Existence of Gibbs measure for a model of T-tessellations

Jonas KAHN

December 13, 2010

1 Introduction

Some man-made landscapes, such as plots of land may be viewed as T-tessellations, that is a tessellations of a subset of the plane, where all vertices are degree three and with one flat angle. Adamczyk and Kiêu have developed a random model aiming at representing such landscapes. This article proves that their model, and extensions thereof, are well-defined.

I shall first shamelessly summarise their arguments for defining this new model. Models with T-vertices had already been studied [Mackisack and Miles, 1996], but they are usually built starting from point seeds from which segments grow until they are blocked: this situation does not really correspond to cultivated fields, and if the segments all grow at the same speed, the parcels cannot have different scales.

At another extreme, Arak et al. [1993] have introduced a very general model for random graphs, directly defined by their measure on the set of graphs. The measure depends on an energy function that can be specified to yield T-tessellations, as detailed by Mackisack and Miles [2002]. These graphs have very nice mathematical properties. As a result, they may be sampled exactly, without resorting to Metropolis algorithm. However these nice properties come at a price: the tessellations are necessarily very random. For example, the intersection with any line is a Poisson process. It is thus hard to generate long-range spatial correlations.

Adamczyk and Kiêu define their model as a non-normalised density with respect to the model of Arak et al. [1993]. Different energies allow different kinds of landscapes. We may for example require that all parcels have a similar area, or penalise sharp angles. The price to pay is harder sampling, requiring Monte-Carlo Markov chain algorithms.

From a theoretical point of view, Adamczyk and Kiêu have not managed to prove that their measure was finite, which is necessary for the model to be well-defined. This article focuses on proving that we have a true probability measure.

In section 2, we describe the mathematical structure of the models, make some general remarks on the structure of T-tessellations, introduce the notations and our strategy. Mainly, it is enough to bound precisely enough the weight of all T-tessellations on a given configuration of lines. This can be achieved by bounding the number of different T-tessellations on those lines. We then count. To do that, we devise algorithms that can rebuild the tessellation from their input, and count the number of different inputs they can have. The algorithms and the final theorem are detailed in section 3. Formal proofs of correctness of the algorithms are all delayed to the appendix A. Finally we discuss the limitations of our proof and hint at possible ameliorations in section 4.

2 Notations and strategy

Let us first recall some definitions related to Poisson line processes on the plane $\mathbb{R}^2$.

We may define a line D by an angle $\alpha \in [0, \pi[$ and a distance $p \in \mathbb{R}$. Let us consider the origin $O = (0, 0)$ and the point P of radial coordinates (p, α). Then the line $D(p, \alpha)$ is the line orthogonal at point P to the line (OP). This parametrisation is usual since the lines corresponding to a homogeneous Poisson process on $[0, \pi[\times\mathbb{R}$ yield a stationary and isotropic process.

We now consider a domain W in the plane. To avoid many complications thereafter, we require it to be convex. We denote by λ_W the measure on the Poisson line process restricted to W, that is the image of the Poisson point process on $D_W \subset [0, \pi[\times\mathbb{R}$ where D_W corresponds to the set of lines in the plane that intersect W. The measure of this process is denoted λ_W^τ where the intensity τ is the expected number of lines in the process. It may be seen as a measure on the set $\mathcal{L}_W$ of sets of line on W. In particular, if we denote by L_W^τ the random set of lines, its cardinal is a Poisson variable with parameter τ:

$$\mathbb{P}\left[\# L_W^\tau = k\right] = \exp(-\tau)\frac{\tau^{-k}}{k!}.$$

A finite polygonal tessellation of W is a finite partition of W into cells such that the border of each cell is an union of a finite number of segments and a part of the border of W itself. A vertex of a tessellation is T-shaped if it is of degree three and two of the incident edges are aligned. A *T-tessellation* is a polygonal tessellation such that :

- All its inner vertices are T-shaped.
- No two segments are aligned.

We denote by $\mathcal{T}_W$ the set of T-tessellations on W. We denote by $L(T)$, or simply L the set of lines that support the segments of T.

The measures defined by Adamczyk and Kiêu, which include the particularization of the model by Arak et al. [1993] to T-tessellations, are of the form:

$$\mathrm{d}\gamma_W(T) = \exp(-H(T))\mathrm{d}\lambda_W^\tau(L(T)),$$

where the energy $H(T) \geq C\#L(T)$ is bounded from below by a linear function of the number of lines in the tessellation $\#L(T)$. As a remark, scaling W is equivalent to scaling τ^2.

We aim at proving that the corresponding partition function is finite. We may bound it by:

$$\begin{aligned}
Z &= \int_{\mathcal{T}_W} \exp(-H(T))\mathrm{d}\lambda_W^\tau(L(T)) \\
&\leq \int_{\mathcal{T}_W} \left[\sup_{T':L(T')=L(T)} \exp(-H(T')) \right] \mathrm{d}\lambda_W^\tau(L(T)) \\
&\leq \int_{\mathcal{L}_W} \exp(-C\#L(T))\# \left\{T : L(T) = L\right\} \mathrm{d}\lambda_W^\tau(L) \\
&\leq \sum_{k=0}^{\infty} \exp(-Ck)\mathcal{N}(k) \exp(-\tau)\frac{\tau^{-k}}{k!}, \qquad (1)
\end{aligned}$$

where we have used the Poisson distribution on the number of lines in L_W^τ and denoted by $\mathcal{N}(k)$ the maximum number of ways to obtain a T-tessellation as segments on k given lines, whatever the position of those lines:

$$\mathcal{N}(k) = \sup_{L:\#L=k} \# \left\{T : L(T) = L\right\}$$

Our strategy consists in counting these T-tessellations on fixed lines, or rather bound their number from above.

Since $k! \geq (k/e)^k$, the sum in the partition function (1) converges if, for any real a:

$$\mathcal{N}(k) = o(k^k a^k). \qquad (2)$$

This implies that k-th powers a^k are free when we are counting the tessellations, as long as there is a bounded number of such factors. In particular, we may label each line with "red" or "blue" (2^k possibilities), or more generally with any properties among a finite set. We may also split k indiscernible objects among the k lines (less than 4^k possibilities).

To bound this number $\mathcal{N}(k)$, we shall use descriptions of the tessellation, essentially labels on lines. These descriptions, or *labelling schemes* will be shown to characterise the tessellation, given the lines L, by rebuilding T with an algorithm using that description as input. A bound on the number of different descriptions then yields a bound on $\mathcal{N}(k)$. Many of the further notations are also thought to be easy of use within an algorithm.

Let us have a closer look at T-tessellations.

A T-tessellation T is built on a set of lines L that support its *segments*. We shall write $s(l)$ for the segment supported by the line $l \in L$, and conversely $l(s)$ for the line that supports s.

The endpoints of those segments can only be an intersection with another line, or with the border of W. So that, knowing L, the only places where something can happen are those intersections of lines and possibly border. We call those intersections *events*. A generic event is noted $\mathfrak{e}$. When specifying the event, through the lines that intersect, we write $\mathfrak{e}(l, m)$, for $l, m \in L$. Naturally the event $\mathfrak{e}(l, m)$ is the same as $\mathfrak{e}(m, l)$. A special case is when a line intersects the border $\mathcal{B}$ of W. Since W is convex, this happens exactly twice, so that $\mathfrak{e}(\mathcal{B}, l)$ and $\mathfrak{e}(l, \mathcal{B})$ are different. Conventionally $\mathfrak{e}(\mathcal{B}, l) < \mathfrak{e}(l, \mathcal{B})$ for the order we define now. We write $\mathcal{E}$ for the set of events.

Given k lines, we can find their $k(k-1)^2$ crossing points, as well as their $2k$ crossing points with the border. They are almost surely distinct. We choose an axis along which each crossing has a different coordinate. This axis will be called the *time axis*, or indifferently *abscissas* axis. The corresponding coordinates are called either *times* or *abscissas*. We order the events according to time. The ordered list of events will be denoted $\mathcal{E}^o$. The reverse-ordered list will be denoted $\mathcal{E}^r$. We shall use the usual vocabulary associated to time, such as saying that a point (or an event defined at that point) happens before another if its abscissa is smaller. We also use *left* and *right* for smaller and larger times.

Since the crossings all happen at different times, the extremal points of each segment of the tessellation have different times. We say that the segment is *born* at its extremity with lower time, and *dies* at the other. We denote by $x_b^T(l)$ and $x_d^T(l)$ the times of birth and death of the segment $s(l)$. The line crossed at the birth is the *parent* of the segment's line, and the segment's line is its *child*. The line crossed at death is the *killer* of the segment's line.

The segment's line is its *victim*.

These relations thus give us two trees, the *tree of births* and the *tree of deaths*. Both have $k+1$ nodes, labelled by the border and the k lines, and both are rooted at the border. For the tree of deaths, parent and child correspond to killer and victim.

The tree of births encode all information about births, that is on leftmost extremities of segment $x_b^T(l)$. Symmetrically, the tree of kills encode all information about deaths, that is on rightmost extremities of segments $x_d^T(l)$. So that rebuilding the two trees is equivalent to rebuilding the tessellation.

Since the segments describe the tessellation, and for algorithmic purposes, we now think of a T-tessellation on a set of lines L as a couple of functions $T = (x_b^T, x_d^T)$ with $x_\bullet^T : L \cup \mathcal{B} \to \mathbb{R}$. The border is added in the domain for the algorithms.

Conventionally, to ease the writing of the algorithms, we now require that:

- W is contained in the band of abscissas $(0,1)$.
- The border is "always alive": $x_b^T(\mathcal{B}) = 0$ and $x_d^T(\mathcal{B}) = 1$.

Even when they follow these requirements, all such couples of functions are not a tessellation, let alone a T-tessellation. We shall dub *prototessellation* any such couple. The notion will be mainly useful for initialisation of the algorithms.

A T-tessellation is a prototessellation P with the following three properties:

- Segments do not cross:

$$\begin{aligned}\forall\, l,m \in L : \neg \big\{ &\big[x_b^P(l), x_b^P(m) < e(l,m) < x_d^P(l), x_d^P(m)\big] \\ &\text{or } \big[x_b^P(l) = x_b^P(m)\big] \text{ or } \big[x_b^P(l) = x_d^P(m)\big] \text{ or } \big[x_d^P(l) = x_d^P(m)\big] \big\}\,. \end{aligned} \tag{3}$$

- Segments are born on the inside of another segment, or on the border:

$$\text{If } x_b^P(l) = e(l,m), \text{ then } x_b^P(m) < e(l,m) < x_d^P(m). \tag{4}$$

- Segments die on the inside of another segment, or on the border:

$$\text{If } x_d^P(l) = e(l,m), \text{ then } x_b^P(m) < e(l,m) < x_d^P(m). \tag{5}$$

A prototessellation where segments do not cross (3) is a *pretessellation*. We deal with such objects within the algorithm, in some cases as output. We shall usually write P for all prototessellation, and x_b^P and x_d^P for the corresponding times of birth and death.

3 Algorithms and Result

3.1 Preliminary algorithm

If we know the tree of births alone, we can almost rebuild the tessellation. We only need the number of murders of each line, which is essentially free (4^k). Hence counting the number of tessellations in the worst case $\mathcal{N}(k)$ is essentially equivalent to counting the highest possible number of trees of births on k lines.

We start with a labelling scheme that only gives existence of Z for low intensities ($\lambda < (4e)^{-1}$ with non-negative energy H). However it is a basis of our final labelling scheme, and the proof of its efficiency introduces ideas that we shall use again, while staying in an easier context.

A first way to label each line so that we may rebuild the tessellation is the following:

- Give for each line its parent. That yields k^k possibilities.
- Give for each line its number of murders. That yields $\binom{2k}{k} \leq 4^k$ possibilities.

The number of murders of a line l is defined as

$$M^T(l) = \# \left\{ m : x_d^T(m) = \mathfrak{e}(m, l) \right\}. \tag{6}$$

Conventionally, we may set $M^T(\mathcal{B})$ to infinity. Though it is not necessary, it avoids to keep track of it in the algorithm.

We may now rebuild the tessellation with Algorithm 1.

Informally, we move along the abscissas axis, while prolongating the segments that are alive. We know when each segment is born, so we add them then. When two segments cross, we look at their remaining number of murders. One of the two must be zero. The corresponding segment is killed. The other segment's number of murders is decreased by one. When a segment hits the border, it is also killed.

Lemma 3.1. *Algorithm 1 yields T.*

Proof in appendix.

The lemma yields $\mathcal{N}(k) \leq (4k)^k$. Putting that back into bound (1) would yield a convergent series if $\lambda < 1/(4e)$ and $H \geq 0$. Compare with calculations in section 3.3 for details.

Algorithm 1 Rebuild from tree of births and number of murders.

Input: The set L of lines of the tessellation, a prototessellation (x_b, x_d) such that $x_b = x_b^T$, a murder function $M : L \cup \mathcal{B} \to \mathbb{N}$ such that $M = M^T$ the number of murders in the real tessellation (6), the ordered set $\mathcal{E}^o$ of events.

```
 1: for all $l \in L$ do
 2:     $x_d(l) \leftarrow 1$                                   ▷ Death time temporarily set to maximum
 3: end for                                                    ▷ End of initialisation
 4: for $\mathfrak{e} \in \mathcal{E}^o$ do                     ▷ Consider potential events timewise
 5:     $l_1, l_2 \leftarrow l(\mathfrak{e})$
 6:     if $x_b(l_1), x_b(l_2) < \mathfrak{e} < x_d(l_1), x_d(l_2)$ then      ▷ Do the lines cross?
 7:         if $M(l_1) = 0$ then                               ▷ Which line is killed?
 8:             $x_d(l_1) \leftarrow \mathfrak{e}$                          ▷ Kill it
 9:             $M(l_2) \leftarrow M(l_2) - 1$                 ▷ Count that $l_2$ killed it
10:         else
11:             $x_d(l_2) \leftarrow \mathfrak{e}$
12:             $M(l_1) \leftarrow M(l_1) - 1$
13:         end if
14:     end if
15: end for
        return P
```

3.2 Main algorithm

The previous labelling scheme still uses too much info for proving existence of our measures. Namely specifying the whole tree of births dooms the effort.

Next algorithm rebuilds T while knowing only part of the times of birth. We call *orphan* a line whose parent is not specified in the labelling scheme. The price to pay is higher complexity: instead of one pass on events, we have to loop back and forth in time, prolongating orphan segments to their birth, and cutting too old segments until stabilisation.

Let us be precise. The algorithm is supposed to take the following input data on the lines. This list will be referred to as *requirements*.

1. Label each line as "leaf" or "not leaf". Leaves will not be allowed to have any child.

2. For each line, its parent is given, and/or it is specified as the parent of its first child.

3. Each line is given a number of *virtual murders*, whose properties we make precise after formulating the algorithm.

4. Each line is given a number of *other children*, also made precise below.

The first set of labels yields a 2^k factor. The third and fourth sets of labels each yield as usual less than 4^k possibilities, as we shall see. We plan on showing that we can rebuild the tessellation while specifying the parent of few enough lines to satisfy bound (2).

We shall denote by U_0 the set of (orphan) lines without an explicit parent.

Informally, we first initialise the algorithm by moving along the time axis, while prolongating the segments that are deemed alive. Namely we prolongate a segment when it is born, if we know the time, and prolongate it when it has a known child otherwise. When two segments cross, we stop each one if its virtual number of murders is zero. Both may be stopped at the same time, and at least one must be. If a segment is not stopped, its virtual number of murders is decreased by one. When a segment hits the border, it is killed. End of initialisation.

Now we loop. Each loop consists of one pass backwards in time, and a cutting phase.

The pass backwards in time is given in Algorithm 3.During the pass backwards in time, we extend the segments whose parent we do not know. Since these segments are extended before their first child (requirement 2) in the real tessellation T, they never cross each other in the process. When such a backwards segment hits another segment, we decrease that segment's number of other children.

The cutting phase is given in Algorithm 4. During the cutting phase, we cut the segments whose number of other children n is negative. More precisely, their rightmost point is now their $(-n)$-th rightmost crossing with an orphan segment. Their number of other children is reset to zero. The consequence of this operation is that $(-n)$ orphan segments will be further extended in the next loop.

End of loop. Stop when all numbers of other children are zero. End of algorithm.

Let us give some intuition. The algorithm hinges on a few facts.

After initialisation, P is a pretessellation.

Most importantly, the following property is true at the end of each loop:

Late Events Property. *Deaths occur after the time they occur in the real tessellation. Same for births.*

In formula, if we denote x_b^P and x_d^P the times of birth and death at the end of a loop, then for all lines $l \in L$:

$$x_b^P(l) \geq x_b^T(l), \tag{7}$$
$$x_d^P(l) \geq x_d^T(l). \tag{8}$$

Finally, within a loop, birth and death times are always non-increasing. That is, if we write x_b^i and x_d^i for times of birth and death after i-th change, then

$$x_b^i(l) \geq x_b^{i+1}(l),$$
$$x_d^i(l) \geq x_d^{i+1}(l)$$

for all lines $l \in L$.

We may now state how the number of virtual murders and other children will be defined as input. A first consequence of these choices will be that the algorithm actually runs.

For other children, we merely use the number in the real tessellation, that is the number of orphan segments born from the segment. In formula:

$$O^T(l) = \#\left\{m \in U_0 : x_b^T(m) = \mathfrak{c}(l,m)\right\}. \tag{9}$$

The definition also holds if l is the border $\mathcal{B}$. Here again, we have the crude bound of k other children, and 4^k ways to split them.

For virtual murders, we define them by running the initialisation phase without them. When two segments cross, we check on the real tessellation

Algorithm 2 Rebuild from final labelling scheme

Input: The lines L, the ordered and reverse-ordered list of events $\mathcal{E}^o$ and $\mathcal{E}^r$, a subset $U_0 \in L$ of orphan lines satisfying requirement 2, a prototessellation $P = (x_b, x_d)$ such that $x_b(l) = x_b^T(l)$ for all non-orphan line $l \in L \backslash U_0$, a function "virtual murders" $V : L \cup \mathcal{B} \to \mathbb{N}$ defined in (13) , and a function $O^T : L \cup \mathcal{B} \in \mathbb{N}$ giving the number of orphan children a line has (9).

```
 1: $U \leftarrow U_0$
 2: for all $l \in L$ do
 3:     $x_d(l) \leftarrow 1$                          ▷ Death time set to maximum for now
 4: end for
 5: for all $l \in U$ do
 6:     $x_b(l) \leftarrow 1$                   ▷ Birth of orphans set to maximum for now
 7: end for                                          ▷ End of "preinitialisation"
 8: for all $\mathfrak{e} \in \mathcal{E}^o$ do
 9:     $l_1, l_2 \leftarrow l(\mathfrak{e})$
10:     if $[x_b(l_1) = \mathfrak{e}] \wedge [x_b(l_2) > \mathfrak{e}]$ then          ▷ Is $l_1$ the first child of $l_2$?
11:         $x_b(l_2) \leftarrow \mathfrak{e}$                 ▷ Temporary maximum birth time for $l_2$
12:     else if $[x_b(l_2) = \mathfrak{e}] \wedge [x_b(l_1) > \mathfrak{e}]$ then
13:         $x_b(l_1) \leftarrow \mathfrak{e}$
14:     else if $x_b(l_1), x_b(l_2) \leq \mathfrak{e} \leq x_d(l_1), x_d(l_2)$ then   ▷ Do the lines cross?
15:         if $V(l_1) = 0$ then                         ▷ Is $l_1$ "virtual-killed"?
16:             $x_d(l_1) \leftarrow \mathfrak{e}$                 ▷ Death time to new maximum
17:             if $V(l_2) = 0$ then                     ▷ Is $l_2$ "virtual-killed"?
18:                 $x_d(l_2) \leftarrow \mathfrak{e}$             ▷ Death time to new maximum
19:             else
20:                 $V(l_2) \leftarrow V(l_2) - 1$       ▷ Count that $l_2$ "virtual-killed" $l_1$
21:             end if
22:         else                                 ▷ In that case $l_2$ is "virtual-killed"
23:             $V(l_1) \leftarrow V(l_1) - 1$           ▷ Count that $l_1$ "virtual-killed" $l_2$
24:             $x_d(l_2) \leftarrow \mathfrak{e}$                 ▷ Death time to new maximum
25:         end if
26:     end if
27: end for                                            ▷ End of initialisation
28: repeat
29:     $P \leftarrow Parent\_seek(L, P, \mathcal{E}^o, U)$                ▷ Extend backwards
30:     Cuts, $U, P \leftarrow Cutting(L, P, \mathcal{E}^r, U_0, O^T)$     ▷ Cut too long segments
31: until Cuts $= 0$
        return $P$
```

Algorithm 3 Parent-seeking loop

Input: The lines L, a pretessellation $P = (x_b, x_d)$, the ordered sequence of events $\mathcal{E}^o$, a subset $U \in L$ of lines whose parent is not currently known.

```
1: for all $\mathfrak{e} \in \mathcal{E}^r$ do                              ▷ Reverse timewise
2:     $l_1, l_2 \leftarrow l(\mathfrak{e})$
3:     if $[l_1 \in U] \wedge [x_b(l_1) > \mathfrak{e}] \wedge [x_b(l_2) < \mathfrak{e} < x_d(l_2)]$ then
4:         $x_b(l_1) \leftarrow \mathfrak{e}$                                ▷ Extend $l_1$ backwards
5:         $U \leftarrow U - l_1$                           ▷ $l_1$ seen as born on $l_2$, for now
6:     end if
7:     if $[l_2 \in U] \wedge [x_b(l_2) > \mathfrak{e}] \wedge [x_b(l_1) < \mathfrak{e} < x_d(l_1)]$ then
8:         $x_b(l_2) \leftarrow \mathfrak{e}$                     ▷ Same as above, $l_1$ and $l_2$ switched
9:         $U \leftarrow U - l_2$
10:    end if
11: end for
        return $P$
```

whether a segment needs to be prolongated, and increase its virtual murders counter accordingly. We use the number of virtual murders we get in the end. An immediate consequence is that there are at most k virtual murders, so that there are less than 4^k ways to split them.

Rather than copying the initialisation phase of Algorithm 2 and replacing the statements on V, let's give another characterisation of these virtual murders. The pretessellation $P(U_0)$ after initialisation has the following properties (we shall not prove it but merely use it for definition):

$$x_b^{P(U_0)}(l) = x_b^T(l) \qquad \text{if } l \notin U_0. \tag{10}$$

$$x_b^{P(U_0)}(l) = \inf\left\{\mathfrak{e}(l,m) : \mathfrak{e}(l,m) = x_b^T(m)\right\} \qquad \text{if } l \in U_0. \tag{11}$$

$$x_d^{P(U_0)}(l) = \inf\left\{\mathfrak{e}(l,m) : x_d^{P(U_0)}(m) \geq \mathfrak{e}(l,m) \geq x_d^T(l), x_b^{P(U_0)}(m)\right\} \quad \text{for all } l \in L. \tag{12}$$

Notice that the times of death are well-defined. Then the number of virtual murders is simply the number of kills in $P(U_0)$, given that simultaneous deaths do not count:

$$V(l) = \#\left\{m \in L : e(l,m) = x_d^{P(U_0)}(m) < x_d^{P(U_0)}(l)\right\}. \tag{13}$$

Algorithm 4 Cutting loop

Input: The lines L, a pretessellation $P = (x_b, x_d)$, the reverse-ordered sequence of events $\mathcal{E}^r$, a subset $U_0 \in L$ of orphan lines, a function $O^T : L \cup \mathcal{B} \to \mathbb{N}$ giving the number of orphan children a line has, and a variable set $U \subset L$ initially empty.

```
 1: Cuts ← 0                                                 ▷ Reset number of cuts
 2: for all $l \in L$ do
 3:     $O(l) \leftarrow 0$                                  ▷ Reset number of other children
 4: end for
 5: for all $\mathfrak{e} \in \mathcal{E}^o$ do                ▷ Timewise
 6:     $l_1, l_2 \leftarrow l(\mathfrak{e})$
 7:     if $[l_1 \in U_0] \wedge [x_b(l_1) = \mathfrak{e}]$ then
 8:         $O(l_2) \leftarrow O(l_2) + 1$
 9:         if $O(l_2) = O^T(l_2) + 1$ then
10:             $x_d(l_2) \leftarrow \mathfrak{e}$           ▷ Death time to new maximum
11:             Cuts ← 1
12:         end if
13:         if $O(l_2) \geq O^T(l_2)$ then                 ▷ $l_2$ has too many other children
14:             $l_1 \in U$                                ▷ We do not know the father of $l_2$
15:         end if
16:     else if $[l_2 \in U_0] \wedge [x_b(l_2) = \mathfrak{e}]$ then   ▷ Same, switching $l_1$ and $l_2$
17:         $O(l_1) \leftarrow O(l_1) + 1$
18:         if $O(l_1) = O^T(l_1) + 1$ then
19:             $x_d(l_1) \leftarrow \mathfrak{e}$
20:             Cuts ← 1
21:         end if
22:         if $O(l_1) \geq O^T(l_1)$ then
23:             $l_2 \in U$
24:         end if
25:     end if
26: end for
        return Cuts, $U, P$
```

Conventionally, we may set $V(\mathcal{B})$ to infinity: the border kills everything that hits it, and it will be easier to keep track in the algorithm.

Lemma 3.2. *With input satisfying the requirements given at the beginning of the section, the Algorithm 2 ends.*

Its output is a pretessellation satisfying the Late Events Property . Moreover, each line has the same number of children as in the real tessellation T.

Proof in appendix.

The lemma states that the algorithm ends, but not that we have the real tessellation. I confess that I do not know whether we may have pathological situations where there are several pretessellations with these numbers of other children and whose segments have the right endpoints to yield the same initialisation. However, we may circumvent the difficulty by carefully choosing the set U_0 of lines whose parents we specify at input.

For a given set U_0, we may write P or $P(U_0)$ for the output pretessellation, $S(U_0)$ the associated segments and $s(U_0, l)$ the segment on line l. We also call $D(U_0) \in L$ the set of lines such that $s(U_0, l) \neq s(l)$. We may also consider the sets $D_b(U_0)$ and $D_d(U_0)$ of lines with different birth and death times, that is $x_b^P(l) \neq x_b^T(l)$ (*resp.* $x_d^P(l) \neq x_d^T(l)$). Those two sets may intersect. Obviously $D_b(U_0) \cup D_d(U_0) = D(U_0)$ and $D_b(U_0) \subset U_0$.

Now, if we choose the right line and give its birth time, then there will be at least two less lines in D_b: that one and another. Namely we consider a line l_1 whose birth time was wrong, and look at its wrong parent l_2. The line l_3 whose birth time we correct is the one that kills this parent l_2 in the real tessellation. Since the first line l_1 had the wrong parent, its fake birth was later than this kill, so that the parent l_2 now lacks a child. Hence there is another line l_4 who will be born on l_2 at the end of the algorithm. Since the death time of l_2 is now right, it will really be the parent of l_4. Formally:

Lemma 3.3. *Let U_0, P and $D(U_0), D_b(U_0), D_d(U_0)$ defined as above.*

Then there is a line l such that

$$\#D_b(U_0 \backslash \{l\}) \leq \#D_b(U_0) - 2 \tag{14}$$

Proof in appendix.

An important remark is that if all the times of birth are right, then P is the real tessellation T. Indeed, since the times of birth are right and P satisfies the Late Events Property , the segments of P contain those of T. Since P is a pretessellation, segments do not cross, hence the times of death cannot be later than in the real tessellation T.

This remark will finally ensure that we may choose a labelling scheme with enough orphan lines to prove existence of our measures. We start by choosing as orphan lines the inner nodes of the tree of birth of odd or even generations, whichever the biggest. We then use the former lemma to obtain a working U_0. We get:

Theorem 3.4. *For any set of k lines, for any $\epsilon > 0$ the number of T-tessellations built on them is at most:*

$$\mathcal{N}(k) \leq C^k \left(\frac{k}{(\ln k)^{1-\epsilon}} \right)^{k-k/(\ln k)}, \tag{15}$$

where C depends only on ϵ.

Proof in appendix.

3.3 The T-tessellation model is well-defined

Our combinatorial result implies existence of the Gibbs measure (**??**) in such cases.

Theorem 3.5. *Let $H(T)$ be an energy on T such that $H(T) \geq -C\#L(T)$, for some real C and any tessellation T. Then for any intensity τ, the Gibbs measure with density*

$$\mathrm{d}\gamma_W(T) = \exp(-H(T))\mathrm{d}\lambda_W^\tau(L(T))$$

is well-defined and finite.

Proof. We have to prove that the measure is finite. We denote by c any absolute constant. Using the bound (1), Stirling formula and Theorem 3.4, we get:

$$\begin{aligned}
\int_{\mathcal{T}} \exp(-H(T))\mathrm{d}\lambda_W^\tau(L(T)) &\leq \sum_{k=0}^{\infty} \exp(-Mk)\exp(-\tau)\frac{\tau^k}{k!}\mathcal{N}(k) \\
&\leq \sum_{k=0}^{\infty} \frac{c^k}{k^k}\left(\frac{k}{(\ln k)^{1-\epsilon}}\right)^{k-k/(\ln k)} \\
&\leq \sum_{k=0}^{\infty} \left(\frac{c}{\sqrt{\ln k}}\right)^k \\
&< \infty.
\end{aligned}$$

□

4 Optimality remarks and perspectives

Though we have used very violent upper bounds at times, there is no way to get a substantially better combinatorial result. Indeed let us consider for some integer $a \leq k$ the following k lines on a square domain $[0,1]^2$:

$$\begin{aligned} y &= \frac{\lambda}{a+1} && \text{for } \lambda \in [1,a] \\ x &= \frac{\lambda}{k-a+1} && \text{for } \lambda \in [1,k-a]. \end{aligned}$$

How many different T-tessellations can we build on those lines? A lower bound is given by supposing that all horizontal segments are maximal, that is go from border to border. Then each of the vertical segments is between two consecutive horizontal lines, and hence of length $1/(k-a+1)$. More significantly, this means each one can be at $(k-a+1)$ different places, independently from each other since the vertical lines do not cross. So that there are at least $(k-a+1)^a$ different T-tessellations that can be built on those lines. If we take $a = k - k/(\ln k)$, we may conclude:

Lemma 4.1. *There are sets of k lines such that the number of T-tessellations on those lines admits the following lower bound:*

$$\mathcal{N}(k) \geq \left(\frac{k}{\ln k}\right)^{k-k/(\ln k)}$$

If we want to get a better result and a tighter upper bound on the partition function, we then need to have a closer look on the usual topologies of the lines. That is an order of magnitude harder, but might be worth the effort. Indeed the previous worst-case example hinges heavily on having many lines crossing many segments, and topologically equivalent sets of lines have very low measure, looking like $(k2^k k!)/(2k)!$ of the space of all sets of k lines.

By contrast, using very sloppy heuristics, we would expect that for most sets of k lines, the number of T-tessellations on those lines behaves like

$$\mathcal{N} = \sqrt{k}^k.$$

The idea is the following: let us take a segment away of the true tessellation. How many different segments may we put on the line to get a tessellation again? Neglecting problems of children and murders, this would be the number of segments that the line cross, plus one. Now the probability of crossing a segment is essentially the length of this segment. So the number of crossed segments would be km, where m is the mean length of a segment. Now that length is the interval between two successive segments a line cross, that is $1/(km)$. So that m should be of order $1/\sqrt{k}$, and for each new line, we have $k/\sqrt{k} = \sqrt{k}$ as many possibilities.

Thus it seems likely that the method in this paper gives little information on the measure, except its very existence.

References

K. Adamczyk and K. Kiêu. Modèle de tessellation pour les paysages. *To be published.*

T. Arak, P. Clifford, and D. Surgailis. Point-based polygonal models for random graphs. *Advances in Applied Probability*, 25:348–372, 1993.

S. M. Mackisack and R. E. Miles. Homogeneous rectangular tessellations. *Advances in Applied Probability*, (28):993–1013, 1996.

S. M. Mackisack and R. E. Miles. A large class of random tessellations with classical poisson polygon distributions. *Forma*, (17):1–17, 2002.

A Proofs for the algorithms

A.1 Proof of Lemma 3.1

At the end of initialisation, we have the following properties:

- Birth times are those of the tessellation for all lines l: $x_b^P(l) = x_b^T(l)$.
- Death times are overestimated for all lines l: $x_d^P(l) \geq x_d^T(l)$.
- The number of murders $M(l)$ for each line is that of the true tessellation.

Indeed the first and third points are merely the input, and the death times are set to an upper bound at stage 1.

What is important is that those properties will remain true throughout the loop that completes the algorithm, and this will yield by recurrence that at the end of the time $\mathfrak{e}$ loop:

- The remaining number of murders for each line $M(l)$ is that of the true tessellation $M^T(l, \mathfrak{e}) = \# \left\{ m \in L : x_d^T(m) = \mathfrak{e}(m, l) \text{ and } \mathfrak{e}(m, l) > \mathfrak{e} \right\}$.

- Death times before $\mathfrak{e}$ are right, that is: $(x_d^T(l) \leq \mathfrak{e}) \Rightarrow (x_d^P(l) = x_d^T(l))$

We have to prove that if this is true before the $\mathfrak{e}(l_1, l_2)$ loop, it will be true after it.

Now, we enter the loop if and only if there is a death in the real tessellation. Indeed, in that case

$$x_b^P(l_1) = x_b^T(l_1) < \mathfrak{e}(l_1, l_2) \leq x_d^T(l_1) \leq x_d^P(l_1),$$

and the same for l_2. If on the contrary there is no death, since segments do not cross (**??**), either $x_b^T(l) \geq \mathfrak{e}(l_1, l_2)$ for one of the two lines, and then this also holds for $x_b^P(l) = x_d^T(l)$, or one of the two lines l is already dead $x_d^T(l) < \mathfrak{e}(l_1, l_2)$. Then $x_d^P(l) < x_d^T(l)$ by recurrence hypothesis.

If we do not enter the loop, there are no changes to M or x_d^P. On the other hand, there is no change to M^T, nor any new line whose death time is required to be right, so the conditions still hold.

If we do enter the loop, then either $x_d^T(l_1) = \mathfrak{e}(l_1, l_2)$, or $x_d^T(l_2) = \mathfrak{e}(l_1, l_2)$. In the first case, using the recurrence hypothesis, $M(l_1) = M^T(l_1, \mathfrak{e}(l_1, l_2)) = 0$, and $x_d^P(l_1)$ is set to $x_d^T(l_1)$, satisfying the second condition. The first condition is also still satisfied, since the only number of murders that changes for the real tessellation is that of l_2, which decreases by one, since l_1 is no more in the set of *remaining* murders. Symmetrically, if $x_d^T(l_2) = \mathfrak{e}(l_1, l_2)$, then $M(l_1) = M^T(l_1, \mathfrak{e}(l_1, l_2)) > 0$ since it contains l_2, and this number of remaining murders is decreased by one while $x_d^P(l_2)$ is set to $x_d^T(l_2)$. So that the recurrence hypothesis is transmitted.

Since death times before $\mathfrak{e}$ are right, after we hit the last event, all death times are right, that is $x_d^P(l) = x_d^T(l)$ for all lines l. Hence the output pretessellation is the real tessellation.

A.2 Proof of Lemma 3.2

The proof is built on the fact that throughout the algorithm, after preinitialisation, three conditions are fulfilled by every line l of the algorithm. The two conditions (7) and (8) of the Late Events Property , and the fact that if we are sure we do not know yet the parent of a line, its birth time is strictly later than its birth time in the real tessellation. In equation:

$$(l \in U) \Rightarrow x_b^P(l) > x_b^T(l). \tag{16}$$

We must then check “the three conditions” still hold each time we change a birth or death time or the set U.

First, let us check "the three conditions" hold after preinitialisation.

Indeed on the one hand at stage (2) we set the death times of all lines to the maximum possible, that is the rightmost point of the domain. So that $x_d^P(l) \geq x_d^T(l)$.

On the other hand, we know the birth times of the lines l not in U_0. For those $x_b^P(l) = x_b^T(l)$. For the lines $l \in U_0$, whose parent we do not know, we set their birth time to more than maximum possible at stage (5). So that $x_b^P(l) > x_b^T(l)$. Since U is initialised as U_0, condition (16) is fulfilled.

The "three conditions" still hold at the end of the whole initialisation. We shall even prove that the prototessellation at the end of initialisation is exactly the pretessellation described as $P(U_0)$ in equations (10), (11) and (12).

Indeed on the one hand the birth times are changed only at stages (10) and (12).

Now these statements are not reached for the lines $l \notin U_0$, whose birth times we know: if $\mathfrak{e}(l,m) = x_b^P(m)$, then either l is really the parent of m and $x_b^T(m) = \mathfrak{e}(l,m) > x_b^T(l) = x_b^P(l)$, and we do not reach the change, or $x_b^P(m)$ has been changed at stage (10) or (12), at which time it was set to $x_b^P(l) = x_b^T(l)$. Contradiction again. Hence times of birth of lines not in U_0 are as described in equation (10) at the end of initialisation.

For lines not in U_0, at stages (10) and (12), we set the birth time of a line $l \in U_0$ to $\mathfrak{e}(l,m)$ for some $m \in L$. This happens once since afterwards the condition $x_b(l) > \mathfrak{e}$ will never be satisfied. This does happen when m is a child of l in P. Since the first child of l is specified, according to requirement 2, this is the event to which the birth time is set, and equation (11) is satisfied at the end of the initialisation.

Since moreover m is the child of l, we know that $x_b^T(l) < \mathfrak{e}(l,m) = x_b^P(l)$. In addition notice that the birth point is always included in the real tessellation segment $s^T(l)$:

$$x_b^P(l) \leq x_d^T(l). \tag{17}$$

This condition also holds for the lines not in U_0 since then $x_b^P(l) = x_b^T(l) < x_d^T(l)$. It will hold throughout what's left of the algorithm, since birth times are never increased. In the latter case, the segment is then even included in the segment of the real tessellation T.

On the other hand, death times are changed during initialisation only at stages (15), (17) and (23). We want to prove that at the end of initialisation, the death time is that mentioned when describing V, that is $x_d^{P(U_0)}$ at equation

(12). Since death and birth times can only be changed to $\mathfrak{e}$, the recurrence hypothesis implies that the **if** condition $x_b(l_1), x_b(l_2) \leq \mathfrak{e} \leq x_d(l_1), x_d(l_2)$ is equivalent to it being true at the end of initialisation. Under that hypothesis, we enter the **if** statements with $l \in l(\mathfrak{e})$ if and only if $x_d^{P(U_0)}(m) = \mathfrak{e}$ or $x_d^{P(U_0)}(l) = \mathfrak{e}$. This happens exactly $V(l) + 1$ times, with the definition (13). Each time save the last, $V(l)$ is decreased by one, so that it really is 0 at the last event, and $x_d^P(l)$ is set to $x_d^{P(U_0)}(l)$, transmitting the recurrence hypothesis and satisfying equation (12).

Hence at the end of initialisation, the prototessellation P is really the pretessellation $P(U_0)$.

We now have to deal with the main loop, separated in functions *Parent_seek* and *Cutting*. **??** Not only do we have to check the "three conditions", but we shall also make use of the fact that P stays a pretessellation from now on: segments do not cross (3).

Within *Parent_seek*, the changes occur at stages (4) and (5), and symmetrically (8) and (9). Since any line whose birth time we change is excluded from U, condition (16) still holds. There are no change in death times, so that condition (8) still holds.

If (7) still holds, P will stay a pretessellation. Indeed, the extended parts are then between times of birth and times of first child. These are included in the real tessellation T, so they cannot cross. A would-be crossing would then be with a segment $s(m)$ that has not changed during the loop. But then the birth time would have been changed at $\mathfrak{e}(l, m)$. Since a line has to be in U to have its birth time changed, and is excluded from U if it is changed, its birth time can change only once through one call of *Parent_seek*, finishing to show that segments still do not cross (3).

We then have to ensure that birth times are still late (7). Since a birth time can change only once during one call and the events are tested reverse timewise, it is enough to prove that if $l \in U$ at event $x_b^T(l)$, then $x_b^P(l)$ is set to $x_b^T(l)$. This is true, since the parent m of a line l in $U \subset U_0$ is in $L \backslash U_0$, thanks to requirement 2. So that this event is in the segment of the pretessellation: $x_b^P(m) = x_b^T(m) < \mathfrak{e}(l, m) < x_d^T(m) \leq x_d^P(m)$. We thus enter the **if** (3) or its symmetrical (7).

The latter point also proves that when entering the function *Cutting*, all lines are born on a segment (4).

Within *Cutting*, the changes happen at stages (17), (18), (23) and the symmetrical (17), (18), (23). No birth time is changed so condition (7) remains valid. To understand what is going on, let us consider an event

$\mathfrak{e}(l,m)$ where $l \in U_0$, $m \in L$, and $x_b^P(l) = \mathfrak{e}(l,m)$ at input. Then since the birth time is included in the real segment (4), either l is really a child of m, that is $x_b^T(l) = \mathfrak{e}(l,m)$, or since moreover segments in T do not cross (3), the segment m dies before: $x_d^T(m) \leq \mathfrak{e}(l,m)$. Now we change death time at stage (10) (or (17)). The counter O is the number of U-children of m at input that are born at $\mathfrak{e}(l,m)$ at the latest. So that when we change its death time (10), there are $O^T(m)$ such children strictly before $\mathfrak{e}(l,m)$. Since it cannot have more children, we obtain $x_d^T(m) \leq \mathfrak{e}(l,m)$ and the condition on death times (8) is still fulfilled. Besides, when we add a line l to U, at stages (14) and (23), we still have $x_b^P(l) < x_b^T(l)$, fulfilling condition (16): indeed this happens at points $\mathfrak{e}(l,m)$ where $O(m) > O^T(m)$, so that $x_d^T(m) \leq \mathfrak{e}(l,m)$ thanks to the former remark, and m cannot be the parent of l.

Notice that $O(\mathcal{B}) > O^T(\mathcal{B})$ is impossible thanks to the late births (7).

Since segments are only shortened during *Cutting*, they will not cross and P stays a pretessellation.

Finally notice that Cuts is set to one, at stages (11) or (18), if and only if at least one line's death time gets earlier.

This remark implies that the algorithm ends: indeed death times can only decrease, and they take values in a finite set, so there will be an iteration when they do not change, there are no cuts. The algorithm ends at this point.

What can we say about the final pretessellation returned by the algorithm?

To begin with, it is really a pretessellation that fulfills the Late Events Property . We have followed the conditions throughout the algorithm.

Moreover, each line has the same number of children as in the true tessellation. Indeed the algorithm ends only when no cuts are made in the *Cutting* function, that is when all lines have at most as many children as in the true tessellation. The other side of the inequality comes from the fact that the sum of other children in that function is bounded from below by the sum of other children in the true tessellation: each line is born on another line or on the border, and since birth times are overestimated, they cannot mistakenly be thought to be born on the border.

A.3 Proof of Lemma 3.3

A first remark is that for any subset U_1 of U_0, the Late Events Property holds for $P(U_0)$ with respect to $P(U_1)$. Indeed there would be no change when running Algorithm 2 with U_0 as input if $P(U_1)$ was the real tessellation. So that $x_b^{P(U_0)}(l) \geq x_b^{P(U_1)}(l) \geq x_b^T(l)$ for all lines. This implies that $D_b(U_1)$ is a subset of $D_b(U_0)$.

Another remark, already made in the proof of the algorithm, is that when $x_b^P(l) = \mathfrak{e}(l,m)$, either m is the real parent of l, or $x_d^T(m) \leq \mathfrak{e}(l,m)$. This implies that if $x_d^T(m) = x_d^P(m)$, all its children in P are real children. Since moreover it has the same number of children as in T, its children are exactly the right children.

We now consider $l_1 \in D_b(U_0)$ and its fake parent l_2, so that $x_b^P(l_1) = l_2$. Then $\mathfrak{e}(l_1,l_2) \geq x_d^T(l_2)$. Moreover $\mathfrak{e}(l_2,l_3) = x_d^T(l_2)$ and $l_3 \in D_b(U_0)$. Indeed $x_b(l_3) \geq \mathfrak{e}(l_2,l_3)$. We take l_3 as the line l in the lemma. Since $l_3 \notin U_0 \setminus \{l_3\}$, we know that $l_3 \notin D_b(U_0 \setminus \{l_3\})$.

Now l_3 is no longer a child of l_2. But the number of children of l_2 at the end of the algorithm is fixed, equal to that in the real tessellation T. So that there is a line l_4 that is a child of l_2 in $P((U_0 \setminus \{l_3\})$ and was not in $P(U_0)$. Since moreover l_2 is killed by the right line in $P(U_0 \setminus \{l_3\})$, we know that $l_4 \in D_b(U_0)$ and $l_4 \notin D_b(U_0 \setminus \{l_3\})$.

Finally, since $D_b(U_0 \setminus \{l_3\}) \in D_b(U_0)$, we may write:

$$\# D_b(U_0 \setminus \{l\}) \leq \# D_b(U_0) - \# \{l_3, l_4\} = \# D_b(U_0) - 2.$$

A.4 Proof of Theorem 3.4

We may rebuild any tessellation T on the lines L with Algorithm 2 and the input given in the "requirements" mentioned at the start section 2. However we have to make it more specific, namely to describe U_0 the set of orphan lines.

This set is obtained by looking at the tree of births. On the one hand, we must specify the parent of each leaf, since we cannot give their first child. We add to this the interior nodes either of even, or of odd generations, whichever the smaller. Thus the lines U_i without specified parents are at most $u_i = (k-z)/2$, where z is the number of leaves. This is a valid labelling as the lines whose parents are not specified have all their children marked as such, since the parity of their generation is different. Hence requirement 2 for the input is satisfied.

Now we run the algorithm. We may not find the true tessellation, and have a set of lines $D_b(U_i)$ whose parent is wrong. This set is included in U_i. We then remove a line from U_0 as in Lemma 3.3 and run the algorithm again. And we iterate until we obtain the true tessellation. Since $D_b(U_0)$ is at least two elements smaller at each steps, we have to remove at most $u/2$ lines from our initial U_i to get a set U_0 we may use as input in Algorithm 2 to obtain the true tessellation. So that its cardinal is at least $u = (k-z)/4$.

Let's now go over each element of the input in order.

First labelling each line as "leaf" or "not leaf". As already mentioned, this yields at most 2^k possibilities.

Second labelling each line not in U_0 with its parent. Since a leaf cannot be a parent, there are at most $(k+1-z)^{k-u} \leq (k+1-z)^{(3k+z)/4}$ possibilities.

Third labelling each line with its number of virtual murders yield at most 4^k possibilities, as already mentioned.

Fourth labelling each line with its number of other children yield at most 4^k possibilities, as already mentioned.

Thus we may give the following upper bound on the number of different T-tessellations on k given lines, using C for any absolute constant:

$$\begin{aligned}\mathcal{N}(k) &\leq C^k \sup_{1\leq z\leq k} (k+1-z)^{(3k+z)/4} \\ &\leq C^k \left(\frac{k}{(\ln k)^{1-\epsilon}}\right)^{k-k/(\ln k)},\end{aligned}$$

where we have used the following bound on the supremum in the right-hand side: take the derivative in z of the logarithm, and we see that the maximum is attained when

$$(1+k-z)(1+\ln(1+k-z)) = 4k-1.$$

For big k, this implies $k/\ln(k)^{1-\epsilon} \geq 1+k-z \geq 4k/\ln(k)$. We then replace by the right bounds in the exponent and the basis.