Laboratoire Bordelais de Recherche en Informatique
UMR 5800 - Université Bordeaux I, 351, cours de la Libération,
33405 Talence CEDEX, France

Research Report `RR-1295-03`

Canonical Decomposition of Outerplanar Maps and Application to Enumeration, Coding and Generation

Nicolas Bonichon, Cyril Gavoille, and Nicolas Hanusse

March, 2003

Canonical Decomposition of Outerplanar Maps and Application to Enumeration, Coding and Generation

Nicolas Bonichon* Cyril Gavoille* Nicolas Hanusse*

Abstract

In this article we define a canonical decomposition of rooted outerplanar maps into a spanning tree and a list of edges. This decomposition, constructible in linear time, implies the existence of bijection between rooted outerplanar maps with n nodes and bicolored rooted ordered trees with n nodes where all the nodes of the last branch are colored white. As a consequence, for rooted outerplanar maps of n nodes, we derive:

- an enumeration formula, and an asymptotic of $2^{3n-\Theta(\log n)}$;
- an optimal data structure of asymptotically $3n$ bits, built in $O(n)$ time, supporting adjacency and degree queries in worst-case constant time;
- an $O(n)$ expected time uniform random generating algorithm.

Keywords: well-orderly tree, outerplanar graphs, maps, one-page graphs, compact encoding, uniform random generation.

*Laboratoire Bordelais de Recherche en Informatique, Université Bordeaux I. {*bonichon, gavoille, hanusse*}*@labri.fr*

1 Introduction

A graph is *outerplanar* if it can be drawn on the plane with non-intersecting edges such that all the nodes lie on the boundary of the infinite face, also called *outerface*. Characterization of outerplanar graphs has been given by Chartrand and Harary [CH67]: a graph is outerplanar if and only if it has neither $K_{2,3}$ nor K_4 as a minor. A linear time recognition algorithm has been given by Mitchell [Mit79]. Labeled and unlabeled outerplanar graphs can be randomly generated in $O(n^4 \log n)$ space and $O(n^2)$ time [BK03] after a preprocessing of $O(n^5)$ time. Among graph properties, outerplanar graphs contain trees, have tree-width at most two, and are exactly the graphs of *pagenumber* one [Bil92]. Recall that a graph G has pagenumber k if k is the smaller integer for which G has a k*-page* embedding, also called *book* embedding. In such an embedding the nodes are drawn on a straight line (the spine of a book), and the edges are partitioned into k *pages*, each page consisting of non-intersecting edges. An outerplanar graph with a 1-page embedding is depicted on Fig. 1c).

A *planar map* is a connected graph drawn on the sphere with non-intersecting edges (see [CM92] for a survey). A planar map is *outerplanar* if all the nodes lie on one face, called the outerface. For convenience, outerplanar maps are drawn on the plane such that the outerface corresponds to the infinite face, see Fig. 1b). A map is *rooted* if one of its edges, the root, is distinguished and oriented. In this case, the map is drawn on the plane in such a way that whenever traveling clockwise around the boundary of the outerface, the tail of the root edge is traversed before its head. A *planted tree* is the rooted planar map of a rooted tree such that the tail of the root of the map coincides with the root of the tree. In the literature, planted trees are also named ordered rooted trees. All the maps considered in this paper are planar, rooted, and are simple (have no loops and multi-edges).

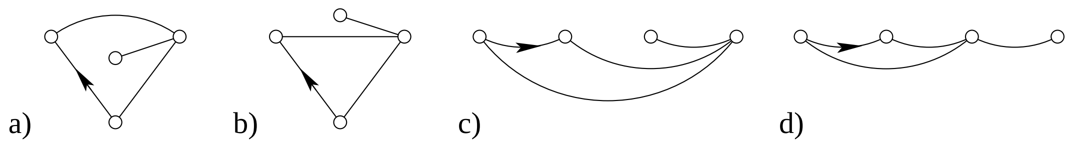

Figure 1: Four drawings for an outerplanar graph G: a) a planar (but not outerplanar) map; b) an outerplanar map; and c) a 1-page embedding. As graphs, a), b), c) and d) are identical, but as maps, only b), c) and d) are identical. As 1-page embedding, we point out that c) and d) are different.

Some sub-classes of outerplanar maps are well-known. Planted trees and maximal outerplanar maps, i.e., the map of an outerplanar graph with the maximum number of edges, are counted by the Catalan numbers. Finally, biconnected outerplanar maps can be seen as dissections of a convex polygon, and their number can be counted by Schröder numbers. For these three sub-classes there exist linear time random generation algorithms [ARS97, BdLP99, ES94].

Besides the combinatorial aspect and random generation, a lot of attention is given in Computer Science to *efficiently* represent discrete objects. By "efficiently", we mean that the representation should be succinct, i.e., the storage of these objects requires few bits, and that the time to compute such representation should be polynomial in its size. Fast manipulation of the so encoded objects and easy access to a part of the code are also desirable properties. Typically, adjacency query, i.e., check if two nodes are neighbors or not, and degree query, i.e., how many neighbors a node has, should be given very fast.

For instance, a folklore encoding of n-edge planted trees, based on a clockwise depth-first traversal, yields a representation with $2n - O(1)$ bits. This coding length is asymptotically optimal since the number of possible n-edge planted trees is the nth Catalan number $\frac{1}{n+1}\binom{2n}{n} \sim 2^{2n-O(\log n)}$. Completing this coding by an efficient data structures of length $o(n)$ bits, it has been shown in [MR01, CLL01] that adjacency and degree queries can then be answered in constant time, assuming that: 1) nodes of the tree are labeled according to the depth-first traversal (i.e., the node i must be the ith node encountered in the clockwise prefix order of the tree); and 2) standard arithmetic operations on integers of $\Omega(\log n)$ bits can be performed in constant time.

Outerplanar graphs are an interesting class of graphs because they are isomorphic to graphs of pagenumber one, cf. Fig. 1. Our contribution is an optimal $3n$-bit encoding for outerplanar maps. We point out that there exist many 1-page embeddings for a graph of pagenumber one. From the asymptotic formula of Flajolet and Noy [FN99], any encoding of 1-page embeddings requires $3.37n$ bits[1].

Let us sketch our technique. First we show that an outerplanar map admits a canonical decomposition into a particular rooted spanning tree (called *well-orderly tree* and defined in Section 2), and a set of additional edges (u, v) such that v is the first node after u (in a clockwise preorder of the tree) that is not a descendant of v. This decomposition can be computed in linear time. Then, we give three applications to this decomposition: enumeration formula (Section 3), efficient encoding (Section 4), and random generation algorithm with experiments (Section 5).

Hereafter we denote by $T_{n,d}$ the number of n-node planted trees where the depth of the clockwise last leaf is d (formulas for these numbers are given in [Fel68, Kre70]). Our canonical decomposition gives a bijection between outerplanar maps and bicolored trees (more precisely planted trees in which the nodes are colored either white or black), and where all the nodes (including the root) of the clockwise last branch are colored white. Clearly these last objects are counted by the following numbers:

$$\sum_{d=1}^{n-1} 2^{n-d-1} T_{n,d} \ .$$

From this bijection, we show that the number of n-node outerplanar maps is asymptotically

$$\frac{2^{3n}}{36n\sqrt{\pi n}} \ .$$

Moreover, the average number of edges of an outerplanar map is $\frac{3}{2}n - O(1)$.

An information-theoretic optimal encoding algorithm is deduced from the previous decomposition. It takes a $O(n)$ time and uses at most $3n - 6$ bits, for $n \geqslant 2$, as follows: $2n - 4$ bits are used to encode the spanning tree, and $n - 2$ bits (at most) are used to encode the additional edges. Adding a standard $o(n)$-bit data structure to this coding [MR01, CLL01], adjacency and degree queries can be then answered in worst-case constant time.

Using a grammar to produce bicolored rooted ordered trees with n nodes where all the nodes of the last branch are colored white, and using Goldwurm's algorithm [Gol95], a random

[1]In their article, 1-page embeddings are called non-crossing graphs.

outerplanar map can be generated uniformly with $O(n)$ space and $O(n^2)$ average time. Using Floating-Point Arithmetic [DZ99], this average time complexity can be reduced to $O(n^{1+\epsilon})$. In Section 5, we propose a $O(n)$ expected time and $O(n)$ space complexity generating algorithm. It can generate outerplanar maps with a given number of nodes, or with a given number of nodes and of edges.

2 The Well-Orderly Tree of an Outerplanar Map

In [BGH03] the authors introduced the *well-orderly trees*, a special case of the *orderly spanning trees* [CLL01]. Let T be a rooted spanning tree of a planar map H. Two nodes are *unrelated* if neither of them is an ancestor of the other in T. An edge of H is unrelated if its endpoints are unrelated. Let $v_1, v_2 \ldots, v_n$ be the clockwise preordering of the nodes in T. A node v_i is *well-orderly* in H w.r.t. T if the incident edges of v_i in H form the following blocks (possibly empty) in clockwise order around v_i:

- $B_P(v_i)$: the edge incident to the parent of v_i;
- $B_<(v_i)$: unrelated edges incident to nodes v_j with $j < i$;
- $B_C(v_i)$: edges incident to the children of v_i;
- $B_>(v_i)$: unrelated edges incident to nodes v_j with $j > i$; and
- the clockwise first edge $(v_i, v_j) \in B_>(v_i)$, if it exists, verifies that v_i is a descendant of the parent of v_j;

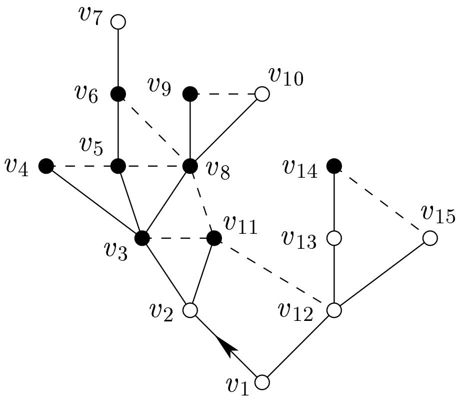

Figure 2: A rooted outerplanar map, and its well-orderly tree depicted with solid edges.

T is a *well-orderly tree* of H if all the nodes of T are well-orderly in H, and if the root of T belongs to the boundary of the outerface of H (see Fig. 2 for an example). Note that a well-orderly tree is necessarily a spanning tree. Observe also that for every edge e of H, with $e = (v_i, v_j)$ and $i < j$, we have either $e \in B_C(v_i)$ (i.e., $e \in T$), or $e \in B_>(v_i)$. For convenience, the clockwise first edge of $B_>(v_i)$, if it exists, is called the *front edge* of v_i.

All planar maps do not admit a well-orderly tree. For instance the map of Fig. 1a) has no well-orderly tree rooted at the tail of the root edge, whereas the map of Fig. 1b) has one. Actually, we have:

Lemma 1 ([BGH03]) *Every rooted planar map admits at most one well-orderly tree rooted at the tail of the root edge of the map.*

We will show that in fact every outerplanar map admits a well-orderly tree. It can be computed by the following recursive algorithm $\mathsf{Traversal}$. H is the rooted outerplanar map, and r is the tail of its root edge. $\mathsf{Traversal}(H, \varnothing, r)$ returns the well-orderly tree T of H rooted at r, the second parameter is the current set of edges of the tree.

Algorithm 1 $\mathsf{Traversal}(H, T, u)$.

```
C ← {(u, v) ∈ H | v ∉ T}
T ← T ∪ C
for all edges (u, v) ∈ C taken in the clockwise order around u do
  T ← Traversal(H, T, v)
end for
return T
```

Theorem 1 *Every rooted outerplanar map H has a well-orderly tree T, computable in linear time, rooted at the tail of the root edge of H. Moreover, for every node u, if $(u, v) \in B_>(u)$, then v is the next unrelated node with u in the clockwise preordering of T. In particular, $|B_>(u)| \leqslant 1$ for every u.*

Proof. Let T be the set of edges returned by $\mathsf{Traversal}(H, \varnothing, r)$ (cf. Algorithm 1), where r is the tail of the root edge of H. Let us denote by T_i and by v_i respectively the second and the third parameters of the ith call of $\mathsf{Traversal}$. We have $T_1 = \varnothing$ and $v_1 = r$. Note that there are exactly n calls to $\mathsf{Traversal}$, where n is the number of nodes of H. By induction, T_i is a tree with i nodes, for every $i \in \{1, \ldots, n\}$. Therefore $T_n = T$ is a spanning tree of H. An important observation is that the clockwise preordering of the nodes of T is precisely $v_1, \ldots, v_n$, and that T_i is a subtree of T_{i+1}, for $i \in \{1, \ldots, n-1\}$.

This algorithm can be easily implemented to run in $O(n)$ time. Let us show that the tree T satisfies the properties of Theorem 1. This is done thanks to the following properties.

Unrelated edges. Every edge of H is either in T, or an unrelated edge. Indeed, let (v_i, v_j) be any edge with $i < j$. Clearly, v_j is not an ancestor of v_i. When v_i is being treated: 1) if $v_j \in T_i$, then v_j is not a descendant of v_i, and thus (v_i, v_j) is an unrelated edge; and 2) if $v_j \notin T_i$, then v_j becomes a child of v_i in T_{i+1}, thus an edge of T_n since T_i is a subtree of T_{i+1}.

Blocks. The incident edges of v_i form clockwise around v_i the four blocks $B_P(v_i)$, $B_<(v_i)$, $B_C(v_i)$, and $B_>(v_i)$. Since T is a spanning tree of a planar map, all the edges of $B_<(v_i)$ are (clockwise) after $B_P(v_i)$ and before the edges of $B_>(v_i)$. Let us show that the edges of $B_C(v_i)$ are after the edges of $B_<(v_i)$ and before the edges of $B_>(v_i)$.

Let us consider the ith call of $\mathsf{Traversal}$. Let (v_i, v_t) be the last edge (in clockwise order around v_i) toward a node that is before v_i in T_i. Let v_k be the nearest common ancestor of v_i and v_t. The path from v_k to v_i, the edge (v_i, v_t) and the path from v_t to v_k defines a region of the plane, R, distinct from the outerface. Since H is outerplanar there is no node inside the region R. So all the neighbors of v_i that are not in T_i follow the edge (v_i, v_t) in the clockwise order around v_i. Hence, the edges of $B_C(v_i)$ are after the edges of $B_<(v_i)$.

The same reasoning can be done to show that the edges of $B_C(v_i)$ are before the edges of $B_>(v_i)$.

Descendant of the parent. For every $(v_i, v_j) \in B_>(v_i)$, v_i is a descendant of the parent of v_j. Assume $(v_i, v_j) \in B_>(v_i)$. In particular $(v_i, v_j) \notin T$. When v_i is being treated, the node v_j must be in T_i, otherwise v_j becomes a child of v_i in T_{i+1}, and thus in T. By construction, the only nodes v_j's of T_i that are after the node v_i in the prefix clockwise preordering of T_i are such that their parent is an ancestor of v_i.

At this point, we have proved that every node in T is well-orderly, and thus T is a well-orderly tree of H.

Next unrelated node. Let $(v_i, v_j) \in B_>(v_i)$. Let v_k be the first unrelated node with v_i clockwise after v_i in T. Assume that $v_j \neq v_k$, so $i < k < j$. Let v_t be the parent of v_j in T. The cycle composed of the path in T from v_t to v_i, and of the edges (v_i, v_j) and (v_j, v_t) defines a region of the plane, R, distinct from the outerface. Because $i < k < j$, v_k must belong to R or to its boundary. As v_j is the only node of the boundary of R that is unrelated with v_i, v_k must belong to R: a contradiction with the fact that H is an outerplanar map. It follows that $v_j = v_k$. Therefore, we have showed that $(v_i, v_j) \in B_>(v_i)$ implies that v_j is the next unrelated node with v_i in T.

This completes the proof of Theorem 1. □

Combining Lemma 1 and Theorem 1, it is clear that an outerplanar map H has an unique decomposition into a well-orderly tree T, and a set of edges of the kind $(u, v) \in B_>(u)$, where v is the next unrelated node after u in T. (Recall that an edge of H belongs either to T or to a $B_>$ block). Conversely, given T, and a piece of information on whether $B_>(u)$ is empty or not for each node u, one can uniquely determine the corresponding rooted outerplanar map. The coding of the cardinality of each $B_>$ block can be done by coloring the nodes of T as follows: if $|B_>(u)| = 0$, u is colored white, and if $|B_>(u)| = 1$, u is colored black. Observing that for every node u of the clockwise last branch of T, $|B_>(u)| = 0$, we obtain:

Corollary 1 *There is a bijection, computable in linear time, between the n-node bicolored rooted trees where all the nodes of the last branch (including the root) are colored white, and the n-node rooted outerplanar maps.*

Recall that a graph (or a map) is k-connected if G has more than k nodes and if, for every subset X of fewer than k nodes, $G \setminus X$ is connected [Die00]. *biconnected* is a synonym for 2-connected.

Theorem 2 *There is a bijection, computable in linear time, between the $(n-1)$-node bicolored rooted trees with a white root, all leaves colored in black, and the set of n-node rooted biconnected outerplanar maps.*

Proof. To show the theorem, let show that a outerplanar H map rooted at v is biconnected if and only if (1) for each leaf u, different from the last leaf, of the well-orderly tree of H rooted at v, $|B_>(u)| = 1$ and (2) the depth of the last leaf of T is 1.

Let H be an outerplanar map and T its well-orderly tree. Suppose that u is a leaf of the tree T such that $|B_>(u)| = 0$. All the neighbors of u belong to the subtree rooted at $P(u)$ and are before u in the clockwise prefix order of T. Let u' be a node of the subtree rooted at

$P(u)$. Let (u'', u') the front edge of u''. By definition the node u' is the first unrelated node of u'' after u'' in the clockwise prefix order of T. So, either u' is equal to $P(u)$ either u'' is in the subtree rooted at $P(u)$. Let consider the graph H' obtained from H when removing the node $P(v)$. If the node $P(v)$ is not the root of T then H is not biconnected. If the node $P(v)$ is the root of T and u is not on the last branch then H is not biconnected.

Assume now that for all leaves u (different from the last leaf) of T $|B_>(u)| = 1$ and that the depth of the last leaf is 1. Let u_1 and u_2 two nodes of H. W.o.l.g. suppose that u_1 is before u_2 in the prefix clockwise order of T. To show that H is biconnected, let us construct a cycle, node disjoint containing u_1 and u_2. The first part C_1 of this cycle contains the edge (l, v) where l is the last leaf of T, and the path of v to u_1 in T. The second part C_2 of the cycle is constructed recursively as follow. Let the variable t initializes with u_1. If u_2 doesn't belong to the subtree of T rooted at t, then we add the path from t to the last leaf l of the subtree rooted at t plus the edge $(l, t') \in B_>(l)$ and t receives now the node t'. If u_2 belongs to the subtree of T rooted at t, then we add the path from t to the last leaf l of the subtree of T rooted at u_2 plus the edge $(l, t') \in B_>(l))$ and t receives now the node t'.

The construction ends when the node t is equal to the last leaf of the tree T. Clearly the cycle $C = C_1 \cup C_2$ is node-disjoint and contains the nodes u_1 and u_2. So there exists two disjoint paths between u_1 and u_2. This property is true for each pair u_1 and u_2 so the outerplanar map H is biconnected. □

Observe that if the well-orderly tree of an outerplanar map H has its clockwise last leaf of depth d, then from Corollary 1 H has no more than $(n-1) + n - (d+1) = 2n - 2 - d$ edges. In particular, the depth of the last leaf of the well-orderly tree of any maximal outerplanar map (i.e., having $2n-3$ edges) must be $d = 1$. As the well-orderly tree is unique, this yields another bijective proof of the well known following result:

Corollary 2 *There is a bijection, computable in linear time, between maximal n-node outerplanar maps and planted trees with $n-1$ nodes.*

3 Enumeration of Outerplanar Maps

Let $T_{n,d}$ be the number of rooted n-node plane trees whose clockwise last leaf is of depth d. These numbers are called the ballot numbers (or Delannoy numbers), and we have [Fel68, Kre70]:

$$T_{n,d} = \frac{d}{2n-2-d}\binom{2n-2-d}{n-1-d}, \quad \text{for all } n > d > 0.$$

So, there are exactly $2^{n-1-d}T_{n,d}$ bicolored trees whose all the nodes of the last branch (including the root of the tree) are colored white. Let M_n be the number of n-node outerplanar maps. The sequence M_n, for $n = 1, 2, \ldots$, is: $1, 1, 3, 13, 67, 381, 2307, 14589, \ldots$ (see Fig. 3 for all the outerplanar maps of 4 nodes).

Theorem 3 *For all $n \geqslant 2$, $M_n = \sum_{d=1}^{n-1} 2^{n-d-1} T_{n,d}$, and $M_n \sim \dfrac{2^{3n}}{36n\sqrt{\pi n}}$.*

Proof. The enumeration formula for M_n follows directly from Corollary 1, and of the above formula for $T_{n,d}$. Let us compute the asymptotic value of M_n. Developing the factorials in

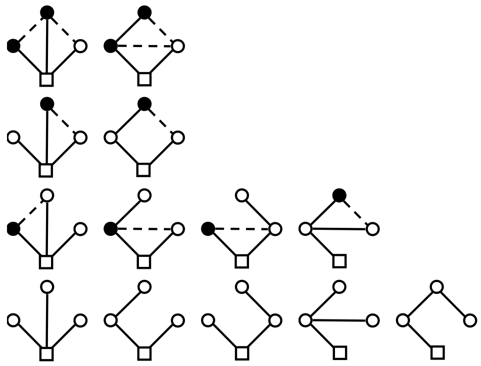

Figure 3: The 13 rooted outerplanar maps with 4 nodes.

binomial terms of $T_{n,d}$, we have:

$$\frac{T_{n,d}}{T_{n,1}} \;=\; d\prod_{j=2}^{d}\frac{n-j}{2n-2-j}$$

Therefore,

$$\frac{M_n}{2^{n-2}T_{n,1}} \;=\; \sum_{d=1}^{n-1}\frac{d}{2^{d-1}}\prod_{j=2}^{d}\frac{n-j}{2n-2-j} \;=\; \sum_{d=1}^{n-1}u_{n,d},\ \text{with } u_{n,d}=\frac{d}{2^{d-1}}\prod_{j=2}^{d}\frac{n-j}{2n-2-j}\ .$$

For every n, let us set $v_d(n)=u_{n,d}$ if $d<n$, and $v_d(n)=0$ if $d\geqslant n$. As $(n-j)/(2n-2-j)\leqslant 1/2$, $|v_d(n)|\leqslant d/2^{d-1}(1/2)^{d-1}=d/2^{2d-2}$, for all d and n. Let $w_d=d/2^{2d-2}$. We have:

$$\sum_{d=1}^{\infty}w_d=\lim_{n\to\infty}\sum_{d=1}^{n}w_d=\lim_{n\to\infty}\left(\frac{16}{9}-\frac{4}{3}\left(\frac{1}{4}\right)^n\left(n+\frac{4}{3}\right)\right)=\frac{16}{9}\ .$$

Because the series w_d is convergent, the series $v_d(n)$ is normally convergent. As $\lim_{n\to\infty}v_d(n)=w_d$, we have:

$$\lim_{n\to\infty}\sum_{d=1}^{n-1}u_{n,d}=\lim_{n\to\infty}\sum_{d=1}^{\infty}v_d(n)=\sum_{d=1}^{\infty}\lim_{n\to\infty}v_d(n)=\sum_{d=1}^{\infty}w_d=\frac{16}{9}\ .$$

We have:

$$T_{n,1} \;=\; \frac{1}{2n-3}\binom{2n-3}{n-2} \;=\; \frac{1}{n}\binom{2(n-2)}{n-2} \;\sim\; \frac{2^{2n}}{16n\sqrt{\pi n}}\ .$$

The result follows. □

We now rely Narayana numbers with biconnected outerplanar maps. The Narayana numbers [Nar59] count rooted plane trees with n nodes and ℓ leaves:

$$Y_{n,\ell} \;=\; \frac{1}{n}\binom{n}{\ell}\binom{n-2}{n-\ell-2}\ , \quad \text{for all } n>\ell>0.$$

Biconnected outerplanar graphs can be seen as dissections of a convex polygon, and their number can be counted by Schröder numbers. From Theorem 2, we have:

Corollary 3 *The number of rooted biconnected outerplanar maps with $n \geqslant 3$ nodes is* $\sum_{\ell=2}^{n-2} 2^{n-1-\ell} Y_{n-1,\ell}$.

Theorem 4 *The number $M_{n,m}$ of rooted outerplanar maps with $n \geqslant 3$ nodes and $m \geqslant n-1$ edges is:*

$$M_{n,m} = \sum_{d=1}^{2n-2-m} \binom{n-d-1}{m-n+1} T_{n,d} .$$

Proof. Outerplanar maps with n nodes and m edges are in bijection with bicolored trees with white nodes on the last branch and $m-n+1$ black nodes (see Corollary 1). On each tree of $T_{n,d}$, there are $\binom{n-d-1}{m-n+1}$ possible ways to color $m-n+1$ nodes that are not on the last branch in black. □

Let $\mathcal{M}_n$ denote the set of n-node outerplanar maps, counted by M_n, and let $\mathcal{M}_{n,m}$ denote the n-node m-edge outerplanar maps, counted by $M_{n,m}$. Thanks to Corollary 1, for each $H \in \mathcal{M}_n$, we can uniquely associate a number d that is the depth of the last leaf of the bicolored tree T in bijection with H. Let $\bar{d}$ be the average value of d in $\mathcal{M}_n$.

Lemma 2 *For every n, $\bar{d} < 3$.*

Proof. Let us prove that

$$\bar{d} = \frac{\sum_{d=1}^{n-1} d 2^{n-1-d} T_{n,d}}{\sum_{d=1}^{n-1} 2^{n-1-d} T_{n,d}} \leqslant \frac{\sum_{d=1}^{n-1} d T_{n,d}}{\sum_{d=1}^{n-1} T_{n,d}} \tag{1}$$

Take the positive functions f and g. For $i, j \in \mathbb{N}$ and f monotonic decreasing, we show that the following expression is negative:

$$\frac{\sum_i i f(i) g(i)}{\sum_i f(i) g(i)} - \frac{\sum_i i g(i)}{\sum_i g(i)} = \frac{\left(\sum_i i f(i) g(i)\right)\left(\sum_j g(j)\right) - \left(\sum_i i g(i)\right)\left(\sum_j f(j) g(j)\right)}{\left(\sum_i f(i) g(i)\right)\left(\sum_j g(j)\right)}$$

Consider only the numerator N of the previous function. N can be rewritten as:

$$\begin{array}{rc} & \left(\sum_i i f(i) g(i)\right)\left(\sum_j g(j)\right) - \left(\sum_i i g(i)\right)\left(\sum_j f(j) g(j)\right) \\ = & \sum_i \sum_j i f(i) g(i) g(j) - i g(i) f(j) g(j) \\ = & \sum_i \left(\sum_{j<i} i g(i) g(j)(f(i) - f(j)) + \sum_{j>i} i g(i) g(j)(f(i) - f(j))\right) \\ = & \sum_i \left(\sum_{j<i} i g(i) g(j)(f(i) - f(j)) + \sum_{j<i} j g(j) g(i)(f(j) - f(i))\right) \\ = & \sum_i \sum_{j<i} (i-j) g(i) g(j)(f(i) - f(j)) \end{array}$$

If f is a monotonic decreasing function, the previous expression is negative. Since $f(d) = 2^{n-1-d}$ is monotonic decreasing, Ineq. (1) is proved.

Observe that the right term of Ineq. (1) is the average depth $\bar{d}_t$ of the last leaf of an n-node planted tree. Let $\mathcal{T}_n$ denote the set of n-node planted trees. Let us show that $\bar{d}_t \leqslant 3\frac{n-1}{n+1}$.

Let us consider the sequence of bijections of the following diagram:

$$\mathcal{T}_n \leftrightarrow \mathcal{D}_{2n-2} \overset{\sim}{\leftrightarrow} \mathcal{D}_{2n-2} \leftrightarrow \mathcal{T}_n \leftrightarrow \mathcal{G}_{n+1}$$

where 1) $\mathcal{D}_{2n}$ is the set of Dyck words[2] of length $2n$, 2) $\mathcal{G}_{n+1}$ is the set of rooted triangulations of a $(n+1)$-gon.

Almost all these bijections are well-known. Let us remind some facts and a new bijection between $\mathcal{T}_n$ and $\mathcal{G}_{n+1}$. Consider the bijections from the left to the right. In the usual edge encoding $w \in \mathcal{D}_{2n-2}$ of a ordered rooted tree $t \in \mathcal{T}_n$, the *clockwise prefix coding*[3], the length of the last branch of t corresponds to the number of consecutive closed parentheses at the end of w. The operation $\sim$ consists in reading w from the right to the left and substituing each open (resp. closed) parenthese by a closed (resp. open) one. One can easily check that the result $\tilde{w}$ is also a dyck word and the distribution of the number of consecutive open parentheses in the beginning of Dyck words is equal to the distribution of closed parentheses at the end of Dyck words.

Take another encoding of $t \in \mathcal{T}_n$, the *degree encoding*: start with the empty Dyck word, perform a prefix traversal of the $n-1$ first nodes of t appending for each node having i children the pattern $\underbrace{\texttt{((}\ldots\texttt{(}}_{i}\texttt{)}$. (The last node of t being a leaf, we do not need to code $\texttt{)}$ for it.) The final word is of length $2n-2$. The number of consecutive open parentheses in the beginning of $\tilde{w}$ represents now the degree of the root of t.

By Corollary 2, there is a bijection between $\mathcal{G}_{n+1}$ and $\mathcal{T}_n$. Let us remind some details of the bijection. If we apply Algorithm Traversal to a rooted triangulation of a $(n+1)$-gon g, the result of the algorithm is of course a bicolored tree t' of $n+1$ nodes. Through this new bijection, the degree of the root of g and of t' are the same. Removing the last branch of length 1 and forgetting the color of the nodes of t', we obtain a tree $t \in \mathcal{T}_n$. The degree of the root of t is equal to the degree of the root of t' minus one.

Through these bijections, the distribution of the degree of the root in $\mathcal{G}_{n+1}$ is in bijection with the distribution of the length of the last branch in $\mathcal{T}_n$ minus one.

Let $\bar{d}_g$ be the average degree of the root in $\mathcal{G}_{n+1}$. $\bar{d}_t = \bar{d}_g - 1 = 2\frac{2n-1}{n+1} - 1 = 3\frac{n-1}{n+1}$. $\square$

Let $\overline{m_n}$ be the average number of edges of the maps belonging to $\mathcal{M}_n$. From Theorems 3, 4, and Lemma 2, we can deduce the following result:

Corollary 4 *For every n, $\frac{3}{2}n - 3 < \overline{m_n} \leqslant \frac{3}{2}n - 2$.*

Proof. Let $\overline{m_{n,d}}$ be the average number of edges for given d.

$$\overline{m_{n,d}} = \frac{\sum_{m=n-1}^{2n-d-2} m \binom{n-d-1}{m-n+1} T_{n,d}}{2^{n-d-1} T_{n,d}}$$

[2] Dyck words or words of balanced parentheses are the set of words belonging to $\{\texttt{(},\texttt{)}\}^*$ with the same amount of open and closed parentheses and so that at least half of the letters of any prefix are a occurence of open parentheses.

[3] Each edge encountered for the first time is coded by an open parenthesis and otherwise by a closed one.

Taking $k = m - n + 1$ and using the well-known binomial identities $\sum_{d=0}^{n} \binom{n}{d} = 2^n$ and $\sum_{d=0}^{n} d\binom{n}{d} = n2^{n-1}$, we have

$$\begin{aligned}
\overline{m_{n,d}} &= \frac{\sum_{k=0}^{n-d-1}(n+k-1)\binom{n-d-1}{k}}{2^{n-d-1}} \\
&= \frac{(n-1)\sum_{k=0}^{n-d-1}\binom{n-d-1}{k} + \sum_{k=0}^{n-d-1} k\binom{n-d-1}{k}}{2^{n-d-1}} \\
&= \frac{(n-1)2^{n-d-1} + (n-d-1)2^{n-d-2}}{2^{n-d-1}} \\
&= \frac{3n-d-3}{2}
\end{aligned}$$

By definition,

$$\begin{aligned}
\overline{m_n} &= \frac{\sum_{d=1}^{n-1} \overline{m_{n,d}} M_{n,d}}{M_n} \\
&= \frac{\sum_{d=1}^{n-1}(3n-d-3)M_{n,d}}{2M_n} \\
&= \frac{(3n-3)\sum_{d=1}^{n-1} M_{n,d} - \sum_{d=1}^{n-1} dM_{n,d}}{2M_n} \\
&= \frac{3n-3}{2} - \frac{\bar{d}}{2}
\end{aligned}$$

From Lemma 2, $1 \leqslant \bar{d} < 3$ and $\frac{3n}{2} - 2 \geqslant \overline{m_n} > \frac{3n}{2} - 3$. □

4 Coding Supporting Adjacency and Degree Queries

From Corollary 1, every outerplanar map with n nodes can be represented by a bicolored tree with n nodes. A standard coding for n-node planted trees uses $2n - 4$ bits if $n \geqslant 2$, and the colors can be stored using $n - 2$ bits at most, observing that the color associated to the last branch (containing the last leaf and the root) is known (white). For the tree encoding, we use a clockwise depth-first traversal of the tree, each edge being traversed twice starting from the root. During the traversal, we output "1" if the edge is traversed for the first time, and "0" otherwise. This leads to a $2(n-1)$-bit encoding. Actually, if the tree has at least one edge ($n \geqslant 2$) two bits can be saved observing that the previous coding always starts we "1" and ends with "0".

This leads to a $3n - 6$ bits encoding. By Lemma 3, the length of this coding is asymptotically optimal, up to an adding factor of $O(\log n)$ bits. It follows:

Theorem 5 *Every n-node rooted outerplanar map or every outerplanar graph, $n \geqslant 2$, can be coded (and decoded) in $O(n)$ time and using a representation on at most $3n - 6$ bits.*

In the following, we show how to extend this coding with an extra $o(n)$ bits (still constructible in linear time) so that the data structure supports adjacency and degree queries in worst-case constant time. For that we present below efficient well-known data structures for binary strings and balanced string of parentheses.

4.1 Constant time queries in strings of parentheses

Let S be a string of symbols defined over a given alphabet. We denote by $S[i]$ the ith symbol of S, $i \geqslant 1$. Let $\mathsf{select}(S, i, \star)$ be the position of the ith $\star$ in S. Let $\mathsf{rank}(S, j, \star)$ be the number of symbols $\star$ before or at the jth position of S. That is if $\mathsf{select}(S, i, \star) = j$ then $\mathsf{rank}(S, j, \star) = i$.

Now, let S be a string of open and closed parentheses, i.e., $\star \in \{\texttt{(}, \texttt{)}\}$. Two parentheses $S[i] = \texttt{(}$ and $S[j] = \texttt{)}$ match if $i < j$ and the difference of the number of open and closed parentheses between them is null. The string S is *balanced* if for each parenthesis, there is a matching parenthesis. Let $\mathsf{match}(S, i)$ be the position in S of the matching parenthesis of $S[i]$. $S[k]$ is *enclosed* by $S[i]$ and $S[j]$ if $i < k < j$. Let $\mathsf{enclose}(S, i_1, i_2)$ be the closest matching parenthesis pair (j_1, j_2) that encloses $S[i_1]$ and $S[i_2]$. Finally, let $\mathsf{wrapped}(S, j)$ denote twice the number of pairs of matching parentheses (i_1, i_2) such that $\mathsf{enclose}(S, i_1, i_2) = (j, \mathsf{match}(S, j))$.

The following results are valid in the word-RAM model, in which standard arithmetic operations on binary words of length $\Omega(\log n)$ can be done in constant time.

Lemma 3 ([MR01, CLL01]) *For every balanced string S of parentheses (or a binary string) of length $O(n)$, the operations* select*,* rank*,* match*,* enclose*, and* wrapped *can be done in worst-case constant time using an auxiliary table of $o(|S|)$ bits and $O(|S|)$ preprocessing time.*

Actually, Lemma 3 holds for the operations select and rank, even if S is not balanced.

4.2 Coding of outerplanar maps

We can associate to any planted tree T a balanced string of parentheses (or Dyck word) as follows: during a clockwise depth-first traversal of T starting at the root, whenever an edge is traversed for the first time, output an open parenthesis and otherwise output an closed parenthesis. For convenience, an open parenthesis (resp. a closed parenthesis) is added at the beginning (resp. at the end) of the output string. One can think this latter transformation as an extra edge entering in the root of T. If T has n nodes, the final string of parentheses contains $2n$ symbols as each of T (plus the extra edge) is traversed twice. The final string is called the *clockwise prefix coding* of T.

Consider T an n-node planted tree, and its clockwise prefix coding S_T. Let $v_1, \ldots, v_n$ be the clockwise preordering of the nodes of T. To perform efficiently queries on T, we label the node v_i by its index i, so that an adjacency query between the nodes v_i and v_j is simply the pair $\{i, j\}$. By construction of the clockwise prefix coding, the ith open parenthesis corresponds to the edge of T entering in v_i. And, the matching parenthesis of the ith open parenthesis corresponds to the edge of T leaving v_i. In other words, each node v_i can be seen as a pair (i, j) of matching parentheses, i.e., such that $S_T[i]$ matches with $S_T[j]$ in S_T. See Example 1.

Lemma 3 and the clockwise prefix coding leads to two useful operations for the tree that can be done in constant time: adjacency and degree.

Two nodes are adjacent in T if one of them is the parent of the other one. One can determine the parent of a node v_i finding the position p of the ith open parenthesis using $p = \mathsf{select}(S_T, i, ()$, and the position of the closest enclosing pair (j_1, j_2) of $(p, \mathsf{match}(S_T, p))$ with $(j_1, j_2) = \mathsf{enclose}(p, \mathsf{match}(S_T, p))$. Then, the parent of v_i is v_j where $j = \mathsf{rank}(S_T, j_1, ()$.

As for the degree, one can check that the number of children of v_i is $\frac{1}{2}\mathsf{wrapped}(S_T, \mathsf{select}(S_T, i, ()))$. We just have to add 1 if v_i is not the root, i.e., if $i \neq 1$.

From the previous explanations, one can claim the following lemma:

Lemma 4 ([CLL01]) *Let T be an n-node planted tree and its clockwise prefix coding S_T. The parent, degree and adjacency of a node can be done in constant time using an auxiliary table of $o(|S_T|)$ bits and a $O(n)$ preprocessing time.*

We encode an outerplanar map H with two binary strings:

1. S_T of length $2n$: the clockwise prefix coding of the well-orderly tree T of H;

2. S_B of length n: a binary string such that $S_B[j] = |B_>(v_i)|$, where v_i is the node corresponding to the jth closed parenthesis of S_T.

Given j, i can be obtained by $i = \mathsf{rank}(S_T, \mathsf{match}(S_T, \mathsf{select}(S_T, j,))), ()$. Observe that if $j > n-2$, then v_i belongs to the last branch of T, and thus $|B_>(v_i)| = 0$. Thus, S_B can reduce to the first $n-2$ entries. To perform $|B_>(v_i)|$ given i, one can compute $j = \mathsf{rank}(S_T, \mathsf{match}(S_T, \mathsf{select}(S_T, i, ())),))$, and return $S_B[j]$.

The sequence of nodes $v_{i_1}, v_{i_2}, \ldots, v_{i_j}$ is a *branch* of T if v_{i_j} is a leaf, and if for every $t \in \{1, \ldots, j-1\}$, $v_{i_{t+1}}$ is the clockwise last child of v_{i_t}. The last branch consists therefore of all the nodes of the path between the root and the clockwise last leaf of T. The branches partition the nodes of T and there is exactly one branch per leaf. One can check that a branch of T corresponds to a maximal block of closed parentheses in S_T.

Example 1 *The outerplanar map of Fig. 2 can be encoded by the following two strings: S_T = `((()((()))(()()))())((())()))` and S_B = 101110111010000. The sequence of nodes v_3, v_8, v_{10} is a branch. To know if node v_3 has a front edge, do the following sequence of operations: $p = \mathsf{select}(S_T, 3, () = 3$, $\mathsf{match}(S_T, p) = 18$, $\mathsf{rank}(S_T, 18,)) = 8$. Since $S_B[8] = 1$, node v_3 has a front edge.*

Lemma 5 *Let $v_{i_1}, \ldots, v_{i_j}$ be a branch of T. Then, $|B_<(v_{i_j+1})| = \sum_{t=1}^{j} |B_>(v_{i_t})|$. Moreover, $|B_<(v_{i_j+1})|$ can be computed in constant time.*

Proof. Clearly, the next unrelated node with any node v_i is the node v_{r+1} such that v_r is the leaf of the branch of v_i. In particular, the next unrelated node of every node v_{i_t}, $t \in \{1, \ldots, j\}$, is v_{i_j+1}. Let $k = i_j + 1$. Let v_i be any unrelated node with v_k and with $i < i_1$. The next unrelated node of v_i is a node v_{r+1} where v_r is the leaf of the branch of v_i. So we have $r < i_1$, that implies $v_k \neq v_{r+1}$. It follows that the only node whose the next unrelated node is v_k are the nodes $v_{i_1}, \ldots, v_{i_j}$. It follows that $|B_<(v_{i_j+1})| = \sum_{t=1}^{j} |B_>(v_{i_t})|$.

To compute $|B_<(v_{i_j+1})|$, we need to find in S_T the positions corresponding to the beginning and the end of the branch. By definition, a branch corresponds to a maximal sequence of closed parentheses in S_T. The position p of v_k in S_T is $p = \mathsf{select}(S_T, k, ()$. The last closed parenthesis of branch b corresponding to v_{i_1} is at position $p-1$ in S_T. To find the position p' of the closed parenthesis of v_{i_j}, we compute $p' = \mathsf{rank}(S_T, k-1, ()$. The cardinality of $B_<(v_k)$ is the number of ones in S_B between and including the positions $p_1 = \mathsf{rank}(S_T, p',))$ and $p_2 = \mathsf{rank}(S_T, p-1,))$. Thus, $|B_<(v_k)| = \mathsf{rank}(S_B, p_2, 1) - \mathsf{rank}(S_B, p_1, 1)$. □

Algorithm 2 *Adjacency*(S_T, S_B, i, j), assuming $i < j$

```
if i is the parent of j in T then
  return true
else
  i' ← match(S_T, select(S_T, i, ())
  r ← rank(S_T, i', ()
  if r ≠ j − 1 then
    return false
  else
    return S_B[r]
  end if
end if
```

Theorem 6 *Every rooted outerplanar map with n nodes admits a $3n + o(n)$-bit encoding, built in linear time, such that adjacency and degree queries can be computed in worst-case constant time.*

Proof. We proposed a $3n$-bit encoding for a rooted outerplanar map. Adding auxiliary tables of size $o(n)$, Lemmas 3 and 4 provide us constant time for computing the parent and the degree of v_i in T.

Adjacency: it remains to check the adjacency for the edges of H that does not belong to T. Nodes v_i and v_j, with $i < j$, are adjacent in H if v_j is the next unrelated node after v_i (that is $j = r - 1$ where v_r is the leaf of the branch of v_i) and if $|B_>(v_i)| = 1$. See Algorithm 2 for details.

Degree: the degree of node v_i is the sum of the degree in the tree T (see Lemma 4), the cardinality of $B_<(v_i)$ (see Lemma 5) and of $B_>(v_i)$ has a front edge. □

Starting from a connected outerplanar graph, we can compute a rooted outerplanar map using the algorithm presented in [CNAO85]. So the previous results on outerplanar maps can also be applied to outerplanar graphs.

5 Uniform Random Generation

To randomly generate an outerplanar map, one can randomly generate a bicolored tree where the nodes of the last branch are colored white. Thanks to Corollary 1, one can then construct in linear time the corresponding random outerplanar map.

Theorem 7 *A rooted outerplanar map with n nodes or with n nodes and m edges can be generated uniformly at random in $O(n)$ expected time.*

Proof. Let $\mathcal{B}_{n,b}$ be the set of all bicolored rooted trees with n nodes such that:

1. the root is colored white;
2. the clockwise last leaf is colored white; and
3. there are exactly b nodes colored black.

Let $\mathcal{B}_n = \bigcup_{b=0}^{n-2} \mathcal{B}_{n,b}$. From the bijection between outerplanar maps and some bicolored trees (cf. Corollary 1), the outerplanar maps with n nodes are in bijection with a subset, say $\tilde{\mathcal{M}}_n$, of bicolored trees. Clearly, $\tilde{\mathcal{M}}_n \subset \mathcal{B}_n$. Similarly, the outerplanar maps with n nodes and m edges are in bijection with a subset, say $\tilde{\mathcal{M}}_{n,m}$, of bicolored trees. The trees of $\tilde{\mathcal{M}}_{n,m}$ have exactly $m-(n-1)$ black nodes, so $\tilde{\mathcal{M}}_{n,m} \subset \mathcal{B}_{n,m-n+1}$.

The algorithm we proposed is an accept-reject algorithm: it consists in repeating a uniformly generation of an element of $\mathcal{B}_n$ (or of $\mathcal{B}_{n,m-n+1}$) until we get an element of $\tilde{\mathcal{M}}_n$ (or of $\tilde{\mathcal{M}}_{n,m}$). This clearly provides a uniform random generation on the set $\tilde{\mathcal{M}}_n$ (or on $\tilde{\mathcal{M}}_{n,m}$), and thus on the corresponding outerplanar maps.

A rooted tree can be generated in linear time using for example the Arnold and Sleep algorithm [AS80]. The colors of the $n-2$ nodes are colored black with probability $1/2$ (recall that the root and the last leaf are forced to be colored white by definition of $\mathcal{B}_n$). This provides an $O(n)$ time algorithm to generate an element of $\mathcal{B}_n$.

To generate an element of $\mathcal{B}_{n,m-n+1}$, we have to select exactly $b = m-n+1$ black nodes among $n-2$ (the root and the last leaf are forced to be colored white). Selecting b elements among $n-2$ can be done in $O(n)$ time [Wil77] (the procedure uses $O(n)$ arithmetic operations on $O(\log n)$ bit integers). This provides an $O(n)$ time algorithm to generate an element of $\mathcal{B}_{n,m-n+1}$.

Testing whether $T \in \mathcal{B}_n$ belongs to $\tilde{\mathcal{M}}_n$ (or whether $T \in \mathcal{B}_{n,m-n+1}$ belongs to $\tilde{\mathcal{M}}_{n,m}$) clearly takes $O(n)$ time: it suffices to test whether all the inner nodes of the last branch are white or not.

As the bijection between bicolored trees and maps is linear, it remains to show that the average number of rejects in the above procedure is bounded by a constant.

Observe that every tree $T \in \mathcal{B}_n$ whose the last leaf is of depth 1 belongs to $\tilde{\mathcal{M}}_n$ and to $\tilde{\mathcal{M}}_{n,m-n+1}$ as well. Thus the probability to accept a tree $T \in \mathcal{B}_n$ in $\tilde{\mathcal{M}}_n$ (or in $\tilde{\mathcal{M}}_{n,m-n+1}$) is at least (we use the fact that $\sum_{d=1}^{n-1} T_{n,d} = c_{n-1}$ and $T_{n,1} = c_{n-2}$, where $c_n = \frac{1}{n+1}\binom{2n}{n}$ is the nth Catalan number counting the number of n-edge rooted trees):

$$\frac{T_{n,1} 2^{n-2}}{|\mathcal{B}_n|} = \frac{T_{n,1} 2^{n-2}}{\sum_{d=1}^{n-1} T_{n,d} 2^{n-2}} = \frac{c_{n-2}}{c_{n-1}} = \frac{n}{4n-6} > \frac{1}{4} .$$

It follows that the average number of rejects is at most 4, that completes the proof. □

The algorithm presented in the proof of Theorem 7 can be enhanced into an anticipated-reject version. In this version, the bicolored tree is constructed from the end (so from the last branch). As soon an inner black node appears in the last branch, we reject. The advantage of this version is that fewer random bits are needed to generate an outerplanar map since the expected length of the last branch is $O(1)$, so only $O(1)$ bits would be wasted before acceptation of the whole bicolored tree. An implementation of the enhanced algorithm is available in the PIGALE Library (*http://pigale.sourceforge.net/*), and an experiments of this algorithm is given in Fig. 4.

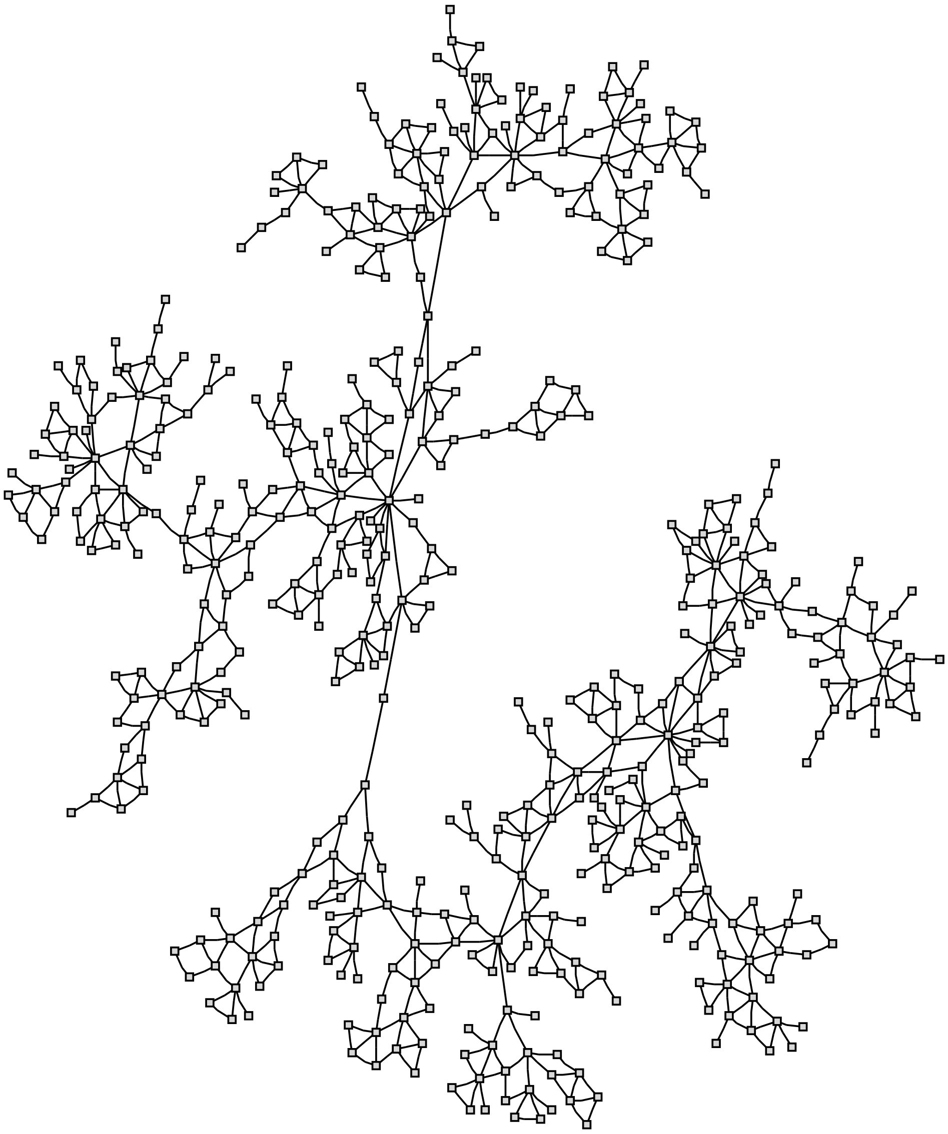

Figure 4: A random outerplanar map with 500 nodes, drawn using a spring embedding algorithm preserving embedding.

References

[ARS97] L. Alonso, J. L. Rémy, and R. Schott. A linear-time algorithm for the generation of trees. *Algorithmica*, 17(2):162–182, 1997.

[AS80] D. B. Arnold and M. R. Sleep. Uniform random generation of balanced parenthesis strings. *ACM Trans. Programming Languages and Systems*, 2(1):122–128, 1980.

[BdLP99] E. Barcucci, A. del Lungo, and E. Pergola. Random generation of trees and other combinatorial objects. *Theoretical Computer Science*, 218(2):219–232, 1999.

[BGH03] Nicolas Bonichon, Cyril Gavoille, and Nicolas Hanusse. An information-theoretic upper bound of planar graphs using triangulation. In 20^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of LNCS, pages 499–510. Springer, February 2003.

[Bil92] T. Bilski. Embedding graphs in books: A survey. *IEE Proceedings-E*, 139(2):134–138, March 1992.

[BK03] Manuel Bodirsky and Mihyun Kang. Generating random outerplanar graphs. In 1^{st} *Workshop on Algorithms for Listing, Counting, and Enumeration (ALICE)*, January 2003.

[CH67] G. Chartrand and F. Harary. Planar permutation graphs. *Ann. Inst. Henry Poincare, Sec. B3*, pages 433–438, 1967.

[CLL01] Yi-Ting Chiang, Ching-Chi Lin, and Hsueh-I Lu. Orderly spanning trees with applications to graph encoding and graph drawing. In 12^{th} *Symposium on Discrete Algorithms (SODA)*, pages 506–515. ACM-SIAM, January 2001.

[CM92] R. Cori and A. Machi. Maps, hypermaps and their automorphisms: a survey i, ii, iii. *Expo. Math.*, 10:403–467, 1992.

[CNAO85] Norishige Chiba, Takao Nishizeki, Shigenobu Abe, and Takao Ozawa. A linear algorithm for embedding planar graphs using pq-trees. *Journal of Computer and System Sciences*, 30(1):54–76, 1985.

[Die00] Reinhard Diestel. *Graph Theory (second edition)*, volume 173 of Graduate Texts in Mathematics. Springer, February 2000.

[DZ99] Alain Denise and Paul Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theoretical Computer Science*, 218:233–248, 1999.

[ES94] P. Epstein and J.-R. Sack. Generating triangulations at random. *ACM Trans. Model. and Comput. Simul.*, 4:267–278, 1994.

[Fel68] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley & Sons, 1968.

[FN99] P. Flajolet and M. Noy. Analytic combinatorics of non-crossing configurations. *Discrete Mathematics*, 204:203–229, 1999.

[Gol95] M. Goldwurm. Random generation of words in an algebraic language in linear binary space. *Information Processing Letters*, 54:229–233, 1995.

[Kre70] G. Kreweras. Sur les éventails de segments. *Cahiers du Bureau Universitaire de Recherche Opérationnelle*, 15:1–41, 1970.

[Mit79] S. L. Mitchell. Linear algorithms to recognize outerplanar and maximal outerplanar graphs. *Inform. Proc. Letters*, pages 229–232, 1979.

[MR01] J. Ian Munro and Venkatesh Raman. Succinct representation of balanced parentheses, static trees and planar graphs. *SIAM Journal on Computing*, 31(3):762–776, 2001.

[Nar59] T. V. Narayana. A partial order and its application to probability. *Sankhya*, 21:91–98, 1959.

[Wil77] H.S. Wilf. A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, 24:281–291, 1977.