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

Research Report `RR-1279-02`

An Information Upper Bound of Planar Graphs Using Triangulation

Nicolas Bonichon, Cyril Gavoille, and Nicolas Hanusse

September, 2002

An Information Upper Bound of Planar Graphs Using Triangulation

Nicolas Bonichon* Cyril Gavoille* Nicolas Hanusse*

Abstract

We propose a new linear time algorithm to represent a planar graph. Based on a specific triangulation of the graph, our coding takes on average 5.03 bits per node, and 3.37 bits per node if the graph is maximal. We derive from this representation that the number of unlabeled planar graphs with n nodes is at most $2^{\alpha n+O(\log n)}$, where $\alpha \approx 5.007$. The current lower bound is $2^{\beta n+\Theta(\log n)}$ for $\beta \approx 4.71$. We also show that almost all unlabeled and almost all labeled n-node planar graphs have at least $1.70n$ edges and at most $2.54n$ edges.

Keywords: compact graph representation, planar graph, triangulation, realizer

1 Introduction

How many information can contain a simple planar graph with n nodes? The question is highly related to the number of planar graphs. Counting the number of (non-isomorphic) planar graphs with n nodes is a well-known long-standing unsolved graph-enumeration problem (cf. [LW87]). There is no known close formula, neither asymptotic nor even an asymptotic on the logarithm of this number. Any asymptotic on the logarithm would give a bound on the number of independent random bits needed to generate randomly and uniformly a planar graph (but not necessary in polynomial time).

Random combinatorial object generation is an important activity regarding average case complexity analysis of algorithms, and testing algorithms on typical instances. Unlike random graphs (the Erdös-Rényi graph Model), still little is known about random planar graphs. This is mainly due to the fact that adding an edge in a planar graph highly depends on the location of all previous edges. Random planar maps, i.e., plane embeddings of planar graphs, have been investigated more successfully. Schaeffer [Sch99], and then Banderier et al. [BFSS00] have showed how to generate in polynomial time several planar map families, e.g. 3-connected planar maps. Unfortunately, this generating gives quite few information about random planar graphs because there are many ways to embed a planar graph into the plane. On the positive side, some families of planar graphs support efficient random generation: Trees [ARS97], maximal outerplanar graphs [ES94, BdLP99] (i.e., triangulation of convex polygons), and more recently labeled and unlabeled outerplanar graphs [BK02].

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

Besides the combinatorial aspect and random generation, a lot of attention is given in Computer Science to *efficiently* represent discrete objects. Efficiently means that the representation is succinct, i.e., the storage of these objects uses few bits, and that the time to compute such representation is polynomial in their size. Fast manipulation of the so encoded objects and easy access to a part of the code are also desirable properties. At least two scopes of applications of high interests are concerned with planar graph representation: Computer Graphics and Networking.

Surface discretization of a 3D object outputs a list of 3D coordinates, and a set of adjacency relations. In the case of convex object, the set of adjacency relations is an unlabeled planar graph. In general, small degree faces are used for surface discretization, with triangle or quad meshes. Then, a compressor is applied on the planar graph. Performances are expressed averaging the number of bits per edge or per node. They are evaluated among a benchmark of standard examples [KADS02], due to the lack of "good" random planar graph generator, or typical instance generator. E.g., King and Rossignac [KR99, Ros99] gave an a triangulation compressor that guarantees 3.67 bits per node encoding, the best possible rate being $\log_2(256/27) \approx 3.24$ bits per node from Tutte's enumerative formula [Tut62].

Routing table design for a network has been investigated in the case of planar networks [FJ89, GH99, Lu02a, Tho01]. The underlying graph of the network is preprocessed in order to optimize routing tables, a data structure dedicated to each node in charge of finding the next output port given the destination address of an incoming message. The main objective is to minimize the size of the routing tables while maintaining routes as short as possible. The strategy used by Gavoille and Hanusse [GH99] based on k-page embedding, and then improved by Lu [Lu02a] with *orderly spanning trees*, demonstrates that a compact planar graph representation helps for the design of compact routing tables, especially when shortest paths are required.

1.1 Related works

Succinct representation of n-node m-edge planar graphs has a long history. Turán [Tur84] pioneered a $4m$ bit encoding, that has been improved later by Keeler and Westbrook [KW95] to $3.58m$. Munro and Raman [MR97] then proposed a $2m + 8n$ bit encoding based of the 4-page embedding of planar graphs (see [Yan89]). In a series of articles, Lu et al. [CGH$^+$98, CLL01] refined the coding to $4m/3 + 5n$ thanks to orderly spanning trees, a generalization of Schnyder's trees [Sch90]. Independently, codings have been proposed for triangulations, where $m = 3n - 6$. A $4n$ bit encoding has been obtained by several authors [CGH$^+$98, Ros99, Bon02], interestingly with rather different techniques. Then improved by the Rossignac's Edgebreaker [KR99], guaranteeing $3.67n$ bits for triangulations, moreover computable in $O(n)$ time. Actually, He, Kao and Lu [HKL00] showed that, in $O(n \log n)$ time, a space optimal encoding for triangulations and for unlabeled planar graphs can be achieved. So, a $O(n \log n)$ time and a $3.24n$ bit encoder for triangulations exists. For that, they use a recursive separator decomposition of the graph, and an exponential coding algorithm for the very end components of sub-logarithmic size. However, the time complexity hidden in the big-O notation could be of limited use in practice. To implement the encoder, one needs, for instance, to implement planar isomorphism and Lipton-Tarjan planar separator [LT79]. Recently the time complexity has been improved to $O(n)$ for planar graphs by Lu [Lu02b]. Although the length of the coding is optimal, the approach of [HKL00] does not give any explicit bound of the number of bits use in the representation.

If we are interested only in the Information-Theoretical bound of planar graphs or in statistical properties of planar graphs (what a random planar graph looks like: number of edges, connectivity, etc.), other tools can be used. Denise et al. [DVW96] specified a Markov chain on the space of all labeled planar graphs whose limit distribution is the uniform distribution. Their experiments show that random planar graphs have approximately $2n$ edges, are connected but not 2-connected. Although the Markov chain converge to the uniform distribution, it is not proved whether this Markov chain becomes sufficiently close to the uniform distribution after a polynomial number of steps. It is however proved that almost all labeled planar graphs have at least $3n/2$ edges, and that the number $p(n)$ of unlabeled planar graphs satisfies that $\frac{1}{n}\log_2 p(n)$ tends to a constant γ such that $\log_2(256/27) \leqslant \gamma \leqslant \log_2(256/27) + 3$. The bounds on γ easily derive from Tutte's formula [Tut62]: Triangulations are planar graphs, and every planar graph is a subgraph of a triangulation, thus having 2^{3n-6} possible subsets of edges. There are also no more than $n!2^{\gamma n+o(n)}$ labeled planar graphs as there are at most $n!$ ways to label the nodes of a graph.

Osthus, Prömel, and Taraz [OPT02] investigated triangulations contain in planar graphs, and they showed that there is no more than $n!2^{5.22n+o(n)}$ labeled planar graphs. They also showed that almost all labeled planar graphs have at most $2.56n$ edges. A lower bound of $13n/7 \approx 1.85n$ has been obtained by Gerke and McDiarmid [GM01], improving the $1.5n$ lower bound of [DVW96]. They also derived that almost all unlabeled planar graphs have at most $2.69n$ edges. Note that there is no evidence that labeled and unlabeled planar graphs have the same growing rate (up to the $n!$ term), unlike general graphs. So upper bounds on labeled planar graphs do not transfer to upper bounds on unlabeled planar graphs, but the reverse is true.

Using generating function techniques, Bender, Gao and Wormald [BGW99] proved that the number of labeled 2-connected planar graphs tends to $n!2^{4.71n+O(\log n)}$. Note that enumeration of planar maps gives an upper bound on planar graphs. Recently, Bousquet-Mélou [BM02], showed that the number of simple planar maps have a $2^{5.098n+O(\log n)}$ asymptotic, therefore given a $2^{5.098n+O(\log n)}$ upper bound for the number of unlabeled planar graphs, and $n!2^{5.098n+O(\log n)}$ for labeled planar graphs.

1.2 Our results

In this paper we show an upper bound of $2^{5.007n+O(\log n)}$ on the number of unlabeled planar graphs. This result implies that the number of labeled planar graphs is no more than $n!2^{5.007n+O(\log n)}$, an improvement upon the bounds of [OPT02].

Since our upper bound can be parameterized with the number of edges, and using the lower bound of [BGW99], we are able to show that almost all unlabeled graphs have at least $1.70n$ edges and at most 2.54 edges, improving the $2.69n$ upper bound of [OPT02]. Moreover the result is valid for labeled planar graphs (slightly improving the $2.56n$ previous bound), connected labeled and connected unlabeled planar graphs.

Beside the fundamental aspect of planar graph enumeration, our technique is based on an *explicit* representation, quite easily and linear time constructible. In addition we give a simple linear time $3.37n$ bit coding of triangulations, and a $5.03n$ bit coding for planar graphs. Our triangulation representation already improves the Edgebreaker compressor [KR99], and there is no doubt that our explicit construction for planar graphs can be used for Networking, in particular to improve the result of [Lu02a].

1.3 Outline of the paper

Let us sketch our technique. A natural approach to represent an n-node planar graph G is to consider a triangulation of G, i.e., a supergraph S of G such that S is planar, has n nodes and $3n-6$ edges. Then, G can be obtained by coding S and a set M_S of edges such that $E(S) \setminus M_S = E(G)$. This way of representing a planar graph is suggested by the $(\log_2(256/27)+3)n = 6.24n$ bit upper bound of [DVW96] given above. To obtain a representation more compact than $6.24n$ bits we need to carefully construct S. In particular, crucial steps are the way we embed G into the plane, and the way we triangulate its faces. We show that, if G is connected, the supergraph S of G, called hereafter *super-triangulation* defined in Section 2, has the following property: given only S and a node $v \in S$, one can perform in a unique manner a traversal of S and find a spanning tree T rooted in v such that T is contained in G. So, given the super-triangulation S of G, the edges of M_S can be described among the possible edges of $S \setminus T$ only, i.e., with at most $2n$ bits. This already provides a $(\log_2(256/27)+2)n = 5.24n$ bit upper bound. Observe that the case G not connected can be easily transformed (in linear time) into a new connected graph $\tilde{G}$, e.g. by merging all the connected components of G into a single node (see Section 4 for more details).

Actually, the $5.24n$ bound mentioned above does not lead to an explicit representation because it is based on a $3.24n$ bit theoretical coding of triangulation, and no such explicit representation is known. To overcome this problem, we represent the super-triangulation S (that is a specific triangulation) by a triple of trees forming a very specific *realizer*, that is a partition of the edges into three particular trees (T_0, T_1, T_2) also called Schnyder's trees [Sch90]. And, we show how to uniquely recover the tree partition from a triangulation. Among the properties of S, we have that T_0 is contained in G (assumed to be connected from the transformation). Moreover every edge (u, v) of S, where u is the parent of v in T_1 and where u is an inner node in T_2, must belong to G. This significantly saves bits in the coding of M_S since many edges of G can be guessed from S. An extra property is that two nodes belonging to the same branch of T_2 have the same parent in T_1 (a branch is a maximal set of related nodes obtained in a clockwise depth-first search of the tree, and such that a node belongs to only one branch at the time, see Section 3). This latter property simplifies a lot the representation of S. Knowing T_2, T_1 does not need to be fully represented. Only one relevant edge per branch of T_2 is enough. As any tree of a realizer can be deduced from the two others, the representation of S can be compacted in a very efficient way, storing for instance T_2 and the relevant edges of T_1.

Finally, the explicit representation of G is done with 8 binary strings of different density (namely the ratio between the number ones it contains and its length): 7 for representing S (5 for T_2 and 2 for the relevant edges of T_1), and 1 for M_S. We compact each string with the standard compressor of Pagh [Pag01] (but we give a simpler linear time construction in this paper). This allows to reach an optimal entropy coding, i.e., with $\log_2 \binom{n}{k} + o(n)$ bit for an n-bit string of k ones. Taken in parameter the number of branches of T_2 (or equivalently its number of leaves), an entropy analysis shows that $3.37n$ bits are sufficient to represent S, and $5.03n$ bits suffice for G. The final $5.007n$ bound is reached with a more sophisticated encoding of S.

In Section 3 we propose a rather simple algorithm to construct a super-triangulation starting from a given plane embedding. The algorithm returns a plane embedding that can be different from the initial one. This is done in two steps: first an intermediate object is constructed, the *well-orderly tree*. It consists in a specific spanning tree of the plane embedding that corresponds to one of the tree of the super-triangulation. Then, faces of the embedding are specifically triangulated to obtain the super-triangulation with all its properties.

2 Embedding and Triangulating Algorithms

A *plane embedding of a graph*, or shortly a *plane graph*, is a mapping of each node to a point of the plane and of each edge to the continuous curve joining the two ends of this edge such that edges do not crossing except, possibly, on a common extremity. A graph that has a plane embedding is a planar graph.

In this paper we deal with simple (no loops and no multi-edges) and undirected graphs. If we cut the plane along the edges, the remainder falls into connected regions of the plane, called *faces*. Each plane graph has a unique unbounded face, called the *outerface*. The *boundary* of a face is the set of incident edges. The edges of the outerface boundary are the *exterior* edges and the others the *interior* edges. Precise definitions can be founded for instance in [NC88].

A *triangulation* is a plane embedding of a maximal planar graph that is a planar graph with n nodes and $3n-6$ edges. There is only one way to embed in the plane (up to a continuous transformation) a maximal planar graph whose three nodes are chosen to lie on the outerface.

2.1 Well-orderly tree, realizer and super-triangulation

Before introducing super-triangulations, we need some preliminaries, in particular well-orderly trees and realizers.

Let T be a rooted spanning tree of a plane graph 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.

We introduce *well-orderly trees*, a special case of *orderly spanning trees* of Chiang, Lin and Lu in [CLL01], referred as simply orderly trees in the next. Let $v_1, \dots, v_n$ be the clockwise preordering of the nodes in T. Recall that a node v_i is *orderly* in H w.r.t. T if the incident edges of v_i in H form the following four blocks (possibly empty) in clockwise order around v_i (see Fig. 1(b)):

- $B_P(v_i)$: 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; and
- $B_>(v_i)$: unrelated edges incident to nodes v_j with $j > i$.

A node v_i is *well-orderly* in H w.r.t. T if it is orderly, and if:

- the clockwise first edge $(v_i, v_j) \in B_>(v_i)$, if it exists, verifies that the parent of v_j is the nearest common ancestor of v_i and v_j in T.

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 (similarly for simply orderly tree). Note that an orderly tree (simply or well-orderly) is necessarily a spanning tree. Observe also that the incident edges in H of a node of T are either in T or unrelated. In particular, if an edge of H is related (i.e., one endpoint is a descendant of the other one in T), then it has to belong to T. It follows that all the neighbors in H of the root of T are in T.

A convenient way to manipulate triangulations is to deal with Schnyder's trees, called also realizer. A *realizer* of a triangulation is a partition of its interior edges in three sets T_0, T_1, T_2 of directed edges such that for each interior node v it holds (see Fig. 1(a)):

- the clockwise order of the edges incident with v is: leaving in T_0, entering in T_1, leaving in T_2, entering in T_0, leaving in T_1 and entering in T_2;
- there is exactly one leaving edge incident with v in T_0, T_1, and in T_2.

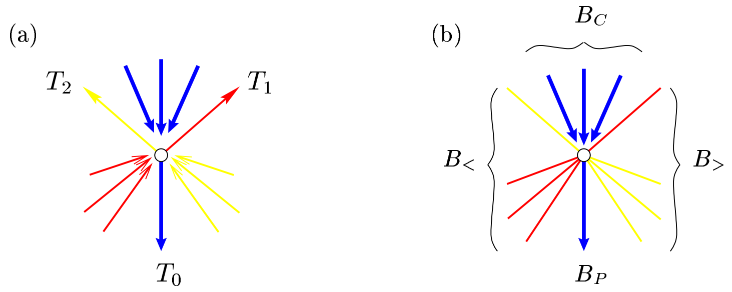

Figure 1: Relationship between realizer and orderly tree: (a) edge-orientation rule around a node for a realizer, and (b) blocks ordering around an orderly node (T is represented by directed edges because the edge (v,w) of T_1).

Observe that if (T_0,T_1,T_2) is a realizer, then (T_1,T_2,T_0) and (T_2,T_0,T_1) are also realizers. Cyclic permutations of a realizer are not in general the only distinct realizers of a given triangulation. Fig. 2 depicts two realizers for a same triangulation. Actually, the number of n-node realizers is asymptotically $2^{4n+O(\log n)}$ (cf. [Bon02]), whereas the number of triangulations is only $(256/27)^{n+O(\log n)}$ (cf. [Tut62]).

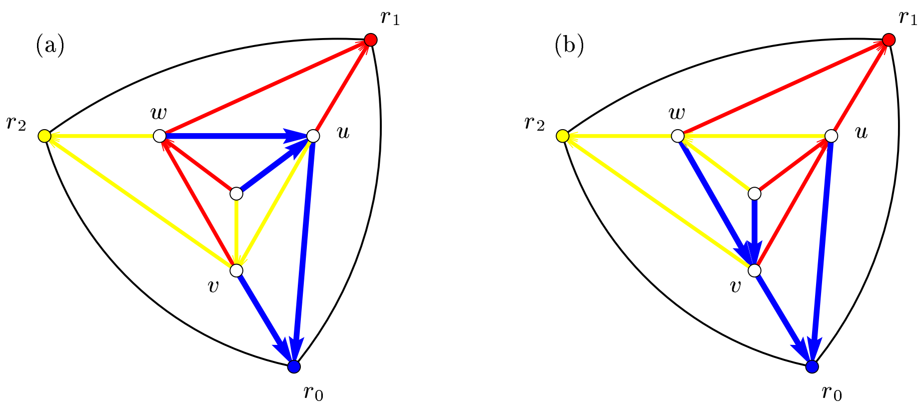

Figure 2: Two realizers for a triangulation. The tree $\overline{T}_0$ rooted in r_0 (the tree with directed blue bold edges augmented with the edges (r_0,r_1) and (r_0,r_2)) is well-orderly in (b), and simply orderly in (a) (the node v is not well-orderly).

Schnyder showed in [Sch90] that each set T_i of a realizer induces a tree rooted in one node of the outerface. Moreover, he described a linear time algorithm to compute such trees. Hereafter, if $R = (T_0, T_1, T_2)$ is a realizer, then for notational convenience R also denotes the underlying triangulation.

There are strong relationships between realizers and orderly trees (see Fig. 1). In every realizer $R = (T_0, T_1, T_2)$, T_0 (and by cyclic permutation each T_i) is an orderly tree of $R \setminus \{r_1, r_2\}$, where r_i denotes the root of T_i. Indeed, the incident edges with any node v that are not in T_0 (thus that are unrelated with T_0) are either clockwise before the entering edges of T_0 or clockwise after. Conversely, let T be an orderly tree of a triangulation. Observe that the root of T has at least two children (because its root is of degree at least two and all its neighbors must be in T), and thus T has at least two leaves. A realizer (T_0, T_1, T_2) can be obtained from T setting $T_0 = T \setminus \{r_1, r_2\}$, where r_1, r_2 are respectively the clockwise last and first leaf of T — actually it is not difficult to see that r_i is the root of T_i —, and setting, for all internal nodes v, that the clockwise first edge of $B_>(v)$ and the clockwise last edge of $B_<(v)$ belongs to T_1 and T_2 respectively (as illustrated on Fig. 1(b)). Observe that this latter assignment for T_1 and T_2 is the only possible realizer with $T_0 = T \setminus \{r_1, r_2\}$.

For each tree T_i of a realizer, we denote by $\overline{T}_i$ the tree composed of T_i augmented with the two edges of the outerface incident to the root of T_i. A node of a rooted tree is *inner* if it is neither the root, nor a leaf. For every non-root node $u \in T_i$, we denote by $p_i(u)$ the parent of u in T_i.

Definition 1 *A realizer $S = (T_0, T_1, T_2)$ is a* super-triangulation *of a graph G if:*

1. *$V(S) = V(G)$ and $E(G) \subseteq E(S)$;*
2. *$E(T_0) \subseteq E(G)$;*
3. *$\overline{T}_0$ is a well-orderly tree of S; and*
4. *for every inner node v of T_2, $(v, p_1(v)) \in E(G)$.*

Theorem 1 *Every connected planar graph with at least three nodes has a super-triangulation, computable in linear time.*

Intuitively, a super-triangulation of a graph G is a specific triangulation of the faces of a specific plane embedding of G. One of the interesting properties of super-triangulations is the following one:

Theorem 2 *For every 3-connected planar graph G and every triple of nodes r_0, r_1, r_2 of a face of G, there exists only one super-triangulation (T_0, T_1, T_2) of G such that T_i has root r_i, for every $i \in \{0, 1, 2\}$.*

In particular, triangulations have exactly one super-triangulation, up to a cyclic permutation of the roots on the outerface. This property will play an important role in Section 4 for our upper bound on the number of planar graphs. Fig. 2(b) is a super-triangulation, whereas Fig. 2(a) is not because the tree $\overline{T}_0$ cannot be well-orderly.

2.2 Computing a super-triangulation

In order to prove Theorem 1 and Theorem 2, we need the next three lemmas The proofs of these lemmas are given after the proofs of Theorem 1 and Theorem 2. A plane graph H is a *well-orderly embedding rooted in v* if H has a well-orderly tree of root v.

Lemma 1 *Every well-orderly embedding rooted in some node v has a unique well-orderly tree of root v.*

Lemma 2 *Let G be a connected planar graph, and let v be any node of G. Then G has a well-orderly embedding of root v. Moreover, the well-orderly tree and the embedding rooted in v can be computed in linear time.*

It is proved in [CLL01] a result similar to Lemma 2 about simply orderly trees and embeddings. However, the extra condition concerning "nearest common ancestor" reduces much more the choice of the embedding for the input planar graph, and leads to the uniqueness of the tree (Lemma 1). In the case of simply orderly embeddings, several orderly trees may exist (cf. Fig. 2 where both orderly trees $\overline{T}_0$ span the same triangulation). Actually, the uniqueness concerns also the way to triangulate the faces of well-orderly embeddings, thanks to the next lemma.

Lemma 3 *Let T be the well-orderly tree of H rooted in some node r_0, and assume that T has at least two leaves. Let r_2 and r_1 be the clockwise first and last leaves of T respectively. Then, there is a unique super-triangulation (T_0, T_1, T_2) of the underlying graph of H, preserving the embedding H, and such that each T_i has root r_i. Moreover, $T_0 = T \setminus \{r_1, r_2\}$ and the super-triangulation is computable in linear time.*

First of all, let us show that lemmas 1, 2, and 3 imply Theorem 1 and Theorem 2.

Proof of Theorem 1. Consider a connected planar graph G with at least three nodes, and let v be any node of G with the only constraint that if G is a path, then v is chosen to be of degree two (this is doable since G has at least three nodes). Thanks to Lemma 2, one can compute in linear time a well-orderly embedding H of G and a well-orderly tree T rooted in v. Let us show that T has at least two leaves r_1, r_2 lying on the outerface of H, r_2 traversed before r_1 in a clockwise preordering of T.

We show that T cannot be a chain, and thus has a node with at least two children (and thus has two leaves). If G is a path, then T rooted in a node of degree two, is not a chain. Assume that G is not a path, but T is a chain. Then there exist an edge of G that is not in T. However, all pairs of nodes of a chain are related, thus must belong to T. Therefore T is not a chain.

Lemma 3 can be therefore applied, and one can compute for G a super-triangulation in linear time. □

Proof of Theorem 2. By Whitney's Theorem (see e.g. [Die00]), there is one way to embed a 3-connected graph when three nodes, say r_0, r_1, r_2, are chosen to lie on the outerface. Let H be such an embedding of G. Assuming that clockwise around the outerface of H the nodes are encountered in the order r_0, r_2, r_1. Let H' (resp. G') be the plane graph (resp. planar graph) obtained from H (resp. from G) and augmented with the edges (r_0, r_2), (r_2, r_1), (r_1, r_0) such that they form the outerface of H'. (Each edge is added only if it does not create multi-edges.)

The well-orderly tree T of H' rooted in r_0 has two leaves lying on the boundary of the outerface H', namely r_2 and r_1, that are respectively the clockwise first and last leaves of T. The tree is unique from Lemma 1, once H' and r_0 have been fixed. One can therefore apply

Lemma 3 to H', and compute in linear time the unique super-triangulation $S' = (T'_0, T'_1, T'_2)$ for G' preserving H' with T'_i rooted in r_i. As G' is 3-connected, H' is the unique embedding for G' with r_i lying on its outerface. Thus S' is actually the only one super-triangulation of G' (Lemma 3). We check that S' is also a super-triangulation of G as the extra edges added on the outerface cannot create new inner nodes in T_2. Conversely, since any super-triangulation $S = (T_0, T_1, T_2)$ of G with T_i rooted in r_i has to contain the edges (r_0, r_2), (r_2, r_1), (r_1, r_0), S is also a super-triangulation of G'. As a consequence, the super-triangulation S for G is S'. S is unique from the uniqueness of S', and can be computed in linear time. That completes the proof. □

Proof of Lemma 1. Assume that H has two well-orderly trees T, T' rooted in v. Let $v_1, \ldots, v_n$ (resp. $v'_1, \ldots, v'_n$) be the clockwise preordering of the nodes of T (resp. T'). Let v_i be the node such that the neighbors of v_i in T and in T' differ, and such that i is minimum. We have $v_t = v'_t$ for all $t \leqslant i$, and $B_C(v_i) \neq B'_C(v_i)$, where $B'_C(v_i)$ denotes the children edge block around v_i in T'.

Let u be a node such that $u \in B'_C(v_i)$ and $u \notin B_C(v_i)$. (The symmetric case $u \notin B'_C(v_i)$ and $u \in B_C(v_i)$ is proved by exchanging the role of T and T'). Let j be such that $u = v_j$, and let v_k be the parent of v_j in T. Obviously, $k < j$. Note that (v_i, v_j) is an unrelated edge of T, but is an edge of T'.

Let us show that $k < i$. Clearly, if $j < i$, then $k < i$. If $j > i$, then the nearest common ancestor between v_i and v_j is the parent in T of v_j, that is v_k. So, $k \leqslant i$. The case $k = i$ is impossible since (v_i, v_j) is not an edge of T. Therefore, $k < i$.

As v_i and v_k are unrelated, the path P in T between v_k and v_i traverses only nodes that are ancestors of v_k or ancestors of v_i. Thus, the nodes of P are v_t's with $t \leqslant \max\{k, i\} \leqslant i$. Hence P is in T' too. It follows that the edge (v_k, v_j) is not in T', otherwise P and the edge (v_i, v_j) of T' would create a cycle in T'. As (v_k, v_j) is not in T' but is in T, the trees T and T' differ at v_k with $k < i$: a contradiction with the definition of i. □

Proof of Lemma 2. We first give a simple algorithm to construct a well-orderly embedding of G. Then we give some hints for an $O(n)$ time implementation.

We start by computing an arbitrary plane embedding H of G such that v belongs to the outerface of H. This can be done in $O(n)$ time [CNAO85]. Then we traverse H from v in order to build a well-orderly tree T. However, not any plane embedding allows the construction of a well-orderly tree. If during the construction, T does not span all the nodes, the embedding of H is modified, and a new traversal is run again. We show that, after a finite number of steps, the number of nodes covered by the traversal increase, and so the construction converges to a well-orderly tree. To describe more precisely the traversal and the modification of H, we need some definitions.

Let T be any tree of H rooted in v, not necessarily a spanning tree. A node is *free* if it does not belong to T. An edge is free if one of its endpoints if free. We extended the notion of well-orderly node as follows: a node is *partially well-orderly* (w.r.t. H and T) if it is well-orderly except that its edge-blocks $B_<$ and $B_>$ (relative to a clockwise preordering of the nodes of T) may contain zero or more free edges. Moreover, the clockwise last edge of $B_<$ and the clockwise first edge of $B_>$ are never free. T is partially well-orderly if all the nodes of T are partially orderly. Every well-orderly tree is a partially well-orderly tree that spans H. The four edge-blocks around a partially well-orderly node u in T are denoted by $B_P(v,T)$, $B_<(v,T)$, $B_C(v,T)$, and $B_>(v,T)$. For convenience, the clockwise last edge of $B_<(v,T)$ is named the *back-edge* of u in T, and the clockwise first edge of $B_>(v,T)$ is named the *front-edge* in T.

We run a procedure TRAVERSAL(H,v) applied on the current embedding H of G, that returns a partially well-orderly tree T of H rooted in v (see Fig. 3 for an example). It initializes $T := \{v\}$, and treats v as follows: 1) List in a clockwise manner around v the edges $(v,u_1),\ldots,(v,u_k)$ that are after the back-edge and before the front-edge of v in T (if the back-edge or the front-edge does not exist, the list consists of all the edges incident with v not already in T); 2) Update T with the edges $(v,u_1),\ldots,(v,u_k)$; and 3) recursively treat the nodes $u_1,\ldots,u_k$ in that order.

Let $v_1,\ldots,v_p$ be the clockwise preordering of the nodes of T, the tree returned by TRAVERSAL(H,v). Consider a node v_i, and let T_{v_i} be the tree obtained by TRAVERSAL(H,v) after the treatment of v_i. A key observation is that $B_<(v_i,T_{v_i}) = B_<(v_i,T)$ and that $B_>(v_i,T_{v_i}) = B_>(v_i,T)$. In particular the back-edge and front-edge of v_i w.r.t. T and w.r.t. T_{v_i} (if they exists) are the same. After the treatment of v_i, the edges around v_i in T_{v_i} form the four (possibly empty) edge-blocks $B_P(v_i,T_{v_i})$, $B_<(v_i,T_{v_i})$, $B_C(v_i,T_{v_i})$, $B_>(v_i,T_{v_i})$. So in T, the edge-blocks around v_i are: $B_P(v_i,T)$, $B_<(v_i,T)$, $B_C(v_i,T)$, and $B_>(v_i,T)$. To show that T is partially well-orderly, it remains to show that if $(v_i,v_j) \in B_>(v_i,T)$ is the front-edge, then the parent of v_j in T is the nearest common ancestor between v_i and v_j. When visiting the node v_i in T, the edges of the tree constructed up to v_i (i.e., $T_{v_{i-1}}$) are either between nodes v_t with $t < i$, or are (v_k,v_j) with $k < i$ and $j > i$. Moreover, v_k belongs to the path from v_i to the root of T, v. Thus a front-edge (v_i,v_j) is such that the parent of v_j is an ancestor of v_i, the nearest common ancestor. Therefore, T is partially well-orderly.

Assume that T does not cover all the nodes (if T is a spanning tree, then we are done). Let v_i be a node of T having an incident free edge. W.l.o.g. we assume that $B_>(v_i,T)$ contains a free edge (cf. Fig. 3). The case where v_i has a free edge in $B_<(v_i,T)$ is symmetric. Let $e_i = (v_i,u)$ be the clockwise last free edge of $B_>(v_i,T)$. (Actually one can choose any free edge that clockwise ends a block of free edges in $B_>(v_i,T)$). By definition, $B_>(v_i,T)$ contains at least one unrelated edge (in particular the front-edge). So, let $e = (v_i,v_j)$ be the clockwise last unrelated edge of $B_>(v_i,T)$ that is before e_i. Finally, let $e_j = (v_j,w)$ be the clockwise first free edge of v_j before e and such that there is no unrelated edge between e_j and e (so e_j is the first edge of the block of free edges just before e). If such edge does not exist, we set $e_j := e$. In other words, e, e_i, and e_j are chosen such that the edges between e and e_i in v_i, and between e and e_j in v_j form a maximal block of free edges. We change the embedding H by running the procedure FLIP(H,e,e_i,e_j) that works as follows: In v_i, e is moved and inserted clockwise after e_i, and in v_j, e is moved and inserted clockwise before e_j. For convenience, we say that a FLIP has been performed around e.

Once H has been updated by FLIP(H,e,e_i,e_j), we reapply TRAVERSAL on the new embedding. Procedures TRAVERSAL and FLIP are so applied up to get a spanning tree. To complete the correctness, we need to show that FLIP keeps planarity, and that the partial well-orderly tree obtained after calling TRAVERSAL converges to a spanning tree.

Let X be the set of free nodes forming the free edges between e and e_i, and between e and e_j. For every $x \in X$, let C_x denote the connected component containing x in the subgraph

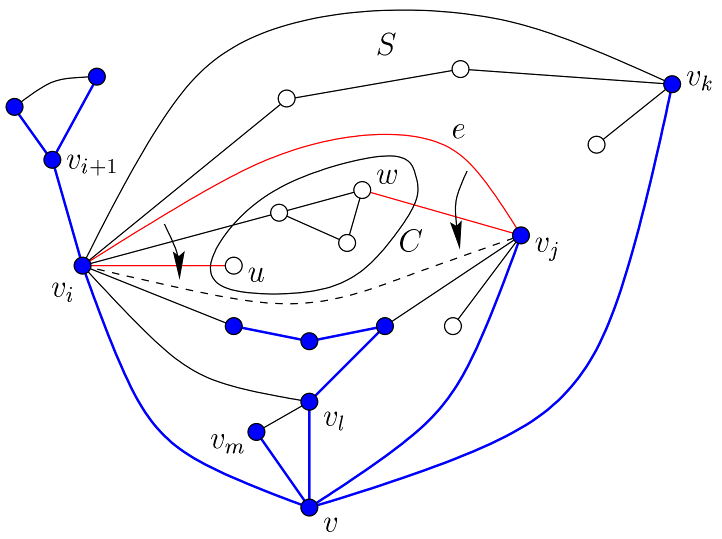

Figure 3: The tree returned by $\textsc{Traversal}(H, v)$ is bold. Free nodes are drawn white. The dotted edge is the new location of e after running $\textsc{Flip}(H, e, (v_i, u), (v_j, w))$.

of H induced by the free nodes. Let $C = \cup_{x \in X} C_x$. To prove that $\textsc{Flip}(H, e, e_i, e_j)$ keeps planarity, we show that every path P from $y \in C$ to the root v contains either v_i or v_j. Let R be the bounded connected region of $\mathbb{R}^2 \setminus B$, where B is the cycle composed of the path in T (the tree obtained before calling $\textsc{Flip}$) from v_i to v_j, and closed by the edge (v_i, v_j). Assume that P contains a node $v_k \in T$ with $v_k \in R \cup B$, and $k \notin \{i, j\}$. W.l.o.g. assume that v_k is the first node T from y in P, and let (v_k, z) be the free edge of P. We have $(v_k, z) \in B_<(v_k, T)$, or $(v_k, z) \in B_>(v_k, T)$. Let us assume $(v_k, z) \in B_<(v_k, T)$, the other case is symmetric. As $B_<(v_k, T) \neq \varnothing$, v_k has a back-edge. The back-edge is (v_k, v_i). Indeed, if the back-edge is (v_k, v_t), $t \neq i$, then the cycle composed of the path in T between v_k and v_t, and closed by (v_k, v_t), would disconnect C_z and $\{v_i, v_j\}$, and v_i or v_j has a free neighbor in C_z. Thus the edge (v_i, v_k) exists, and is clockwise after (v_i, v_j) since it belongs to R. As (v_i, v_k) disconnects v_j from C_z, there must exist a free edge (v_i, s), for some $s \in C_z$. This edge is clockwise after (v_i, v_k): a contradiction with the definition of (v_i, v_j).

Therefore the embedding returned by $\textsc{Flip}(H, e, e_i, e_j)$ is a plane embedding of G. Observe that whenever e has been moved below C thanks to $\textsc{Flip}$, the tree returned by a new call to $\textsc{Traversal}$ contains T. Indeed, C is connected to $G \setminus C$ only by v_i and v_j. Thus the move of e cannot create an unrelated edge (v_t, x) with $x \in C$ and $t \notin \{i, j\}$. Assume that after moving e, $\textsc{Traversal}$ does not visit any new node. Then, either v_i has a front-edge and a free edge clockwise after its front-edge, or v_j has a back-edge and a free edge clockwise before its back-edge. Indeed, if not, all the neighbors of v_i and of v_j would be related, and the size of T would increase. Assume that v_i has front-edge e' and a free edge. It follows that a next call to $\textsc{Flip}$ can be applied on an edge e' clockwise before e. So, in at most $\deg(v_i)$ calls to $\textsc{Flip}$, a new free neighbor of v_i (or of v_j) will be visited. Observe that there is at most one $\textsc{Flip}$ around the edge e. On Fig. 3, after running $\textsc{Flip}$ around e, and calling $\textsc{Traversal}$, the tree T is augmented with (at least) the edge (v_j, w).

This completes the correctness of the well-orderly embedding algorithm for G. Let us evaluate its time complexity. There is at most $O(n)$ calls to $\textsc{Traversal}$ and to $\textsc{Flip}$ (as there is at most two calls to FLIP per edge, per in each direction), each one taking $O(n)$ time. Thus a naive implementation of that algorithm gives a $O(n^2)$ time algorithm.

We first remark that FLIP can be implemented in $O(1)$ worst-case time, by the use of double pointers for the incident edge list of a node, and using for each edge a pointer for each endpoint towards the edge pointer in the incident list. Moreover, as the tree grows by adding edges, the construction of the whole tree costs $O(n)$ time. The only difficulty is to efficiently manage the edges e, e_i, and e_j for preparing the call to FLIP.

Note that when one treats a node v_i with a front-edge (v_i, v_k), we have choice to continue the construction of T (the notations refer to the example depicted on Fig. 3): either one can continue T from v_i, by treating v_{i+1}, or we can consider the subgraph S delimited by the cycle composed of the path from v_i to v_k in T and close by (v_i, v_k), and one can recursively apply the treatments of the nodes of S (by treating v_m, v_l, and v_j in Fig. 3). If in T_{v_i} the nodes of S have been visited in the order $v_{i_1}, \ldots, v_{i_r}$, then the nodes are recursively treated in the order $v_{i_r}, \ldots, v_{i_1}$. Indeed both parts of the embedding (the part after v_i, and the part inside S) cannot interact because of the edge e. The part of the tree composed of the nodes after v_j can be computed after computing all the trees for S, and after applying FLIP(H, e, e_i, e_j). It is not difficult to see that the recursive version of the algorithm allows to manage e, e_i, and e_j on the fly with a total of $O(\sum_{i=1}^{n} \deg(v_i)) = O(n)$ time. Fig. 4 shows an example of the recursive algorithm. □

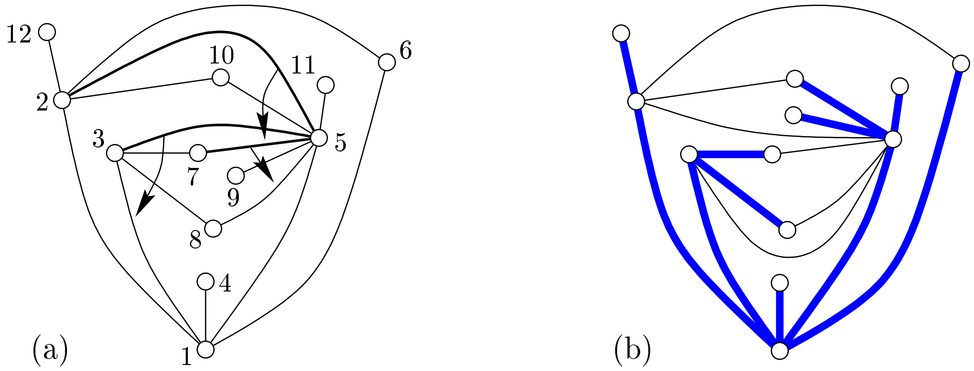

Figure 4: An application of the recursive algorithm with the visit ordering of nodes (a), and the final embedding with the well-orderly tree (b). Node labeled i is visited at step i, that is the edge between i and its parent is created at step i. When node 6 is reached, a recursive step is run on the nodes $3, 4, 5$, below the edge $(2, 6)$. In node 3, after a trivial recursive step (treatment of node 4), a FLIP around $(3, 5)$ is performed. This allows to visit the nodes 7 and 8. At the time node 5 is treated, a FLIP around $(5, 7)$, and then around $(5, 2)$ is performed, allowing to visit the nodes 9, 10, and 11, completing the recursive step below the edge $(2, 6)$. Then, the last node 12 is visited.

Proof of Lemma 3. Let T be the well-orderly tree of H rooted in r_0, and let $T_0 = T \setminus \{r_1, r_2\}$, where r_2, r_1 are the clockwise first and last leaves of T.

We first show that r_1 and r_2 belong to the boundary of the outerface of H. Consider P_i the path in T from r_0 to r_i, for $i \in \{1, 2\}$. All the nodes of P_i must belong to the outerface, in particular r_i. Indeed, by induction (this is true for r_0), a node v of P_2 (resp. P_1) has an empty edge-bock $B_<(v)$ (resp. $B_>(v)$). Thus the clockwise first (resp. last) children of v (if it exists) must belong to the outerface.

Let H' (resp. G') be the plane graph (resp. planar graph) obtained from H (resp. from G) and augmented with the three edges between the r_i's — keeping planarity as r_1, r_2 lie on the outerface — such that they form the boundary of the outerface of H'. (Each edge is added only if it does not create multi-edges.) As the edges between the r_i's belong to any super-triangulation of G, it suffice to show that the super-triangulation $S = (T_0, T_1, T_2)$ for G' preserving H' and with r_i root of T_i, is unique. The super-triangulation for G must be same, S (these three edges cannot create inner node for T_2). We will first show how to construct S, and then show that it is unique.

First observe that T_0 is a tree rooted in r_0 (removing two leaves from T maintains its connectivity). Clearly, $E(T_0) \subseteq E(G) \subseteq E(G')$, and $\overline{T}_0$ is a well-orderly tree of H' rooted in r_0. By Lemma 1, $\overline{T}_0$ is unique.

Let us construct the set T_1. Let $v_i \notin \{r_0, r_1\}$ be a node of $\overline{T}_0$. We apply the following assignment rules (for the induction we assume that the assignment has been applied to $r_0 = v_1, \ldots, v_{i-1}$, and that $p_1(r_0) = r_1$):

1. The clockwise first edge of $B_>(v_i)$ (this edge-block is relative to $\overline{T}_0$), if it exists, belongs to T_1;
2. If $B_>(v_i) = \varnothing$, then the edge leaving v_i to the child of $p_0(v_i)$ immediately after v_i clockwise around $p_0(v_i)$, if it exists, belongs to T_1;
3. If $B_>(v_i) = \varnothing$, and if v_i is the clockwise last child of $p_0(v_i)$, then $(v_i, p_1(p_0(v_i)))$ belongs to T_1.

Let us check that $H' \cup T_1$ is still a plane graph. No edge is added if Rule 1 applies. For Rule 2, there are no edges incident to $p_0(v_i)$ between v_i and its next sibling, keeping planarity of the embedding. And for Rule 3, $p_0(v_i)$ has no children between v_i and $p_1(v_i)$, allows to freely connected v_i with $p_1(v_i)$ since, by induction, one can assume that the property holds for every node v_j, $j < i$.

Let us check that $\{T_0, T_1\}$ are two sets of a realizer. We have seen that $\overline{T}_0$ is well-orderly. Every non-root node v_i has assigned a parent in T_1. So T_1 is connected. We check that in each of the three rules, the parent of v_i in T_1 is assigned to a node v_j of $\overline{T}_0$ with $j > i$. As a consequence, T_1 has no cycle and is connected, so it is a tree. We also check that the edge between v_i and its parent in T_1 is clockwise after the children (if they exist) of v_i in T_0, and clockwise before the edge to $p_0(v_i)$. Hence such T_1 set is compatible with the edge-orientation rule of realizers.

At this step $H' \cup T_1$ may contain edges that are assigned neither to $\overline{T}_0$ nor to $\overline{T}_1$. Let $X = E(H') \setminus (E(\overline{T}_0) \cup E(\overline{T}_1))$ be this edge set. Constructing T_2 can be done using Property 3. As there is only one way to select T_2 from $\{T_0, T_1\}$, we need to check that the edges of X are compatible with such set T_2. Let e be an arbitrary edge of X. Assume $e = (v_i, v_j)$ with $i < j$. Since $e \notin T_0$, then $e \in B_>(v_i)$. Moreover, $e \notin T_1$ implies that e is not the clockwise first edge of $B_>(v_i)$. Indeed, the clockwise first edge of $B_>(v_i)$ has been assigned to T_1 by Rule 1. Therefore, $e \in T_2$ fulfills the edge-orientation rule of realizers.

It follows that $S = (T_0, T_1, T_2)$ is a realizer of the plane graph $H' \cup T_1 \cup T_2$. We have seen that $E(T_0) \subseteq E(G) \subseteq E(G')$ and that $\overline{T}_0$ is well-orderly in H'. By the assignment rules, we remark that if the edge $(v, p1(v)) \notin E(G')$ (Rule 2 or 3), then v cannot have any child in T_2 (the edge $(v, p_1(v))$ forms a triangle with some T_0 and T_1 edges). In other words, for every inner node v of T_2, $(v, p_1(v)) \in E(G')$ (actually $(v, p_1(v)) \in E(G)$). Thus S is a super-triangulation for G'.

It remains to show that S is the unique super-triangulation of G' that preserve H' and with r_i root of T_i. As T_0 is unique, and as, given $\{T_0, T_1\}$, T_2 is unique, it suffices to prove that T_1 is unique.

The clockwise first edge of $B_>(v_i)$ must belong to T_1 since the parent of v_i in T_1 must be clockwise before the edges of T_2 entering in v_i. If $B_>(v_i) = \varnothing$, and if v_i has a sibling v_j immediately clockwise after v_i around $p_0(v_i)$, then (v_i, v_j) must be in T_1. Otherwise, (v_i, v_j) would be in T_2 and $v_i = p_2(v_j)$ would create an inner node of T_2 in v_i. However, as $B_>(v_i) = \varnothing$, $(v_i, p_1(v_i)) \notin E(G')$, contradicting the super-triangulation definition. Finally, if $B_>(v_i) = \varnothing$, and if v_i is the clockwise last child of $p_0(v_i)$, then $(v_i, p_1(p_0(v_i)))$ must be in T_1. Otherwise, $(v_i, p_1(p_0(v_i)))$ would be in T_2 and $v_i = p_2(p_1(p_0(v_i)))$ would create an inner node of T_2 in v_i. This contradicts the super-triangulation definition since, as $B_>(v_i) = \varnothing$, $(v_i, p_1(v_i)) \notin E(G')$.

Therefore T_1 is unique, and S is unique, completing the proof. $\square$

3 Encoding a Planar Graph with a Super-Triangulation

We start this section with some useful properties for coding super-triangulations.

3.1 Properties of super-triangulations

Straightforward from the equivalence between realizers and orderly trees of triangulations, we have the following basic property:

Property 1 *Let (T_0, T_1, T_2) be any realizer, and let $v_1, \ldots, v_n$ be the clockwise preordering of the nodes of $\overline{T}_0$. For every v_j with $v_i = p_2(v_j)$ and $v_k = p_1(v_j)$, then $i < j < k$ and neither v_i nor v_k is related with v_j in $\overline{T}_0$.*

A *cw-triangle*, is a triple a nodes (u, v, w) of a realizer such that $p_2(u) = v$, $p_1(v) = w$, and $p_0(w) = u$. In the realizer depicted in Fig. 2(a), (u, v, w) forms a cw-triangle, whereas the realizer of Fig. 2(b) has no cw-triangle.

Let $v_1, \ldots, v_n$ be the clockwise preordering of the nodes of T. The subsequence $v_i, \ldots, v_j$ is a *branch* of T if it is a chain (i.e., v_t is the parent of v_{t+1} for every $i \leqslant t < j$), and if $j - i$ is maximal. Branches partition the nodes of T, and there is exactly one branch per leaf.

The tree $\overline{T}_0$ of a realizer (T_0, T_1, T_2) has the *branch property* if for all nodes v_j and $v_i = p_0(v_j)$, either $p_2(v_j) = p_2(v_i)$, or $v_k = p_2(v_j)$ with $i < k < j$ (that is $p_2(v_j)$ is a descendant of v_i clockwise before v_j in $\overline{T}_0$). An important feature of the branch property is that all the nodes of a given branch of T_0 (maybe except the root of T_0) must have the same parent in T_2. Indeed, v_j and $v_i = p_0(v_j)$ belong to the same branch implies that $j = i + 1$, and thus, because there is no index k such that $i < k < j$, $p_2(v_j) = p_2(v_i)$ must hold.

Property 2 *Let $S = (T_0, T_1, T_2)$ be any realizer. The following statements are equivalent:*

1. *S is a super-triangulation for some graph G.*
2. *S has no cw-triangle.*
3. *$\overline{T}_i$ is well-orderly in S, for every $i \in \{0, 1, 2\}$.*
4. *$\overline{T}_i$ has the branch property in S, for every $i \in \{0, 1, 2\}$.*

Proof. Let $3'$ (resp. $4'$) denote the property 3 (resp. 4) for $\overline{T}_0$ only. To prove $1 \Leftrightarrow 2 \Leftrightarrow 3 \Leftrightarrow 4$, once can restrict our attention to $1 \Leftrightarrow 2 \Leftrightarrow 3' \Leftrightarrow 4'$ since "S has/has no cw-triangle" is a property stable by cyclic permutation of the trees. Thus if $2 \Leftrightarrow 3'$ then $2 \Leftrightarrow 3$, and similarly for $4'$.

By definition of a super-triangulation, $1 \Rightarrow 3'$. Moreover, if S is a realizer with $\overline{T}_0$ well-orderly, then the realizer S is a super-triangulation of the underlying triangulation S (the four conditions trivially hold). Hence, $1 \Leftrightarrow 3'$.

To prove that $3' \Rightarrow 2$, assume that S has a cw-triangle (u, v, w), and that $\overline{T}_0$ is well-orderly in S. As $w = p_1(v)$, then the nearest common ancestor between v and w in $\overline{T}_0$ is u. It follows that $v = p_2(u)$ is a descendant of u in $\overline{T}_0$, contradicting Property 1. So $3' \Rightarrow 2$.

To prove that $4' \Rightarrow 2$, assume that S has a cw-triangle (u, v, w), and that $\overline{T}_0$ has the branch property. $u = p_0(w)$ and thus either $p_2(u) = p_2(w)$ or $p_2(w)$ is a descendant of u contained inside the region bounded by the cw-triangle (u, v, w). Clearly, $p_2(u) = p_2(w)$ is impossible, and by the edge-ordering rule of realizers $p_2(w)$ is outside the region bounded by the cw-triangle (u, v, w). So $4' \Rightarrow 2$.

Let $v_1, \ldots, v_n$ be the clockwise preordering of the nodes of $\overline{T}_0$. To prove that $4' \Rightarrow 3$, assume that $\overline{T}_0$ is orderly, but not well-orderly in S. Thus there is an edge (v_p, v_j) with $v_j = p_1(v_p)$ such that the nearest common ancestor between v_p and v_j in $\overline{T}_0$, say v_t, is not $v_i = p_0(v_j)$. Consider the cycle C formed by the path in $\overline{T}_0$ between v_p and v_j, and closed by the edge of (v_p, v_j). Let B be the bounded connected region of $\mathbb{R}^2 \setminus C$. Let us calculate $p_2(v_i)$ and $p_2(v_j)$. By the edge-ordering rule of realizers: 1) $p_2(v_j) \notin B$; 2) $p_2(v_j)$ cannot belong to the path from v_p to v_t in $\overline{T}_0$ with the edge $(v_j, p_2(v_j))$ lying outside B; and 3) $p_2(v_j)$ cannot belong to the path from v_j to v_t in $\overline{T}_0$ since $p_2(v_j)$ is never a descendant of v_j in $\overline{T}_0$ (Property 1). Thus $p_2(v_j) \notin B \cup C$. Again, from the edge-ordering rule of realizers, $p_2(v_i)$ must belong to $B \cup C$. It follows that $p_2(v_i) = p_2(v_j)$ or $v_k = p_2(v_j)$ descendant of v_i such that $i < k < j$ is impossible, contradicting the branch property. So $4' \Rightarrow 3$.

It remains to show that $2 \Rightarrow 4'$. We assume that S has no cw-triangle. Let v_j be any node, and let $v_i = p_0(v_j)$ and let $v_l = p_2(v_i)$. We assume that $k < l$ ($\overline{T}_0$ has not the branch property), and we will show a contradiction.

Let $v_h = p_0(v_l)$, and let P be the path in $\overline{T}_0$ between v_h and v_k. Note that $v_l \notin P$ because v_l is not an ancestor of v_k ($k < l$). Consider C the cycle obtained by traveling v_i, v_l, v_h, P, v_k, v_j, v_i, and let B be the bounded connected region of $\mathbb{R}^2 \setminus C$. Let Q_1 be the path in $\overline{T}_1$ from v_l to r_1, the root of $\overline{T}_1$. By the edge-ordering rule of realizers, the first edge of Q_1 must belong to $B \cup C$. Because $r_1 \notin B$, Q_1 must intersect C. By Property 1, the intersection must be in some v_t with $t > l$. This intersection cannot be in v_i from the edge-ordering rule of realizers. Since the nodes of P are v_t's with $t \leqslant \max\{h, k\} < l$, we have also that $P \cap Q_1 = \varnothing$. It follows that Q_1 intersect C in v_j.

Let C_1 be the cycle composed of Q_1, v_j, v_i, v_l, and let B_1 be the bounded connected region of $\mathbb{R}^2 \setminus C_1$. Let us show that $B_1 \cup C_1$ contains a cw-triangle. Either C_1 is a cw-triangle and we are done with a contradiction, or consider the path Q_2 in $\overline{T}_0$ from $p_1(v_l)$ to r_0 (the root of $\overline{T}_0$). Similarly, Q_2 must intersect C_1 in v_i. The cycle C_2 traveling v_i, v_l, $p_1(v_l)$, Q_2 defines a connected region B_2 of $\mathbb{R}^2 \setminus C_2$ with a number of faces at least one lower than the number of faces of B_1. Proceeding as previously, we can construct either a cw-triangle, or a smaller region with the same inductive property. As the number of faces is finite, it follows that S has a cw-triangle: a contradiction completing the proof of $2 \Rightarrow 4'$, and thus of Property 2. $\square$

Property 3 *A tree of a realizer is uniquely determined by given the two others. Moreover, given the embedding of the two trees, it takes a linear time to construct the third one.*

Proof. By cyclic permutation of the trees, it suffices to prove that the pair $\{T_0, T_1\}$ uniquely determines T_2. Consider the plane graph H composed of $T_0 \cup T_1$. Constructing T_2 consists to triangulate all the faces of H, except the outerface. Let F be any face of H, and let B be its boundary. We assume that F is not the outerface, and that $T_2 \cap F$ contains at least one edge, i.e., that B has at least four nodes.

Once F has been triangulated, B contains only three types of nodes depending on whether the directed edges of $T_2 \cap F$ are leaving, entering, or whether no edge of $T_2 \cap F$ is incident with them. Indeed, a node of B having leaving and entering edges of $T_2 \cap F$ would contradict the edge-ordering rule of realizers. Let us call a *sink* a node of B having at least one entering edge of $T_2 \cap F$. Observe that B contains exactly one sink: zero sink is clearly impossible ($T_2 \cap F \neq \varnothing$), and two or more sinks would provide a node with leaving and entering edges of $T_2 \cap F$, contradicting the edge-ordering rule of realizers. Clearly, once the sink has been located on B, there is only one way to triangulate F. It remains to show that there is only one place on B for a sink.

Traveling B clockwise around F, the two adjacent edges of B at every node v form some cases named X_iY_j, where $X, Y \in \{E, L\}$ and $i, j \in \{0, 1\}$, with the following interpretation: v is in the Case E_0L_1 if the edge clockwise before v in B is entering (E) in v and belongs to T_0, and if the edge clockwise after v in B is leaving (L) v and belongs to T_1. The other cases are defined similarly.

Let us show that v is a sink if and only if v is in the Case L_0L_1. If v is a sink, then two adjacent edges of B with v must be leaving, and L_0L_1 is the only possibility satisfying the edge-orientation rule of realizers. Now, assume that v occurs in the Case L_0L_1. Then, $T_2 \cap F$ has no edge leaving v. If v has no entering and leaving edges in $T_2 \cap F$, then a node $u \in B$ adjacent to v must be a sink. The node u is therefore in the Case E_1X_i (if u is clockwise after v in B), or in the Case Y_jE_0. However, we have already seen that a sink must be in the Case L_0L_1: a contradiction. Thus v is a sink.

It follows that the place of the unique sink on B is entirely determined by the edge pattern of B induced by the pair $\{T_0, T_1\}$. Clearly, triangulate all the faces (by determining the sinks) takes a linear time, completing the proof. $\square$

Property 4 *Let (T_0, T_1, T_2) be a realizer, and let F be any subset of edges of T_2. Let $v_1, \ldots, v_n$ be the clockwise preordering of the nodes of $\overline{T}_0$, let L be the set of the v_i's having a leaving* *edge in F, and let $D = (d_2, d_3, \ldots, d_{n-1})$ be the sequence such that d_i is the number of edges of F entering in v_i. Then, given (T_0, L, D) one can compute F in linear time.*

Proof. Observe that v_1 and v_n have no entering edges of T_2. All the edges of F are of the form $(v, p_2(v))$ with $v \in L$. Let $v_i \in L$ with i minimum, and let $v_j = p_2(v_i)$. Let us show that v_j has an entering edge of F, v_j is unrelated with v_i with, and that j is the largest index $< i$.

Observe that if such a property holds, this give us a recursive procedure to construct F from (T_0, L, D): For all $v_i \in L$, with increasing i do: 1) scan the largest $j < i$ such that $d_j > 0$ and such that v_j is unrelated with v_i; and 2) remove (v_i, v_j) from F, and update d_j and L. So, let us show that the property holds.

By Property 1, v_j must be unrelated with v_i and $j < i$. Assume that j is not maximal. So there exists an edge $(v_t, v_k) \in F$, $v_k = p_2(v_t)$, v_t unrelated with v_i, and $j < t < i$. Let C be the cycle composed of the path from v_j to v_i in $\overline{T}_0$, and closed by the edge (v_i, v_j). Let R be the bounded connected region of $\mathbb{R}^2 \setminus C$. As $j < t < i$, $v_t \in R \cup C$. From the edge-orientation rule of realizers, any edge of T_2 entering in v_t belongs to R, in particular (v_t, v_k). Because $v_k \in L$ and i is minimal such that $v_i \in L$, we must have $k > i$. It follows that $v_k \notin R \cup C$, contradicting the planarity. Therefore j is maximal, completing the proof. □

3.2 Representation of planar graphs with binary strings

Along this section, we consider $S = (T_0, T_1, T_2)$ be any super-triangulation of G, a connected planar graph with n nodes and m edges. To show how to use S to efficiently represent G, we define two sets of edges. The *relevant-edge* set is:

$$R_S \ := \ \{(v, p_1(v)) \mid v \text{ leaf of } T_2\}$$

The *remaining-edge* set is the set M_S of edges of G that are neither in T_0, nor defined by Rule 4 of Definition 1. More formally:

$$M_S \ := \ E(G) \setminus (E(T_0) \cup \{(v, p_1(v)) \mid v \text{ inner node of } T_2\}) \ .$$

Theorem 3 *Let $S = (T_0, T_1, T_2)$ be any super-triangulation of a graph G.*

1. *Given T_2 and R_S, one can determine S in linear time.*
2. *Given S and M_S, one can determine G in linear time.*

Proof. By the branch property of T_2 (Property 2(4)), it suffices to determine $p_1(v)$ for each leaf of T_2 to construct $p_1(w)$ for every non-root node w of T_2 (recall that branches partition the nodes of T_2). So, T_2 and the set of relevant-edges are enough to construct T_1 in linear time.

Now, given T_1 and T_2, the edges of T_0 can be uniquely constructed thanks to Property 3. Thus S can be reconstructed in linear time given T_2 and R_S proving the first claim.

From Definition 1, to entirely determine G given S and M_S, it suffices to: 1) set the edges of T_0; 2) set the edges $(v, p_1(v))$ such that v is an inner node of T_2; 3) set the edges of the remaining-edge set, M_S. This can be done in linear time proving the second claim. $\square$

Let $v_1, \ldots, v_n$ be the clockwise preordering of the nodes of a tree T. A leaf v_i of T is a *bud* if the parent of v_i and of v_{i+1} is v_{i-1} (cf. Fig. 5). Informally, a bud is a leaf that is clockwise a first child and having at least a sibling. Observe that for every bud v_i, v_{i+1} belongs to a branch ending with some leaf v_j, $j > i$. So a tree has at least one leaf more than buds. Buds allow a compact encoding of the relevant-edge leaving the leaf located just after a bud v_i (the node v_j in Fig. 5). Indeed, as we will see in more details in Lemma 5 how to save bits, if $T = T_0$ and has the branch property, then $p_2(v_j) \in \{v_i, p_2(v_{i-1})\}$.

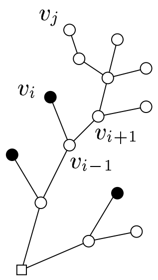

Figure 5: A rooted plane tree with 8 leaves and 3 buds. The Eulerian string of this tree is (in bold the 4-bit subsequence corresponding to the bud v_i): 110 **1101** 11100101001000011010O.

Consider the Eulerian tour $e_1, \ldots, e_{2(n-1)}$ of edges of T obtained clockwise around the outerface, and starting from the root. The *Eulerian string* of T is the binary string such that the ith bit is 1 if and only if $e_i = (u, v)$ with u parent of v (see Fig. 5 for an example). Observe that T has a bud for every 1101-sequence of the Eulerian string. A 1101101-sequence corresponds to two consecutive leaves that are buds.

Let S be a binary string. We denote by $\#S$ the number of binary strings having the same length and the same number of ones than S. More precisely, if S is of length x and has y ones, then we set $\#S := \binom{x}{y}$. Along this section, we assume that T_2 has $n-2$ nodes (i.e., $\overline{T}_2$ has n nodes), has ℓ leaves and b buds.

Lemma 4 *Knowing S and m, the set M_S can be coded by a binary strings C_1 such that: $\#C_1 = \binom{n+\ell}{3n-m-6}$. Moreover, knowing S and m, coding and decoding M_S take a linear time.*

Proof. M_S is composed of some edges of the outerface of S, of some edges of T_2 and of some edges of the form $(v, p_1(v))$ with v leaf of T_2. The string C_1 is constructed as follows. The first three bits represent the edges of the outerface of S (setting to 1 if the edge belongs to G). The next $n-3$ bits represent the edges of T_2 that are in M_S (recall that T_2 has $n-2$ nodes). For instance, perform a preordering of T_2, and set to 1 the ith next bit if and only if the ith encountered edge of T_2 is in M_S. The ℓ next bits of C_1 represent the edges $(v, p_1(v))$ with v leaf of T_2 that are in M_S. (For instance, performing a preordering of T_2 and set to 1 the ith next bit if and only if the ith encountered leaf v of T_2 has an edge $(v, p_1(v))$ in M_S.) C_1 can be coded and decode in linear time.

The length of C_1 is $n + \ell$, and the number of ones is $|M_S|$. We have, $|M_S| = |E(G)| - |E(T_0)| - i$, where $i = |V(T_2)| - \ell - 1 = n - \ell - 3$ is the number of inner nodes of T_2. Thus

$|M_S| = m - 2n + \ell + 6$. Note that, as S is known, n and ℓ are known. $\square$

Lemma 5 *Knowing T_2, the set R_S can be coded by two binary strings B_1, B_2, and by an integer $t \in [0, b]$, such that: $\#B_1 = \binom{b}{t}$, and $\#B_2 = \binom{n-t+\ell-b-3}{\ell-b-1}$. Moreover, knowing T_2, coding and decoding R_S take a linear time.*

Proof. We take into account the buds of T_2 as follows. Let $v_1, \ldots, v_n$ be the clockwise preordering of the nodes of $\overline{T}_2$. For every bud v_i, let us denote by $l(v_i)$ the leaf of the branch containing v_{i+1} (on Fig. 5, $l(v_i)$ is the node named v_j), and let $B = \{l(v_i) \mid v_i \text{ bud of } T_2\}$. Since $\overline{T}_2$ has the branch property (Property 2), either $p_1(l(v_i)) = p_1(v_{i-1})$, or $p_1(l(v_i))$ is a descendant of v_{i-1} clockwise before v_{i+1}. This descendant is unique and is the bud v_i. So for every bud v_i, either $p_1(l(v_i)) = p_1(v_{i-1})$ or $p_1(l(v_i)) = v_i$.

Let $A = \{(v, p_1(v)) \mid v \in B\} \subset R_S$. We represent differently the edges of A and of $R_S \setminus A$. Let t be the number of buds v_i such that $p_1(l(v_i)) = p_1(v_{i-1})$. The binary string B_1 is of length b, and is defined as follows: the jth bit of B_1 is set to 1 if and only if for the jth bud v_i of T_2 the case $p_1(l(v_i)) = p_1(v_{i-1})$ occurs. Clearly, $\#B_1 = \binom{b}{t}$. As there are only two case for $p_1(l(v_i))$, one can fully reconstruct R_S knowing T_2, $R_S \setminus A$, and B_1.

To represent the edges of $R_S \setminus A$, we consider Property 4 applied on the realizer (T_2, T_0, T_1) and for $F = R_S \setminus A$. F can be determined in linear time from the triple (T_2, L, D) where L is the set of leaves of T_2 that are not in B, and $D = d_2, \ldots, d_{n-1}$ is the degree sequence of entering edges of F. As T_2 is known, L can be computed, and only D needs to be represented to reconstruct $F = R_S \setminus A$. We remark that for every bud v_i such that $p_1(l(v_i)) = p_1(v_{i-1})$, v_i has no entering edge of F, i.e., $d_i = 0$. Such cases can be detected by knowing T_2 and B_1. So it suffices to represent D in which we have removed all the cases d_i for which $p_1(l(v_i)) = p_1(v_{i-1})$. The resulting subsequence D' is composed of $n-2-t$ integers (eventually with null integers), whose sum is $|F| = |R_S \setminus A| = |R_S| - |B| = \ell - b$.

Any sequence of $k \geqslant 1$ integers (eventually of null integers) whose sum is $s \geqslant 1$ can be represented (in $O(s+k)$ time) by a binary string of length $s+k-1$ and with $k-1$ ones: each element corresponding to the number of zeros delimited by two ones, or by the extremities of the string. E.g., here is an example with $k = 8$ and $s = 9$.

$$\begin{array}{c} 2,1,0,1,1,0,1,3 \\ 0010110101101000 \end{array}$$

Similarly, a sequence of k non-null integers whose sum is s can be represented by a binary string of length s and with $k-1$ ones. It follows that D' can be represented by a binary string B_2 such that $\#B_2 = \binom{n-t+\ell-b-3}{\ell-b-1}$. Thus, knowing T_2, R_S can be represented by B_1 and B_2, completing the proof. $\square$

Lemma 6 *Knowing n, ℓ, b, the tree T_2 can be coded by five binary strings $A_1, \ldots, A_5$, and by some integers $p \in [1, \ell]$ and $w \in [b, n-\ell]$, such that: $\#A_1 = \binom{\ell-1}{p-1}$, $\#A_2 = \binom{p}{b}$, $\#A_3 = \binom{w}{b-1}$, $\#A_4 = \binom{n-\ell-w+p-b-3}{p-b-1}$, and $\#A_5 = \binom{n-\ell-2}{p-b-1}$. Moreover, knowing n, ℓ, b, coding and decoding T_2 take a linear time.*

Proof. Let S be the Eulerian string of T_2 in which 4 bits have been added: a 01-sequence at the beginning and at the end of S. The length of S is $2n-2$ (recall that T_2 has $n-2$ nodes), and has $n-1$ zeros and ones. S contains exactly ℓ 10-sequences (the Eulerian string starts with a one and ends with a zero, and a leaf in encountered at each 10-sequence), and has $\ell+1$ 01-sequences (the Eulerian string contains $\ell-1$ 01-sequences). The number of 1101-sequences in S (potentially with coverings) is b, the number of buds.

Let X_p denote the set of binary strings of length $2n-2$, and of the form $(U_i V_i)^p U_{p+1}$ where U_i's are maximal blocks of 01-sequences, and with $U_1 = U_{p+1} = 01$. For instance (U_i's are boxed):

$$W = \boxed{01}\, V_1 \boxed{010101}\, V_2 \boxed{01\cdots}\, \cdots V_p \boxed{01}, \quad W \in X_p$$

Note that $S \in X_p$ for some parameter $p \geqslant 1$. The strings $A_1, \ldots, A_5$ are used to represent the members of X_p, thus in particular S and the Eulerian string of T_2.

Let $u_i \geqslant 1$ be the integer such that $U_i = (01)^{u_i}$, for every $i \in \{1, \ldots, p+1\}$. The sequence of non-null integers $u_2, \ldots, u_p$ (recall that $u_1 = u_{p+1} = 1$) can be used to represent all the U_i blocks. This sequence can be described by a binary string A_1 such that $\#A_1 = \binom{s}{p-1}$ with $s = \sum_{i=2}^{p} u_i$. Because all the 01-sequences of S are located in the U_i's, $s = (\ell+1)-2$. Thus $\#A_1 = \binom{\ell-1}{p-1}$. It remains to describe the V_i blocks.

Only two cases occur for the V_i's. Either $V_i = 1^+$ consists of ones only, or $V_i = 1^*0^+$ ends by at least a zero (obviously a V_i cannot contain any 01-sequence). Case 1 ($V_i = 1^+$) provides, with the adjacent blocks U_i and U_{i+1}, exactly one bud (a 1101-sequence), and provides exactly one 10-sequence (between the V_i and U_{i+1} blocks). Case 2 ($V_i = 1^*0^+$) provides only one 10-sequence.

A binary string A_2 indicates whether V_i occurs for Case 1 or 2. As Case 1 occurs exactly b times, $\#A_2 = \binom{p}{b}$. Let w be the number of ones used by the V_i's occurring in Case 1. The length of V_i's occurring in Case 1 can be described by a sequence of b non-null integers whose sum is w precisely. So, a binary string A_3 such that $\#A_3 = \binom{w}{b-1}$ suffices to represent this sequence.

To complete the description of a member of X_p, it remains to defined V_i's occurring in Case 2. There are $p-b$ blocks in Case 2, and there are on the form 1^*0^+. It suffices to describe the number of ones and of zeros of each V_i's in Case 2. The number of ones remaining for these blocks is $(n-1)-(\ell+1)-w$. Case 2 occurs $p-b$ times, so the sequence of one length (possibly with null lengths) can be represented by a binary string A_4 such that $\#A_4 = \binom{(n-1)-(\ell+1)-w+p-b-1}{p-b-1}$. The zero length sequence can be represented by a binary string A_5 such that $\#A_5 = \binom{n-\ell-2}{p-b-1}$ as the number of zeros for Case 2 is $(n-1)-(\ell+1)$ (there are $\ell+1$ zeros used for U_i's).

This binary string representation is clearly linear time constructible. $\square$

Lemma 7 *Every binary string S of length n can be coded into a binary string of length $\log_2(\#S) + O(n \log\log n / \log n)$. Moreover, knowing n, coding and decoding S can be done in linear time and linear space, assuming a* RAM *model of computation on words of $\omega \geqslant \log_2 n$ bits.*

Proof. The main idea is to split S into blocks of equal size b, and to code each block optimally. Each block encoding takes a time exponential in b. However the code of all possible blocks can be tabulated once in time $2^{O(b)} = O(n)$, for suitable b small enough. Optimality of the coding of S' derive from optimality of each block by super-additivity of binomials. More precisely we proceed as follows.

Let $b = \lfloor \log_2 n - \log_2 \log_2 n \rfloor$. So standard arithmetic operations on integers in the range $[0, 2^b)$ can be done in constant time since $\omega \geqslant b$, by loading each integer into a word of length ω and padded with left extra zeros. Let k_p denote $\binom{b}{p}$. We need the construction of some tables.

We first construct a table L such that for every $p \in [0, b]$, $L[p] = \lceil \log_2 k_p \rceil$. All the numbers $k_0, \ldots, k_b$ can be computed dynamically with $O(b^2)$ b-bit numbers (Pascal's method). In total, the construction of L costs $O(b^2 + \sum_p \log \log k_p) = O(\log^2 n)$ of time, where $O(\log \log k_p)$ is the cost to compute $\lceil \log k_p \rceil$ from the binary representation of k_p (using a binary search and some masks).

We construct a table P of integers in the range $[0, b]$ such that for every $i \in [0, 2^b)$, $P[i]$ is the number of ones in the binary representation of i. The table P can be constructed in time and space $O(b2^b) = O(n)$. However, the time can be reduced to $O(b2^{b/2}) = O(\sqrt{n \log n})$ (and even smaller) using a table P' for half-words of $\lceil b/2 \rceil$ bits. Indeed, we have $P[i] = P'[i/2^{\lfloor b/2 \rfloor}] + P'[i \bmod 2^{\lfloor b/2 \rfloor}]$.

For each $p \in \{0, \ldots, b\}$, we compute a table D_p (used for decoding) such that, for every $i \in [0, k_p)$, $D_p[i]$ is a distinct binary string of length b having p ones. Strings of D_p are lexicographically ordered. Generating all D_p's costs $O(2^b) = O(n/\log n)$ time and $O(b2^b) = O(n)$ space by running all binary strings $s \in [0, 2^b)$ by increasing value, and filling the right entry $D_{P[s]}[i_p]$ (and updating the current index i_p).

Finally, we construct a table C (used for coding) such that for every $s \in [0, 2^b)$, $C[s]$ denotes the index i such that $D_p[i] = s$, where $p = P[s]$. The index $i = C[s]$ is stored on b bits, although only the $L[p] = \lceil \log_2 k_p \rceil$ least significant bits of i are useful since $i \in [0, k_p)$. To construct C is suffices to iterate for all $p \in [0, b]$ and all $i \in [0, k_p)$: $C[D_p[i]] = i$. Once D_p and P have been computed, constructing C costs $O(\sum_{p=0}^{b} k_p) = O(2^b) = O(n/\log n)$ of time and $O(b2^b) = O(n)$ of space.

Let $q = \lfloor n/b \rfloor$ be the number of blocks of b bits in S. If b does not divide n, the last $n \bmod b$ bits will be treated separately. For the coding and decoding procedure we iterate q times the following steps:

1. read from S the next block s of b bits (s can be manipulated as an index of $[0, 2^b)$);
2. write in S' the value $P[s]$ as a binary number on $\lceil \log_2 b \rceil$ bits; and
3. write in S' the string composed of the $L[s]$ most significant bits of $C[s]$.

We end the coding process by writing in S' the $n \bmod b$ remaining bits of S (if any).

The decoding procedure is:

1. read from S' the $\lceil \log_2 b \rceil$ bits to form the value p;
2. read from S' the next $L[p]$ bits, representing an integer $i \in [0, k_p)$; and
3. write in S the string $D_p[i]$.

We end the decoding process by writing in S the $n \bmod b$ remaining bits of S' (if any).

Coding and decoding procedures clearly take $O(q) = O(n/\log n)$ time, once the tables L, P, D_p, C have been generated. The correctness of the coding and decoding are clear from symmetry of the above procedures.

It remains to show that the length of S' does not exceed $\log_2(\#S) + O(n \log\log n/\log n)$. Let p_i be the number of ones in the ith b-bit block of S, for $i \in \{1, \ldots, q\}$. From the coding procedure, the number of bits written in S' for the ith block is: $\lceil \log_2 b \rceil + \lceil \log_2(k_{p_i}) \rceil$. Summing over all the blocks, we obtain the following upper bound for the length of S':

$$\sum_{i=1}^{q} (\lceil \log_2 b \rceil + \lceil \log_2(k_{p_i}) \rceil) + (n \bmod b) \;=\; \left(\sum_{i=1}^{q} \log_2(k_{p_i}) \right) + O(b + q \log b) \;.$$

Observe that by super-additivity $\binom{a}{b} \cdot \binom{a'}{b'} \leqslant \binom{a+a'}{b+b'}$, so

$$\prod_{i=1}^{q} k_{p_i} \;=\; \prod_{i=1}^{q} \binom{b}{p_i} \;\leqslant\; \binom{bq}{\sum_i p_i} \;=\; \#\overline{S},$$

where $\overline{S}$ is the string composed of the first bq bits of S. Since the length and the number of ones between S and $\overline{S}$ differ by at most b, it follows that $|\log_2(\#S) - \log_2(\#\overline{S})| = O(b \log n)$. Therefore, we have that the length of S' is no more than:

$$\left(\log_2 \prod_{i=1}^{q} k_{p_i} \right) + O(b + q \log b) \;\leqslant\; \log_2(\#S) + O(n \log\log n/\log n)$$

as claimed, completing the proof. □

4 Entropy Analysis

4.1 The number of unlabeled planar graphs

This subsection concerns the proof of the following two results.

Theorem 4 *Triangulations and connected planar graphs on n nodes can be represented by a binary string of length at most $3.37n$ and $5.03n$ respectively. Moreover, coding and decoding can be done in linear time.*

Theorem 5 *The number $p(n)$ of unlabeled planar graphs on n nodes satisfies, for every n large enough:*

$$\beta n - \Theta(\log n) \;\leqslant\; \log_2 p(n) \;\leqslant\; \alpha n + O(\log n)$$

with the constants $\alpha \approx 5.007$ and $\beta \approx 4.710$.

The lower bound in Theorem 5 results from the number $g(n)$ of labeled 2-connected planar graphs. Clearly, $p(n) \geqslant g(n)/n!$. The following asymptotic is proved in [BGW99]:

$$g(n) \;\sim\; \Theta(n^{-7/2})\, c^{-n}\, n!$$

where $c \approx 0.03819$ is a constant. It follows that for every n large enough

$$\log_2 p(n) \geqslant \log_2 \frac{g(n)}{n!} = \beta n - \frac{7}{2}\log_2 n + O(1), \quad \text{with } \beta = -\log_2 c \approx 4.71066.$$

Let us show the upper bound. Let $q(n)$ denote the number of unlabeled connected planar graphs on n nodes. To rely $p(n)$ and $q(n)$, we represent every planar graph G with $k \geqslant 1$ connected components by a triple $(k, \tilde{G}, v)$, where $\tilde{G}$ and v are defined as follows. Let v_i be any non cut-vertex of the ith component of G. (Recall that a *cut-vertex* of a graph is a node whose removal strictly increases the number of connected components. A leaf of a tree being not a cut-vertex, it is clear that every connected graph has a non cut-vertex). We merge all the connected components of G by identifying all the v_i's into a single node v. Clearly $\tilde{G}$ is planar and connected. One can obtained G from $(k, \tilde{G}, v)$ by splitting v in $\tilde{G}$. All the $k' \leqslant k$ connected components obtained by this way are included in G (there is no risk to disconnect a single connected component of G as v_i's are not cut-vertex of each connected components of G). To fully recover G, we may add $k - k'$ isolated nodes because $\tilde{G}$. The number of nodes of $\tilde{G}$ is $n - (k' - k) - k' + 1 = n - k + 1$. From this representation, it turns out

$$p(n) \leqslant \sum_{k=1}^{n} k \cdot q(n-k+1) \cdot (n-k+1) \leqslant n^3 q(n) \tag{1}$$

as $q(n)$ clearly increases with n. So to prove Theorem 5, it remains to prove that, for every n large enough,

$$\log_2 q(n) \leqslant \alpha n + O(\log n), \quad \text{for some constant } \alpha \approx 5.007. \tag{2}$$

We have the well-known upper bound (see for instance [Bol78, p. 255]), for all $x > y > 0$ where $r = y/x$:

$$\binom{x}{y} < \left(\frac{x}{y}\right)^y \left(\frac{x}{x-y}\right)^{x-y} = \left(\frac{x}{rx}\right)^{rx} \left(\frac{x}{x-rx}\right)^{x-rx} = \left(\left(\frac{1}{r}\right)^r \left(\frac{1}{1-r}\right)^{1-r}\right)^x = \left(2^{\mathrm{H}(r)}\right)^x$$

where H is the *entropy function* of r, defined for all $r \in [0,1]$ by:

$$\mathrm{H}(r) := -r\log_2 r - (1-r)\log_2(1-r), \quad \text{and with} \quad \mathrm{H}(0) := \mathrm{H}(1) := 0\ .$$

So that, for all $n, c_1, c_2 > 0$:

$$\frac{1}{n}\log_2 \binom{c_1 n}{c_2 n} \leqslant c_1 \mathrm{H}(c_2/c_1)\ . \tag{3}$$

From Theorem 3, any connected planar graph G with n nodes and m edges can be represented with the eight binary strings, namely $A_1, \ldots, A_5$, B_1, B_2, C_1 described by lemmas 4, 5, 6, and with a constant number of $O(\log n)$ bit integers. Let us define

$$Q := \max_{\ell,b,t,w,p} \log_2 (\#A_1 \cdots \#A_5 \cdot \#B_1 \cdot \#B_2 \cdot \#C_1)$$

where ℓ, b, t, w, p are some integers ranging in $[0, n]$ and defined by lemmas 4, 5, and 6, and let $q(n, m)$ the number of connected planar nodes with n nodes and m edges. From our representation, it turns out

$$\log_2 q(n,m) \leqslant Q + O(\log n) \tag{4}$$

As $q(n) \leqslant \sum_{m=n-1}^{3n-6} q(n,m) \leqslant 2n \cdot \max_m q(n,m)$, to prove Eq. (2) and to compute α, it remains to analyze the value Q. For this purpose, let $\lambda := \ell/n$, and let

$$f(\lambda) := \max_{b,t,w,p} \left\{ \frac{1}{n} \log_2 \left(\#A_1 \cdots \#A_5 \cdot \#B_1 \cdot \#B_2 \right) \right\}$$

Roughly speaking $f(\lambda)n$ represents the number of bits to code the pair (T_2, R_S), that is by Theorem 3, the number of bits to code the super-triangulation S. Note that only n, m and ℓ (i.e., λn) appear in the expression of $\#C_1$. So, by definition of Q we have:

$$Q \;=\; \max_{0 \leqslant \lambda \leqslant 1} \left\{ f(\lambda)n + \max_{n < m \leqslant 3n-6} \log_2(\#C_1) \right\} + O(\log n) \;. \tag{5}$$

Let $\mu := m/n$. Note that $\#C_1 = \binom{n+\ell}{3n-m-6} \leqslant (n+\ell)^6 \binom{n+\ell}{3n-m} \leqslant (2n)^6 \binom{(1+\lambda)n}{(3-\mu)n}$. By Eq. (3), the upper bound on $\#C_1$ with the entropy function gives:

$$\log_2(\#C_1) \;\leqslant\; \left((1+\lambda) \cdot \mathrm{H}\left(\frac{3-\mu}{1+\lambda} \right) \right) n + 6 \log_2 (2n) \;.$$

Hence, plugging in Eq. (5) the latter upper bound on $\#C_1$, we obtain:

$$\frac{1}{n} \left(Q - O(\log n) \right) \;\leqslant\; \max_{\substack{0 \leqslant \lambda \leqslant 1 \\ 1 \leqslant \mu \leqslant 3}} \left\{ f(\lambda) + (1+\lambda) \cdot \mathrm{H}\left(\frac{3-\mu}{1+\lambda} \right) \right\} \;. \tag{6}$$

The study of f can be done by expressing each binomial with the entropy function, given a linear combination of entropy function terms. Derivating and solving the equation is a very heavy calculation. The closed expression of $f(\lambda)$ is quite large. By a formal solver (see [BGH02] for the full sheet of calculations we have used), f has the following behavior:

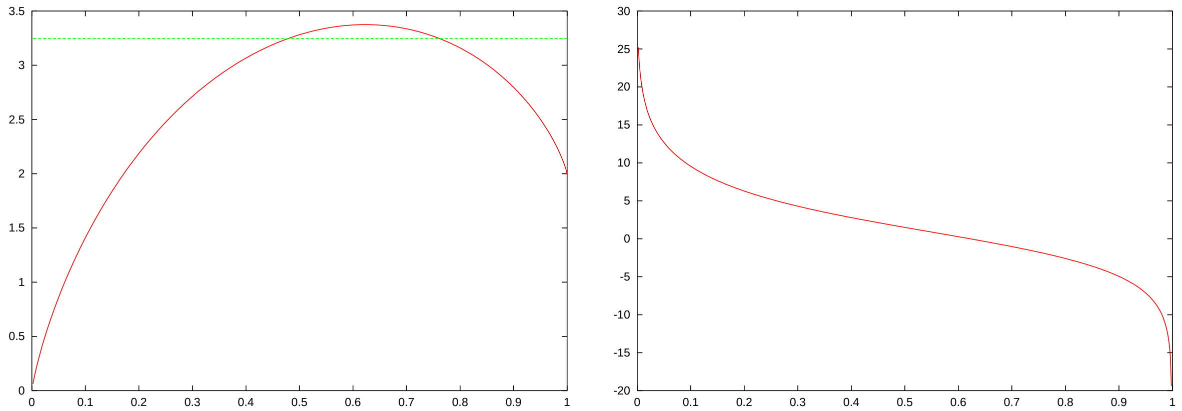

Figure 6: The behavior of $f(\lambda)$ (left side), and of $f'(\lambda)$ (right side).

The maximum of $f(\lambda)$ is reached for $\lambda = \lambda_0$, solution of the following degree-5 equation:

$$21\lambda^5 - 49\lambda^4 + 47\lambda^3 - 24\lambda^2 + 7\lambda - 1 = 0$$

In the range $[0,1]$, we obtain only one solution $\lambda_0 \approx 0.622900$, and $f(\lambda_0) \approx 3.374449$.

Proof of Theorem 4. By Lemma 7, each string appearing in Q can be compressed into an asymptotic entropy optimal number of bits. The graph G can be therefore encoded in linear time with no more than $Q + O(n \log\log n / \log n)$ bits, as the sum of string lengths appearing in Q is $O(n)$.

If G is a triangulation, then $m = 3n - 6$. So, with $\mu = 3$, Eq. (6) rewrites in

$$\frac{1}{n}\left(Q - O(\log n)\right) \;\leqslant\; \max_{0 \leqslant \lambda \leqslant 1} f(\lambda) \;=\; f(\lambda_0) \;\approx\; 3.37$$

proving that triangulations can be represented in linear time with asymptotically $3.37n$ bits as claimed.

The maximum of the upper bound on Q in Eq. (6) is attained for $\mu = \frac{1}{2}(5 - \lambda)$, since $\mathrm{H}(r)$ is maximum for $r = 1/2$ (and we have $\mathrm{H}(1/2) = 1$). So connected planar graphs can be represented in linear time with asymptotically Q bits such that

$$\frac{1}{n}\left(Q - O(\log n)\right) \;\leqslant\; \max_{0 \leqslant \lambda \leqslant 1} \{f(\lambda) + 1 + \lambda\}$$

Calculation shows that we have to solve the following degree-7 equation:

$$275\lambda^7 - 1430\lambda^6 + 2884\lambda^5 - 2986\lambda^4 + 1767\lambda^3 - 614\lambda^2 + 118\lambda - 10 = 0$$

Numerically, there are three possible roots, but only one ranges in $[0, 1]$, namely $\lambda_1 \approx$

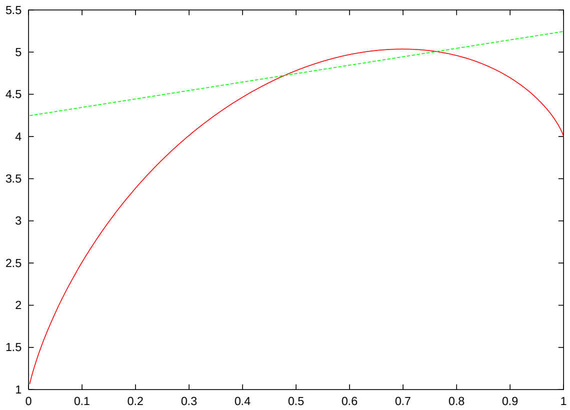

Figure 7: The behavior of $f(\lambda) + 1 + \lambda$ vs. $\lambda \mapsto \log_2(256/27) + 1 + \lambda$.

0.699017. We have $f(\lambda_1) + 1 + \lambda_1 \approx 5.036013$ (cf. Fig. 7). So connected planar graphs can be represented in linear time with $5.03n$ bits as claimed. Note that the graphs attaining the $5.03n$ bit encoding have asymptotically $\frac{1}{2}(5 - \lambda_1)n \approx 2.15n$ edges. $\square$

Proof of Theorem 5. By Theorem 4, we have already a 5.03 upper bound on α. To improve slightly this bound, we observe that triangulations have only one super-triangulation (Theorem 2). Therefore, we have an alternative to represent a super-triangulation: either we use our representation (given by Theorem 3) thanks to the pair (T_2, R_S) (this is can be coded by $f(\lambda)n + O(\log n)$ bits), or we can use a theoretical optimal encoding of the underlying triangulation of S, that is within $\lceil \log_2(T_n) \rceil$ bits if T_n denotes the number of n-node triangulations. Recover the super-triangulation from the triangulation can be done in a unique way (in linear time) by Theorem 2.

By Tutte's enumeration formula [Tut62], for every $n \geqslant 3$,

$$T_n \;=\; \frac{2(4n-11)!}{(n-2)!(3n-7)!} \;=\; \frac{2}{(3n-8)(3n-7)}\binom{4n-11}{n-2} \;\leqslant\; \binom{4n}{n} \;.$$

So that,

$$\frac{1}{n}\log_2(T_n) \;\leqslant\; 4\mathrm{H}(1/4) \;=\; 8 - 3\log_2 3 \;\approx\; 3.245112$$

From the two representations of a triangulation, Eq. (6) rewrites in

$$\frac{1}{n}\left(Q - O(\log n)\right) \;\leqslant\; \max_{\substack{0\leqslant\lambda\leqslant 1\\ 1\leqslant\mu\leqslant 3}} \left\{ \min\left\{f(\lambda), 4\mathrm{H}(1/4)\right\} + (1+\lambda)\cdot \mathrm{H}\left(\frac{3-\mu}{1+\lambda}\right)\right\} \;. \tag{7}$$

Bounding $\mathrm{H}\left(\frac{3-\mu}{1+\lambda}\right)$ by 1, and since $\log_2 q(n,m) \leqslant Q + O(\log n)$, Eq. (7) rewrites in

$$\frac{1}{n}\left(\log_2 q(n) - O(\log n)\right) \;\leqslant\; \max_{0\leqslant\lambda\leqslant 1} \left\{\min\left\{f(\lambda), 4\mathrm{H}(1/4)\right\} + 1 + \lambda\right\} \;.$$

In the range $[0,1]$, the equation $f(\lambda) = 4\mathrm{H}(1/4)$ has two solutions (see Fig. 6). Numerically we have $\lambda_2 \approx 0.478207$ and $\lambda_3 \approx 0.762116$. The values λ_0 and λ_1 maximize respectively $f(\lambda)$ and $f(\lambda)+1+\lambda$, so $f'(\lambda_0) = 0$ and $f'(\lambda_1) = -1$. The functions f and f' decrease for all $\lambda > \lambda_0$ (cf. Fig. 6). Hence, $f'(\lambda_3) < -1$ since $\lambda_3 > \lambda_1$. It means that, for $\lambda > \lambda_3$, the term $f(\lambda)$ decreases more than the term $1+\lambda$ increases. I.e.,

$$\forall\, \lambda > \lambda_3, \quad f(\lambda) + 1 + \lambda \;<\; f(\lambda_3) + 1 + \lambda_3 \;=\; 4\mathrm{H}(1/4) + 1 + \lambda_3 \;.$$

Thus,

$$\max_{0\leqslant\lambda\leqslant 1} \left\{\min\left\{f(\lambda), 4\mathrm{H}(1/4)\right\} + 1 + \lambda\right\} \;=\; 4\mathrm{H}(1/4) + 1 + \lambda_3 \approx 5.007228 \;.$$

Setting $\alpha := 4\mathrm{H}(1/4) + 1 + \lambda_3$, we have therefore shown that $\log_2 q(n) \leqslant \alpha n + O(\log n)$, and also that $\log_2 p(n) \leqslant \alpha n + O(\log n)$ with $\alpha \approx 5.007$. This completes the proof of Theorem 5. $\square$

4.2 The number of edges of random planar graphs

Theorem 6 *Almost all unlabeled and almost all labeled planar graphs on n nodes have at least* $1.70n$ *edges and at most* $2.54n$ *edges. Moreover, the result holds also for unlabeled connected and labeled connected planar graphs.*

Proof. Let $m_1 = 1.70n$, $m_2 = 2.54n$, and let $I = [m_1, m_2]$. Let $p(n,m)$ (resp. $l(n,m)$) be the number of unlabeled (resp. labeled) planar graphs with n nodes and m edges. Finally, let $l(n)$ be the number of labeled planar graphs with n nodes.

We first concentrate our attention on the first claim of Theorem 6. We have to prove that:

$$\lim_{n\to+\infty} \sum_{m\in I} \frac{p(n,m)}{p(n)} \;=\; 1 \quad \text{and} \quad \lim_{n\to+\infty} \sum_{m\in I} \frac{l(n,m)}{l(n)} \;=\; 1$$

As $p(n) = \sum_{m \in I} p(n,m) + \sum_{m \notin I} p(n,m)$, and the same with $l(n)$, we have to prove that

$$\sum_{m \notin I} p(n,m) \; = \; o(p(n)) \quad \text{and} \quad \sum_{m \notin I} l(n,m) \; = \; o(l(n)) \; . \tag{8}$$

To prove Eq. (8), we rely $p(n,m)$ and $l(n,m)$ to $q(n,m)$, number introduced in Eq. (4).

In the representation of a unlabeled planar graph G into the triple $(k, \tilde{G}, v)$ as seen previously (cf. Eq. (1)), we observe that the number of edges is invariant in G and in $\tilde{G}$. Because the number of edges of a connected component of G with η nodes ranges in $[\eta - 1, 3\eta - m]$, the number k of connected components of G is at least $k_1 = \max\{1, n-m\}$, and at most $k_2 = n - \lceil m/3 \rceil - 1$ if $m \geqslant 2$ and $k_2 = n - \lceil m/3 \rceil$ otherwise. Thus Eq. (1) extends in: $\forall n, m \geqslant 0$,

$$p(n,m) \; \leqslant \; \sum_{k=k_1}^{k_2} k \cdot q(n-k+1, m) \cdot (n-k+1) \; \leqslant \; n^2 \sum_{k=k_1}^{k_2} q(n-k+1, m) \; .$$

For labeled planar graphs, we use a representation of G into a quadruple $(k, \tilde{G}, v, L)$, where L is an n-entry array of integers ranging in $[1, n]$. Each entry is a distinct node label of G. The construction is similar to the unlabeled case. All the k connected components of G are merged into a single node v, the nodes merged of each component are chosen to be not a cut-vertex. The graph $\tilde{G}$ is a connected unlabeled planar graph, and has $n-k+1$ nodes and m edges. Let $v_1, \ldots, v_{n-k+1}$ be any canonical ordering of the nodes of $\tilde{G}$. Let $C_1, \ldots, C_{k'}$ be the connected components of G that are distinct from an isolated node. We need to order these components on the base of $\tilde{G}$, without any information on the labels of G. The C_i's are ordered such that $C_i \setminus \{v\}$ contains the node v_z having the lowest index z among the nodes of $\{v_1, \ldots, v_{n-k+1}\} \setminus \bigcup_{j=1}^{i-1} C_{j-1}$, setting by convention $C_0 = \{v\}$. For instance, $C_1 \setminus \{v\}$ must contain the node v_1, or v_2 if $v \neq v_1$. For every $i \in \{1, \ldots, n-k+1\}$, we set $L[i]$ as the label of the node v_i in G, with the convention that the label of $v = v_{i_0}$ is done as v would belong to C_1 in G. Then, for every $i \in \{2, \ldots, k'\}$, $L[n-k+i]$ is the label of v considered as v belongs to C_i in G. Finally, the entries $L[n-k+k'+1], \ldots, L[n]$ are the labels of the isolated nodes of G. The reconstruction of the underlying unlabeled graph of G can be done as previously from the triple $(k, \tilde{G}, v)$. Then, it is not difficult to see that L suffices to correctly assign all the labels of G. From this representation, we have therefore:

$$l(n,m) \; \leqslant \; \sum_{k=k_1}^{k_2} k \cdot q(n-k+1, m) \cdot (n-k+1) \cdot n! \; \leqslant \; n^2 \cdot n! \cdot \sum_{k=k_1}^{k_2} q(n-k+1, m)$$

Now, let us upper bound $q(n,m)$, for every $m \in [n-1, 3n-6] \setminus I$. For convenience, let

$$h(\mu) \; := \; \max_{0 \leqslant \lambda \leqslant 1} \left\{ \min\{f(\lambda), 4\mathrm{H}(1/4)\} + (1+\lambda) \cdot \mathrm{H}\left(\frac{3-\mu}{1+\lambda}\right) \right\} \; .$$

From Eq. (4), $\log_2 q(n,m) \; \leqslant \; Q + O(\log n)$, and from Eq. (7) in Theorem 4 we obtain

$$\log_2 q(n,m) \; \leqslant \; h(m/n) \cdot n + O(\log n)$$

We check that $h(m_1/n)$ and $h(m_2/n)$ are some constant strictly lower than $\beta \approx 4.71066$, the constant defined in Theorem 5 (see Fig. 8). Moreover, for every $m \in [n-1, m_1)$, $h(m/n) \leqslant h(m_1/n)$ and for every $m \in (m_2, 3n-6]$, $h(m/n) \leqslant h(m_2/n)$.

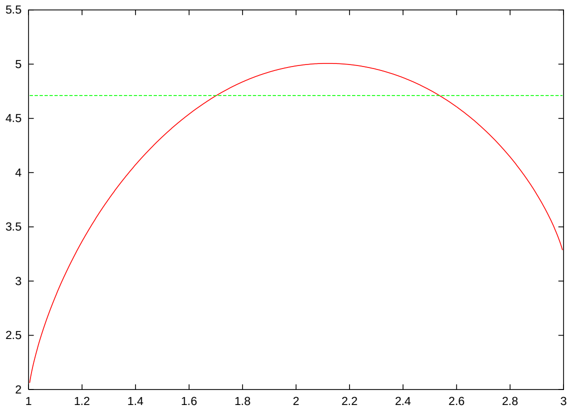

Figure 8: Behavior of $h(\mu)$ vs. $\lambda \mapsto 4.71066$.

Thus,

$$\forall\, m \in [n-1, 3n-6] \setminus I, \quad q(n,m) \;\leqslant\; 2^{(\beta-\delta)n} \;\leqslant\; \frac{g(n)}{n!} \cdot 2^{-\delta n}$$

where δ is a constant such that $0 < \delta < \beta$. Observe that for $n' = n - k + 1$ and $k \in [k_1, k_2]$, $m \in [n'-1, 3n'-6]$. Since the function $(g(n)/n!) \cdot 2^{-\delta n}$ increases with n, and since $k_2 - k_1 \leqslant 3n$, we have:

$$\sum_{k=k_1}^{k_2} q(n-k+1, m) \;\leqslant\; 3n \cdot \frac{g(n-k+1)}{(n-k+1)!} \cdot 2^{-\delta(n-k+1)} \;\leqslant\; 3n \cdot \frac{g(n)}{n!} \cdot 2^{-\delta n} \ .$$

Therefore, using the facts that $|I| \geqslant 2n$, and that $p(n) \geqslant g(n)/n!$,

$$\begin{aligned} \sum_{m \notin I} p(n,m) \;&\leqslant\; \sum_{m \notin I} \left(n^2 \sum_{k=k_1}^{k_2} q(n-k+1, m) \right) \;\leqslant\; \sum_{m \notin I} 3n^3 \cdot \frac{g(n)}{n!} \cdot 2^{-\delta n} \\ &\leqslant\; 6n^4 \cdot \frac{g(n)}{n!} \cdot 2^{-\delta n} \;=\; o\left(\frac{g(n)}{n!}\right) \;=\; o(p(n)) \ . \end{aligned}$$

Similarly, because $l(n) \geqslant g(n)$,

$$\sum_{m \notin I} l(n,m) \;\leqslant\; 6n^4 \cdot g(n) \cdot 2^{-\delta n} \;=\; o(g(n)) \;=\; o(l(n)) \ .$$

Hence we have proved Eq. (8) as required.

Obviously, the result extends to every set $A_{n,m}$ (resp. $B_{n,m}$) of unlabeled (resp. labeled) planar graphs with n nodes and m edges that fulfills:

$$\sum_{m \geqslant 0} |A_{n,m}| \;=\; \Omega\left(\frac{g(n)}{n!}\right) \quad \text{or} \quad \sum_{m \geqslant 0} |B_{n,m}| \;\geqslant\; \Omega\left(g(n)\right) \ .$$

Indeed, we have similarly,

$$\sum_{m \notin I} |A_{n,m}| \;\leqslant\; \sum_{m \notin I} p(n,m) \;=\; o\left(\frac{g(n)}{n!}\right) \;=\; o\left(\sum_{m \geqslant 0} |A_{n,m}|\right)$$

and similarly for the $B_{n,m}$. In particular this holds for connected unlabeled and connected labeled planar graphs. $\square$

5 Conclusion

An interesting problem left open in this paper is to know if the approach consisting in representing a planar graph by a suitable triangulation and a set of edges to delete would lead to an optimal information bound.

We think that our upper bound could be improved, still using a triangulation approach, because we have proposed a $3.37n$ bit encoding for triangulations, whereas $3.24n$ is possible. An hypothetic gain of $0.13n$ bits should be possible, leading to a $(5.00 - 0.13)n = 4.87n$ bit upper bound for general planar graphs. Independently, if the number of leaves ℓ of T_2 in a random triangulation is close to the value we have obtained here, that is $\ell \approx 0.62n$, the remaining-edge set M_S could be stored with $n + \ell = 1.62n$ bits (we need n bits to describe the edges of T_2 that are in G, and of ℓ bits for the edges of T_1 that are in G but that are leaving a leaf of T_2). This would give a $(3.24 + 1.62)n = 4.84n$ bit random planar graph encoding. By the Occam's razor principle, a conjecture must be simple, and since we have the numerical coincidence that is $4.85 \approx 5\mathrm{H}(2/5)$, we propose:

Conjecture 1 *For every n large enough, $p(n) \leqslant \binom{5n}{2n}$.*

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.

[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.

[BFSS00] Cyril Banderier, Philippe Flajolet, Gilles Schaeffer, and Michèle Soria. Planar maps and airy phenomena. In 27^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1853 of LNCS, pages 388–402. Springer, July 2000.

[BGH02] Nicolas Bonichon, Cyril Gavoille, and Nicolas Hanusse. Maple worksheet, 2002. Available at: `http://dept-info.labri.fr/~gavoille/article/BGH02.mws`.

[BGW99] Edward A. Bender, Zhicheng Gao, and Nicholas C. Wormald. The number of labeled 2-connected planar graphs, 1999. Manuscript.

[BK02] Manuel Bodirsky and Mihyun Kang. Generating random outerplanar graphs, 2002. Submitted.

[BM02] Mireille Bousquet-Mélou, 2002. Private communication.

[Bol78] Béla Bollobás. *Extremal Graph Theory*. Academic Press, New York, 1978.

[Bon02] Nicolas Bonichon. A bijection between realizers of maximal plane graphs and pairs of non-crossing Dyck paths. In *Formal Power Series & Algebraic Combinatorics (FPSAC)*, July 2002.

[CGH$^+$98] Richie Chih-Nan Chuang, Ashim Garg, Xin He, Ming-Yang Kao, and Hsueh-I Lu. Compact encodings of planar graphs via canonical orderings and multiple parentheses. In 25^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1443 of LNCS, pages 118–129. Springer, July 1998.

[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.

[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.

[DVW96] Alain Denise, M. Vasconcellos, and D.J.A. Welsh. The random planar graph. *Congressus Numerantium*, 113:61–79, 1996.

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

[FJ89] Greg N. Frederickson and Ravi Janardan. Efficient message routing in planar networks. *SIAM Journal on Computing*, 18(4):843–857, August 1989.

[GH99] Cyril Gavoille and Nicolas Hanusse. Compact routing tables for graphs of bounded genus. In 26^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1644 of LNCS, pages 351–360. Springer, July 1999.

[GM01] S. Gerke and Colin McDiarmid. Adding edges to planar graphs, 2001. Preprint.

[HKL00] Xin He, Ming-Yang Kao, and Hsueh-I Lu. A fast general methodology for information-theoretically optimal encodings of graphs. *SIAM Journal on Computing*, 30(3):838–846, 2000.

[KADS02] Andrei Khodakovsky, Pierre Alliez, Mathieu Desbrun, and Peter Schröder. Near-optimal connectivity encoding of 2-manifold polygon meshes. *Graphical Models*, 2002. To appear in a special issue.

[KR99] Davis King and Jarek Rossignac. Guaranteed 3.67V bit encoding of planar triangle graphs. In $11th$ *Canadian Conference on Computational Geometry (CCCG)*, pages 146–149, August 1999.

[KW95] Kenneth Keeler and Jeffery Westbrook. Short encodings of planar graphs and maps. *Discrete Applied Mathematics*, 58:239–252, 1995.

[LT79] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, April 1979.

[Lu02a] Hsueh-I Lu. Improved comact routing tabels for planar networks via orderly spanning trees. In 8^{th} *Annual International Computing & Combinatorics Conference (COCOON)*, volume 2387 of LNCS. Springer, August 2002.

[Lu02b] Hsueh-I Lu. Linear-time compression of bounded-genus graphs into information-theoretically optimal number of bits. In 13^{th} *Symposium on Discrete Algorithms (SODA)*, pages 223–224. ACM-SIAM, January 2002.

[LW87] Valer A. Liskovets and Timothy R. Walsh. Ten steps to counting planar graphs. *Congressus Numerantium*, 60:269–277, 1987.

[MR97] J. Ian Munro and Venkatesh Raman. Succinct representation of balanced parentheses, static trees and planar graphs. In 38^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 118–126. IEEE Computer Society Press, October 1997.

[NC88] Takao Nishizeki and Norishige Chiba. *Planar Graphs: Theory and Algorithms.* North-Holland Mathematics Studies 140, 1988.

[OPT02] Deryk Osthus, Hans Jürgen Prömel, and Anusch Taraz. On random planar graphs, the number of planar graphs and their triangulations. Manuscript submitted for publication, 2002.

[Pag01] Rasmus Pagh. Low redundancy in static dictionaries with constant query time. *SIAM Journal on Computing*, 31(2):353–363, 2001.

[Ros99] Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, 1999.

[Sch90] Walter Schnyder. Embedding planar graphs on the grid. In 1^{st} *Symposium on Discrete Algorithms (SODA)*, pages 138–148. ACM-SIAM, 1990.

[Sch99] Gilles Schaeffer. Random sampling of large planar maps and convex polyhedra. In 31^{st} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 760–769, Atlanta, Georgia, May 1999.

[Tho01] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In 42^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, October 2001.

[Tur84] György Turán. Succinct representations of graphs. *Discrete Applied Mathematics*, 8:289–294, 1984.

[Tut62] William T. Tutte. A census of planar triangulations. *Canadian Journal of Mathematics*, 14:21–38, 1962.

[Yan89] Mihalis Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38:36–67, 1989.

Contents