CR12: transfer matrix and power iteration

The goal of the work is to approximate a growth rate (physicists sometimes talk about "entropy").

We want to tile the 2D square grid with 3 types of tiles: squares 1×1 □, rectangles 2×1 □□ and rectangles 1×2 ▯, and we want to estimate how many there are such tilings. Equivalently, it's the number of (not necessarily perfect) matchings in the (infinite) square grid graph.

Note that, for the method presented here, the set of tiles is not very important: it works for any finite set of tiles, with only few adaptations.

We are not very interested in the "theoretical" part here. We will focus on programming and computation.

We denote $f(n, m)$ the number of possible tilings of a rectangle $n \times m$ with these 3 tiles. One can see that $\lim_{n\to\infty} \sqrt[n\times n]{f(n, n)}$ converges to a constant, the *growth rate*, which we will denote by c, and we want to approximate this constant. (Note that you have to be careful what happens at the "border" of the rectangle we want to tile. But here, everything is going pretty well.)

It is generally difficult to find exact growth rates in 2D (there are only few cases where the exact value is known, more informations at the end of the document if you are interested). We therefore want to approximate it. To do this, we will use a standard technique of "transfer matrix", and then of power iteration to get the largest eigenvalue.

- You can use the programming language you want.
- Send me your programs / results / comments / questions to michael.rao@ens-lyon.fr
  - at the end of the TD (mostly to get your emails, and to have something if you forgot to send it after, or if you do not have time to work more on this, or if it is the best you can do :)).
  - and you can send "updates" up to Wednesday 24 November.

1 1D world

Note that in one dimension, the problem becomes really easy, so easy that you don't need a computer...

If you want to tile a strip $n \times 1$ □□□□□□□□□□... with the 3 tiles, the tile 1×2 ▯ cannot be used.

Now we count the number of tilings $f(n, i)$ of $n \times 1$. If the leftmost tile is a □□, then the remainder will be a tiling of $(n-2) \times 1$. Otherwise, the leftmost tile is □, and the rest will be a tiling of $(n-1) \times 1$. We therefore have $f(n, 1) = f(n-2, 1) + f(n-1, 1)$, and we directly have the recurrence of the Fibonacci sequence. So we know that $\lim_{n\to\infty} \sqrt[n]{f(n, 1)}$ is the golden ratio.

2 Bands

But for fixed height bands, things are not "theoretically" more complicated (but becomes tricky to do it by hand...)

There are always finitely many cases to handle. For example, for $n \times 2$:

- either the leftmost tile is ▯, and then the rest is a tiling of $(n-1) \times 2$,
- either there are two □ on the left, and the rest is also a tiling of $(n-1) \times 2$,
- if there are two □□, the rest is a tiling of $(n-2) \times 2$,
- but if there are is only one □□ (w.l.o.g on the first line), then we have a □ on the second line, and the rest is a tiling of $(n-1) \times 2 \setminus (0, 0)$ (i.e. ▯▯▯▯▯▯▯▯▯...). So one can define $f'(n, 2)$ to be the number of tilings of $n \times 2 \setminus (0, 0)$, and find formulas for $f'(n, 2)$ too. This can be done without adding a new parameter.

But at the end (and for every k), one can always find vectors A_k, B_k and matrix M_k such that $f(n,k) = A_k^T \times (M_k)^n \times B_k$.

Let $c_k = \lim_{n\to\infty} \sqrt[n]{f(n,k)}$. Note that c_k is the largest eigenvalue of M_k. And we can easily see that $c = \lim_{k\to\infty} \sqrt[k]{c_k}$.

Your goal now is to write a program which constructs such M_k for every k. (A_k and B_k are not very important for the following.)

Note that there are several possibilities to "encode" things. Try to find one such that M_k is not too big...

3 Power iteration method

Now that we have a matrix M_k, and we want to find the largest eigenvalue. Note that our matrices are (almost) Perron-Frobenius matrices. The largest eigenvalue is unique, real and positive. One can easily find this eigenvalue by the power iteration method. Note also that c_k is the largest eigenvalue of (M_k)...

Let v_0 be a random positve vector (for example $(1, 1, 1..., 1)$) and let $v_{k+1} = \frac{Mv_k}{|v_k|}$. v_k will naturally converges to "the" eigenvector associated with the largest eigenvalue, and it's norm will converges to the largest eignevalue.

Program the power iteration method.

4 Go as far as possible

Now you know how to compute c_k, for every k. The problem is that the complexity is exponential in k.

Try to computes c_k for the largest k you can. How far you go ?

How many digits you can guess for $c = \lim \sqrt[k]{c_k}$?

5 Ideas to go faster

- Note that the matrix can be sparse. Can this help you?
- You can also use symetries to reduce the size of the matrix.
- What happens "on the border" can affect the speed of convergence. For example, one can tile all the strip, but authorizing to bypass the border. Or one can tiles cylinders instead of that stips.
- Note that we do not look too much at the theoretical part. But, nothing tells us how far we are from the limit. If we manage correctly what happends on the border, on can get lower or upper bounds of c.

6 More information on what we are trying to do

Few of the structures of 2D models are known exactly. Two " classic " examples of known models:

- "Dimere model": This is equivalent to tiling with 1×2 and 2×1 rectangles. Equivalently, it is the number of perfect matchings in the square grid. Our exercise wanted to estimate the number of matchings (not necessarily perfect) of a square grid.
- "Hard hexagon model": number of stable sets in a triangular grid.
- On the other hand, the "hard square model" (stable sets in a square grid) is a long standing open problem in the field of statistical mechanics.