アルゴリズム 67−2
（１９９９．３．１５）

# 平面グラフの共通面埋め込み問題とその応用

陳 致中　　Xin He　　Ming-Yang Kao

東京電機大学理工学部数理学科　　SUNY at Buffalo　　Yale University

平面グラフ$G = (V, E)$と列$\mathcal{C}_1, \ldots, \mathcal{C}_q$（各$\mathcal{C}_i$は$V$の部分集合の族の列）が与えられたとき、$G$の平面への埋め込みの中から、各$\mathcal{C}_i$に対してある面$F_i$の境界が$\mathcal{C}_i$に属する各集合と空ではない共通部分を持つようなものを求めたい。この問題がＮＰ困難であることを示す。主な結果は、各$\mathcal{C}_i$に属する各集合が$G$の連結である部分グラフを導出するという制限を加えた場合、この問題を解く$O(\alpha \log \alpha)$-時間のアルゴリズムを設計できたことである（$\alpha$は入力のサイズ)。この特別な場合の問題はトポロジカル推論やＶＬＳＩ設計などへの応用がある。

# Common-face Embeddings of Planar Graphs with Applications

Zhi-Zhong Chen　　Xin He　　Ming-Yang Kao

Tokyo Denki University　　SUNY at Buffalo　　Yale University

Given a planar graph $\mathcal{G}$ and a sequence $\mathcal{C}_1, \ldots, \mathcal{C}_q$, where each $\mathcal{C}_i$ is a family of vertex subsets of $\mathcal{G}$, we wish to find a plane embedding of $\mathcal{G}$ such that for each $i \in \{1, \ldots, q\}$, there is a face $F_i$ in the embedding whose boundary intersects every set in $\mathcal{C}_i$. We show that this problem is NP-complete. Our main result is an $O(\alpha \log \alpha)$-time ($\alpha$ is the input size) algorithm for an important special case where for each input family $\mathcal{C}_i$, each set in $\mathcal{C}_i$ induces a connected subgraph of the input graph $\mathcal{G}$. This special case naturally arises from topological inference and VLSI layout problems.

## 1 Introduction

It is a fundamental problem in mathematics to embed a graph into a given space while optimizing certain objectives required by applications. A graph is called *planar* if it can be embedded on the plane so that any pair of edges can only intersect at their endpoints; a *plane* graph is a planar one together with such an embedding. A classical variant of the problem is to test whether a given graph is planar and in case it is, to find a planar embedding. This planarity problem can be solved in linear time.

In this paper, we initiate the study into the following new planarity problem. Let $\mathcal{G}$ be a planar graph. Assume that $\mathcal{G}$ is *simple*. Let $\mathcal{M}$ be a sequence $\mathcal{C}_1, \ldots, \mathcal{C}_q$, where each $\mathcal{C}_i$ is a family of vertex subsets of $\mathcal{G}$. A plane embedding $\Phi$ of $\mathcal{G}$ *satisfies* $\mathcal{C}_i$ if the boundary of some face in $\Phi$ intersects every set in $\mathcal{C}_i$. $\Phi$ *satisfies* $\mathcal{M}$ if it satisfies all $\mathcal{C}_i$. $\mathcal{G}$ *satisfies* $\mathcal{M}$ if $\mathcal{G}$ has an embedding that satisfies $\mathcal{M}$. The CFE problem is the following:

- **Input**: $\mathcal{G}$ and $\mathcal{M}$.
- **Question**: Does $\mathcal{G}$ satisfy $\mathcal{M}$?

We show that the CFE problem is NP-complete. For the special case where every vertex subset in $\mathcal{M}$ induces a connected subgraph of $\mathcal{G}$, we give an $O(\alpha \log \alpha)$-time algorithm, where $\alpha$ is the input size. The CFE problem arises naturally from topological inference [2]. For instance, a less general and efficient version of our algorithm for the special case has been employed to design fast algorithms for reconstructing maps from scrambled partial data in geometric information systems. In this application, each vertex subset in $\mathcal{M}$ describes a recognizable geographical feature, and each family in $\mathcal{M}$ is a set of features that are known to be near each other. Similarly, our algorithm for the special case can compute a constrainted layout of VLSI modules, where each vertex subset consists of the ports of a module, and each subset family specifies a set of modules that are required to be close to each other.

## 2 The main results

**Theorem 2. 1** *The CFE problem is NP-complete.*

The *size* $|\mathcal{C}_i|$ of $\mathcal{C}_i$ is the total cardinality of the sets in $\mathcal{C}_i$. The *size* $|\mathcal{M}|$ of $\mathcal{M}$ is $|\mathcal{C}_1|+\cdots+|\mathcal{C}_q|$. $|G|$ denotes the total number of vertices and edges in a graph $G$. Let $\alpha = |\mathcal{G}| + |\mathcal{M}|$. The next theorem is the main theorem of this paper.

**Theorem 2. 2** *If every vertex subset in $\mathcal{M}$ induces a connected subgraph of $\mathcal{G}$, then the CFE problem can be solved in $O(\alpha \log \alpha)$ time.*

**Proof:** We consider three special cases:
Case M1: $\mathcal{G}$ is connected.
Case M2: $\mathcal{G}$ is biconnected.
Case M3: $\mathcal{G}$ is triconnected.
Theorem 3. 8 solves Case M3. Theorem 4. 1 reduces this theorem to Case M1. §5 reduces Case M1 to M2. Theorem 6. 1 uses Theorem 3. 8 to solve Case M2. □

The remainder of the paper assumes that every vertex subset of $\mathcal{G}$ in $\mathcal{M}$ induces a connected subgraph of $\mathcal{G}$.

## 3 Case M3

This section assumes that $\mathcal{G}$ is triconnected. Then, $\mathcal{G}$ has a unique combinatorial embedding up to the choice of the exterior face. Thus, the CFE problem reduces in linear time to that of finding all the faces in the embedding whose boundaries intersect every set in some $\mathcal{C}_i$. The naive algorithm takes $\Theta(|\mathcal{G}||\mathcal{M}|)$ time. We solve the latter problem more efficiently by recursively solving the ACF problem defined below.

Throughout this section, for technical convenience, the vertices of a plane graph are indexed by distinct positive integers. The faces are indexed by positive integers or $-1$. The faces indexed by positive integers have distinct indices and are called the *positive* faces. Those indexed by $-1$ are the *negative* faces.

Let $\mathcal{H}$ be a plane graph. A *vf-set* of $\mathcal{H}$ is a set of vertices and positive faces in $\mathcal{H}$. A *vf-family* is a family of vf-sets; a *vf-sequence* is a sequence of vf-families. Let $\mathcal{N}$ be a vf-sequence $\mathcal{D}_1, \ldots, \mathcal{D}_q$ of $\mathcal{H}$. Also, let $\mathcal{D} = \{S_1, \ldots, S_d\}$ be a vf-family of $\mathcal{H}$. Let $\mathcal{F}_i$ be the set of faces in $S_i$. Let $U_i$ be the set of vertices in $S_i$.

- $|\mathcal{D}| = |S_1| + \cdots + |S_d|$; similarly, $|\mathcal{N}| = |\mathcal{D}_1| + \cdots + |\mathcal{D}_q|$.
- $\Lambda_v(\mathcal{D})$ is the set of vertices in the intersection of the vf-sets in $\mathcal{D}$.
- $\Lambda_f(\mathcal{H}, \mathcal{D})$ is the set of positive faces $F$ of $\mathcal{H}$ such that for each $S_i$, $F$ is a face in $S_i$ or its boundary intersects $S_i - \Lambda_v(\mathcal{D})$.
- $\mathrm{ACF}(\mathcal{H}, \mathcal{D}) = \Lambda_v(\mathcal{D}) \cup \Lambda_f(\mathcal{H}, \mathcal{D})$.

The ACF problem is the following:

- **Input**: $\mathcal{H}$ and $\mathcal{N}$.
- **Output**: $\mathrm{ACF}(\mathcal{H}, \mathcal{D}_1), \ldots, \mathrm{ACF}(\mathcal{H}, \mathcal{D}_q)$.

To solve the ACF problem recursively, $\mathcal{H}$ need not be simple or triconnected. Furthermore, those faces that are indexed by $-1$ are ruled out as final output during recursions. To solve the problem efficiently, each vertex in $\Lambda_v(\mathcal{D}_i)$ is meant as a succinct representation of all the faces whose boundaries contain that vertex. Similarly, the positive faces in the input $\mathcal{D}_i$ and the output are represented by their indices.

The next lemma relates the CFE problem and the ACF problem.

**Lemma 3. 1** *Let the faces of $\mathcal{G}$ be indexed by positive integers. Then, the output to the CFE problem is "yes" iff for all $C_i$,* $\mathrm{ACF}(\mathcal{G}, C_i) \neq \emptyset$.

## 3.1 A counting lemma

**Lemma 3. 2** *1. Let $v_1$ and $v_2$ be distinct vertices in $\mathcal{G}$. Let $F_1$ and $F_2$ be distinct faces in $\mathcal{G}$. Then, both $v_1$ and $v_2$ are on the boundaries of both $F_1$ and $F_2$ iff $v_1$ and $v_2$ form a boundary edge of both $F_1$ and $F_2$.*

*2. Given a set $U$ of vertices in $\mathcal{G}$, there are $O(|U|)$ faces in $\mathcal{G}$ whose boundaries each contain at least two vertices in $U$.*

*3. Given a set $\mathcal{F}$ of faces in $\mathcal{G}$, there are $O(|\mathcal{F}|)$ vertices in $\mathcal{G}$ which are each on the boundaries of at least two faces in $\mathcal{F}$.*

**Corollary 3. 3** *If $\mathcal{H}$ is simple and triconnected, then the output of the ACF problem has size $O(|\mathcal{N}|)$.*

## 3.2 A simplification technique

To solve the ACF problem efficiently, we simplify the input graph by removing unnecessary edges and vertices as follows.

For a vf-set $S$ of $\mathcal{H}$, the topological subgraph $\mathcal{H}\Diamond S$ of $\mathcal{H}$ constructed as follows is said to *simplify* $\mathcal{H}$ over $S$.

Let $U$ and $\mathcal{F}$ be the sets of vertices and positive faces in $S$, respectively. Let $\mathcal{F}_U$ be the set of the positive faces in $\mathcal{H}$ whose boundaries each contain at least two distinct vertices in $U$. Let $V'$ and $E'$ be the sets of boundary vertices and edges of the faces in $\mathcal{F} \cup \mathcal{F}_U$, respectively. Let $\mathcal{H}'$ be the plane subgraph of $\mathcal{H}$ consisting of $V' \cup U$ and $E'$.

Let $U'$ be the set of vertices which are of degree at least three in $\mathcal{H}'$; note that each vertex in $U'$ appears on the boundaries of at least two faces in $\mathcal{F} \cup \mathcal{F}_U$. A *critical* path $P$ in $\mathcal{H}'$ is a maximal path such that (1) every internal vertex of $P$ appears only once in it, and (2) no internal vertex of $P$ is in $U \cup U'$. By the choice of $U'$, every internal vertex of a critical path is of degree 2 in $\mathcal{H}'$. We use this property to further simplify $\mathcal{H}'$. Let $\mathcal{H}\Diamond S$ be the plane graph obtained from $\mathcal{H}'$ by replacing each critical path with an edge between its endpoints. This edge is embedded by the same curve in the plane as the path is. For technical consistency, if a critical path forms a cycle and its endpoint is not in $U \cup U'$, then we replace it with a self-loop for the vertex of the cycle with the smallest index.

Each vertex in $\mathcal{H}\Diamond S$ is given the same index as in $\mathcal{H}$. The closure of the interior of each face of $\mathcal{H}\Diamond S$ is the union of those of several faces or just one in $\mathcal{H}$. Let $F$ be a face in $\mathcal{H}\Diamond S$ and $F'$ be one in $\mathcal{H}$. Let $\sigma$ (resp., $\sigma'$) denote the closure of the interior of $F$ (resp., $F'$). If $\sigma = \sigma'$, then $F$ and $F'$ are regarded as the same face, and $F$ is assigned the same index in $\mathcal{H}\Diamond S$ as $F'$ is in $\mathcal{H}$. For technical conciseness, these two faces are identified with each other. If $\sigma$ is the union of the closures of the interiors of two or more faces in $\mathcal{H}$, $F$ is not the same as any face in $\mathcal{H}$ and is indexed by $-1$.

**Lemma 3. 4** *1. Given $\mathcal{H}$ and $S$, we can compute $\mathcal{H}\Diamond S$ in $O(|\mathcal{H}| + |S|)$ time.*

*2. Let $S'$ be a vf-set of $\mathcal{H}\Diamond S$. If $S' \subseteq S$, then $\mathcal{H}\Diamond S' = (\mathcal{H}\Diamond S)\Diamond S'$.*

*3. If $\mathcal{H}$ simplifies $\mathcal{G}$ over a vf-set $S^*$ with $S \subseteq S^*$, then $|\mathcal{H}\Diamond S| = O(|S|)$.*

## 3.3 Algorithms for ACF

To solve the ACF problem recursively, we use simplification to reduce the number of $\mathcal{D}_i$ and the number of sets in each $\mathcal{D}_i$.

For brevity, let $\mathcal{H}\Diamond\mathcal{D} = \mathcal{H}\Diamond(S_1 \cup \cdots \cup S_d)$; similarly, $\mathcal{H}\Diamond\mathcal{N} = \mathcal{H}\Diamond(\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_q)$. Given a vf-set $S^*$ of $\mathcal{H}$, we say $\mathcal{D} \leq S^*$ if $S_i \subseteq S^*$ for all $S_i$; we say $\mathcal{N} \leq S^*$ if $\mathcal{D}_i \leq S^*$ for all $\mathcal{D}_i$.

Lemmas 3. 5 and 3. 6 below reduce to 1 the number of $\mathcal{D}_i$ in $\mathcal{N}$ in the ACF problem.

**Lemma 3. 5** *Assume $q \geq 2$. Let $\mathcal{N}_\ell = \mathcal{D}_1, \ldots, \mathcal{D}_{\lceil q/2 \rceil}$. Let $\mathcal{N}_r = \mathcal{D}_{\lceil q/2 \rceil+1}, \ldots, \mathcal{D}_q$. Let $\mathcal{H}_\ell = \mathcal{H}\Diamond\mathcal{N}_\ell$ and $\mathcal{H}_r = \mathcal{H}\Diamond\mathcal{N}_r$.*

*1. Given $\mathcal{H}$ and $\mathcal{N}$, we can compute $\mathcal{H}_\ell$ and $\mathcal{H}_r$ in $O(|\mathcal{H}| + |\mathcal{N}|)$ total time.*

*2. For $1 \leq i \leq \lceil q/2 \rceil$, $\mathcal{H}\Diamond\mathcal{D}_i = \mathcal{H}_\ell\Diamond\mathcal{D}_i$. Similarly, for $\lceil q/2 \rceil + 1 \leq i \leq q$, $\mathcal{H}\Diamond\mathcal{D}_i = \mathcal{H}_r\Diamond\mathcal{D}_i$.*

*3. If $\mathcal{H}$ simplifies $\mathcal{G}$ over a vf-set $S^*$ with $\mathcal{N} \leq S^*$, then $|\mathcal{H}_\ell| = O(|\mathcal{D}_1| + \cdots + |\mathcal{D}_{\lceil q/2 \rceil}|)$ and $|\mathcal{H}_r| = O(|\mathcal{D}_{\lceil q/2 \rceil+1}| + \cdots + |\mathcal{D}_q|)$.*

**Lemma 3. 6** *Assume $q \geq 1$. Let $\mathcal{H}_i = \mathcal{H}\Diamond\mathcal{D}_i$.*

*1.* $\mathrm{ACF}(\mathcal{H}, \mathcal{D}_i) = \mathrm{ACF}(\mathcal{H}_i, \mathcal{D}_i)$.

*2. If $\mathcal{H}$ simplifies $\mathcal{G}$ over a vf-set $S^*$ with $\mathcal{N} \leq S^*$, then $|\mathcal{H}_i| = O(|\mathcal{D}_i|)$.*

*3. If $\mathcal{H}$ simplifies $\mathcal{G}$ over a vf-set $S^*$ with $\mathcal{N} \le S^*$, then given $\mathcal{H}$ and $\mathcal{N}$, we can compute all $\mathcal{H}_i$ in $O(|\mathcal{H}| + |\mathcal{N}| \log(q+1))$ total time.*

**Lemma 3. 7** *Let $\mathcal{D}_\ell = \{S_1, \dots, S_{\lceil d/2 \rceil}\}$ and $\mathcal{D}_r = \{S_{\lceil d/2 \rceil+1}, \dots, S_d\}$. Let $\mathcal{H}_\ell = \mathcal{H} \Diamond \mathcal{D}_\ell$ and $\mathcal{H}_r = \mathcal{H} \Diamond \mathcal{D}_r$. Let $\mathcal{D}' = \{\mathrm{ACF}(\mathcal{H}_\ell, \mathcal{D}_\ell), \mathrm{ACF}(\mathcal{H}_r, \mathcal{D}_r)\}$.*

*1. $\mathrm{ACF}(\mathcal{H}, \mathcal{D}) = \mathrm{ACF}(\mathcal{H}, \mathcal{D}')$.*

*2. If $\mathcal{H}$ simplifies $\mathcal{G}$ over a vf-set $S^*$ with $\mathcal{D} \le S^*$, then given $\mathcal{H}$ and $\mathcal{D}$, $\mathrm{ACF}(\mathcal{H}, \mathcal{D})$ can be computed in $O(|\mathcal{H}| + |\mathcal{D}| \log(d+1))$ time.*

**Theorem 3. 8** *1. Let $d$ be the maximum number of vf-sets in any $\mathcal{D}_i$ in $\mathcal{N}$. If $\mathcal{H}$ simplifies $\mathcal{G}$ over a vf-set $S^*$ with $\mathcal{N} \le S^*$, then the ACF problem can be solved in $O(|\mathcal{H}| + |\mathcal{N}| \log(d+q))$ time.*

*2. Let $d$ be the maximum number of vertex sets in any $C_i$ in $\mathcal{M}$. Case M3 of the CFE problem can be solved in $O(|\mathcal{G}| + |\mathcal{M}| \log(d+q))$ time.*

## 4 Reduction to Case M1

Let $G$ be a graph. Let $V(G)$ denote the set of vertices in $G$. A *cut* vertex of $G$ is one whose removal increases the number of connected components in $G$; a *block* is a maximal subgraph with no cut vertex. Let $\Psi(G)$ denote the forest whose vertices are the cut vertices $v$ and the blocks $B$ of $G$ and whose edges are those $\{v, B\}$ with $v \in B$. $\Psi(G)$ is a tree if $G$ is connected. A set $U$ is *$G$-local* if $U \subseteq V(G)$. A family $C$ of sets is *$G$-local* if every set in $C$ is $G$-local. For a vertex subset $W$ of $G$, let $G - W$ denote the graph obtained from $G$ by deleting the vertices in $W$.

Let $\mathcal{G}_1, \dots, \mathcal{G}_k$ be the connected components of $\mathcal{G}$. A family $\mathcal{C}_h$ in $\mathcal{M}$ is *global* if for every $i \in \{1, \dots, k\}$, $\mathcal{C}_h$ is not $\mathcal{G}_i$-local. Let $H$ be an edge-labeled graph defined as follows. The vertices of $H$ are $\mathcal{G}_1, \dots, \mathcal{G}_k$. For each global $\mathcal{C}_h$, $H$ contains a cycle $C$ possibly of length 2 where (1) the vertices of $C$ are those $\mathcal{G}_i$ such that some set in $\mathcal{C}_h$ is $\mathcal{G}_i$-local and (2) the edges of $C$ are all labeled $\mathcal{C}_h$.

Let $B_1, \dots, B_p$ be the blocks of $H$. Then, for each global $\mathcal{C}_h$, exactly one $B_j$ contains all the edges labeled $\mathcal{C}_h$. For every $B_j$, let $\mathcal{U}_j$ be the family consisting of all sets $U$ such that some edge of $B_j$ is labeled $\mathcal{C}_h$ with $U \in \mathcal{C}_h$. For each $\mathcal{G}_i$, let $\mathcal{M}_i$ be the sequence consisting of the $\mathcal{G}_i$-local families in $\mathcal{M}$ as well as the family $\mathcal{U}_{j,i} = \{U \in \mathcal{U}_j \mid U \text{ is } \mathcal{G}_i\text{-local}\}$ for each $B_j$ with $\mathcal{G}_i \in V(B_j)$.

**Theorem 4. 1** *$\mathcal{G}$ satisfies $\mathcal{M}$ iff every $\mathcal{G}_i$ satisfies $\mathcal{M}_i$. Consequently, Theorem 2. 2 holds if it holds for Case M1.*

## 5 Reducing Case M1 to M2

This section assumes that $\mathcal{G}$ is connected.

Let $w$ be a cut vertex of $\mathcal{G}$. Let $W_1, \dots, W_k$ be the vertex sets of the connected components of $\mathcal{G} - \{w\}$. Let $\mathcal{G}_i$ be the subgraph of $\mathcal{G}$ induced by $\{w\} \cup W_i$. $\mathcal{G}_1, \dots, \mathcal{G}_k$ are the *augmented components* induced by $w$. For each $\mathcal{C}_h$ in $\mathcal{M}$, let $U_{h,1}, \dots, U_{h,t_h}$ be the sets in $\mathcal{C}_h$ containing $w$; possibly $t_h = 0$. $\mathcal{C}_h$ is *$w$-global* if for each $\mathcal{G}_i$, $\mathcal{C}_h - \{U_{h,1}, \dots, U_{h,t_h}\}$ is not $\mathcal{G}_i$-local; otherwise, $\mathcal{C}_h$ is *$w$-local.*

**Lemma 5. 1** *1. Assume $\mathcal{C}_h - \{U_{h,1}, \dots, U_{h,t_h}\}$ is $\mathcal{G}_i$-local for some $\mathcal{G}_i$. Then, $\mathcal{G}$ satisfies $\mathcal{M}$ iff $\mathcal{G}$ satisfies $\mathcal{M}$ with $\mathcal{C}_h$ replaced by $(\mathcal{C}_h - \{U_{h,1}, \dots, U_{h,t_h}\}) \cup \{U_{h,1} \cap V(\mathcal{G}_i), \dots, U_{h,t_h} \cap V(\mathcal{G}_i)\}$.*

*2. Assume that $\mathcal{C}_h$ is $w$-global. Then, $\mathcal{G}$ satisfies $\mathcal{M}$ iff $\mathcal{G}$ satisfies $\mathcal{M}$ with $\mathcal{C}_h$ replaced by $\mathcal{C}_h - \{U_{h,1}, \dots, U_{h,t_h}\}$.*

By Lemma 5. 1 , we may assume that (1) each set in a $w$-global family in $\mathcal{M}$ does not contain $w$ and (2) each set in a family in $\mathcal{M}$ is $\mathcal{G}_i$-local for some $\mathcal{G}_i$. Let $H$ be an edge-labeled graph constructed as follows. The vertices of $H$ are $\mathcal{G}_1, \dots, \mathcal{G}_k$. For each $w$-global family $\mathcal{C}_h$, $H$ has a cycle $C$ possibly of length 2 where (1) the vertices of $C$ are those $\mathcal{G}_i$ such that at least one set in $\mathcal{C}_h$ is $\mathcal{G}_i$-local and (2) the edges of $C$ are all labeled $\mathcal{C}_h$.

Let $B_1, \dots, B_p$ be the blocks of $H$. Clearly, for each $w$-global family $\mathcal{C}_h \in \mathcal{M}$, exactly one block of $H$ contains all the edges labeled $\mathcal{C}_h$. For each $B_j$, let $\mathcal{U}_j$ be the family consisting of $\{w\}$ and all $U \subseteq V(\mathcal{G})$ such that some edge of $B_j$ is labeled $\mathcal{C}_h$ with $U \in \mathcal{C}_h$. For each $\mathcal{G}_i$, let $\mathcal{M}_i$ be the sequence consisting of the $\mathcal{G}_i$-local families in $\mathcal{M}$ as well as the family $\mathcal{U}_{j,i} = \{U \in \mathcal{U}_j \mid U \subseteq V(\mathcal{G}_i)\}$ for each $B_j$ with $\mathcal{G}_i \in V(B_j)$.

**Lemma 5. 2** *$\mathcal{G}$ satisfies $\mathcal{M}$ iff every $\mathcal{G}_i$ satisfies $\mathcal{M}_i$.*

By Lemma 5. 2 , Case M1 can be reduced to Case M2 in quadratic time. The inefficiency comes from the one-by-one removal of cut vertices. Using the union-find data structure and splay trees, we can remove all the cut vertices in almost linear time.

## 6 Case M2

We here assume $\mathcal{G}$ is biconnected, and prove:

**Theorem 6. 1** *Theorem 2. 2 holds for Case M2.*

## 6.1 SPQR decompositions

A *planar st-graph* $G$ is an acyclic plane digraph such that $G$ has exactly one source $s$ and exactly one sink $t$, and both vertices are on the exterior face. These two vertices are the *poles* of $G$. A *split pair* of $G$ is either a pair of adjacent vertices or a pair of vertices whose removal disconnects $G$. A *split component* of a split pair $\{u, v\}$ is either an edge $(u, v)$ or a maximal subgraph $C$ of $G$ such that $C$ is a planar $uv$-graph and $\{u, v\}$ is not a split pair of $C$. A split pair $\{u, v\}$ of $G$ is *maximal* if there is no other split pair $\{u', v'\}$ in $G$ with $\{u, v\}$ in a split component of $\{u', v'\}$.

The *decomposition tree* $T$ of $G$ is a rooted ordered tree recursively defined in four cases as follows. The nodes of $T$ are of four types $S, P, Q$, and $R$. Each node $\mu$ of $T$ has an associated planar $st$-graph ske($\mu$), called the *skeleton* of $\mu$. Also, $\mu$ is associated with an edge in the skeleton of the parent $\phi$ of $\mu$, called the *virtual edge* of $\mu$ in ske($\phi$).

*Case Q*: $G$ is a single edge from $s$ to $t$. Then, $T$ is a Q-node whose skeleton is $G$.

*Case S*: $G$ is not biconnected. Let $c_1, \dots, c_{k-1}$ with $k \geq 2$ be the cut vertices of $G$. Since $G$ is a planar $st$-graph, each $c_i$ is in exactly two blocks $G_i$ and $G_{i+1}$ with $s \in G_1$ and $t \in G_k$. Then, $T$'s root is an S-node $\mu$, and ske($\mu$) consists of the chain $e_1, \dots, e_k$ and the edge $(s, t)$, where the edge $e_i$ goes from $c_{i-1}$ to $c_i$, $c_0 = s$, and $c_k = t$.

*Case P*: $\{s, t\}$ is a split pair of $G$ with at least two split components. Then, $T$'s root is a P-node $\mu$, and ske($\mu$) consists of $k + 1$ parallel edges $e_1, \dots, e_{k+1}$ from $s$ to $t$.

*Case R*: Otherwise. Let $\{s_1, t_1\}, \dots, \{s_k, t_k\}$ with $k \geq 1$ be the maximal split pairs of $G$. Let $G_i$ be the union of the split components of $\{s_i, t_i\}$. Then, $T$'s root is an R-node $\mu$, and ske($\mu$) is the simple triconnected graph obtained from $G$ by replacing each $G_i$ with an edge $e_i$ from $s_i$ to $t_i$ and inserting the edge $(s, t)$.

In the last three cases, $\mu$ has children $\chi_1, \dots, \chi_k$ in this order, such that each $\chi_i$ is the root of the decomposition tree of $G_i$. The virtual edge of $\chi_i$ is the edge $e_i$ in ske($\mu$). $G_i$ is called the *pertinent graph* pert($\chi_i$) of $\chi_i$ as well as the *expansion graph* of $e_i$. $G$ is the pertinent graph of $T$'s root. Also, no child of an S-node is an S-node, and no child of a P-node is a P-node.

The *allocation nodes* of a vertex $v$ of $G$ are the nodes of $T$ whose skeleton contains $v$; note that $v$ has at least one allocation node.

In the above description of $T$, for each non-leaf node $\mu$, an additional edge is added into ske($\mu$) between its two poles (which does not correspond to any child of $\mu$). This additional edge has no effect on our algorithm. From now on, we ignore this edge in ske($\mu$).

For each non-S-node $\mu$ in $T$, pert($\mu$) is called a *block* of $G$ [1], which differs from that in §4 and §5. For a block $B$ = pert($\mu$), let node($B$) = $\mu$. For an ancestor $\phi$ of node($B$), the *representative* of $B$ in ske($\phi$) is the edge in ske($\phi$) whose expansion graph contains $B$.

Let $\mu$ be an R- or P-node in $T$ with children $\chi_1, \dots, \chi_b$. For each $k \in \{1, \dots, b\}$, let $e_k$ be the virtual edge of $\chi_k$ in ske($\mu$). If $\chi_k$ is an S-node, pert($\chi_k$) is a chain consisting of two or more blocks. If $\chi_k$ is an R-nod or P-node, pert($\chi_k$) is a single block. For each $k \in \{1, \dots, b\}$, we say that the blocks in pert($\chi_k$) are *on edge* $e_k$. The *minor blocks* of pert($\mu$) are the blocks on $e_1$, ..., those on $e_b$.

## 6.2 Basic ideas

A *st-orientation* of a planar graph is an orientation of its edges together with an embedding such that the resulting digraph is a planar $st$-graph.

**Fact 1 (see [1])** If an $n$-vertex planar graph has an $st$-orientation, then every embedding, where $s$ and $t$ are on the exterior face, of this graph can be obtained from this orientation through a sequence of $O(n)$ following operations:

- Select one of the two possible flips of an R-node's skeleton around its poles.
- Permute the skeletons of a P-node's children with respect to their common poles.

Let $\{s, t\}$ be an edge of $\mathcal{G}$. Since $\mathcal{G}$ is biconnected, we convert $\mathcal{G}$ to a planar $st$-graph in $O(n)$ time for technical convenience. For the remainder of §6, let $T$ be the decomposition tree of $\mathcal{G}$. Also, let $\mathcal{C}_i = \{U_{i,1}, \dots, U_{i,r_i}\}$.

Let $\mu$ be a node of $T$. $T_\mu$ denotes the subtree of $T$ rooted at $\mu$ and dep($\mu$) denotes the distance from $T$'s root to $\mu$.

- $U_{i,j}$ is *contained* in pert($\mu$) if the vertices of $U_{i,j}$ are all in pert($\mu$); $U_{i,j}$ is *strictly contained* in pert($\mu$) if in addition, no pole of pert($\mu$) is in $U_{i,j}$.
- Let done($U_{i,j}$) be the deepest node $\mu$ in $T$ such that $U_{i,j}$ is strictly contained in pert($\mu$), if such a node exists. If no such $\mu$ exists, then $U_{i,j}$ contains a pole of $\mathcal{G}$ and let done($U_{i,j}$) be $T$'s root.
- A family $\mathcal{C}_i$ *straddles* pert($\mu$) if at least one set in $\mathcal{C}_i$ is strictly contained in pert($\mu$), and at least one set in $\mathcal{C}_i$ has no vertex in pert($\mu$).
- Let done($\mathcal{C}_i$) be the deepest node $\mu$ in $T$ such that for every $U_{i,j} \in \mathcal{C}_i$, at least one vertex of $U_{i,j}$ is in pert($\mu$).

• Let $\text{sub}(\mu) = \{U_{i,j} \mid \text{done}(U_{i,j}) = \mu\}$ and $\text{fam}(\mu) = \{\mathcal{C}_i \mid \text{done}(\mathcal{C}_i) = \mu\}$.

• If $\mu$ is a P- or R-node, let $\text{xfam}(\mu) = \text{fam}(\mu) \cup (\cup_{\chi_k}\text{fam}(\chi_k))$ and $\text{xsub}(\mu) = \text{sub}(\mu) \cup (\cup_{\chi_k}\text{sub}(\chi_k))$, where $\chi_k$ ranges over all S-children of $\mu$.

In a fixed embedding of a block $B$, the poles of $B$ divide the boundary of its exterior face into two paths $\text{side}_1(B)$ and $\text{side}_2(B)$, called the two *sides* of $B$. $U_{i,j}$ is *two-sided* for $B$ if both $\text{side}_1(B)$ and $\text{side}_2(B)$ intersect $U_{i,j}$. In particular, $U_{i,j}$ is two-sided for $B$ if it contains a pole of $B$. $U_{i,j}$ is *side-1* (resp., *side-2*) for $B$ if only $\text{side}_1(B)$ (resp., $\text{side}_2(B)$) intersects $U_{i,j}$. Assume that $B$ is a minor block of $\text{pert}(\mu)$ for some $\mu$. Let $e_k$ be the representative of $B$ in $\text{ske}(\mu)$. In a fixed embedding of $\text{ske}(\mu)$, $e_k$ separates two faces $F$ and $F'$. When embedding $\text{pert}(\mu)$, we can embed $\text{side}_1(B)$ towards either $F$ or $F'$, referred to as the two *orientations* of $B$ in $\text{pert}(\mu)$.

A family $\mathcal{C}_i$ is *side-0* (resp., *side-1* or *side-2*) *exterior-forcing* for $B$ if $\text{done}(\mathcal{C}_i)$ is an ancestor of $\text{node}(B)$ in $T$ and some $U_{i,j} \in \mathcal{C}_i$ strictly contained in $B$ is two-sided (resp., side-1 or side-2) for $B$. For $p = 0$, 1, 2, define

• $\text{ext}_p(B) = \min\{\text{dep}(\text{done}(\mathcal{C}_i)) \mid \mathcal{C}_i$ is side-$p$ exterior-forcing for $B\}$, if there is side-$p$ exterior-forcing for $B$;

• $\text{ext}_p(B) = \infty$ otherwise.

Assume $\text{ext}_p(B) \neq \infty$. Let $\mu = \text{node}(B), \phi_1, \phi_2, \dots, \phi_h$ be the path in $T$ from $\mu$ to $\phi_h$, where $\text{dep}(\phi_h) = \text{ext}_p(B)$. For each $\ell \in \{1, \dots, h-1\}$, the representative of $B$ in $\text{ske}(\phi_\ell)$ must be an exterior edge in any satisfying embedding of $\text{ske}(\phi_\ell)$. In addition, if $p = 1$ or 2, $\text{side}_p(B)$ must be embedded towards the exterior face of the embedding of $\text{pert}(\phi_\ell)$.

Since $(s, t)$ is an edge of $\mathcal{G}$, the root $\rho$ of $T$ is a P-node and has a child Q-node $\phi$ representing $(s, t)$. A subtle difference between $\rho$ and each non-root node of $T$ is that the two sides of $\mathcal{G} = \text{pert}(\rho)$ is actually on the same face. To eliminate this difference, we delete $\phi$ from $T$; afterwards, if $\rho$ has only one child, we further delete $\rho$ from $T$. From here onwards, $T$ denotes this modified tree.

## 6.3 The CFE algorithm

The CFE algorithm processes $T$ from bottom up. A *ready* node $\mu$ of $T$ is either (1) a leaf node or (2) a P- or R-node such that the non-S-children of $\mu$ and the children of every S-child of $\mu$ all have been processed. The CFE algorithm processes the ready nodes of $T$ in an arbitrary order. An S-node is processed when its parent is processed. We detail how to process $\mu$ as follows.

For the case where $\mu$ is a leaf node of $T$, note that $\text{pert}(\mu)$ is a single edge of $\mathcal{G}$. Since no $U_{i,j}$ is strictly contained in $\text{pert}(\mu)$, $\text{fam}(\mu) = \text{sub}(\mu) = \emptyset$. Therefore, we simply set $\text{ext}_p(\text{pert}(\mu)) = \infty$ for $p = 0, 1, 2$,

We next consider the case where $\mu$ is a non-leaf ready node. Before $\mu$ is processed, an embedding of every minor block of $\text{pert}(\mu)$ is already fixed, except for a possible flip around its poles. Moreover, for each minor block $B$ of $\text{pert}(\mu)$ and each $p \in \{0, 1, 2\}$, $\text{ext}_p(B)$ is known. When processing $\mu$, the CFE algorithm checks whether some embedding $\Phi_\mu$ of $\text{pert}(\mu)$ satisfies the following two conditions:

• $\Phi_\mu$ satisfies every $\mathcal{C}_i$ in $\text{xfam}(\mu)$.

• For each $\mathcal{C}_i$ straddling $\text{pert}(\mu)$ and each $U_{i,j} \in \mathcal{C}_i$ strictly contained in $\text{pert}(\mu)$, at least one vertex of $U_{i,j}$ is embedded on the exterior face of $\Phi_\mu$. (*Remark.* This ensures the existence of an embedding of $\text{pert}(\text{done}(\mathcal{C}_i))$ satisfying $\mathcal{C}_i$ later.)

If no such $\Phi_\mu$ exists, then $\mathcal{G}$ cannot satisfy $\mathcal{M}$ and the CFE algorithm outputs "no" and stops. Otherwise, it finds such an $\Phi_\mu$ and fixes it except for a possible flip around its poles. It also computes $\text{ext}_p(\text{pert}(\mu))$ for $p = 0$, 1, 2.

To detail how to process $\mu$, we classify the sets $U_{i,j}$ in each $\mathcal{C}_i$ into four types and define a set $\text{img}(U_{i,j}, \mu)$ for each type as follows.

*Type 1*: $U_{i,j}$ contains at least one pole of $\text{ske}(\mu)$. Then, $\text{done}(U_{i,j})$ is an ancestor of $\mu$. Let $\text{img}(U_{i,j}, \mu) = \{v \in U_{i,j} \mid v \text{ is a vertex in } \text{ske}(\mu)\}$.

*Type 2*: $U_{i,j}$ contains at least one vertex but no pole of $\text{ske}(\mu)$. Then, $\text{done}(U_{i,j}) = \mu$. Let $\text{img}(U_{i,j}, \mu)$ as in the case of type 1.

*Type 3*: $U_{i,j}$ is strictly contained in $\text{pert}(\chi)$ for some S-node $\chi$ of $\mu$ and $U_{i,j}$ contains at least one vertex in $\text{ske}(\chi)$. Then, $\text{done}(U_{i,j}) = \chi$. Let $\text{img}(U_{i,j}, \mu) = \{\text{virtual edge of } \chi \text{ in } \text{ske}(\mu)\}$.

*Type 4*: $U_{i,j}$ is strictly contained in a minor block $B$ of $\text{pert}(\mu)$. Then, $\text{done}(U_{i,j})$ is $\text{node}(B)$ or its descendent. Let $\text{img}(U_{i,j}, \mu) = \{$ representative of $B$ in $\text{ske}(\mu)\}$.

Each element of $\text{img}(U_{i,j}, \mu)$ is called an *image* of $U_{i,j}$ in $\text{ske}(\mu)$. The remainder of §6.3 details how to process of $\mu$.

### 6.3.1 Processing an S-child

When processing $\mu$, for each S-child $\chi$ of $\mu$, we need to find an embedding of $\text{pert}(\chi)$ satisfying certain conditions. We call this process the *S-procedure* and describe it below.

Let $\chi$ be an S-child of $\mu$. Then, ske($\chi$) is a path. Let $e_1$, ..., $e_b$ be the edges in ske($\chi$). For each $k \in \{1, \ldots, b\}$, let $B_k$ be the expansion graph of $e_k$. Before the S-procedure is called on $\chi$, the following requirements are met:

• For each $k \in \{1, \ldots, b\}$, an embedding of $B_k$ has been fixed, except for a possible flip around its poles.

• For some $k \in \{1, \ldots, b\}$ and $p \in \{1,2\}$, side$_p(B_k)$ is required to face either the left or the right side of ske($\chi$).

Our only choice for embedding pert($\chi$) is to flip $B_1$, ..., $B_b$ around their poles. We need to check whether for some combination of flippings of $B_1$, ..., $B_b$, (1) the resulting embedding satisfies every $C_i \in$ fam($\chi$) and (2) the second requirement above is met.

The S-procedure consists of the following five stages:

Stage S1 constructs an auxiliary graph $D = (V_D, E_D)$ with $V_D = \{k_p \mid 1 \le k \le b,\ p = 1,2\}$ as follows. For each $C_i \in$ fam($\chi$), insert a path $P_i$ into $D$ to connect all $k_p \in V_D$ such that for some type-4 $U_{i,j} \in C_i$, (a) img$(U_{i,j}, \chi) = \{e_k\}$ and (b) $U_{i,j}$ is side-$p$ for $B_k$. To avoid confusion, we call the elements of $V_D$ *points*, and the connected components of $D$ *clusters*. Those points $k_p \in V_D$ such that side$_p(B_k)$ is required to face the left side of ske($\chi$) are called *L-points*. *R-points* are defined similarly. For each cluster $C$ of $D$, all side$_p(B_k)$ where $k_p$ ranges over all the points in $C$ must be embedded toward the same side of ske($\chi$). Also, each type-3 $U_{i,j}$ in $C_i$ contains a vertex in ske($\chi$) which is on both sides of ske($\chi$). For this reason, such sets were not considered when constructing $D$.

Stage S2 checks whether there is a cluster of $D$ containing both an $L$-point and an R-point. If such a cluster exists, then S2 outputs "no" and stops. Otherwise, each cluster $C$ consists of either $L$-points only or R-points only. In the former (resp., latter) case, we call $C$ an *L-cluster* (resp., *R-cluster*).

Stage S3 constructs another auxiliary graph $RD = (V_{RD}, E_{RD})$ from $D$ as follows. The vertices of $RD$ are the clusters of $D$. For each $k \in \{1, \ldots, b\}$, there is an edge $\{C_1, C_2\}$ in $RD$, where $C_1$ (resp., $C_2$) is the cluster of $D$ containing point $k_1$ (resp., $k_2$). $RD$ may have self-loops.

Stage S4 checks whether $RD$ is bipartite. If it is not, then S4 outputs "no" and stops. Otherwise, for each connected component $K$ of $RD$, the clusters in $K$ can be uniquely partitioned into two independent subsets $V_{K,1}$ and $V_{K,2}$ of clusters. If $V_{K,1}$ or $V_{K,2}$ contains both an $L$-cluster and an R-cluster, S4 outputs "no" and stops. Otherwise, $V_{RD}$ can be partitioned into two independent subsets $V^L_{RD}$ and $V^R_{RD}$ of clusters such that all $L$-clusters are in $V^L_{RD}$ and all R-clusters are in $V^R_{RD}$. Let $V^L_D = \{k_p \mid k_p$ is in a cluster in $V^L_{RD}\}$ and $V^R_D = \{k_p \mid k_p$ is in a cluster in $V^R_{RD}\}$.

Stage S5 embeds side$_p(B_k)$ toward the left side of ske($\chi$) for each $k_p \in V^L_D$.

### 6.3.2 $\mu$ is an R-node

In this case, ske($\mu$) is a simple triconnected graph and has a unique embedding. Let $\chi_1$, ..., $\chi_b$ be the children of $\mu$ in $T$. For each $k \in \{1, \ldots, b\}$, let $B_{k,1}$, ..., $B_{k,s_k}$ be the minor blocks of pert($\mu$) in pert($\chi_k$). When $\chi_k$ is an R- or P-node, $s_k = 1$. To process $\mu$, the CFE algorithm proceeds in five stages:

Stage R1 first computes $C'_i = \{\text{img}(U_{i,j}, \mu) \mid U_{i,j} \in C_i\}$ for every $C_i \in$ fam($\mu$). Let $\mathcal{M}'(\mu)$ be the sequence of all $C'_i$ with $C_i \in$ fam($\mu$). Then R1 calls Theorem 3. 8 (??) to solve the CFE problem on input ske($\mu$) and $\mathcal{M}'(\mu)$. If the output is "no", R1 outputs "no" and stops. Otherwise, for each $C'_i$ in $\mathcal{M}'(\mu)$, there is a face $F_i$ in the unique embedding of ske($\mu$) whose boundary intersects each img$(U_{i,j}, \mu) \in C'_i$. $F_i$ must be unique or else done($C_i$) would be a descendent of $\mu$, contradicting the fact $C_i \in$ fam($\mu$).

Stage R2 computes the minor block $B_{k,l}$ of pert($\mu$) strictly containing $U_{i,j}$ for each $C_i \in$ fam($\mu$) and each type-4 $U_{i,j} \in C_i$. If $U_{i,j}$ is two-sided for $B_{k,l}$, either side of $B_{k,l}$ may be embedded toward the face $F_i$; otherwise, for some $p \in \{1,2\}$, $U_{i,j}$ is side-$p$ for $B_{k,l}$ and it requires that side$_p(B_{k,l})$ be embedded towards $F_i$.

Stage R3 makes sure that for every $C_i$ straddling pert($\mu$) and for every $U_{i,j} \in C_i$ strictly contained in pert($\mu$), a vertex in $U_{i,j}$ is embedded on the exterior face of pert($\mu$). This is done by checking whether the following statements are all false.

• There are an exterior edge $e_k$ of ske($\mu$) and a minor block $B_{k,l}$ of pert($\mu$) on $e_k$ with $\max_{p \in \{1,2\}} \text{ext}_p(B_{k,l}) < \text{dep}(\mu)$; thus, both side$_1(B_{k,l})$ and side$_2(B_{k,l})$ must be embedded towards the exterior face of ske($\mu$).

• There are an interior edge $e_k$ of ske($\mu$) and a minor block $B_{k,l}$ of pert($\mu$) on $e_k$ with $\min_{p \in \{0,1,2\}} \text{ext}_p(B_{k,l}) < \text{dep}(\mu)$; thus, at least one of side$_1(B_{k,l})$ and side$_2(B_{k,l})$ must be embedded towards the exterior face of ske($\mu$).

• There is a $U_{i,j} \in$ sub($\mu$) with dep(done($C_i$)) $<$ dep($\mu$) and neither side of the embedding of ske($\mu$) contains an image in img$(U_{i,j}, \mu)$.

• There are an S-child $\chi_k$ of $\mu$ and a $U_{i,j} \in \mathrm{sub}(\chi_k)$ such that $\mathrm{dep}(\mathrm{done}(\mathcal{C}_i)) < \mathrm{dep}(\mu)$ and the virtual edge $e_k$ of $\chi_k$ is an interior edge in $\mathrm{ske}(\mu)$.

If any statement above holds, R3 outputs "no" and stops. Otherwise, for each minor block $B_{k,l}$ of $\mathrm{pert}(\mu)$ such that $\mathrm{ext}_p(B_{k,l}) < \mathrm{dep}(\mu)$ for some $p \in \{1,2\}$, it requires that $\mathrm{side}_p(B_{k,l})$ be embedded towards the exterior face of $\mathrm{ske}(\mu)$. Since the above 2 is false, the representative $e_k$ of $B_{k,l}$ in $\mathrm{ske}(\mu)$ must be an exterior edge of $\mathrm{ske}(\mu)$.

Stage R4 first checks whether for some minor block $B_{k,l}$ of $\mathrm{pert}(\mu)$, the orientation requirements imposed on $B_{k,l}$ in Stage R2 or R3 are in conflict. If they are, R4 outputs "no" and stops. Otherwise, for each R-child or P-child $\chi_k$ of $\mu$, the minor block $\mathrm{pert}(\chi_k)$ can be oriented according to the requirements imposed on it, or arbitrarily if no requirement was imposed on it. Afterwards, for each S-child $\chi_k$ of $\mu$, it calls the S-procedure on input $\chi_k$ together with the orientation requirements that were imposed on the minor blocks in $\mathrm{pert}(\chi_k)$ in Stage R2 or R3. If the S-procedure on any $\chi_k$ outputs "no", R4 outputs "no" and stops because $\mathrm{pert}(\chi_k)$ cannot be successfully embedded; otherwise, it has found a satisfying embedding.

Stage R5 computes $\mathrm{ext}_p(\mathrm{pert}(\mu))$ for $p = 0$, 1, 2 as follows. Let $\mathrm{xsub}'(\mu) = \{U_{i,j} \in \mathrm{xsub}(\mu) \mid \mathrm{dep}(\mathrm{done}(\mathcal{C}_i)) < \mathrm{dep}(\mu)\}$. Partition $\mathrm{xsub}'(\mu)$ into $A_0$, $A_1$, $A_2$ where $A_0$ (resp., $A_1$ or $A_2$) consists of all $U_{i,j} \in \mathrm{xsub}'(\mu)$ such that $U_{i,j}$ is two-sided (resp., side-1 or side-2) for $\mathrm{pert}(\mu)$. For $i \in \{1,2\}$, let $\beta_i = \min_{p,B_{k,l}} \mathrm{ext}_p(B_{k,l})$ where $p$ ranges over all integers in $\{0,1,2\}$ and $B_{k,l}$ ranges over all minor blocks on an edge of $\mathrm{side}_i(\mathrm{pert}(\mu))$. Then, set

$$\begin{aligned}
\mathrm{ext}_0(\mathrm{pert}(\mu)) &= \min_{U_{i,j} \in A_0} \mathrm{dep}(\mathrm{done}(\mathcal{C}_i));\\
\mathrm{ext}_1(\mathrm{pert}(\mu)) &= \min\{\beta_1, \min_{U_{i,j} \in A_1} \mathrm{dep}(\mathrm{done}(\mathcal{C}_i))\};\\
\mathrm{ext}_2(\mathrm{pert}(\mu)) &= \min\{\beta_2, \min_{U_{i,j} \in A_2} \mathrm{dep}(\mathrm{done}(\mathcal{C}_i))\}.
\end{aligned}$$

### 6.3.3 $\mu$ is a P-node

In this case, $\mathrm{ske}(\mu)$ consists of parallel edges $e_1, e_2, \ldots, e_b$ between its two poles with $b \geq 2$. Let $\chi_1$, ..., $\chi_b$ be the children of $\mu$ in $T$. For each $k \in \{1, \ldots, b\}$, let $B_{k,1}$, ..., $B_{k,s_k}$ be the minor blocks of $\mathrm{pert}(\mu)$ in $\mathrm{pert}(\chi_k)$. When embedding $\mathrm{ske}(\mu)$, edges $e_1$ through $e_b$ can be embedded in any order. The CFE algorithm first finds a proper embedding of $\mathrm{ske}(\mu)$ in three stages:

Stage P1 constructs an auxiliary graph $H = (V_H, E_H)$ with $V_H = \{e_1, \ldots, e_b\}$ by performing the following steps in turn for every $\mathcal{C}_i \in \mathrm{fam}(\mu)$:

• Compute $S_i = \cup_{U_{i,j}} \mathrm{img}(U_{i,j}, \mu)$, where $U_{i,j}$ ranges over all type-3 or type-4 sets in $\mathcal{C}_i$. Let $m_i$ be the number of edges in $S_i$. Then, $m_i \geq 2$; otherwise $\mathcal{C}_i$ would be in $\mathrm{fam}(\chi_k)$ for some $k \in \{1, \ldots, b\}$.

• If $m_i \geq 3$, then output "no" and stop since $\mathrm{pert}(\mu)$ does not satisfy $\mathcal{C}_i$. • Insert edge $\{e_k, e_{k'}\}$ to $H$, where $e_k$ and $e_{k'}$ are the two edges in $S_i$.

For each $\mathcal{C}_i \in \mathrm{fam}(\mu)$, no set in $\mathcal{C}_i$ is type 2, and each type-1 set in $\mathcal{C}_i$ contains a pole of $\mathrm{pert}(\mu)$, which is on every face of all embeddings of $\mathrm{ske}(\mu)$. For this reason, neither type-1 nor type-2 set in $\mathcal{C}_i$ is considered in the construction of $H$.

Stage P2 checks whether both statements below are false in order to ensure that for every $\mathcal{C}_i$ straddling $\mathrm{pert}(\mu)$ and every $U_{i,j} \in \mathcal{C}_i$ strictly contained in $\mathrm{pert}(\mu)$, a vertex in $U_{i,j}$ is embedded on the exterior face of $\mathrm{pert}(\mu)$.

1. There is a minor block $B_{k,l}$ of $\mathrm{pert}(\mu)$ with $\max_{p \in \{1,2\}} \mathrm{ext}_p(B_{k,l}) < \mathrm{dep}(\mu)$.

2. There are at least three edges $e_k$ in $\mathrm{ske}(\mu)$ such that (a) there is a minor block $B_{k,l}$ on $e_k$ with $\min_{p \in \{0,1,2\}} \mathrm{ext}_p(B_{k,l}) < \mathrm{dep}(\mu)$; and (b) $\chi_k$ is an S-node and there exists $U_{i,j}$ in $\mathrm{sub}(\chi_k)$ with $\mathrm{dep}(\mathrm{done}(\mathcal{C}_i)) < \mathrm{dep}(\mu)$.

If Statement 1 or 2 holds, P2 outputs "no" and stops. Otherwise, it marks each $e_k \in V_H$ for which Statement 2(a) or 2(b) holds. At most two $e_k \in V_H$ are marked, and each marked $e_k \in V_H$ must be an exterior edge in any satisfying embedding of $\mathrm{ske}(\mu)$.

Stage P3 outputs "no" and stops if an $e_k \in V_H$ has degree at least three in $H$ or a marked $e_k \in V_H$ has degree 2 in $H$. Otherwise, P3 finds and fixes an embedding of $\mathrm{ske}(\mu)$ where (1) each marked $e_k \in V_H$ is in the exterior face and (2) for every $\{e_k, e_{k'}\} \in E_H$, $e_k$ and $e_{k'}$ form the boundary of a face. For each $\mathcal{C}_i \in \mathrm{fam}(\mu)$, let $F_i$ be the face in the fixed embedding of $\mathrm{ske}(\mu)$ whose boundary is formed by the two edges in $S_i$. For each $U_{i,j} \in \mathcal{C}_i$, the boundary of $F_i$ intersects $\mathrm{img}(U_{i,j}, \mu)$.

Next, the CFE algorithm tries to embed $\mathrm{pert}(\mu)$ based on the embedding of $\mathrm{ske}(\mu)$ fixed in Stage P3 through the same stages as Stages R2 through R5 in §6.3.2 except that in the stage corresponding to R5, $A_0 = \emptyset$ and $\mathrm{ext}_0(\mathrm{pert}(\mu)) = \infty$.

## References

[1] G. Di Battista and R. Tamassia. On-Line Planarity Testing. SIAM J. Comput., 25(5):956-997, 1996.

[2] Z.-Z. Chen, M. Grigni and C.H. Papadimitriou. Planar Map Graphs. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 514-523, 1998.