A Simple Linear-Time Algorithm for Computing the Centroid and Canonical Form of a Plane Graph and Its Applications

Tatsuya Akutsu
Bioinformatics Center, Institute for Chemical Research, Kyoto University
Kyoto 611-0011, Japan
takutsu@kuicr.kyoto-u.ac.jp

Colin de la Higuera
LINA, UMR CNRS 6241, Universitéde Nantes
44322 Nantes Cedex 03, France
cdlh@univ-nantes.fr

Takeyuki Tamura
Bioinformatics Center, Institute for Chemical Research, Kyoto University
Kyoto 611-0011, Japan
tamura@kuicr.kyoto-u.ac.jp

Abstract

We present a simple linear-time algorithm for computing the topological centroid and the canonical form of a plane graph. Although the targets are restricted to plane graphs, it is much simpler than the linear-time algorithm by Hopcroft and Wong for determination of the canonical form and isomorphism of planar graphs. By utilizing a modified centroid for outerplanar graphs, we present a linear-time algorithm for a geometric version of the maximum common connected edge subgraph (MCCES) problem for the special case in which input geometric graphs have outerplanar structures, MCCES can be obtained by deleting at most a constant number of edges from each input graph, and both the maximum degree and the maximum face degree are bounded by constants.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Plane graph, Graph isomorphism, Maximum common subgraph

Digital Object Identifier 10.4230/LIPIcs.CPM.2018.10

Funding TA was partially supported by JSPS KAKENHI #18H04113. TT was partially supported by JSPS KAKENHI #25730005. A part of CDLH's work was done when he was a visiting professor at Kyoto University.

1 Introduction

Comparison of geometric objects is an important topic in various fields including pattern recognition, computational geometry, and combinatorial pattern matching [7, 8, 17, 18]. In many cases, geometric objects are given as graphs having geometric information. Therefore, comparison of geometric objects are often modeled as pattern matching problems on graphs possibly with geometric information.

Among various problems on graph pattern matching, the most fundamental one is the *graph isomorphism* problem, which asks whether or not two given graphs are isomorphic. Although extensive studies have been done on determining the complexity class of graph

isomorphism, it is still unclear for general graphs [5], Polynomial-time algorithms are known for special graph classes, which include graphs of bounded degree [14] and graphs of bounded treewidth [16]. In particular, it is known that *planar graph isomorphism* can be tested in $O(n \log n)$ time [10] and in linear time [11], where n is the number of vertices on each given graph, and the former one was modified for testing the congruity of polyhedra [19]. Furthermore, both algorithms can be modified for computation of the *canonical form* of a given graph, where the canonical form is a unique representation of a graph that is invariant under isomorphic transformations. Although the algorithm in [10] is conceptually simple, that in [11] is complicated. In this paper, we focus on *plane graphs*, which is a planar graph with planar embedding, and present a (conceptually) simple linear-time algorithm for computing the canonical form of a plane graph. The algorithm first computes the topological *centroid* of a given graph, then transforms the graph into a *circular string*, and finally computes a canonical form of this circular string [6, 12]. Since it is known that there is a close relationship between planar and related graph isomorphism problems and the circular string problem [11, 15], the approach is reasonable. Of course, it seems possible to modify the algorithm in [11] for computation of the canonical form of a plane graph with keeping the linear time complexity. However, our algorithm is much simpler and, as far as we know, no simple algorithm has been known for determination of the canonical form or isomorphism of plane graphs. In addition, our algorithm constructs a tree representation of an input plane graph. By combining with grammar-based tree compression algorithm [3, 9], plane graphs having many local symmetries might be efficiently compressed.

We also apply the (modified) centroid to the maximum common connected edge subgraph (MCCES) problem for two geometric plane graphs, which seeks for a graph with the maximum number of edges that is identical to a subgraph of each input graph under isometric transformations. Note that MCCES for geometric plane graphs is practically important because various kinds of maps (e.g., roadmaps) are often represented as geometric planar graphs. Although the MCCES problem is NP-hard for considerably restricted classes of graphs [1, 2, 4, 13], it can trivially be solved in polynomial time if graphs are restricted to geometric graphs and isomorphism is restricted to those by isometric transformations. We present a linear-time algorithm for the MCCES problem for the special case in which input geometric graphs have outerplanar structures, MCCES can be obtained by deleting at most a constant number of edges from each input graph, and both the maximum degree and the maximum face degree are bounded by constants.

2 Preliminaries

Let $G(V, E)$ be an undirected graph. We assume that G is connected and its planar embedding is given. Such a graph is called a *plane graph*. We use n to denote the size of V (i.e., $n = |V|$). Since we only consider plane graphs, $|E|$ is $\Theta(n)$. Two plane graphs G_1 and G_2 are called *isomorphic* if there exists an isomorphic mapping from G_1 to G_2 such that outer faces correspond to each other and the circular ordering of edges connected to each vertex is preserved. If G_1 and G_2 are isomorphic, we write $G_1 \cong G_2$.

Let $\phi(G)$ be a function that maps a given undirected graph $G(V, E)$ to a string over an alphabet of size $O(n)$. Then, $\phi(G)$ is called a *canonical form* of $G(V, E)$ if the following conditions are satisfied:

- $|\phi(G)|$ (i.e., the length of $\phi(G)$) is $O(n)$,
- G can be reconstructed from $\phi(G)$,
- $\phi(G_1) = \phi(G_2)$ if and only if $G_1 \cong G_2$.

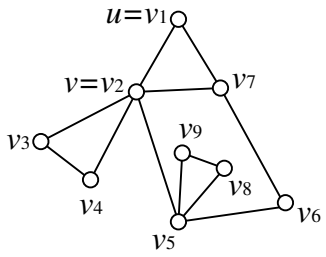

Figure 1 Example for $cano(G, u, v)$.

We assume that a plane graph is given in a form of the doubly-connected-edge-list (DCEL) [17] so that deletion of an edge can be done in a constant time and deletion of a face can be done in time proportional to the number of surrounding edges.

For a string $A = a_0a_1 \dots a_{n-1}$, $a_ia_{i+1} \dots a_{i+n-1}$ is called the *canonical form* of a circular string A if it is lexicographycally smallest among $i = 0, \cdots, n-1$, where indices are calculated modulo n [6, 12].

3 $O(n^2)$ time canonical form computation

We begin with a simple $O(n^2)$ time algorithm for computing a canonical form. It is a known fact (e.g., see [18]) but a part of this algorithm is used as a subroutine in the next section.

Let $G(V, E)$ be a plane graph. Suppose that we are given a pair of vertices (u, v) such that $\{u, v\} \in E$. We construct a string $cano(G, u, v)$ by using the following procedure.

1. Perform depth first search (DFS) starting from u with choosing (u, v) as the first edge (with the direction from u to v) and regarding it as the leftmost edge from u. In DFS, we visit edges emanating from each vertex from left to right (i.e., anti-clockwise order).
2. Rename the vertices according to the visited DFS order. Let the resulting vertices be $v_1, v_2, \dots, v_n$.
3. Construct the Euler string by concatenating edges in the visited order, where each edge is represented as (i, j) when v_j is visited just after v_i.

▶ **Example 1.** Consider a plane graph $G(V, E)$ shown in Figure 1. If DFS is started from (u, v), vertices are renamed as in Figure 1. The resulting $cano(G, u, v)$ will be

$$(1,2)(2,3)(3,4)(4,2)(2,4)(4,3)(3,2)(2,5)(5,6)(6,7)(7,1)(1,7)(7,2)(2,7)$$
$$(7,6)(6,5)(5,8)(8,9)(9,5)(5,9)(9,8)(8,5)(5,2)(2,1)$$

It is seen from Example 1 that in $cano(G, u, v)$, each edge appears exactly twice in the opposite directions, which further means that $cano(G, u, v)$ is a kind of Euler string. Clearly, $cano(G, u, v)$ can be computed in $O(n)$ time and $|cano(G, u, v)|$ is $O(n)$. Since the original plane graph is reconstructed from $cano(G, u, v)$, we can see that $cano(G_1, u_1, v_1) = cano(G_2, u_2, v_2)$ holds if and only if there exists an isomorphic mapping from G_1 to G_2 that maps u_1 and v_1 to u_2 and v_2, respectively. $cano(G, u, v)$ is called an *edge-specified canonical form*.

Let $cano_0(G)$ be the edge-specified canonical form which is lexicographycally smallest among $cano(G, u', v')$s such that $\{u', v'\} \in E$. Then, we have

▶ **Proposition 2.** *$cano_0(G)$ is a canonical form of a plane graph G and can be computed in $O(n^2)$ time.*

4 Linear time canonical form computation

As shown in Section 3, we can have a canonical form in $O(n)$ time if some unique edge is identified. However, it is quite difficult to efficiently identify the unique edge because the canonical form problem intrinsically includes the canonical form problem on circular strings. Therefore, we will reduce the canonical form problem on plane graphs into the canonical form problem on circular strings.

Our idea is to identify the (topological) *centroid* of a given plane graph G, where the centroid is either a vertex, an edge, or a face. Once the centroid is found, we can create a circular string using the method given in Section 3.

For an unordered tree T, v is called a *centroid* if the longest path from v to leaves is the shortest. It is known that there exist either one centroid, or two centroids connected by an edge. In the former case, this unique vertex is called the *centroid vertex*. In the latter case, this unique edge is called the *centroid edge*. It is well known that for a tree T, the centroid vertex/edge can be determined in linear time. We extend this concept for plane graphs.

In the following, we show how to construct a canonical form of a plane graph based on the centroid and the trees connected to the centroid that are constructed in the determinization process of the centroid. To this end, we first determine the centroid of a plane graph. It is known that the connected plane graph has the unique outer face. We consider the directed cycle consisting of the edges of the outer face, where edges are visited in the clockwise order. Let C be this directed cycle (see Figure 2 (A)). An inner face (i.e., a face that is not the outer face) of a plane graph G is called *exposed* if it includes an edge in C (with ignoring the direction). Furthermore, an inner face is called *singly exposed* if the outer edges in this face are connected in C. Similarly, an edge not belonging to an inner face is called *singly exposed* if one of its endpoints is of degree 1. A graph consisting of a vertex, an edge, or a face (including adjunct subgraphs) is not regarded as *singly exposed*. In addition, each maximally connected subgraph G_S that is surrounded by a singly exposed face with sharing only one vertex v that is an outer one is called an *adjunct subgraph* (see Figure 2 (B)). Each adjunct subgraph is ignored and removed along with its surrounding face because information on this part can be easily included in the final canonical form as follows. According to the ordering of C, we can define the parent vertex u of v. Then, the leftmost child w of v in the subgraph can be uniquely determined and thus $cano(G_S, v, w)$ can be computed (see Figure 3). We can insert this $cano(G_S, v, w)$ (delimited by a special symbol '&' not appearing in other parts) into a part the canonical form corresponding to removed outer edges.

After identification of the singly exposed faces and edges, all of them will be removed. This removal is done by deleting outer edges included in these faces and edges. However, we keep information about all these edges in order to reconstruct tree structures at the final stage. To this end, we virtually detach the last outer edge from the last endpoint in each singly exposed face, where the last means that in the order of C (see Figure 2 (B)).

We use the following procedure (denoted as **PEELING**) to identify the centroid vertex/edge/face (see Figure 4).

1. Repeat Steps 2-3 until there does not exist a singly exposed face/edge.
2. Identify all singly exposed faces and edges.
3. Delete outer edges and adjunct subgraphs in these exposed faces and edges.

The correctness of **PEELING** is guaranteed by the following two propositions.

▶ **Proposition 3.** *After each removal step, the resulting graph is connected.*

Proof. Since each face is doubly connected by its surrounding edges and only consecutive outer edges are removed from each face, we do not lose the connectedness. ◀

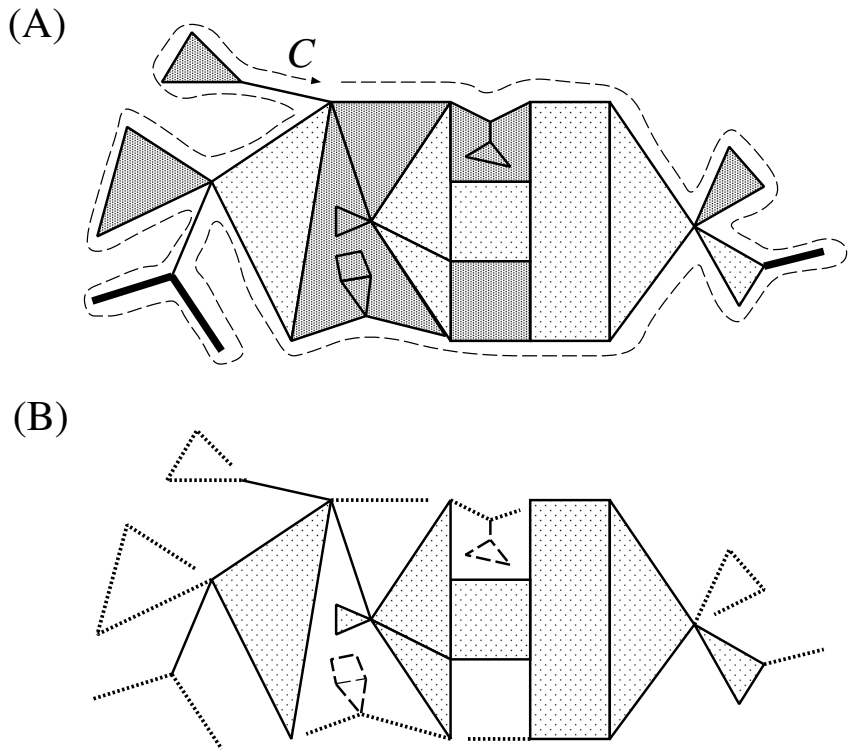

Figure 2 Example of face removal operations. (A) Directed cycle C is shown by a dashed curve. Gray regions and bold edges correspond to singly exposed faces and edges, respectively. (B) Deleted outer edges and adjunct subgraphs are shown by dotted lines and dashed lines, respectively.

▶ **Proposition 4.** *If there does not exist a singly exposed face or edge in G, G is either a vertex, an edge, or a face (possibly including adjunct subgraphs inside).*

Proof. Suppose that G is not a vertex, an edge, or a face. Then, consider the directed cycle C consisting of outer edges. Each edge in C belongs to a face or an edge (not in a face). Then, C must include at least one edge in a singly exposed face or edge. ◀

After determining the centroid of a plane graph, we construct a canonical form using the centroid as follows. Since the other cases are easier, we assume w.l.o.g. (without loss of generality) that a single face f_c (possibly including adjunct subgraphs inside) is finally left. Then, we add trees and adjunct subgraphs deleted by **PEELING** to the centroid f_c. Let G' be the resulting graph. As mentioned before, we assume w.l.o.g. that there does not exist any adjunct subgraph. Therefore, the resulting graph consists of the centroid and trees. Let $v^1, v^2, \ldots, v^d$ be the vertices of f_c arranged in the clockwise order, starting from an arbitrary one (see Figure 5). For each vertex v^i, let $v^i_1, \ldots, v^i_{d_v}$ be the neighboring vertices (not in f_c) in the clockwise order. For each subtree T^i_j rooted at (v^i, v^i_j), we construct $cano(T^i_j, v^i, v^i_j)$ and then construct the string $cano(v^i)$ by concatenating these as

$$cano(v^i) \quad = \quad \#cano(T^i_1, v^i, v^i_1)\#cano(T^i_2, v^i, v^i_2)\#\cdots\#cano(T^i_{d_v}, v^i, v^i_{d_v})\#,$$

where '#' is a special symbol not appearing in other parts.

Since trees in the canonical form may be obtained by decomposing cycles of the original plane graph, leaves may need information about from which vertices they are detached.

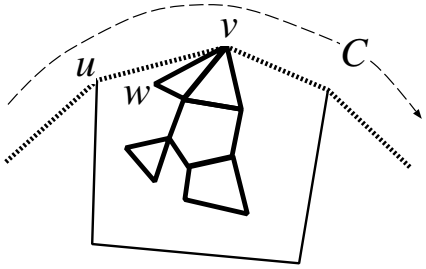

Figure 3 Example of an adjunct subgraph.

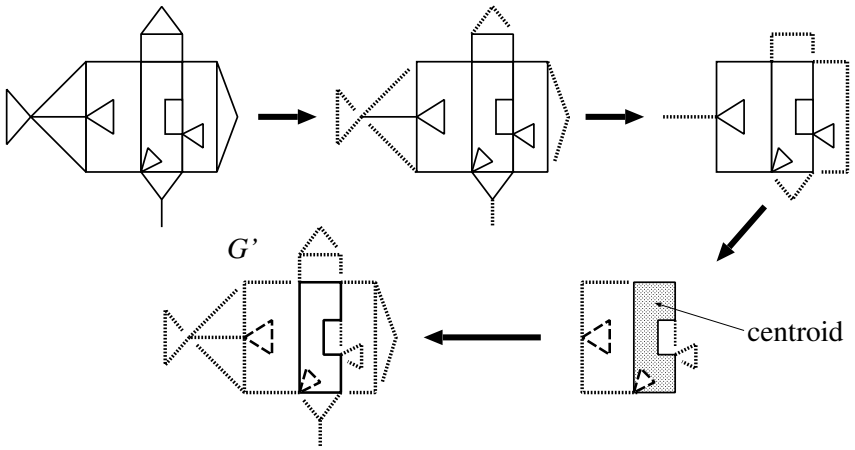

Figure 4 Illustration of the **PEELING** procedure. Deleted edges are shown by dotted lines. Adjunct subgraphs are shown by dashed lines.

Therefore, in computation of $cano(T_j^i, v, v_j^i)$, we need to add the information about other trees because the disconnected edge shares a vertex in T_j^i, another tree, or the centroid. In order to cope with this problem, we renumber T_j^is to be $T_1, \ldots, T_m$ in the clockwise order starting from an arbitrary tree (we only use the difference of the indices modulo m). We consider the following three cases (see Figure 5):

- if the disconnected endpoint v_j of an edge (v_h, v_j) in T_i is actually a vertex $v_{j'}$ in the same subtree T_i, we replace j in $cano(...)$ with (T, j'),
- else if the disconnected endpoint v_j of an edge (v_h, v_j) in T_i is actually a vertex $v_{j'}$ in $T_{i'}$, we replace j in $cano(...)$ with $(T + (i' - i), j')$, where $i' - i$ is computed modulo m,
- otherwise (i.e., v_j is actually a vertex $v^{k'}$ in the centroid), we replace j in $cano(...)$ with $(C + (k' - k))$. where v^k is the root of T_i and $k' - k$ is computed modulo d.

Then, we construct $cano(F)$ by concatenating $cano(v_i)$s by

$$cano(F) \quad = \quad cano(v_1)!cano(v_2)!\cdots!cano(v_k)!,$$

where '!' is a special symbol not appearing in other parts. Finally, we regard $cano(F)$ as a circular string and define $cano(G)$ to be the canonical form of this circular string.

▶ **Theorem 5.** *$cano(G)$ is a canonical form of a plane graph G and can be computed in $O(n)$ time.*

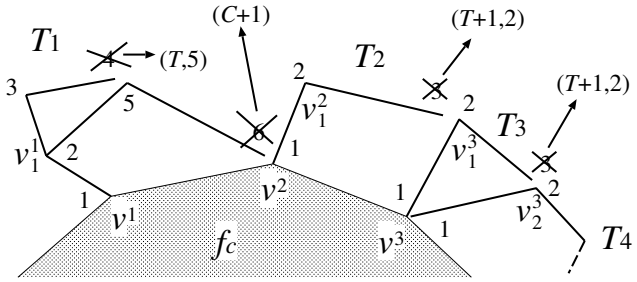

Figure 5 Illustration of replacement of labels for disconnected vertices.

Proof. The correctness follows from the following facts:

- The peeling process is invariant under isomorphic transformations, that is, the same set of edges is always deleted at each time step for isomorphic plane graphs.
- After the peeling process, only one face, edge, or vertex remains.
- $cano(v_i)$ is invariant under isomorphic transformations.

Next, we analyze the time complexity. We maintain plane graphs using DCEL data structure with adding information about exposed/not exposed. We also maintain lists of consecutively exposed edges and pointers from each list to the corresponding face and from each face to the corresponding lists. Each list/face also has a flag showing whether or not it is singly exposed one. The peeling process can be done by deleting edges in lists with singly exposed flags. Of course, all data structures must be updated, which can be done in time proportional to the number of deleted edges and the number of newly exposed edges. Since each edge is newly exposed only once and is deleted only once, the total time complexity is proportional to the number of edges (i.e., the total complexity is $O(n)$). It is straightforward to see that $cano(F)$ can be obtained in $O(n)$ time. Since the canonical form of a circular string over a general alphabet can be computed in $O(n)$ time [6, 12], the total time complexity is $O(n)$. ◀

5 Canonical form of geometric plane graphs

The algorithm in Section 4 can be modified for computing the canonical form of a given geometric plane graph so that the canonical form is invariant under isometric transformations. We say that G_1 and G_2 are isomorphic under isometric transformations if there is an isometric transformation T (i.e., combination of translation, rotation, and mirror image) such that $T(shape(G_1)) = shape(G_2)$, where $shape(G)$ denotes the set of line segments in G. We may omit $shape(...)$ and write this relation as $T(G_1) = G_2$. Since inclusion of mirror images in isometric transformation can be done by multiplying a constant factor to the time complexity, we ignore them in the following.

In order to include geometric information, it is enough to add geometric information to $cano(G, u, v)$. Suppose that (u, v) and (v, w) are consecutive edges. Let $L^2(v, v')$ denote the square of the Euclid distance between v and v', the exact value of which can be computed from the coordinates of v and v'. Then, we add the following information to (v, w).

- $L^2(u, v)$, $L^2(v, w)$, $L^2(u, w)$,
- whether w is located left or right of $\vec{uv}$.

It is straightforward to see the correctness of this modified procedure to define the canonical form. Since it does not increase the order of the size of the canonical form and the time complexity, the following holds.

▶ **Proposition 6.** *The canonical form of a geometric plane graph be computed in $O(n)$ time.*

It might be possible to use the geometric centroid (which can be easily computed), in place of the topological centroid, to compute the canonical form in linear time. However, it is unclear whether or not the circular string can be constructed easily.

6 Maximum common connected edge subgraph of geometric plane graphs

We consider the problem of finding a maximum common connected edge subgraph (MCCES) G_c of two geometric plane graphs G_1 and G_2: G has the maximum number of edges such that $G = T(G_1) \cap G_2$ for some isometric transformation T. For simplicity, we assume that G_1 and G_2 have $O(n)$ edges. We also assume Real RAM (Random Access Machine) as a computation model in which each arithmetic computation can be done in a constant time. This problem can be solved by the following procedure (**SimpleMCCES**):

1. Let G_0 be an empty graph.
2. For all directed edge pairs $(e_1, e_2) \in E(G_1) \times E(G_2)$, repeat steps 3-6.
3. Determine isometric transformation T uniquely (except mirror image) that maps e_1 to e_2.
4. Let $G \leftarrow T(G_1) \cap G_2$.
5. Let $E^c(G)$ be the set of edges in the connected component of G having the maximum number of edges.
6. If $|E^c(G)| > |E^c(G_0)|$, then let $G_0 \leftarrow G$.
7. Output G_0.

In computation of $T(G_1) \cap G_2$, edges remain only if two corresponding edges completely overlap. Furthermore, T is examined only if the lengths of e_1 and e_2 are the same. Then, the correctness of the procedure is obvious.

Next, we analyze the time complexity. Suppose that G_c has $O(k)$ edges. Step 4 can be done in $O(n)$ time by performing DFS using edges common to $T(G_1)$ and G_2. If the maximum degree is bounded by a constant, it can be done in $O(k)$ time. Since we may examine all edge pairs, Steps 3-6 are repeated $O(n^2)$ times. Therefore, the total time complexity is $O(n^3)$ in a general case and is $O(kn^2)$ if the maximum degree is bounded by a constant.

When the maximum degree is bounded by a constant, we can improve the time complexity to $O(n^2 \log n)$ (it is an improvement when $k > c \log n$ for some constant c) using a *geometric hashing* [20]. For each directed edge pair $(e_1, e_2) \in E(G_1) \times E(G_2)$ having the same length, we compute the unique isometric transformation T such that $T(e_1) = e_2$, and put this pair into the bin labeled with T. Then, we find the bin containing the maximum number of pairs connected in both G_1 and G_2, which corresponds to MCCES. Since $O(n^2)$ pairs are examined and finding the bin (where a respective edge pair is to be put in) needs $O(\log n)$ time using binary search, the total computation time to create all bins is $O(n^2 \log n)$. Since we assumed that the maximum degree is bounded by a constant, connected components in all bins can be computed in linear time of the total number of edge pairs. Therefore, the total time complexity is $O(n^2 \log n)$.

The above results are almost trivial. Here we present a faster algorithm for a special case of geometric MCCES in which graphs are outerplanar, both the maximum degree and the maximum face degree (i.e., maximum number of edges of a face) are bounded by constants, and G_c is very similar to G_1 and G_2 (precisely, G_c is obtained by deleting at most K edges from G_1 and also by deleting at most K edges from G_2). In the following, a plane graph with outerplanar graph structure is called an *outer-plane graph*. Suppose that the maximum

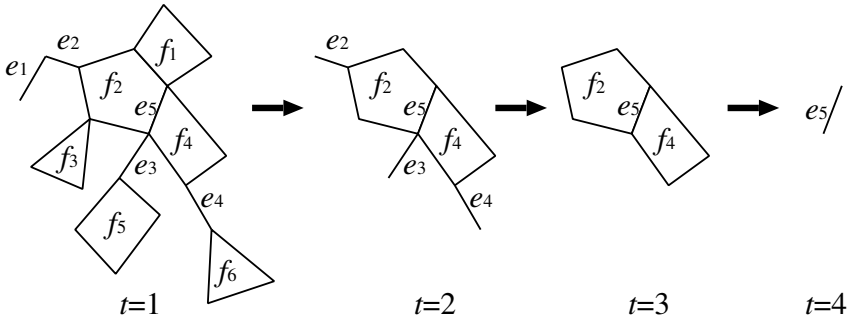

Figure 6 Determination of the centroid for an outer-plane graph. In this case, $t_G(f_1) = t_G(f_3) = t_G(f_5) = t_G(f_6) = t_G(e_1) = 1$, $t_G(e_2) = G(e_3) = t_G(e_4) = 2$, $t_G(f_2) = G(f_4) = 3$, and e_5 is the centroid.

degree and the maximum face degree are bounded by constants D_v and D_f, respectively. We will show that the position of the centroid changes for a constant amount by addition (or deletion) of an edge.

To this end, we use a simpler definition of the centroid for an outer-plane graph G, where it can also be applied to an outerplanar graph. We use the following simple procedure (see Figure 6), where the resulting face/edge/vertex is the centroid and is denoted by $c_O(G)$.

1. Repeat Step 2 until there remains only one face, edge, or vertex.
2. Identify all faces and edges each of which overlaps with other face(s)/edge(s) at one edge (including its endpoints) or one vertex.
3. Delete all faces and edges identified in Step 2.

It is straightforward to see that this procedure works in $O(n)$ time and the centroid is determined uniquely for isomorphic outer-plane graphs. For two edges e_1 and e_2, $d(e_1, e_2)$ denotes the shortest distance between endpoints of e_1 and endpoints of e_2, where the distance between vertices is defined as the length of the shortest path connecting u and v. For the centroid C in a graph G, let $e(x)$ denote the set of edges in x if x is a face or an edge, and the set of edges connecting to x otherwise (i.e., x is a vertex). In addition, let $e(x, d)$ denote the set of edges each of which has an edge in x within distance d in G. Then, we apply **SimpleMCCES** only for the edge pairs (each with two directions) between $e(c_O(G_1), d_K)$ and $e(c_O(G_2), d_K)$, where d_K is to be determined later so that at least one edge in $e(c_O(G_c))$ is included in both $e(c_O(G_1), d_K)$ and $e(c_O(G_2), d_K)$.

▶ **Lemma 7.** *Suppose that G_2 is obtained by adding an edge to G_1 with keeping outerplanarity and connectivity. Then, the minimum distance between $e(c_O(G_1))$ and $e(c_O(G_2))$ is at most ${D_f}^2/2$.*

Proof. For each face or edge (not in a face), we consider the time step (the number of repeats) at which the face/edge is deleted in the procedure determining the centroid, where the time step for the firstly deleted faces/edges is regarded to be 1. For each face or edge x in graph G, $t_G(x)$ denotes this deletion time step (see Figure 6).

We classify an addition of an edge into the following three cases (see Figure 7).

(a) One endpoint is a new vertex.
(b) An existing face is divided into two faces.
(c) A new face (not in an existing face) is created.

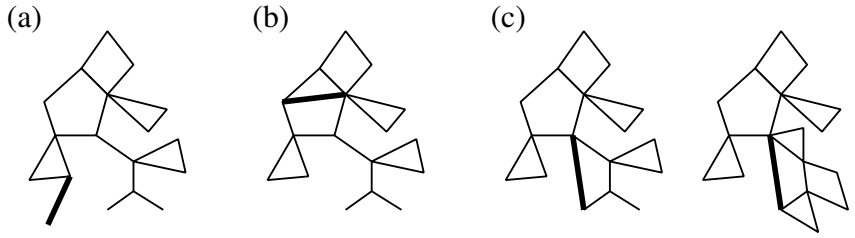

Figure 7 Classification of edge addition patterns. Added edges are shown by bold lines.

Let G_2 be the graph obtained by addition of an edge to G_1.

It is straightforward to see that the following properties hold.

- In case (a), $|t_{G_2}(x) - t_{G_1}(x)| \leq 1$ holds for each face/edge x.
- In case (b), $|t_{G_2}(x) - t_{G_1}(x)| \leq 1$ holds for each face/edge x.
- In case (c), $|t_{G_2}(x) - t_{G_1}(x)| \leq D_f$ holds for each face/edge x.

Since the centroid must have an overlap with the lastly deleted face(s)/edge(s) (i.e., face(s)/edge(s) with the maximum $t_{G_i}(x)$) and the distance between two vertices in the same face is at most $D_f/2$, the lemma holds. ◀

▶ **Theorem 8.** *Suppose that both the maximum degree and the maximum face degree of geometric outer-plane graphs G_1 and G_2 are bounded by constants D_v and D_f, respectively. Suppose also that a maximum common connected edge subgraph is obtained by deletion of at most K edges from each of G_1 and G_2. Then, a maximum common connected edge subgraph can be computed in $O(f(D_f, D_v, K)n)$ time, where $f(D_f, D_v, K) = D_f^2 \cdot D_v^{KD_f{}^2+D_f+4}$.*

Proof. From Lemma 7 and the assumption on G_c, the minimum distance between edges in $c_O(G_c)$ and $c_O(G_i)$ is at most $KD_f{}^2/2$. Then, all edges in $e(c_O(G_c))$ are included in $e(c_O(G_i), (KD_f{}^2 + D_f)/2)$ for each G_i. Therefore, by letting $d_K = (KD_f{}^2 + D_f)/2$, a maximum common connected edge subgraph can be found for two geometric graphs G_1 and G_2, Since the vertex degree is bounded by D_v, the number of edges in $e(c_O(G_i), (KD_f{}^2 + D_f)/2)$ is at most $D_f \cdot D_v^{(KD_f{}^2+D_f)/2+1}$. Since we examine all pairs in $e(c_O(G_1), (KD_f{}^2 + D_f)/2)$ and $e(c_O(G_2), (KD_f{}^2 + D_f)/2)$, the number of directed edge pairs examined is at most $2D_f^2 \cdot D_v^{KD_f{}^2+D_f+2}$. For each pair, computation of $T(G_1) \cap G_2$ can be done in $O(D_v^2 n)$ time using DFS. Therefore, the theorem holds. ◀

It is unclear whether Lemma 7 or a similar lemma holds for outer-plane graphs if the centroid $c(G_i)$ defined in Section 3 is used. However, it is easy to see that a similar lemma does not hold for plane graphs by considering a graph including large adjunct subgraphs. Therefore, defining a centroid for plane graphs so that a property similar to Lemma 7 holds is left as an open problem.

References

1 Faisal N. Abu-Khzam. Maximum common induced subgraph parameterized by vertex cover. *Inf. Process. Lett.*, 114(3):99–103, 2014. doi:10.1016/j.ipl.2013.11.007.

2 Tatsuya Akutsu. A polynomial time algorithm for finding a largest common subgraph of almost trees of bounded degree. *IEICE Transactions*, 76-A(9):1488–1493, 1993.

3 Tatsuya Akutsu. A bisection algorithm for grammar-based compression of ordered trees. *Inf. Process. Lett.*, 110(18-19):815–820, 2010. doi:10.1016/j.ipl.2010.07.004.

4 Tatsuya Akutsu and Takeyuki Tamura. On the complexity of the maximum common subgraph problem for partial k-trees of bounded degree. In Kun-Mao Chao, Tsan-sheng Hsu, and Der-Tsai Lee, editors, *Algorithms and Computation - 23rd International Symposium, ISAAC 2012, Taipei, Taiwan, December 19-21, 2012. Proceedings*, volume 7676 of *Lecture Notes in Computer Science*, pages 146–155. Springer, 2012. doi:10.1007/978-3-642-35261-4_18.

5 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

6 Kellogg S. Booth. Lexicographically least circular substrings. *Inf. Process. Lett.*, 10(4/5):240–242, 1980. doi:10.1016/0020-0190(80)90149-0.

7 Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298, 2004. doi:10.1142/S0218001404003228.

8 Andreas Fischer, Kaspar Riesen, and Horst Bunke. Improved quadratic time approximation of graph edit distance by combining hausdorff matching and greedy assignment. *Pattern Recognition Letters*, 87:55–62, 2017. doi:10.1016/j.patrec.2016.06.014.

9 Moses Ganardi, Danny Hucke, Markus Lohrey, and Eric Noeth. Tree compression using string grammars. *Algorithmica*, 80(3):885–917, 2018. doi:10.1007/s00453-017-0279-3.

10 John E. Hopcroft and Robert Endre Tarjan. A V log V algorithm for isomorphism of triconnected planar graphs. *J. Comput. Syst. Sci.*, 7(3):323–331, 1973. doi:10.1016/S0022-0000(73)80013-3.

11 John E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In Robert L. Constable, Robert W. Ritchie, Jack W. Carlyle, and Michael A. Harrison, editors, *Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1974, Seattle, Washington, USA*, pages 172–184. ACM, 1974. doi:10.1145/800119.803896.

12 Costas S. Iliopoulos and William F. Smyth. Optimal algorithms for computing the canonical form of a circular string. *Theor. Comput. Sci.*, 92(1):87–105, 1992. doi:10.1016/0304-3975(92)90137-5.

13 Nils Kriege, Florian Kurpicz, and Petra Mutzel. On maximum common subgraph problems in series-parallel graphs. *Eur. J. Comb.*, 68:79–95, 2018. doi:10.1016/j.ejc.2017.07.012.

14 Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, 25(1):42–65, 1982. doi:10.1016/0022-0000(82)90009-5.

15 Joseph Manning and Mikhail J. Atallah. Fast detection and display of symmetry in outerplanar graphs. *Discrete Applied Mathematics*, 39(1):13–35, 1992. doi:10.1016/0166-218X(92)90112-N.

16 Jiří Matoušek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992. doi:10.1016/0012-365X(92)90687-B.

17 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Texts and Monographs in Computer Science. Springer, 1985. doi:10.1007/978-1-4612-1098-6.

18 Christine Solnon, Guillaume Damiand, Colin de la Higuera, and Jean-Christophe Janodet. On the complexity of submap isomorphism and maximum common submap problems. *Pattern Recognition*, 48(2):302–316, 2015. doi:10.1016/j.patcog.2014.05.019.

19 Kokichi Sugihara. An n log n algorithm for determining the congruity of polyhedra. *J. Comput. Syst. Sci.*, 29(1):36–47, 1984. doi:10.1016/0022-0000(84)90011-4.

20 KHaim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE Comput. Sci. & Eng.*, 4(4):10–21, 1997. doi:10.1109/99.641604.