IEICE TRANS. INF. & SYST., VOL.E94–D, NO.2 FEBRUARY 2011

PAPER Special Section on Foundations of Computer Science — Mathematical Foundations and Applications of Algorithms and Computer Science —

Generation of Symmetric and Asymmetric Biconnected Rooted Triangulated Planar Graphs

Bingbing ZHUANG[†a)], ***Nonmember and*** **Hiroshi NAGAMOCHI**[†b)], ***Member***

SUMMARY In a rooted triangulated planar graph, an outer vertex and two outer edges incident to it are designated as its root, respectively. Two plane embeddings of rooted triangulated planar graphs are defined to be equivalent if they admit an isomorphism such that the designated roots correspond to each other. Given a positive integer n, we give an $O(n)$-space and $O(1)$-time delay algorithm that generates all biconnected rooted triangulated planar graphs with at most n vertices without delivering two reflectively symmetric copies.

key words: *enumeration, reflective symmetry, triangulation, plane graphs, planar graphs, biconnectivity, graph algorithms*

1. Introduction

To generate a certain class of graphs is one of the fundamental and important issues in graph theory. The common idea behind most of the recent efficient enumeration algorithms (e.g., [7]) is to define a unique representative graph for each graph in a class of graphs as its "parent," which induces a rooted tree that connects all graphs in the class, called the *family tree* $\mathcal{F}$, where each node in $\mathcal{F}$ corresponds to a graph in the class. Then all graphs in the class will be generated one by one according to the depth-first traversal of the family tree $\mathcal{F}$. This is similar to the reverse research method [1], which first requires to introduce an adjacency on pairs of graphs before choosing a graph onto which we move from a given graph. Time delay of an enumeration algorithm is a time bound between two consecutive outputs. Enumerating graphs with a polynomial time delay would be rather easy since we can examine the whole structure of the current graph anytime. However, an algorithm with a constant time delay in the worst case is a hard target to achieve without a full understanding of the graphs to be generated, since not only the difference between two consecutive outputs is required to be $O(1)$, but also any operation for examining symmetry and identifying the edges/vertices to be modified to get the next output needs to be executable in $O(1)$ time.

Our research group has been developing algorithms for enumerating chemical graphs that satisfy given various constraints [2]–[4]. We aim to continue providing efficient enumeration algorithms for wider classes of graphs than trees such as cacti and outerplanar graphs on our web server.

Manuscript received March 31, 2010.
Manuscript revised August 3, 2010.

[†]The authors are with the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
a) E-mail: zbb@amp.i.kyoto-u.ac.jp
b) E-mail: nag@amp.i.kyoto-u.ac.jp

DOI: 10.1587/transinf.E94.D.200

The authors recently proposed a general enumeration scheme for classes $\mathcal{H}$ of "rooted graphs with a reflective block structure" [11], [12]. A *reflective block* means a rooted biconnected component which may admit reflective symmetry around its root. A *rooted graph with reflective block structure* consists of reflective blocks, where we consider the free symmetry among the siblings of each cut vertex; i.e., we do not distinguish any permutation of the siblings from others. The proposed scheme [11], [12] allows us to develop only an efficient enumeration algorithm for the class $\mathcal{B}$ of the reflective blocks without designing any algorithm for enumerating entire graphs; i.e., plugging an enumeration algorithm for the class $\mathcal{B}$ into our scheme automatically yields an enumeration algorithm for the class $\mathcal{H}$ of the graphs that consist of those reflective blocks without repetition. For example, a cactus is a graph which consists of cycles such that no two cycles share more than one vertex, where each rooted cycle can be regarded as a reflective block. The scheme can yield an algorithm that generates rooted cacti in $O(1)$-time delay just by designing an $O(1)$-time delay algorithm for the class of rooted cycles. Therefore, to develop algorithms for various classes of graphs with a reflective block structure, it is important to study the structure of reflective blocks from an algorithmic view point.

In this paper, we study how to generate the class $\mathcal{B}$ of rooted biconnected planar graphs with internally triangulated faces, which has reflective block structure. As for enumeration of triangulations, Li and Nakano [5] presented an $O(1)$-time delay algorithm for the class of all biconnected rooted triangulated plane graphs with at most n vertices. Afterwards Nakano [6] presented an $O(1)$-time delay algorithm for the class of all triconnected rooted triangulated plane graphs with at most n vertices. In these algorithms, two reflectively symmetric copies around the root may be output. Contrary to these, we show the next main result in this paper.

Theorem 1: Let $\mathcal{B}_1$ (resp., $\mathcal{B}_2$) be the class of symmetric (resp., asymmetric) biconnected rooted planar graphs with internally triangulated faces. For each integer $n \geq 2$ and $i \in \{1, 2\}$, all blocks in $\mathcal{B}_i$ with at most n vertices can be generated without duplication in $O(1)$ time delay in the worst case and in $O(n)$ space.

Theorem 1 together with the scheme [11], [12] implies that rooted *connected* planar graphs with internally triangulated faces can be generated in $O(1)$ time delay in the worst case.

During our study, we realized a new method of avoid-

ing duplications of symmetric copies of the same block by introducing a notion of "pseudo-symmetry." Unlike the previous approach to avoiding duplications of symmetric copies [7], [9], we do not encode graphs into sequences of labels to define "canonical form" among symmetric copies, and no comparison of two encoded subgraphs is executed in our algorithm. In fact, we propose two algorithms, one generates only symmetric blocks and the other generates asymmetric ones. We also developed a "balanced orientation" of edges to check whether two vertices are adjacent or not in $O(1)$ time and $O(n)$ space, which is crucial to establish Theorem 1. These new ideas of "pseudo-symmetry" and "balanced orientation" are used in our companion papers on $O(1)$-time delay enumerations of outerplanar graphs and biconnected plane graphs [13].

The rest of the paper is organized as follows. Section 3 defines parents of blocks based on pseudo-symmetry. Sections 4 and 5 give algorithms for enumerating only symmetric blocks and asymmetric blocks, respectively.

2. Preliminaries

Throughout the paper, a graph stands for a simple undirected graph, which is denoted by a pair $H = (V, E)$ of a vertex set V and an edge set E. A graph is treated as a labeled graph in which all vertices receive distinct *vertex names* unless stated otherwise. The set of vertices and the set of edges of a given graph H are denoted by $V(H)$ and $E(H)$, respectively.

A vertex in a connected graph is called a *cut-vertex* if its removal results in a disconnected graph. A connected graph is called *biconnected* if it has no cut-vertex.

We define a *block* to be a rooted biconnected graph with a configuration [11], [12], in other words, a block is a plane embedding of a graph. Two blocks are called *equivalent* if the biconnected graphs of these blocks admit a rooted-isomorphic bijection under the configuration. We assume that, for each block B, either (i) no other block B' is equivalent to B under the configuration, where B is called *asymmetric* or (ii) there is exactly one distinct block B' which is equivalent to B, and B and B' admit a symmetry of order 2 which is given by an automorphism ψ such that $V_1(B) = \{\psi(v) \mid v \in V_2(B)\}$ and $\psi(v) = v$, $v \in V_3(B)$ for a partition $V_1(B)$, $V_2(B)$ and $V_3(B)$ of the vertex set $V(B)$. Intuitively, $V_1(B)$ and $V_2(B)$ are the reflective symmetric vertex partitions while $V_3(B)$ is the set of self symmetric vertices which can be consider as the middle of a symmetric block.

A graph is called *planar* if its vertices and edges can be drawn as points and curves on the plane so that no two curves intersect except for their endpoints. In such a drawing of a planar graph, the plane is divided into several connected regions, each of which is called a *face*. A face is called *outer face* if it is the unbounded region, and it is called *inner face* otherwise. By definition, any drawing of a planar graph has only one outer face. A cycle of a graph is called a *facial cycle* if it is the boundary of a face. We call such a cycle the *outer facial cycle* (resp., an *inner facial cycle*) if it is the boundary of the outer (resp., an inner) face. A set F of facial cycles in a drawing defines a combinatorial embedding of a planar graph which gives an order of neighbours of each vertex. A planar graph with a fixed combinatorial embedding is called a *plane* graph if a facial cycle in the embedding is designated as the outer facial cycle. Note that two distinct plane graphs can be isomorphic to the same planar graph, and hence both of them can be treated as plane embeddings (i.e., drawings) of this planar graph.

A graph is called a *triangulated planar graph* if it admits a plane embedding in the plane such that all inner faces are triangle, and we call such a plane embedding a *proper*. If a triangulated planar biconnected graph with at least three vertices has a proper embedding such that the outer face is not a triangle, then the graph has only two proper embeddings, where each of them can be obtained from the other by reflection since the outer face is decided. Otherwise, a triangulated planar biconnected graph is a maximal planar graph, which has $2f$ proper embeddings, where f is the number of faces in the graph.

Let G be a biconnected and triangulated planar graph. We call G *rooted* if it designates a vertex v and two edges e and e' incident to it such that v, e and e' appear as an outer vertex and outer edges of a proper embedding of G. Such a proper embedding of G can be treated as a *block* of G. Two blocks are *equivalent* if they admit an isomorphism such that the roots (resp., outer vertices) correspond to each other. Hence a block B has at most one distinct block B' equivalent to B. A block B is called *asymmetric* if no other block is equivalent to B, and is called *symmetric* otherwise, where a symmetric block has a reflectional symmetry around its root vertex. Let $\mathcal{B}$ denote the set of all blocks of rooted triangulated planar biconnected graphs.

In this paper, we give an algorithm of enumerating all blocks in $\mathcal{B}$ without repetition.

3. Parents of Blocks

A block in $\mathcal{B}$ may have a reflectional symmetry around the root. One possible approach to enumeration of all blocks without outputting their symmetric copies is to encode the structures by $V_1(B)$ and $V_2(B)$ of a block B separately and to maintain a "canonical form" of B by comparing the two codes whenever a new child of the current block is generated. The algorithm for rooted outerplanar graphs [9] exploits such a method which decomposes a rooted biconnected outerplanar graph G into three subgraphs induced by the subsets $V_i(B)$, $i = 1, 2, 3$ to obtain three encoded sequences of G

However, in this paper, we use the following novel idea to avoid such a comparison of two encoded subgraphs. We introduce "pseudo-symmetry" of blocks in $\mathcal{B}$, which can be tested in $O(1)$ time based on a local structure of B. Pseudo-symmetry of a block G satisfies the property that a symmetric block G is always pseudo-symmetric while an asymmetric block G can be pseudo-symmetric or not. Then we introduce (i) a "bilateral" operation to pseudo-symmetric blocks that converts a symmetric (resp., asymmetric) block B into

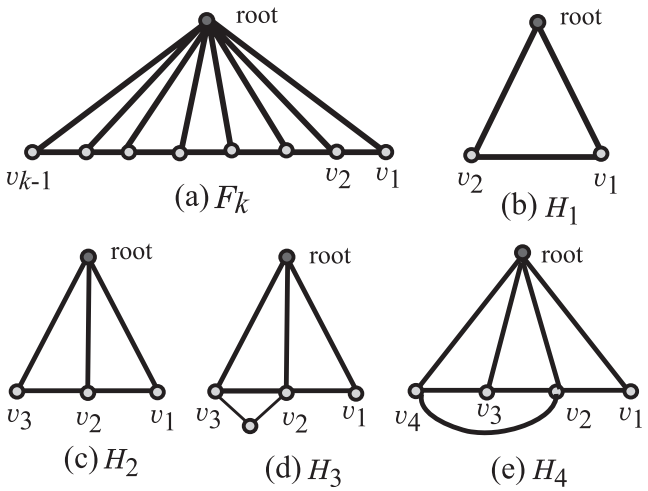


Fig. 1 (a) A fan F_k, (b)–(e) seed blocks H_i, $i = 1, 2, 3, 4$.

another symmetric (resp., asymmetric) block B'; (ii) a "unilateral" operation to non pseudo-symmetric blocks that converts an asymmetric block B into another asymmetric block B'. By defining such B' as the parent of B, we can obtain two family trees of blocks separately, one for all symmetric blocks, and the other for all asymmetric blocks. Generate all the blocks in a family tree one by one. No comparison between blocks is needed.

Let $deg(v; G)$ denote the degree of a vertex v in a block G. A block $G \in \mathcal{B}$ is called a *fan* if it is obtained from a path P with at least one vertex by adding a new vertex v together with an edge incident to each vertex in the path (see Fig. 1 (a)), where v is regarded as the root r_G of the fan G. A fan with k vertices is denoted by F_k. For each block $G \in \mathcal{B}$, we define the *core* $F(G)$ of G to be the maximum fan F_k with $r(F_k) = r_G$ contained in G. A vertex (resp., an inner face) in the core $F(G)$ is called a *core vertex* (resp., a *core* inner face).

Each block $G \in \mathcal{B}$ is treated as a plane drawing such that the root r_G of G appears on the top of the drawing and the nonroot vertices $v_1, v_2, \ldots, v_k$ in $F(G)$ along a path from right to left (see Fig. 1 (a)).

Let v_k and v_{k-1} be the leftmost and second leftmost neighbours of the root of a block G with the root r_G. We define the following three *states* for the leftmost/rightmost neighbours of the root r_G. The leftmost neighbour v_k of the root r_G is called *3-outer* (resp., *3-inner*) if $deg(v_k; G) \geq 3$ and the second leftmost neighbour v_{k-1} is an outer (resp., inner) vertex; and v_k is called *2-outer* if $deg(v_k; G) = 2$ (where v_{k-1} is always outer). We define state 2-outer, 3-inner and 3-outer rightmost neighbours analogously. A block G is called *pseudo-symmetric* if both the leftmost and rightmost neighbours of the root r_G take the same state. See Fig. 3 (a), Fig. 4 (a) and Fig. 2 (a) for three such cases.

We define a total order for the three states to be 3-inner>3-outer>2-inner. We generate pseudo-symmetric blocks such that the left side is heavier than the right side with respect to their states without generating their symmetric copies.

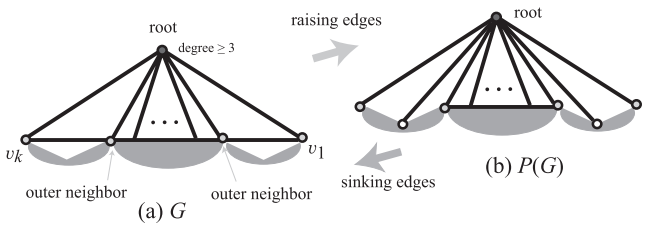


Fig. 2 (a) Block G with the 3-outer leftmost and rightmost neighbours of r_G; (b) Parent $\mathcal{P}_{\mathcal{B}}(G)$.

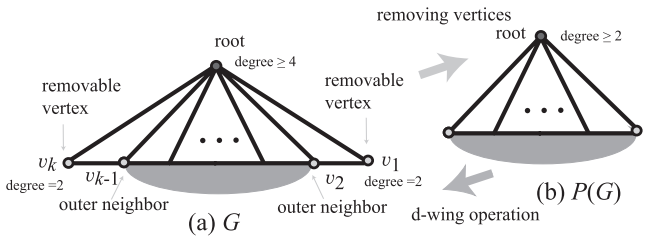


Fig. 3 (a) Block G with the 2-outer leftmost and rightmost neighbours; (b) Parent $\mathcal{P}_{\mathcal{B}}(G)$.

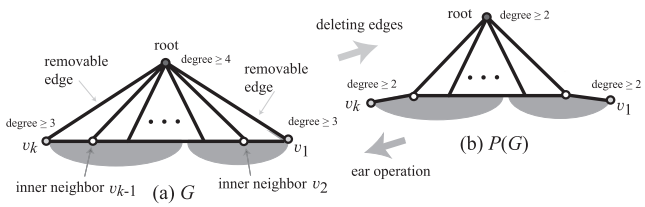


Fig. 4 (a) Block G with the 3-inner leftmost and rightmost neighbours of r_G; (b) Parent $\mathcal{P}_{\mathcal{B}}(G)$.

In a block G, an edge is called *removable* if the block obtained from G by deleting the edge remains biconnected Fig. 5 (a) and (b), and a vertex v is called *removable* if $deg(v; G) = 2$ and the block obtained from G by removing v together with the two incident edges remains biconnected (Fig. 5 (c)). Let β denote the boundary of G, and let $\beta[u, v]$ denote the path from a vertex u to a vertex v obtained by traversing β in the clockwise order. We call such a path $\beta[u, v]$. Let $\beta'[u, v]$ denote the path obtained by traversing the boundary of G from u to v in the anti-clockwise order.

Lemma 2: Let v_i and v_j $(i < j)$ be two neighbours of the root. If $E(\beta[v_i, v_j]) - E(F(G)) \neq \emptyset$, then path $\beta[v_i, v_j]$ contains a removable edge or vertex.

Proof. If $V(\beta[v_i, v_j]) - V(F(G))$ contains a vertex of degree 2, then we see that such a vertex is an outer vertex and is removable since its two neighbours are joined by an edge. Hence it suffices to show that if all edges in $E(\beta[v_i, v_j]) - E(F(G))$ are irremovable then $V(\beta[v_i, v_j]) - V(F(G))$ contains a vertex of degree 2. Since all inner faces are triangulated in G, removal of any irremovable edge $e \in E(\beta[v_i, v_j]) - E(F(G))$ creates exactly one cut-vertex, which separates G into two subgraphs G' and G'' such that they share only the cut-vertex and one of them,

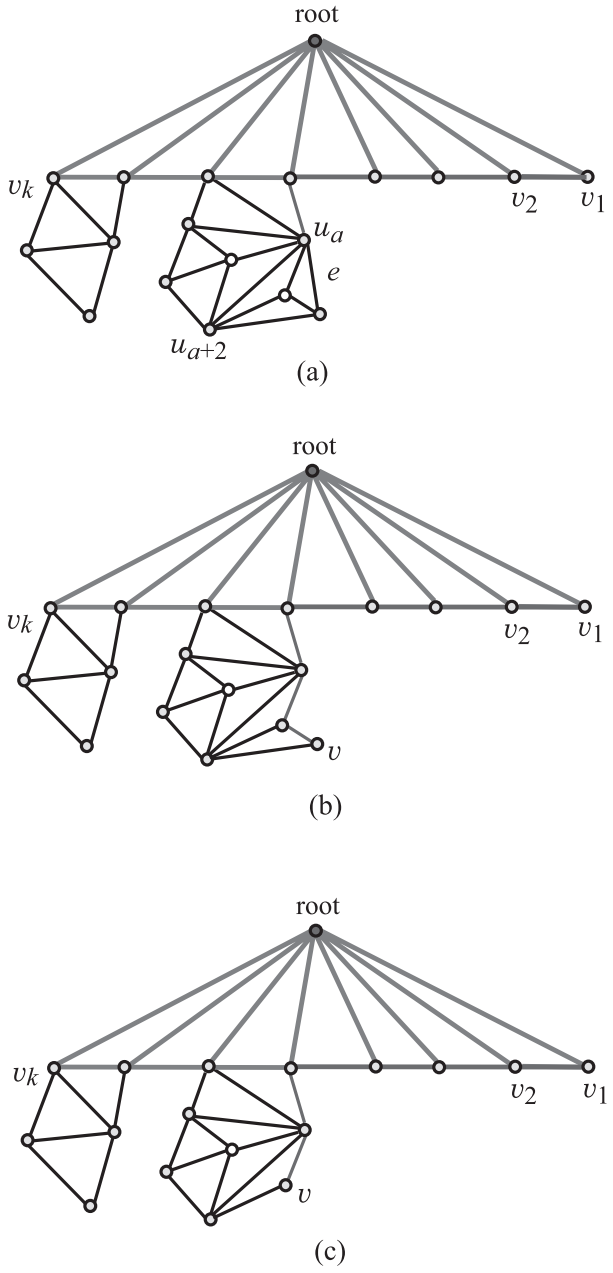


Fig. 5 (a), (b) Removable edges e; (c) A removable vertex v.

say, G' is disjoint with $F(G)$. Let $e = (u, u')$ be an edge in $E(\beta[v_i, v_j]) - E(F(G))$ that minimizes the number of vertices in such a subgraph G', where $u' \in V(G')$ is assumed without loss of generality. We claim that $deg(u'; G) = 2$. Assume that $deg(u'; G) \geq 3$, i.e., G' contains an edge $e' = (u', u'') \in E(\beta[v_i, v_j]) - E(F(G))$ which is adjacent to e. Note that all inner faces in G' are triangulated. Since e' belongs to a triangle inner face of G', the cut-vertex created by removing e' is contained in G', which contradicts the choice of e. Therefore, $deg(u'; G) = 2$, as required. ∎

We introduce a parent-child relationship among blocks in $\mathcal{B}$. We first choose *seed* blocks, for which no parent is defined, indicating that each of seed blocks will be generated before enumerating other blocks in $\mathcal{B}$

1. Seed blocks H_0, H_1, and H_2 are defined to be F_2, F_3, and F_4, respectively. See Fig. 1 (b) and (c) for H_1 and H_2
2. Seed block H_3: The block obtained from F_4 by adding a vertex of degree 2 adjacent to the leftmost and second leftmost neighbours of the root. See Fig. 1 (d).
3. Seed block H_4: The block obtained from F_5 by adding an edge between the leftmost and the second rightmost neighbours of the root. See Fig. 1 (e).

Note that each seed block H_i with $i = 0, 1, 2$ is symmetric while H_i with $i = 3, 4$ is asymmetric.

A block G is represented by a seed block followed by a sequence of operations.

When G is not a seed block, we define the *parent* $\mathcal{P}_{\mathcal{B}}(G)$ of each block G so that $\mathcal{P}_{\mathcal{B}}(G)$ remains symmetric (resp., asymmetric) if G is symmetric (resp., asymmetric). If a block G is pseudo-symmetric, we define $\mathcal{P}_{\mathcal{B}}(G)$ by a reduction from G based on "bilateral" operations, where G may not be symmetric. If G is not pseudo-symmetric, then we use "unilateral" operations to keep the asymmetry of G.

Raising an edge (v_i, v_{i+1}) between two adjacent neighbours of the root r_G of a block G is an operation that replaces edge (v_i, v_{i+1}) with a new edge (r_G, u) for the nonroot vertex u in the inner face (u, v_i, v_{i+1}).

For the next five cases, we introduce "bilateral" operations.

1. $deg(r_G; G) = 2$ for the root r_G, and $deg(v_2; G) \geq 3$ and $deg(v_1; G) \geq 3$ for the leftmost and rightmost neighbours v_2 and v_1 of r_G, as shown in Fig. 6 (b), (d), (f) and (i): Then $\mathcal{P}_{\mathcal{B}}(G)$ is defined to be the block obtained by raising edge (v_2, v_1), as shown in Fig. 6 (c), (e), (g) and (j).
2. $deg(r_G; G) = 3$, and the middle neighbour v_2 of r_G is an inner vertex: Then $\mathcal{P}_{\mathcal{B}}(G)$ is defined to be the block obtained by removing the root together with the three incident edges, designating v_2 as the new root, as shown in Fig. 6 (g) and (h).
3. $deg(r_G; G) \geq 3$, and G is a pseudo-symmetric block such that the leftmost and rightmost neighbours v_k and v_1 of the root are both 3-outer: Let v_{k-1} and v_2 be the second leftmost and second rightmost neighbours of the root r_G. Then $\mathcal{P}_{\mathcal{B}}(G)$ is defined to be the block obtained by raising the edges $e_L = (v_k, v_{k-1})$ and $e_R = (v_2, v_1)$, as shown in Fig. 2.
4. $deg(r_G; G) \geq 4$, and G is a pseudo-symmetric block such that the leftmost and rightmost neighbours v_k and v_1 of the root r_G are both 2-outer, where vertices v_k and v_1 are removable: Then $\mathcal{P}_{\mathcal{B}}(G)$ is defined to be the block obtained by removing v_k and v_1 together with the incident edges, as shown in Fig. 3.
5. $deg(r_G; G) \geq 4$, and G is a pseudo-symmetric block such that the leftmost and rightmost neighbours v_k and v_1 of the root r_G are both 3-inner, where edges (r_G, v_k)

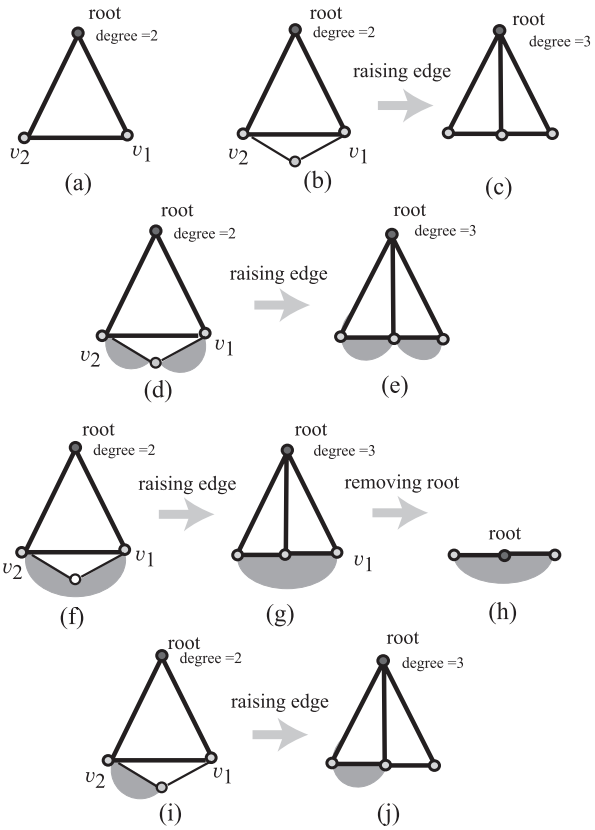


Fig. 6 Parents of blocks with small cores.

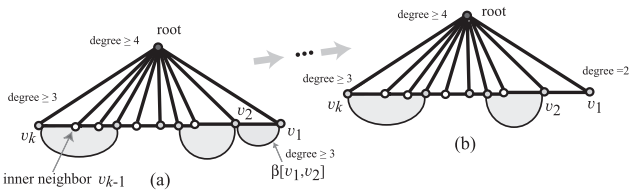


Fig. 7 (a) Block G with the 3-inner leftmost neighbour and 3-outer rightmost neighbour of r_G; (b) An ancestor of G.

and (r_G, v_1) are removable: Then $\mathcal{P}_{\mathcal{B}}(G)$ is defined to be the block obtained by deleting the removable edges (r_G, v_k) and (r_G, v_1), as shown in Fig. 4.

For the next three cases, we introduce "unilateral" operations.

1. $deg(r_G; G) \geq 4$, and the leftmost and rightmost neighbours of the root are 3-inner and 3-outer, respectively: Let v_1 and v_2 denote the rightmost and second rightmost neighbours of the root, respectively. Then $\mathcal{P}_{\mathcal{B}}(G)$ is defined to be the block obtained by deleting the first removable element (edge or vertex) on the path $\beta'[v_1, v_2]$ from v_1 to v_2, as shown in Fig. 7 (a).
2. $deg(r_G; G) \geq 3$, $deg(v_k; G) \geq 3$ for the leftmost neighbour v_k of the root r_G, the rightmost neighbour v_1 is 2-outer, and G has more than one noncore inner face: Then $\mathcal{P}_{\mathcal{B}}(G)$ is defined to be the block obtained by deleting the first removable element on the path $\beta[v_2, v_k]$ from the second rightmost neighbour v_2 of r_G to v_k, as shown in Fig. 8 (a) and (b).
3. $deg(r_G; G) \geq 3$, $deg(v_k; G) \geq 3$ for the leftmost neighbour v_k of the root r_G, the rightmost neighbour v_1 is 2-outer, and G has exactly one noncore inner face: Then $\mathcal{P}_{\mathcal{B}}(G)$ is defined to be the block obtained by deleting the rightmost neighbour v_1 of the root r_G, as shown Fig. 8 (c) to (e) and Fig. 8 (d) to (f).

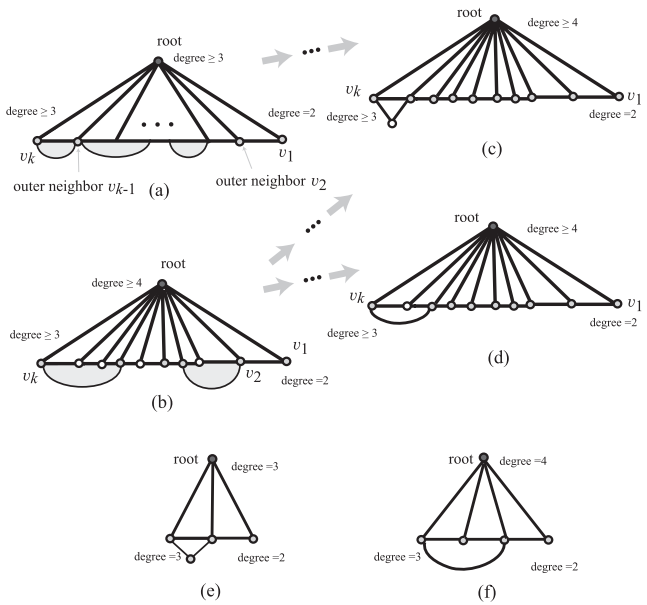


Fig. 8 (a) The leftmost neighbour v_k is 3-inner and the rightmost neighbour v_1 is 2-outer; (b) The leftmost neighbour v_k is 3-outer and the rightmost neighbour v_1 is 2-outer; (c) G has exactly one noncore inner face; (d) G has exactly one noncore inner face; (e) H_4; (f) H_3.

It is easy to see that any non-seed block G satisfies exactly one of the above cases, implying that the parent $\mathcal{P}_{\mathcal{B}}(G)$ of a non-seed block G is unique.

A block G' is called a *child* of G if $G = \mathcal{P}_{\mathcal{B}}(G')$. Let $\mathcal{C}_{\mathcal{B}}(G)$ denote the set of all children of G. Note that if G is symmetric then any child of G is also symmetric.

4. Generating Children of Symmetric Blocks

In this section, we see that all the symmetric blocks can be generated by performing only "bilateral" operations starting from symmetric seed blocks H_i, $i = 1, 2$. Actually, the following five bilateral operations are just the reverse operation for generating parents (in different order).

Sinking an edge (r_G, v) between the root r_G and an inner neighbour v in a block G is an operation that replaces the edge (v, r_G) with a new inner edge (v', v'') for the two common neighbours of v and r_G, where G is not allowed to contain edge (v', v'') before sinking (v, r_G).

We introduce the following bilateral operations.

1. `Dual-wing` (`d-wing` for short): Create two new vertices v_L and v_R as the new leftmost and rightmost neighbours of r_G in the resulting block G' by adding four

new edges (v_L, r_G), (v_L, v_k), (v_R, r_G) and (v_R, v_1) for the leftmost and rightmost neighbours v_k and v_1 of r_G in G (Fig. 3).

2. `Stretch`: Add a new vertex v together with three new edge (v, r_G), (v, v_1) and (v, v_k) joining v to the root r_G, the leftmost and rightmost neighbours v_k and v_1 of r_G, where the root of the resulting block G' is designated by v (Fig. 6 (h) to (g)).
3. `Ear`: Applicable only when the second vertex u_2 and the second last vertex u_{p-1} in the path $\beta[v_1, v_k]$ are not adjacent to the root r_G. Add two new outer edges (r_G, u_2) and (r_G, u_{p-1}). The core $F(G')$ of the resulting block G' contains two more triangles than $F(G)$ (Fig. 4).
4. `Sink`: Applicable only when $deg(r_G; G) = 3$ and the rightmost and leftmost neighbours of r_G are not adjacent. Sink the unique inner edge incident to the root r_G (Fig. 6 (c) to (b), (e) to (d), (g) to (f) and (j) to (i)).
5. `Dual-sink` (`d-sink` for short): Applicable only when $deg(r_G; G) \geq 5$ and the third rightmost neighbour v_3 (resp., third leftmost neighbour v_{k-2}) of r_G is an outer vertex not adjacent to rightmost neighbour v_1 (resp., the leftmost neighbour v_k). Sink two edges (v_{k-1}, r_G) and (v_2, r_G) (Fig. 2).

We generate $H_0 = F_2$ separately from the other blocks in $\mathcal{B}$. To generate all symmetric blocks in $\mathcal{B} - \{H_0\}$, we start with one of the symmetric seed blocks H_1 and H_2 and perform only bilateral operations `stretch`, `d-wing`, `ear`, `sink` and `d-sink`. For any symmetric block $G \in \mathcal{B} - \{H_0\}$, the set $C_{\mathcal{B}}(G)$ of children is given by the set of all blocks that can be constructed from G by applying one of `stretch`, `d-wing`, `ear`, `sink` and `d-sink`. Thus, by considering applicability of these bilateral operations on a given block G, we obtain the following algorithm for computing all descendant of G recursively.

Procedure BILATERAL(G)
Input: A block G with at most n vertices.
Output: All descendant of G with at most n vertices that can be obtained from G by applying bilateral operations.
begin
  /* Let $v_1, v_2, \ldots, v_k$ be the neighbours of the root r_G of G, where $k = deg(r_G)$ in the order from left to right. */
  if $deg(r_G; G) = 3$ and $(v_1, v_k) \notin E(G)$ **then**
    Let G' be the block obtained from G by applying `sink`; BILATERAL(G')
  endif;
  if $deg(v; G) \geq 5$, v_3 and v_{k-2} are outer vertices, and $(v_1, v_3), (v_k, v_{k-2}) \notin E(G)$ **then**
    Let G' be the block obtained from G by applying `d-sink`; BILATERAL(G')
  endif;
  if the nonroot outer neighbour u_{p-1} (resp., u_2) of v_k (resp., v_1) is not adjacent to the root r_G **then**
    Let G' be the block obtained from G by applying `ear`; BILATERAL(G')
  endif;
  if $|V(G)| = n$ **then** return
  else
    Let G' be the block obtained from G by applying `stretch`; BILATERAL(G');
    if $|V(G)| = n - 1$ **then** return
    else
      Let G' be the block obtained from G by applying `d-wing`; BILATERAL(G'); return
  endif endif
  Return
end.

Note that only a constant number of outer vertices increases/decreases when a child G' of G is generated. It is not difficult to implement BILATERAL(G) so that each line of the procedure can be executed in $O(1)$ time and $O(|V(G)|)$ space except for testing if $(v_1, v_k) \notin E(G)$ or $(v_1, v_3), (v_k, v_{k-2}) \notin E(G)$. Each of such edges $(v_1, v_k), (v_1, v_3), (v_k, v_{k-2}) \in E(G)$ to tested is an edge such that each of its end-vertices is a nonroot outer vertex, which we call an *exposed-edge*. We show that whether given two outer vertices u and v has an exposed-edge between them or not can be tested in $O(1)$ time in Sect. 6.

5. Generating Children of Asymmetric Blocks

In this section, we show that all asymmetric blocks can be generated by both "unilateral" and bilateral operations starting from asymmetric seed blocks H_i, $i = 3, 4$.

To generate asymmetric blocks, we start with one of the asymmetric seed blocks H_3 and H_4 and perform both bilateral operations and/or unilateral operations, each of which never generates a symmetric block from any asymmetric block.

We further introduce the following three unilateral operations to construct a child from a given asymmetric block. These three operations are the reverse operation for generating asymmetric parents.

1. `Wing`: Create a new vertex v_R together with two new edges (v_R, r_G) and (v_R, v_1) for the rightmost neighbour v_1 of the root r_G of G. The vertex v_R is the rightmost neighbour of the root $r_{G'} = r_G$ of the resulting block G' (Fig. 8 (e) to (c) and (f) to (d)).
2. `v-add`: Create a new outer vertex v together with two new edges (v, u) and (v, u') for an outer edge $e = (u, u')$ in G. The vertex v is said to be `v-added` to edge $e = (u, u')$.
3. `e-add`: For two adjacent outer edges (u_a, u_b) and (u_b, u_c) such that the two end vertices u_a and u_c are not adjacent, add a new outer edge (u_a, u_c). The edge (u_a, u_c) is said to be `e-added` to u_b.

Let $v_1 = v_R, v_2, \ldots, v_k = v_L$ be the neighbours of the root which appear in this order from right to left, and denote

the sequence of vertices in $\beta[v_1, v_k]$ by

$$\beta[v_1, v_k] = (u_1 = v_1, u_2, \ldots, u_p = v_k).$$

For a block G such that v_2 is an outer vertex, denote the sequence of vertices in $\beta'[v_2, v_1]$ by $u'_1 = v_2, u'_2, \ldots, u'_q = v_1$. We define the *right bank-path* of G to be the path $\beta'[u'_1, u'_{s+1}]$ if the first removable element on $\beta'[v_2, v_1]$ is an edge (u'_s, u'_{s+1}) (resp., a vertex u'_s), where we say that the right bank-path ends up with a removable edge (resp., vertex) (Fig. 9 (a)).

Note that `wing` is applicable to any non-pseudo-symmetric block G to obtain a child of G. On the other hand, `e-add` (resp., `v-add`) is applicable to an outer vertex (resp., an outer edge) only when the newly added edge (resp., vertex) by `e-add` (resp., `v-add`) is the first removable element $\beta'[v_2, v_1]$ or $\beta[v_2, v_k]$ in the resulting block.

(I) Operation `e-add` to the ith vertex u'_i in the right bank-path is allowed to be performed if and only if the second leftmost neighbour of the root is an outer vertex, u'_{i-1} and u'_{i+1} are not adjacent, and (i) $2 \le i \le p$ or (ii) $i = p + 1$ and the first removable element is an edge.

(II) Operation `v-add` to the ith edge (u'_i, u'_{i+1}) in the right bank-path is allowed to be performed if and only if $1 \le i \le p$ and the second leftmost neighbour of the root is an outer vertex.

In (II), the newly introduced vertex is clearly the first removable element in the right bank-path in the resulting block G'. In (I), we show that the newly introduced edge $e = (u'_{i-1}, u'_{i+1})$ is the first removable element in the right bank-path in the resulting block G'. For this, it suffices to prove that no irremovable edge $e' = (u'_{j-1}, u'_j)$ with $j \le i - 1$ never becomes removable in G'. If such edge $e' = (u'_{j-1}, u'_j)$ exists, then G has a triangle inner face (u'_{j-1}, u'_j, u'_i) and we see that $\beta'[u'_j, u'_i]$ contains a removable element, as in the proof of Lemma 2, contradicting the choice of p.

We also denote the sequence of vertices in $\beta[v_2, v_k]$ by $u_1 = v_2, u_2, \ldots, u_m = v_k$, and define the *left bank-path* of G to be path $\beta[u_1, u_{p+1}]$ if the first removable element on $\beta[v_2, v_k]$ is an edge (u_p, u_{p+1}) (resp., a vertex u_p), where we say that the left bank-path ends up with a removable edge (resp., vertex) ((Fig. 9 (b) and (c)).

(III) Operation `e-add` to the ith vertex u_i in the left bank-path is allowed to be performed if and only if the second rightmost neighbour of the root is an outer vertex, u_{i-1} and u_{i+1} are not adjacent, and (i) $2 \le i \le p$ or (ii) $i = p + 1$ and the first removable element is an edge.

(IV) Operation `v-add` to the ith edge (u_i, u_{i+1}) in the left bank-path is allowed to be performed if and only if $1 \le i \le p$ and the right bank-path contains no outer vertex.

For (III) and (IV), we see that the newly introduced vertex/edge is the first removable element in the left bank-path in the resulting block, as observed in (I) and (II).

Note that once one of `e-add` and `v-add` is performed on the right bank-path, none of `e-add` and `v-add` is performed on the left bank-path in the resulting child.

Given a block G, the following shows what operations can be applied to G.

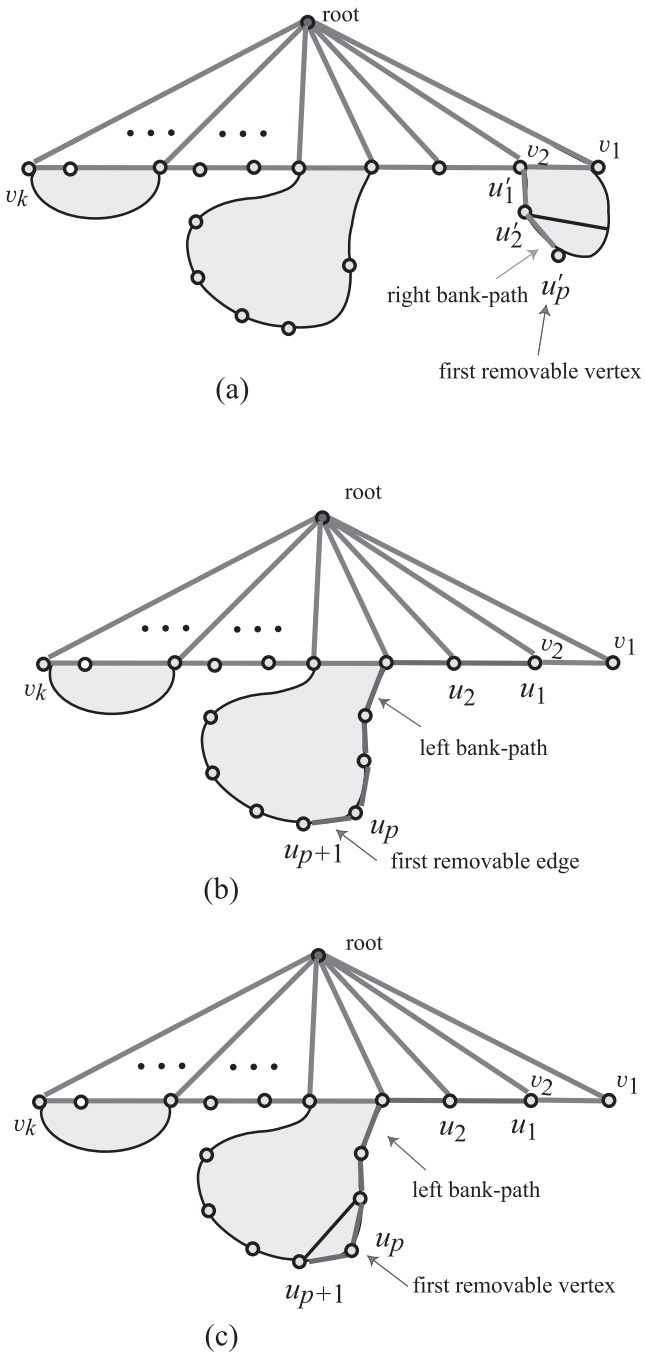


Fig. 9 (a) The first removable vertex in the right bank-path; (b) The first removable edge in the left bank-path; (c) The first removable vertex in the left bank-path.

Case-1. An asymmetric block G is not pseudo-symmetric and has exactly one non-core inner face: Then the set $C_{\mathcal{B}}(G)$ consists of the blocks that can be constructed from G by any applicable operation from `wing`, `v-add`, `e-add`, `stretch`, `d-wing` and `d-sink`.

Case-2. An asymmetric block G is not pseudo-symmetric and contains at least two non-core inner faces: the set $C_{\mathcal{B}}(G)$ consists of the blocks that can be constructed from G by any

applicable operation from `v-add`, `e-add`, `sink`, `stretch`, `d-wing`, `ear` and `d-sink`.

Case-3. An asymmetric block G is pseudo-symmetric: the set $C_{\mathcal{B}}(G)$ consists of the blocks that can be constructed from G by any applicable bilateral operation from `sink`, `stretch`, `d-wing`, `ear` and `d-sink`.

Any bilateral operation in `sink`, `stretch`, `d-wing`, `ear` and `d-sink` makes a block pseudo-symmetric. Hence once a bilateral operation is applied to a block in Case-1 or 2, the resulting block G' satisfies Case-3 and any descendant of G' remains pseudo-symmetric after applying any sequence of bilateral operations. Also once one of `v-add`, `e-add`, `stretch`, `d-wing` and `d-sink` is applied to a block G in Case-1, `wing` will never be applied to the resulting block G' and any descendant of G' never satisfies Case-1.

Procedure UNILATERAL(G)
Input: An asymmetric block G with at most n vertices.
Output: All descendants of G with at most n vertices.
begin
  BILATERAL(G);
  if the right bank-path contains no outer vertex (none of `v-add` and `e-add` has been applied to the right bank-path) **then**
    /* Let $(u_1, u_2, \ldots, u_{p+1})$ denote the left bank-path of G. */
    for $i = 2, 3, \ldots, p+1$ **do**
      Let G' be the block obtained from G by applying `e-add` to vertex u_i according to rule (I); UNILATERAL(G')
    endfor
  endif;
  /* Let $(u'_1, u'_2, \ldots, u'_{p'+1})$ denote the right bank-path of G. */
  if the second leftmost neighbour of the root of G is an outer vertex **then**
    for $i = 2, 3, \ldots, p'+1$ **do**
      Let G' be the block obtained from G by applying `e-add` to vertex u'_i according to rule (III); UNILATERAL(G')
    endfor
  endif;
  if $|V(G)| = n$ **then** return **endif**;
  if G has exactly one noncore inner face **then**
    Let G' be the block obtained from G by applying `wing`; UNILATERAL(G');
  endif;
  if the right bank-path contains no outer vertex (none of `v-add` and `e-add` has been applied to the right bank-path) **then**
    /* Let $(u_1, u_2, \ldots, u_{p+1})$ denote the left bank-path of G. */
    for $i = 1, 2, \ldots, p$ **do**
      Let G' be the block obtained from G by applying `v-add` to edge (u_i, u_{i+1}) according to rule (II); UNILATERAL(G')
    endfor
  endif;
  /* Let $(u'_1, u'_2, \ldots, u'_{p'+1})$ denote the right bank-path of G. */
  if the second leftmost neighbour of the root of G is an outer vertex **then**
    for $i = 1, 2, \ldots, p'$ **do**
      Let G' be the block obtained from G by applying `v-add` to edge (u'_i, u'_{i+1}) according to rule (IV); UNILATERAL(G')
    endfor
  endif;
  Return
end.

By maintaining each of the left and right bank-paths, it is not difficult to implement UNILATERAL so that each line of the procedure can be executed in $O(1)$ time.

6. Algorithm and Analysis

The entire algorithm is described as follows.

Algorithm GENTRIANGLE(n)
Input: An integer $n \geq 2$.
Output: All rooted triangulated blocks with at most n vertices.
begin
  Let $G := H_0$;
  for $i = 1, 2, 3, 4$ **do**
    if $|V(H_i)| \leq n$ **then** $G := H_i$;
      if $G = H_i$ is symmetric (i.e., $i \in \{1, 2\}$) **then**
        BILATERAL(G)
      else
        UNILATERAL(G)
      endif
    endif
  endfor
end.

For the recursive process, we use the odd-even output method [6]. Output before the recursion if the recursive depth is odd, otherwise output after the recursion. Hence there will only be gap of $O(1)$ time during the recursion.

In what follows, we show how to test whether given two outer vertices u and v has an exposed-edge between them.

Lemma 3: Let G be a block constructed during an execution of GENTRIANGLE(n), and let G_0 be the ancestor of G constructed after finishing all applications of unilateral operations. If $deg(r_G; G)$ is odd, then the middle neighbour v_M of r_G in G was the root of an ancestor of G or the middle neighbour of the root of G_0.

Proof. Immediate from construction of blocks by symmetric operations. ∎

For a vertex v in a block G constructed constructed during an execution of GenTriangle(n), let `ear`(v) denote an application of operation `ear` by which v becomes adjacent to the root of G or its ancestor, and let `sink`(v) (resp., `d-sink`(v)) denote an application of operation `sink` (resp., `d-sink`) by which an edge joining v and the root of G or its ancestor is eliminated.

We observe that a vertex created as a core vertex during an execution of GenTriangle(n) can change into a non-core vertex only by applying `sink`(v), `d-sink`(v) or `stretch`, and that a non-core vertex can change into a core vertex only by applying `ear`(v) to the vertex.

We easily see that an exposed-edge in a block originally belongs to a seed block, or can be introduced only by operations `wing`, `v-add`, `e-add`, `sink`, `d-wing` and `d-sink`. We show how to store all exposed-edges created during an execution of GenTriangle(n) as long as they remain exposed-edges in the current block.

For two adjacent vertices u and v, define `mark`(u, v) to be a procedure that assigns the directed edge (u, v) to u as an *exposed-edge* of vertex u. If an exposed-edge is marked by both end vertices, then there may exist some vertex with $O(n)$ exposed-edges. To avoid this, for most of the exposed-edges we only let one of its vertices remember it. We denote the set of expose-edges assigned to a vertex u by $Ex(u)$, where some edge in $Ex(u)$ may be replaced with another edge and some edge in $Ex(u)$ may become a non-exposed-edge during an execution of GenTriangle(n). However we will show that, for a block G at any stage of an execution of GenTriangle(n), $|Ex(u)| \leq 4$ holds for all $u \in V(G)$, and all exposed-edges in G are contained in the union of $Ex(u)$, $u \in V(G)$. This enables us to test whether two outer vertices has an exposed-edge between them in $O(1)$ time as long as $Ex(u)$, $u \in V(G)$ can be updated in $O(1)$ time per bilateral/unilateral operation.

We show how to maintain $Ex(u)$, $u \in V(G)$ as follows.

- When G is set to be the seed block H_1, we apply `mark`(v_2, v_1) for the leftmost and rightmost neighbours v_1 and v_2 of the root (see Fig. 10 (a)).
- When G is set to be the seed block H_2, we apply `mark`(v_1, v_2) and `mark`(v_3, v_2) for the leftmost, second rightmost and rightmost neighbours v_1, v_2 and v_3 of the root (see Fig. 10 (b)).
- When G is set to be the seed block H_3, we apply `mark`(v_1, v_2), `mark`(v_3, v_2), `mark`(u, v_2) and `mark`(u, v_3) for the leftmost, second rightmost and rightmost neighbours v_1, v_2 and v_3 of the root and the unique non-core vertex u (see Fig. 10 (c)).
- When G is set to be the seed block H_4, we apply `mark`(v_4, v_2) and `mark`(v_1, v_2) for neighbours v_1, v_2, v_3 and v_4 of the root from right to left (see Fig. 10 (d)).
- When `wing` is applied in a block with the rightmost and second rightmost neighbours v_1 and v_2 of the root, we apply `mark`(v, v_1) to the resulting new edge joining v_1 and the new rightmost neighbour v, and `mark`(v_2, v_1), deleting edge (v_1, v_2) from $Ex(v_1)$.

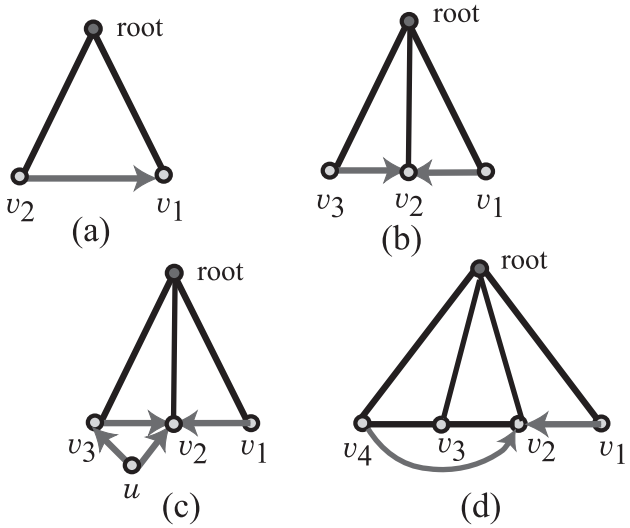


Fig. 10 Storing exposed-edges in seed blocks H_i, $i = 1, 2, 3, 4$.

- When `v-add` is applied to an edge (u_i, u_{i+1}) to create a new non-core vertex v in an execution of Unilateral(G), we apply `mark`(v, u_i) and `mark`(v, u_{i+1}).
- When `e-add` is applied to a vertex u_i to create a new edge (u_{i+1}, u_{i-1}) in an execution of Unilateral(G), we replace $(u_{i+1}, u_i) \in Ex(u_{i+1})$ with (u_{i+1}, u_{i-1}).
- When `d-wing` is applied to G, we apply `mark`(v, v_k) and `mark`(v', v_1) to the two new vertices v and v', where v_1 and v_k are the leftmost and rightmost neighbours of G.
- When `d-sink` is applied to G, where v_k, v_{k-1} and v_{k-2} denote the first, second and third leftmost neighbours of the root, we apply `mark` as follows: replace edge (v_k, v_{k-1}) in $Ex(v_k)$ with (v_k, v_{k-2}) and `mark`(v_{k-1}, v_k). For the three rightmost neighbours of the root, we apply the same procedure symmetrically.
- When `sink` is applied, where v_1, v_2 and v_3 denote the first, second and third leftmost neighbours of the root, we apply `mark` in the same manner of `d-sink`.
- When `ear` is applied to G, where v_1 (resp., v_k) denote the rightmost (resp., leftmost) neighbours of the root, we delete any edge in $Ex(v_1)$ (resp., $Ex(v_k)$) one of which end-vertices is an inner vertex at this point.

For a core vertex v in a block, let `sink`(v) (resp., `d-sink`(v)) denote an application of `sink` (resp., `d-sink`) that removes the edge between v and the root of the block. Let `ear`(v) denote an application of `ear` by which v becomes adjacent to the root.

Lemma 4: Let $G \in \mathcal{B}$ be a block constructed during an execution of GenTriangle(n). Then two outer vertices u and u' are adjacent if and only if $(u, u') \in Ex(u)$ or $(u, u') \in Ex(u')$ holds. Moreover $|Ex(v)| \leq 4$ holds for every vertex v.

Proof. It is easy to see by induction that during an execution of GenTriangle(n) each exposed-edge in the current block is stored in $Ex(v)$ of a vertex v.

We show the second part of the lemma. After setting the initial block as a seed block, $|Ex(u)| \leq 1$ for each core vertex u and $|Ex(u)| \leq 2$ for the unique non-core vertex u (if any). Applying `v-add` creates a new non-core vertex v with $|Ex(v)| = 2$, without changing the set size of $Ex(u)$ of any other existing vertex u. When a new exposed-edge is added to $Ex(v)$ for a vertex v by `e-add`, one of the edges $Ex(v)$ (which is no longer exposed) will be deleted from $Ex(v)$, without changing the set size of $Ex(u)$ of each existing vertex u. Applying `wing` creates a new core vertex v with $|Ex(v)| = 1$, without changing the set size of $Ex(u)$ of any other existing vertex u. After setting an initial block to be a seed block and applying `wing`, `e-add` and `v-add`, we see that (i) $|Ex(v)| \leq 1$ for every core vertex v; and (ii) $|Ex(v)| \leq 2$ for every non-core vertex v.

We show that $|Ex(v)|$ can increase at most by 2 after applying any sequence of symmetric operations.

We see that `sink`(v) is applied at most once since v can be the middle neighbour of the root at most once by Lemma 3.

When `d-sink`(v) is applied for the first time, $|Ex(v)|$ increases by 1. After this, `d-sink`(v) may be applied many times as long as v remains an outer vertex. We show that at least one edge in $Ex(v)$ will be deleted by an operation `ear`(v) between any two applications of `d-sink`(v). Let (v, u) be the edge added to $|Ex(v)|$ by the first of these applications of `d-sink`(v) (see Fig. 11 (a), (b)). We observe that u appears before v along the boundary $\beta'[v_L, v_R]$ from the leftmost neighbour v_L to the rightmost neighbour v_R in any block (see Fig. 11 (c)). Note that u cannot appear after v along the boundary of any block constructed later, since no operation can change the order of any two outer vertices along the boundary.

Before the next application of `d-sink`(v), vertex v must be an outer core vertex, which implies that v remains an outer vertex after the first of application of `d-sink`(v).

If (v, u) remains an exposed-edge, then u cannot appear after v along the boundary $\beta[v_L, v_R]$ from the leftmost neighbour v_L to the rightmost neighbour v_R in any block. This means that u is an inner vertex after `ear`(v) is applied, and that edge (v, u) will be deleted from $Ex(v)$ by the application of `ear`(v).

Therefore, $|Ex(v)|$ never exceeds 4. This proves the second part of the lemma. ∎

We briefly show that $O(n)$ space suffices to implement GenTriangle(n). Define *signature* $\gamma(G)$ of each block G to be the sequence of operations that generates G. We show that the length of $\gamma(G)$ is $O(n)$. The total number of applications of `sink` and `d-sink` are bounded from above by that of `wing`, `d-wing`, `stretch`, `v-add`, `e-add` and `ear`. The total number of applications of the operations is bounded by $|V(G)| + |E(G)|$ since each of those operations introduce at least one new vertex or edge. Since $|E(G)| = O(|V(G)|)$ for planar graphs G, the length of $\sigma(G)$ is $O(n)$.

Each operation done to a parent block G adds or changes at most two vertices and at most four edges. Hence it costs $O(1)$ time each operation. Form a parent to a child takes one operation from a child to the parent also takes only one reverse operation. Thus all rooted triangulated planar blocks can be generated in constant time.

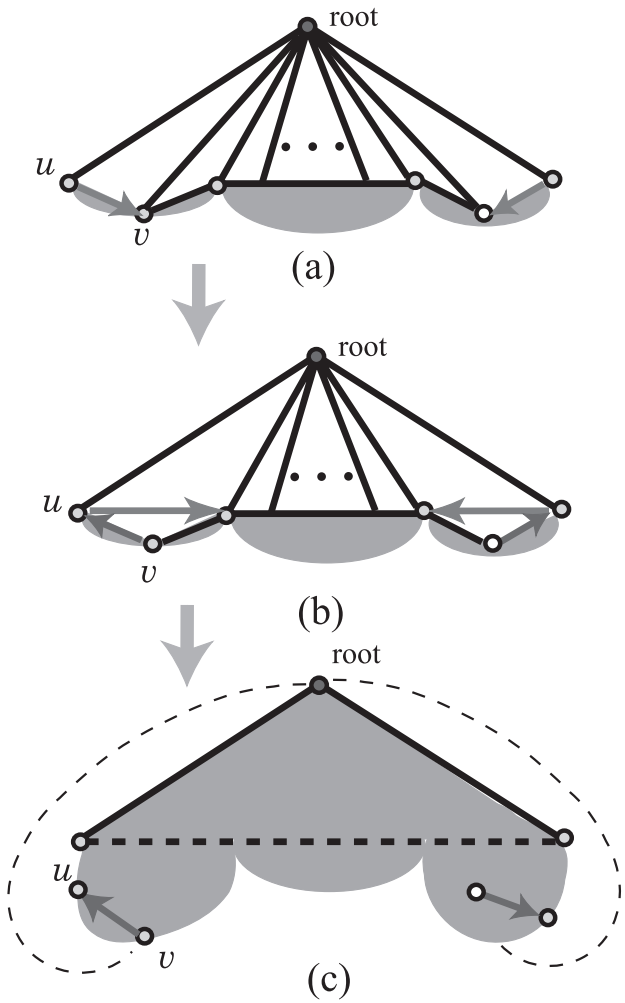


Fig. 11 (a), (b) Application of `d-sink`(v) for a vertex v; (c) An exposed-edge (v, u).

Theorem 5: For an integer $n \geq 2$, all blocks in $\mathcal{B}$ with at most n vertices can be generated without duplication in $O(n)$ space by an algorithm that outputs the difference between two consecutive blocks in $O(1)$ time in a series of all outputs.

7. Concluding Remarks

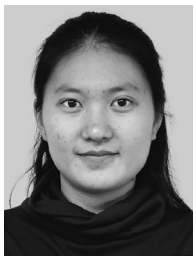
In this paper, we gave an enumeration algorithm for the class of rooted *biconnected* planar graphs with internally triangulated faces. By combining the algorithm with the general framework for rooted connected graphs [11], [12], rooted *connected* planar graphs with internally triangulated faces can also be generated efficiently.

Unlike the previous approach to enumeration of graphs with reflectional symmetry such as the algorithm for rooted

outerplanar graphs [9], our algorithm never compares any two encoded subgraphs. It is our future work to design new enumeration algorithms for other classes of blocks using bi- and unilateral operations on pseudo-symmetry.

References

[1] D. Avis and K. Fukuda, "Reverse search for enumeration," Discrete Appl. Math., vol.65, pp.21–46, 1996.
[2] H. Fujiwara, J. Wang, L. Zhao, H. Nagamochi, and T. Akutsu, "Enumerating tree-like chemical graphs with given path frequency," J. Chem. Inf. Mod., vol.48, pp.1345–1357, 2008.
[3] T. Imada, S. Ota, H. Nagamochi, and T. Akutsu, "Enumerating stereoisomers of tree structured molecules using dynamic programming," ISAAC 2009, LNCS 5878, pp.14–23. 2009.
[4] Y. Ishida, L. Zhao, H. Nagamochi, and T. Akutsu, "Improved algorithm for enumerating tree-like chemical graphs," GIW2008, Genome Informatics, vol.21, pp.53–64, 2008.
[5] Z. Li and S. Nakano, "Efficient generation of plane triangulations without repetitions," ICALP 2001, LNCS 2076, pp.433–443, 2001.
[6] S. Nakano, "Efficient generation of triconnected plane triangulations," Comput. Geom., Theory Appl., vol.27, no.2, pp.109–122, 2004.
[7] S. Nakano and T. Uno, "Efficient generation of rooted trees," NII Technical Report, NII-2003-005, 2003.
[8] S. Nakano and T. Uno, "Constant time generation of trees with specified diameter," Proc. WG 2004, Lect. Notes Comput. Sci., vol.3353, pp.33–45, 2004.
[9] J. Wang and H. Nagamochi, "Constant time generation of rooted and colored outerplanar graphs," Dept. of Applied Mathematics and Physics, Kyoto University, Technical Report 2010-007, 2010. http://www-or.amp.i.kyoto-u.ac.jp/members/nag/Technical_report/TR2010-007.pdf
[10] B. Zhuang and H. Nagamochi, "Enumerating rooted biconnected planar graphs with internally triangulated faces," Dept. of Applied Mathematics and Physics, Kyoto University, Technical Report 2009-018, 2009. http://www-or.amp.i.kyoto-u.ac.jp/members/nag/Technical_report/TR2009-018.pdf
[11] B. Zhuang and H. Nagamochi, "Enumerating rooted graphs with reflectional block structures," Dept. of Applied Mathematics and Physics, Kyoto University, Technical Report 2009-019, 2009. http://www-or.amp.i.kyoto-u.ac.jp/members/nag/Technical_report/TR2009-019.pdf
[12] B. Zhuang and H. Nagamochi, "Enumerating rooted graphs with reflectional block structures," CIAC 2010, LNCS 6078, pp.49–60, 2010.
[13] B. Zhuang and H. Nagamochi, "Enumerating biconnected rooted plane graphs," Dept. of Applied Mathematics and Physics, Kyoto University, Technical Report 2010-001, 2010. http://www-or.amp.i.kyoto-u.ac.jp/members/nag/Technical_report/TR2010-001.pdf



Bingbing Zhuang was born in Guang Dong, China, on Auguest 15, 1985. She received the B.Sc. degree from the University of Hong Kong, Hong Kong, in 2008. She is a Master course student in the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University. Her research interests include graph algorithms and combinatorial design problems.

Hiroshi Nagamochi was born in Tokyo, on January 1, 1960. He received the B.A., M.E. and D.E. degrees from Kyoto University, in 1983, in 1985 and in 1988, respectively. He is a Professor in the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University. His research interests include network flow problems and graph connectivity problems. Dr. Nagamochi is a member of the Operations Research Society of Japan and the Information Processing Society.