平面三角分割グラフを列挙するアルゴリズムの改良

中野 眞一[1] and 宇野 毅明[2]
[1] 群馬大学 〒376-8515 群馬県桐生市, nakano@cs.gunma-u.ac.jp
[2] 国立情報学研究所 〒101-8430 東京都千代田区一ツ橋, uno@nii.jp

抄録: 本稿では, 頂点数がちょうど n であり, 外平面の頂点数がちょうど r であるような2連結な平面三角分割グラフを列挙するアルゴリズムの改良を示す. このアルゴリズムは, $O(n)$ の空間しか使用せず, すべてのグラフを重複なく, ちょうど1度ずつ, 1つあたり $O(rn)$ の計算時間で出力する. 既存の最速なアルゴリズムの1つあたりの計算時間は $O(r^2n)$ である.

Improved Algorithm for Enumerating Plane Triangulations

Shin-ichi Nakano[1] and Takeaki Uno[2]
[1] Gunma University, Kiryu 376-8515, Japan, nakano@cs.gunma-u.ac.jp
[2] National Institute of Informatics, Tokyo 101-8430, Japan, uno@nii.jp

abstract: In this paper we give an algorithm to generate all biconnected plane triangulations having exactly n vertices including exactly r vertices on the outer face. The algorithm uses $O(n)$ space in total and generates such triangulations without duplications in $O(rn)$ time per triangulation, while the previous best algorithm generates such triangulations in $O(r^2n)$ time per triangulation.

1 Introduction

Generating all graphs with some property without duplications has many applications, including unbiased statistical analysis [9]. Many algorithms to solve these problems are already known [1, 2, 9, 11]. Many nice textbooks have been published on the subject[5, 7].

In this paper we wish to generate all biconnected plane triangulations having exactly n vertices including exactly r vertices on the outer face. Such triangulations play an important role in many algorithms, including graph drawing algorithms [3, 4, 10].

Recentry we have given an algorithm to generate all biconnected "based" plane triangulations having exactly n vertices including exactly r vertices on the outer face [8]. A *based* plane triangulation means a plane triangulation with one designated "base" edge on the outer face. For instance, four biconnected based plane triangulations are shown in Fig. 1, where the base edges are depicted by thick lines. Note that, however, those based triangulations are isomorphic as non-based plane triangulations. The algorithm uses $O(n)$ space in total and runs in $O(f(n,r))$ time, where $f(n,r)$ is the number of nonisomorphic biconnected based plane triangulations having exactly n vertices including exactly r vertices on the outer face. The algorithm generates triangulations without duplications. So the algorithm generates each

triangulation in $O(1)$ time on average, while the previous best algorithm generates such triangulations in $O(n^2)$ time per triangulation [1]. The algorithm does not output entire triangulations but the difference from the previous triangulation.

The strategy of the algorithm in [8] is as follows. Given n and r, we first define a tree T such that the inner vertices of T correspond to the biconnected based plane triangulations having at most $n-1$ vertices including at most $n-r$ inner vertices, the leaves of T correspond to the biconnected based plane triangulations having exactly n vertices including exactly r vertices on the outer face, and the edges of T correspond to some relation between the biconnected based plane triangulations. T is called the genealogical tree, and the genealogical tree for $n = 5$ and $r = 4$ is shown in Fig. 2. In the figure we can observe that if we remove a vertex depicted by a white circle from a biconnected based plane triangulation, then we can have a "parent" biconnected based plane triangulation. Also we can prove the number of vertices of T is within 3 times the number of leaves of T. Since the size of T is huge in general, so we cannot construct whole part of T at once. However, we can simply traverse T in $O(1)$ time per edge of T by partially constructing T. We need only $O(n)$ space in total. And on the traversal we can find all the vertices of T, which correspond to all the triangulations.

By modifying the algorithm, we can also generate without duplications all biconnected (non-based) plane triangulations having exactly n vertices including exactly r vertices on the outer face in $O(r^2n)$ time per triangulation on average [8]. Another algorithm with $O(n^2)$ time per triangulation is also claimed in [9] without detail but using a complicated theoretical linear-time plane graph isomorphism algorithm [6], while the algorithm in [8] is simple and does not need the isomorphism algorithm.

The strategy of the algorithm in [8] is as follows. The genealogical tree T has many biconnected based plane triangulations which are distinct as based plane triangulations, but isomorphic as (non-based) plane triangulations. The only difference is the choice of the base edge on the outer face. See Fig. 1. On the traversal of T, we have to output exactly one biconnected (non-based) plane triangulations for each isomorphic class. By giving a unique sequence of letters for each biconnected based plane triangulations, we can define a representative triangulation among each isomorphic class as the triangulation having the lexicographically-first sequence of letters. The algorithm in [8] needs $O(rn)$ time computation at each leaf v of T to decide whether the sequence of letters for the based triangulation corresponding to v is the lexicographically-first one among the isomorphic class, and only in such case the based plane triangulation is output as the representative plane triangulation. Otherwise the based triangulation is not output. For each output triangulation, T may contain r isomorphic ones corresponding to the r choices of the base edge. Thus the algorithm generates each triangulation in $O(r^2n)$ time on average.

In this paper we improve the running time of the algorithm in [8] as follows. We define a new unique sequence of letters for each biconnected based plane triangulation. Given a biconnected based plane triangulation, the new sequence of letters needs less computation to decide whether the sequence of letters for the based plane triangulation is the lexicographically-first one among the isomorphic class. Our algorithm needs only $O(n)$ time computation at each leaf of T. Again, for each output triangulation, T may contain r isomorphic ones corresponding to the r choices of the base edge. Thus our algorithm generate each triangulation in $O(rn)$

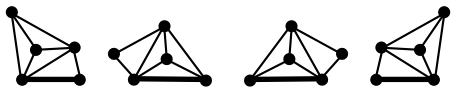

Figure 1: Biconnected based plane triangulations.

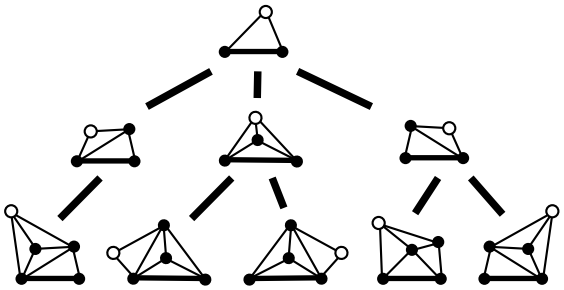

Figure 2: The genealogical tree for $n = 5$ and $r = 4$.

time on average.

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 defines our new sequence of letters. Finally Section 4 is a conclusion.

2 Preliminaries

In this section we give some definitions.

Let G be a connected graph with n vertices. An edge connecting vertices x and y is denoted by (x, y). The *degree* of a vertex v is the number of neighbors of v in G. A *cut* is a set of vertices whose removal results in a disconnected graph or a single-vertex graph K_1. The *connectivity* $\kappa(G)$ of a graph G is the cardinality of the minimum number of vertices consisting a cut. G is $k-$*connected* if $k \leq \kappa(G)$.

A graph is *planar* if it can be embedded in the plane so that no two edges intersect geometrically except at a vertex to which they are both incident. A *plane* graph is a planar graph with a fixed planar embedding. A plane graph divides the plane into connected regions called *faces*. The unbounded face is called *the outer face*, and other faces are called *inner faces*. We regard *the contour* of a face as the clockwise cycle formed by the vertices and edges on the boundary of the face. We denote the contour of the outer face of plane graph G by $C_o(G)$. An edge connecting two vertices on $C_o(G)$ but not on $C_o(G)$ is called a *chord* of G. A plane graph is called *a plane triangulation* if each inner face has exactly three edges on its contour. A *based* plane triangulation is a plane triangulation with one designated edge on the contour of the outer face. The designated edge is called *the base edge*.

3 The sequence of letters

Let G be a biconnected based plane triangulation, $C_o(G) = w_0, w_1, \cdots, w_\ell$ and (w_0, w_ℓ) be the base edge of G. In this section we assign a sequence of letters to G, and show that the sequence is unique for G.

Let $G_0 = G$. Let G_1 be the plane triangulation derived from G_0 by removing all ver-

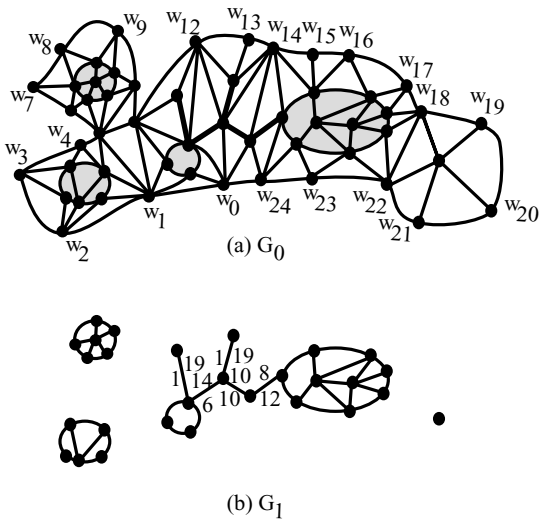

Figure 3: (a) G_0 and (b) G_1.

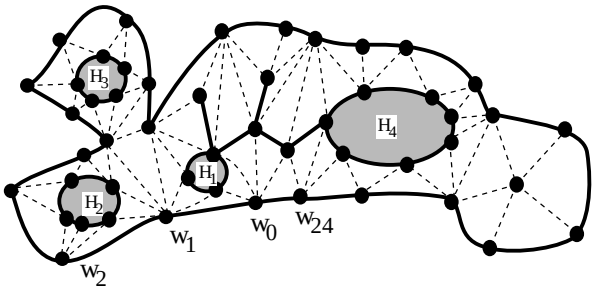

Figure 4: D_0.

tices on the outer face and edges incident to them. An example is shown in Fig. 3. Note that G_1 may be disconnected, but each inner face has exactly three edges on its contour, so G_1 is still a plane triangulation. Let D_0 be the plane subgraph of G_0 induced by the edges of G_0 located outside of $C_o(G_1)$ including $C_o(G_1)$. An example is shown in Fig. 4. We are going to assign a sequence of letters to D_0. By recursively assigning a sequence of letters to each biconnected component of G_1, then marging derived sequences appropriately, and append it to the sequence for D_0, we can assign a sequence of letters for G.

We first explain about the order of the marge, then explain how to assign a sequence of letters to D_0.

Assume G_1 has biconnected components $H_1, H_2, \cdots$. Each biconnected component has at least one neighbor vertex w_i on $C_o(G_0)$. For each $H_1, H_2, \cdots$ we choose one neighbor vertex w_i on $C_o(G_0)$ having the minimum index i. We can marge the sequences of $H_1, H_2, \cdots$ in the increasing order of i. If two or more biconnected components share the same i, then we can break the tie by checking the edges connecting w_i and the

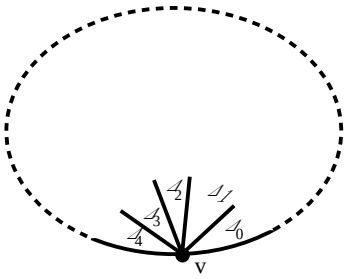

Figure 5: The triangle sequence.

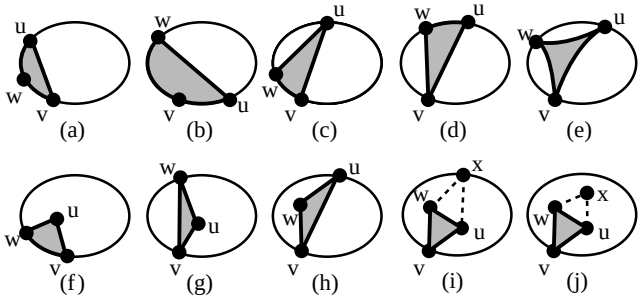

Figure 6: The ten types of triangles.

biconnected components, and order them counterclockwise around w_i. For instace, in Fig. 4, we order them H_1, H_2, H_3, H_4.

For recursion we define the base edge of each $H \in \{H_1, H_2, \cdots, \}$ as follows. Assume w_i on $C_o(G_0)$ is the neighbor of H having the minimum i, and (w_i, u) is the first edge connecting w_i and H around w_i in clockwise order. Let $u^{'}$ be the preceding vertex of u on $C_o(H)$. We define $(u, u^{'})$ the base edge of H.

Now we explain how to assign a sequence of letters to D_0. Let v be a vertex on $C_o(G_0)$. Let $\Delta_0, \Delta_1, \cdots, \Delta_k$ be the inner faces of G_0 appearing around v in counterclockwise order, and assume Δ_0 and Δ_k have at least one edge on $C_o(G_0)$ on its contour. See Fig. 5. We can observe that each $\Delta_1, \Delta_2, \cdots, \Delta_k$ is one of the ten types shown in Fig. 6, where the inner face is shaded. Note that we do not mention about Δ_0 here, since Δ_0 always appears as the last inner face around the preceding vertex of v on $C_o(G_0)$. In Fig. 6, u, v, w is the three vertices on the contour of the inner face, and the three vertices appear on $C_o(G_0)$ for (a)–(e), exactly two of the three vertices appear on $C_o(G_0)$ for (f)–(h), and exactly one of the three vertices appears on $C_o(G_0)$ for (i) and (j). For (i) and (j), x is the vertex on the contour of the other inner face having edge (u, w), and $x \in C_o(G_0)$ for (i), and $x \notin C_o(G_0)$ for (j). We assign to v a sequnece of letters consisting of a left parenthesis, the type names of Δ_1 (possibly with a parameter), a right parenthesis, a left parenthesis, the type names of Δ_2 (possibly with a parameter), a right parenthesis, $\cdots$, a left parenthesis, the type names of Δ_k (possibly with a parameter), and a right parenthesis.

For some type we append a parameter p to the type name as follows.

Assume $C_o(G_0) = w_0(= v), w_1, \cdots, w_\ell$. For (c),(d),(e),(h), we define $p = q$ where $w_q = u$. Here p is the length of the subpath of $C_o(G_0)$ from v to u. For (g), we define $p = q$ where $w_q = w$. Those parameters are called the *chord parameters*. Intutively those parameters uniquely define the structure of the chords of $C_o(G_0)$.

For (i), assume G_1^x is the connected component of G_1 containing u and w. See Fig. 6 (i). Now if we remove edge (u, w) from G_1^x then the resulting graph is disconnected. We can observe (u, w) and (w, u) appear on $C_o(G_1^x)$. Assume $C_o(G_1^x) = v_0(= u), v_1(= w), v_2, v_3, \cdots, v_q(= w), v_{q+1}(= u), \cdots$. We define $p = q$. An example for value of p is shown in Fig. 3(b). Those parameters are called the *bridge parameters*. Intutively those parameters
uniquely define the structure of $C_o(G_1)$.

For instance, we assign to w_0, w_1, w_2 in Fig. 7, $(i)(i)(j)(f)$, $(j)(g, 10)(e, 10)(h, 4)(j)(f)$, $(j)(j)(f)$, respectively. Note that, for any choice of the base edge, the parameters are always identical.

Let $C_o(G) = w_0, w_1, \cdots, w_\ell$ and (w_0, w_ℓ) be the base edge of G. By concatenating those sequences for vertices on $C_o(G)$, we can assign a sequence of letters to D_0. By recursively assigning a sequence of letters to each biconnected component of G_1, then marging derived sequences appropriately, we can assign a sequence of letters for a biconnected based plane triangulation G with the base edge (w_0, w_ℓ). For instance, we assign to the biconnected based plane triangulation with the base edge (w_0, w_{24}), shown in Fig. 7, sequence $(i)(i)(j)(f)$, $(j)(g, 10)(e, 10)(h, 4)(j)(f)$, $(j)(j)(f)$, $(j)(f)$, $(j)(f)(g, 21)(e, 21)(d, 6)(h, 5)(j)(f)$, $(j)(f)$, $(f), \cdots$. Also we assign to the biconnected based plane triangulation with the base edge (w_1, w_0), shown in Fig. 7, sequence $(j)(g, 10)(e, 10)(h, 4)(j)(f)$ $(j)(j)(f)$, $(j)(f)$, $(j)(f)(g, 21)(e, 21)(d, 6)(h, 5)(j)(f)$, $(j)(f)$, $(f), \cdots$. By a suitable "lotated shift" those sequences are easily translated each other. We have the following theorem.

Theorem 1 *Let G_a and G_b be two biconnected based plane triangulations, and S_a and S_b be the sequence of letters for them, respectively. G_a and G_b are isomorphic if and only if $S_a = S_b$.*

Let G be a biconnected plane triangulation with r vertices on the outer face. Assume $G_o(G) = w_0, w_1, \cdots, w_{r-1}$. By choosing a base edge from the r edges on $C_o(G)$, r of based triangulations are derived from G. Let $G_{(i)}$ be the biconnected based plane triangulation with the base edge (w_i, w_{i+1}), for i, $0 \leq i \leq r-1$. Let $S_{(i)}$ be the sequence of letters for $G_{(i)}$ for i, $0 \leq i \leq r-1$. We have the following theorem.

Theorem 2 *Given G, in $O(n)$ time, we can find i such that $S_{(i)}$ is the lexicographically-first one among the r sequences $S_{(0)}$, $S_{(1)}, \cdots, S_{(r-1)}$.*

Proof: (Sketch) By a simple dynamic programming we can solve the problem in $O(n)$ time as follows.

We need some notations here. Let Δ_0, $\Delta_1, \cdots, \Delta_k$ be the inner faces of G_0 appearing around a vertex v on $C_o(G_0)$. in counterclockwise order, and assume Δ_0 and Δ_k have at least one edge on $C_o(G_0)$ on its contour. See Fig. 5. We call Δ_1 *the first face* of v, and $\Delta_1, \Delta_2, \cdots, \Delta_k$ the faces of v. By concatenating the faces of the vertices on $C_o(G_0)$, we are going to consider the (cyclic) sequence of faces in this algorithm. Note that some faces may appear two or three times on the sequence, for instance a face of type (e) in Fig. 6 appear three times on the sequence. All we need to do is to find a

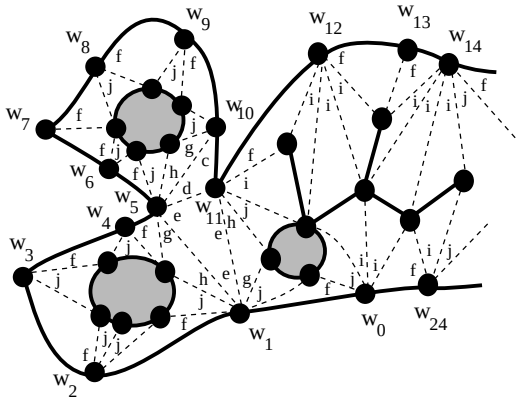

Figure 7: The triangle sequence.

starting vertex on $C_o(G_0)$ which produce the lexicographically-first sequence of letters.

First we compute all parameters in D_0 in $O(|E_D|)$ time, where E_D is the set of edges in D_0. Then we assign the sequence to each vertices on $C_o(G_0)$.

We start with stage 0.

Now in $O(n)$ time we find vertices having the first face with the lexicographically-first sequence of letters among the vertices on $C_o(G_0)$. We call those vertices the *stage* 0 *winners*. Let V_0 be the set of such vertices, and let F_0 be the set of such first faces. If there is only one such vertex in V_0, say v, then we have done, and we have found $w_i = v$.

Otherwise, we have two or more such vertices in V_0. Then we proceed to stage 1.

We check the succeeding faces of the faces in F_0. Among those succeeding faces we choose faces having the lexicographically-first sequence of letters. Let F_1 be the set of such faces. Only the vertices in V_0 having the first face succeeded by faces in F_1 still have a chance to be w_i.

If $|F_1| = 1$, then we have done, and w_i is the vertex in V_0 having the first face succeeded by the face in F_1.

Otherwise, F_1 contains two or more such faces. We concatenate each alternating occurence of faces in F_0 and faces in F_1. If such a alternating occurence contains $2k$ faces, then we regard them as one (virtual) face having letters $(1, k)$. Only the vertices in V_0 having the first face succeeded by virtual faces having the maximum k still has a chance to be w_i. Let V_1 be the set of such vertices. We call those vertices the *stage* 1 *winners*.

We repeat this process until we find a unique w_i. If two or more winners remain after analyzing D_0, we proceed to each biconnected component of G_1, and then proceed recursively.

If after completely analyzing G we still have two or more winners for w_i then we can choose any of them. (This is a symmetry case.)

Since each face is re-checked after the preceding (virtual) face is concatenated to some virtual face, the total number of such recheck is at most $3f$, where f is the number of faces in G. Also we check each edge at most a constant number of times, and for planar graph $f \leq m$ and $m \leq 3n$ holds. Thus the running time of the algorithm is $O(n)$.

By replacing the $O(rn)$ time computation

at each leaf of the genealogical tree T in [8] to the $O(n)$ time computation above, we can improve the running time of the algorithm. We have the following theorem.

Theorem 3 *One can generate all the biconnected plane triangulations having n vertices including r vertices on the outer face in $O(rn)$ time per triangulation on average.*

4 Conclusion

In this paper we have given an algorithms to generate all biconnected plane triangulations without duplications. Our idea is to define a unique sequence of letters for each biconnected based plane triangulations, and by the sequence we can efficiently decide for each biconnected based plane triangulation whether the based triangulation should be output as a biconnected plane triangulations.

References

[1] D. Avis, *Generating rooted triangulations without repetitions*, Algorithmica, 16, (1996), pp.618-632.

[2] T. Beyer and S. M. Hedetniemi, *Constant time generation of rooted trees*, SIAM J. Comput., 9, (1980), pp.706-712.

[3] M. Chrobak and S. Nakano, *Minimum-width grid drawings of plane graphs*, Computational Geometry: Theory and Applications, 10, (1998), pp.29-54.

[4] H. de Fraysseix, J. Pach and R. Pollack, *How to draw a planar graph on a grid*, Combinatorica, 10, (1990), pp.41-51.

[5] L. A. Goldberg, *Efficient algorithms for listing combinatorial structures*, Cambridge University Press, New York, (1993).

[6] J. E. Hopcroft and J.K. Wong, *Linear time algorithm for isomorphism of planar graphs*, Proc. of 6th STOC, (1974), pp.172-184.

[7] D. L. Kreher and D. R. Stinson, *Combinatorial algorithms*, CRC Press, Boca Raton, (1998).

[8] Z. Li and S. Nakano, *Efficient generation of plane triangulations without repetitions*, Proc. ICALP2001, LNCS 2076, (2001), pp.433–443.

[9] B. D. McKay, *Isomorph-free exhaustive generation*, J. of Algorithms, 26, (1998), pp.306-324.

[10] W. Schnyder, *Embedding planar graphs on the grid*, Proc. 1st Annual ACM-SIAM Symp. on Discrete Algorithms, San Francisco, (1990), pp.138-148.

[11] R. A. Wright, B. Richmond, A. Odlyzko and B. D. McKay, *Constant time generation of free trees*, SIAM J. Comput., 15, (1986), pp.540-548.