Sampling domino tilings

Last time, using Thurston's tiling group, we assigned a height to each point on the border of a Cartesian lattice. There is an easy method to compute these heights. First, color lattice cells with black and white like in a chessboard. Assign a height of 0 to the reference point, then follow the border of the lattice computing heights for the vertices as follows: the height increases by 1 whenever we traverse an edge with a black square to the left, and the height decreases by 1 when the square to the left is white.

For a given tiling, we can actually use this method to assign heights to all lattice points, not only to border ones. For interior vertices, use any path around the tiles to compute the height (all of them give the same value). The heights associated with lattice points define a piecewise linear surface (try to visualize it!), we will use this surface to analyze the mixing time of our Markov Chain algorithms. One fact we will use is that, among the many surfaces compatible with the heights associated with border vertices, there is a unique "highest" surface, and a unique "lowest" surface.

A natural Markov Chain for generating domino tilings is the following:

1. Start with an arbitrary tiling (we can construct such a tiling in $O(n)$ time using Thurston's algorithm).
2. Pick a 2×2 window uniformly at random.
   If the window looks like  , change it to  w.p. 1/2.
   If the window looks like  , change it to  w.p. 1/2.
   Otherwise, do nothing.
3. Repeat Step 2.

Figure 1 shows the change in height after one chain move, the chain works on the surface associated with the current tiling by pushing/popping by 4 units the center of the configuration (a local maximum or a local minimum).

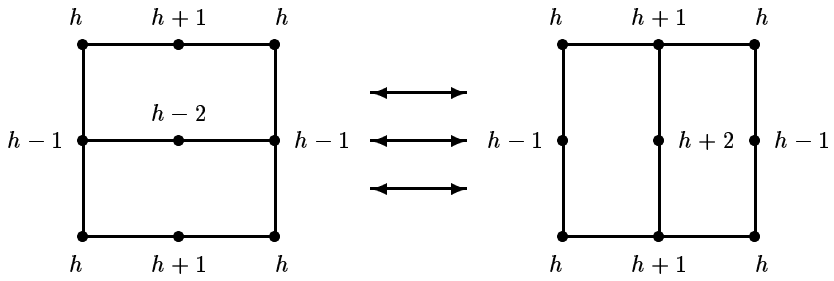


Figure 1: Height change after one move

We will use the following idea for analyzing the mixing time: start one copy of the Markov Chain algorithm on the tiling corresponding to the highest surface, and another coupled copy on

the tiling corresponding to the lowest surface. We can use the volume between the two surfaces as a measure of the distance between them, when the volume becomes zero the two chains have coupled.

Consider the following modified algorithm:

1. Pick a 2×2 window uniformly at random and flip an unbiased coin c.
   If, in T_i, the window looks like  and $c = heads$, change it to  .
   If, in T_i, the window looks like  and $c = tails$, change it to  .
   Otherwise, do nothing.
2. Repeat Step 1.

We define a coupling on (T_1, T_2) by choosing the same window and the same coin flips in the above algorithm. Note that the "above" relation for surfaces defines a partial order on tilings; moreover, the partial order has a structure of distributive lattice. The coupling we just defined preserves the partial order, so we can apply a general result by Propp and Wilson.

Theorem : For a coupling that preserves the partial order in a distributive lattice, the coupling time is

$$T_c = T(top, bottom),$$

where *top* (*bottom*) is the unique smallest (largest) element of the lattice, and $T(top, bottom)$ is the expected time for the smallest and largest element to agree.

This result does not give an apriori bound on the coupling time, but provides a good *stopping rule*. To estimate T_c, just run several times the coupled Markov Chains starting from the highest/lowest surfaces, and take the largest coupling time. Although this method does not guarantee a polynomial running time, it is very efficient in practice. In fact, it is often used to speed-up algorithms for which polynomial time bounds are known.

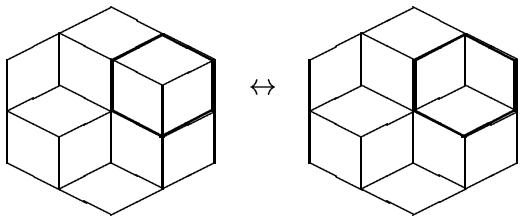
Sampling lozenge tilings

An algorithm for generating random lozenge tilings based on local moves is the following:

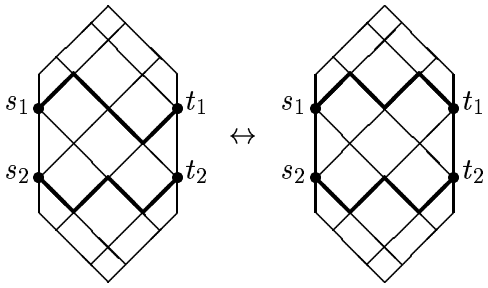
1. Start with an arbitrary lozenge tiling.
2. Pick a hexagonal window uniformly at random and flip an unbiased coin c.
   If the window looks like  and $c = heads$, change it to  .
   If the window looks like  and $c = tails$, change it to  .
   Otherwise, do nothing.
3. Repeat Step 2.

The moves of this Markov Chain are illustrated in Figure 2(a). As in the domino case, this chain can also be shown to be connected, and to converge to the uniform distribution over tilings.

Let us now interpret this Markov Chain in the world of routings (see Figure 2(b)). It is clear that a lozenge rotation induces a local move on the corresponding routing, in which a "peak" or a "valley" is inverted by moving two edges.



(a) Tiling rotation

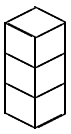


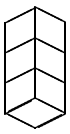
(b) Routing rotation

Figure 2: Lozenge rotations

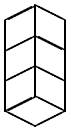
It turns out that the Markov Chain in the routings world becomes considerably easier to analyze if *every* peak and valley can give rise to a rotation: note that this is not the case for the above chain, since sometimes when we try to invert a point the move will be blocked by the presence of another path. (Recall that the paths in a routing are not allowed to cross.) This motivates the introduction of a more general set of moves in which entire *towers* of cubes are rotated. The original moves will simply correspond to the special case of rotating a tower of height 1. The modified algorithm is:

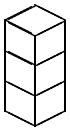
1. Start with an arbitrary lozenge tiling
2. Pick a hexagonal window uniformly at random and $r \in_u (0, 1)$.

   If the window looks like 

, change it to 

if $r \leq 1/2k$.

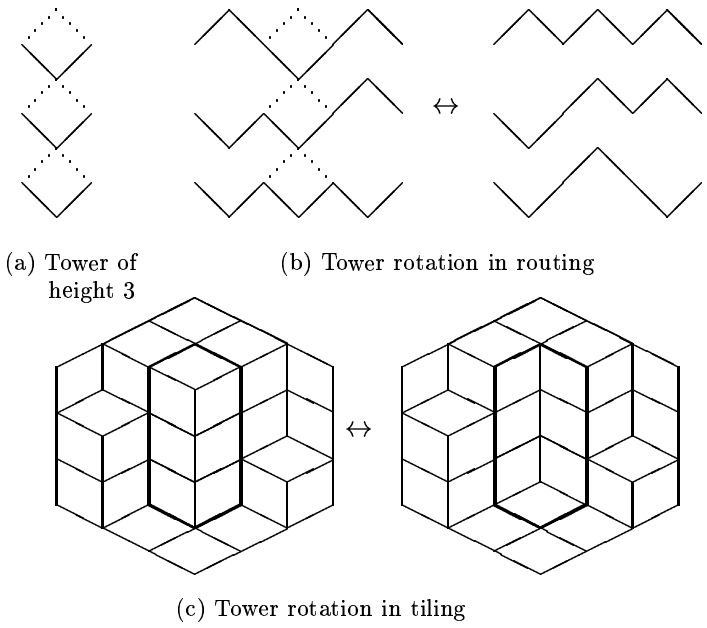
   If the window looks like 

, change it to 

if $r \geq 1 - 1/2k$.

3. Repeat Step 2.

An illustration of the tower moves is shown in Figure 3.



(a) Tower of height 3 (b) Tower rotation in routing

(c) Tower rotation in tiling

Figure 3: A move in the modified Markov chain for lozenge routings, and its counterpart for tilings