, board)
        return red_tilings + blue_tilings
```

Sources

Information on the history of tiling in math

- https://www.quantamagazine.org/a-brief-history-of-tricky-mathematical-tiling-20231030/

Images from slide 2:

- https://en.wikipedia.org/wiki/Triangular_tiling
- https://en.wikipedia.org/wiki/Square_tiling
- https://en.wikipedia.org/wiki/Hexagonal_tiling

Images from slide 3:

- https://www.researchgate.net/figure/Tiling-the-plane-with-isosceles-trianglesa-Close-packed-spheres-are-separated-by-one_fig3_23673931
- https://blog.wolfram.com/2019/03/07/shattering-the-plane-with-twelve-new-substitution-tilings-using-2-phi-psi-chi-rho/
- https://cnewsliveenglish.com/news/23392/einstein-tile-the-13-sided-shape-that-forms-an-unrepeatable-pattern

Tiling NP-Completeness Proof Info and Image from slide 5

- https://people.irisa.fr/Francois.Schwarzentruber/publications/arxiv1907.00102.pdf

Another proof for tiling NP-Completeness

- https://cs.stackexchange.com/questions/147776/prove-tiling-is-np-complete

Image for slide 6

- https://en.wikipedia.org/wiki/Wang_tile

Explanation of the DP Solution for a simple example

- https://www.youtube.com/watch?v=L1x3an2pl3U&ab_channel=WilliamFiset