Synchronized Planarity with Applications to Constrained Planarity Problems

Thomas Bläsius
Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Germany

Simon D. Fink
Faculty of Informatics and Mathematics, Universität Passau, Germany

Ignaz Rutter
Faculty of Informatics and Mathematics, Universität Passau, Germany

Abstract

We introduce the problem Synchronized Planarity. Roughly speaking, its input is a loop-free multi-graph together with synchronization constraints that, e.g., match pairs of vertices of equal degree by providing a bijection between their edges. Synchronized Planarity then asks whether the graph admits a crossing-free embedding into the plane such that the orders of edges around synchronized vertices are consistent. We show, on the one hand, that Synchronized Planarity can be solved in quadratic time, and, on the other hand, that it serves as a powerful modeling language that lets us easily formulate several constrained planarity problems as instances of Synchronized Planarity. In particular, this lets us solve Clustered Planarity in quadratic time, where the most efficient previously known algorithm has an upper bound of $O\left(n^8\right)$.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Graphs and surfaces

Keywords and phrases Planarity Testing, Constrained Planarity, Cluster Planarity, Atomic Embeddability

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.19

Related Version *Full Version*: `https://arxiv.org/abs/2007.15362`

Funding Work partially supported by DFG-grant Ru-1903/3-1.
Simon D. Fink: DFG-grant Ru-1903/3-1.
Ignaz Rutter: DFG-grant Ru-1903/3-1.

1 Introduction

A graph is *planar* if it admits an embedding into the plane that has no edge crossings. Planarity is a well-studied concept that facilitates beautiful mathematical structures [20, 9], allows for more efficient algorithms [19], and serves as a cornerstone in the context of network visualization [26]. It is not surprising that various generalizations, extensions, and constrained variants of the Planarity problem have been studied [24]. Examples are Clustered Planarity, where the embedding has to respect a laminar family of clusters [22, 8]; Constrained Planarity, where the orders of edges incident to vertices are restricted, e.g., by PQ-trees [6]; and Simultaneous Planarity, where two graphs sharing a common subgraph must be embedded such that their embeddings coincide on the shared part [5].

For planar embeddings, there is the important notion of *rotation*. The rotation of a vertex is the counter-clockwise cyclic order of incident edges around it. Many of the above planarity variants come down to the question whether there are embeddings of one or multiple graphs such that the rotations of certain vertices are in sync in a certain way. Inspired by this observation, by the Atomic Embeddability problem [15], and by the cluster decomposition tree (CD-tree) [8], we introduce a new planarity variant. *Synchronized Planarity* has a loop-free multi-graph together with two types of synchronization constraints as input. Each *Q-constraint* is given as a subset of vertices together with a fixed reference rotation for each of these vertices. The Q-constraint is satisfied if and only if either all these vertices have their reference rotation or all these vertices have the reversed reference rotation. Vertices appearing in Q-constraints are called *Q-vertices* and all remaining vertices are *P-vertices*.[1] A *P-constraint* between two P-vertices u and v defines a bijection between the edges incident to u and v. It is satisfied if and only if u and v have the opposite rotation under this bijection. We require that the P-constraints form a matching, that is, no vertex appears in more than one P-constraint. The decision problem Synchronized Planarity now asks whether the given graph can be embedded such that all Q- and all P-constraints are satisfied.

Synchronized Planarity serves as a powerful modeling language that lets us express various other planarity variants using simple linear-time reductions. Specifically, we provide such reductions for Clustered Planarity, Atomic Embeddability, Partially PQ-constrained Planarity, and Simultaneous Embedding with Fixed Edges with a connected shared graph (Connected SEFE). Our main contribution is an algorithm that solves Synchronized Planarity, and thereby all the above problems, in quadratic time.

1.1 Technical Contribution

Our result impacts different planarity variants that have been studied previously. Before discussing them individually in the context of previous publications, we point out a common difficulty that has been a major barrier for all of them, and briefly sketch how we resolve it.

Consider the following constraint on the rotation of a single vertex. Assume its incident edges are grouped and we only allow orders where no two groups alternate, that is, if e_1, e_2 are in one group and e_3, e_4 are in a different group, then the circular subsequence e_1, e_3, e_2, e_4 and its inverse are forbidden. Such restrictions have been called *partition constraints* before [8], and they naturally emerge at cut-vertices where each incident 2-connected component forms a group. A single partition constraint is not an issue by itself, but it becomes difficult to deal with in combination with further restrictions. This is why cut-vertices and disconnected clusters are a major obstacle for SEFE [5] and Clustered Planarity [8], respectively.

The same issues appear for Synchronized Planarity, when we have a cut-vertex v that is involved in P-constraints, that is, its rotation has to be synchronized with the rotation of a different vertex u. We deal with these situations as follows, depending on whether u is also a cut-vertex or not. If not, it is rather well understood which embedding choices impact the rotation of u and we can propagate this from u to v.[2] This breaks the synchronization of u and v down into the synchronization of smaller embedding choices. This is a well-known technique that has been used before [6, 17]. If u is also a cut-vertex, we are forced to actually deal with the embedding choices emerging at cut-vertices. This is done by "encapsulating" the restrictions on the rotations of u and v that are caused by the fact that they are cut-vertices. All additional restrictions coming from embedding choices in the 2-connected components are pushed away by introducing additional P-constraints. After this, the cut-vertices u and v have very simple structure, which can be resolved by essentially joining them together. This procedure is formally described in Section 3.2 and illustrated in Figures 2 and 3.

[1] The names are based on PQ-trees, where Q- and P-nodes have fixed and arbitrary rotation, respectively.
[2] We can also do this if v is not a cut-vertex.

This solution can be seen as combinatorial perspective on the recent breakthrough result by Fulek and Tóth [15], who resolved the cut-vertex issue by applying an idea coming from Carmesin's work [10]. While Carmesin works with 2-dimensional simplicial complexes, Fulek and Tóth achieve their result by transferring Carmesin's idea to the setting of topological graphs on surfaces and combining it with tools from their work on thickenability. Our work transfers the problem and its solution back to an entirely combinatorial treatment of topological graphs in the plane. This further simplification allows us to more clearly highlight the key insight that makes the algorithm tick and at the same time provides access to a wide range of algorithmic tools for speeding up the computations. Due to space constraints, proofs of statements marked with a star are only given in the full version.

1.2 Related Work

Clustered Planarity was first considered by Lengauer [22] and later rediscovered by Feng et al. [13]. In both cases, the authors give polynomial-time algorithms for the case that each cluster induces a connected graph. The complexity of the general problem that allows disconnected clusters has been open for 30 years. In that time, many special cases have been shown to be polynomial-time solvable [3, 11, 14, 16] before Fulek and Tóth [15] recently settled Clustered Planarity in P. The core ingredient for this is their $O(n^8)$ algorithm for the Atomic Embeddability problem. It has two graphs G and H as input. Roughly speaking, H describes a 3-dimensional molecule structure with atoms represented by spheres and connections (a.k.a. pipes) represented by cylinders. The other graph G comes with a map to the molecule structure that maps each vertex to an atom such that two neighboring vertices lie on the same atom or on two atoms connected by a pipe. Atomic Embeddability then asks whether G can be embedded onto the molecule structure such that no edges cross.

Atomic Embeddability has been introduced as a generalization of the Thickenability problem that appears in computational topology [1]. It can be shown that Atomic Embeddability and Thickenability are linear-time equivalent [15]. Thus, the above $O(n^8)$ algorithm for Atomic Embeddability also solves Thickenability and Synchronized Planarity. In a preprint, Carmesin [10] proves a Kuratowski-style characterization of Thickenability, which he claims yields a quadratic algorithm as a byproduct. While it is believable that the running time of his algorithm is polynomial, a detailed runtime analysis is missing. In light of this, we only compare our algorithm to the $O(n^8)$-algorithm by Fulek and Tóth. For a detailed comparison of their solution to Atomic Embeddability and our solution to Synchronized Planarity, we refer to the full version.

To finally solve Clustered Planarity, Fulek and Tóth [15] use the reduction of Cortese and Patrignani [12] to Independent Flat Clustered Planarity, which they then reduce further to Thickenability. The last reduction to Thickenability is based on a combinatorial characterization of Thickenability by Neuwirth [23], which basically states that multiple graphs have to be embedded consistently, that is, such that the rotation is synchronized between certain vertex pairs of different graphs. Via the reduction from Connected SEFE to Clustered Planarity given by Angelini and Da Lozzo [2], the above result extends to Connected SEFE, which was a major open problem in the context of simultaneous graph representations [7]. We flatten this chain of reductions by giving a simple linear reduction from each of the problems Connected SEFE, Clustered Planarity, and Atomic Embeddability to Synchronized Planarity, yielding quadratic-time algorithms for all of them. Due to space constraints, the reductions are only given in the full version. Moreover, the problem Partially PQ-constrained Planarity, for which we also give a linear reduction to Synchronized Planarity, has been solved in polynomial time before, but only for biconnected graphs [6] and in the non-partial setting where either all or none of the edges of a vertex are constrained [17].

2 Preliminaries

A *partition* of a base set X is a grouping of its elements into non-empty subsets, the *cells*, so that every element is contained in exactly one cell. We assume a set implementation allowing constant-time insertion and removal of elements, such as doubly-linked lists with pointers stored with the elements. When referring to graphs, we generally mean *loop-free multi-graphs.* A *(multi-)star* consists of a *center vertex* connected by multiple, possibly parallel, edges to its *ray vertices.* A *k-wheel* is a k-cycle, where each node is also connected to an additional central node. Furthermore, we assume a graph representation that allows efficient manipulation, such as an adjacency list with doubly-linked lists.

Drawings, Embeddings and Cyclic Orders. A *(topological) drawing* Γ of a graph is a mapping of every vertex v to a point $p_v \in \mathbb{R}^2$ in the plane and a mapping of every edge $\{u, v\}$ to a Jordan arc having p_u and p_v as endpoints. A drawing uniquely defines cyclic orders of edges incident to the same vertex. Drawings with the same cyclic orders are considered equivalent, their equivalence class is called *(combinatorial) embedding.* For an embedding $\mathcal{E}$, we use $\mathcal{E}(u)$ to denote the cyclic order of the edges incident to u as given by $\mathcal{E}$, which is also called the *rotation* of u. For a (cyclic) order $\sigma = \langle x_1, \ldots, x_k \rangle$ of k elements, we use $\overline{\sigma} = \langle x_k, \ldots, x_1 \rangle$ to denote its reversal.

The Synchronized Planarity Problem. An instance is a tuple $I = (G, \mathcal{P}, \mathcal{Q}, \psi)$, where

1. $G = (P \cup Q, E)$ is a (loop-free) multi-graph with P-vertices P and Q-vertices Q,
2. $\mathcal{Q}$ is a partition of Q,
3. ψ is a mapping that assigns a rotation to each Q-vertex, and
4. $\mathcal{P}$ is a set of triples (u, v, φ_{uv}), where u and v are P-vertices of the same degree, φ_{uv} is a bijection between their incident edges, and each P-vertex occurs at most once in $\mathcal{P}$.

We call the triples $\rho = (u, v, \varphi_{uv})$ in $\mathcal{P}$ *pipes.* Pipes are not directed and we identify (u, v, φ_{uv}) and (v, u, φ_{vu}) with $\varphi_{vu} = \varphi_{uv}^{-1}$. We also define $\deg(\rho) = \deg(u) = \deg(v)$. If two P-vertices are connected by a pipe, we call them *matched*; all other P- and Q-vertices are *unmatched.*

The planar embedding $\mathcal{E}$ of G *satisfies the cell* $X \in \mathcal{Q}$ if it is either $\mathcal{E}(v) = \psi(v)$ for all $v \in X$ or $\mathcal{E}(v) = \overline{\psi(v)}$ for all $v \in X$. We say that the embedding satisfies the *Q-constraints* if it satisfies all cells, that is, vertices in the same cell of the partition $\mathcal{Q}$ are consistently oriented. The embedding $\mathcal{E}$ *satisfies the pipe* $\rho = (u, v, \varphi_{uv})$ if $\varphi_{uv}(\mathcal{E}(u)) = \overline{\mathcal{E}(v)}$, that is, they have opposite rotations under the bijection φ_{uv}. We say that the embedding satisfies the *P-constraints* if it satisfies all pipes. The embedding $\mathcal{E}$ is called *valid* if it satisfies the P-constraints and the Q-constraints. The problem SYNCHRONIZED PLANARITY asks whether a given instance $I = (G, \mathcal{P}, \mathcal{Q}, \psi)$ admits a valid embedding.

PQ-Trees and Embedding Trees. A *PQ-tree* represents a set of circular orders of its leaves by partitioning its inner nodes into two classes: For *Q-nodes* the rotation of incident edges is fixed up to reversal, for *P-nodes*, this order can be chosen arbitrarily. Rooted PQ-trees have initially been studied by Booth and Lueker [9]. There is an equivalence between rooted and unrooted PQ-trees [21], where the latter are also called PC-trees [25]. We thus do not distinguish them and simply use the term PQ-trees. Note that a P-node with three or less neighbors allows the same permutations as a Q-node of the same degree. We thus assume P-nodes to have degree at least 4. We consider a PQ-tree *trivial* if it consists of a single inner P-node (with at least four leaves). Otherwise, it consists of a single Q-node with at least two leaves, or it contains at least two inner nodes, all of which have degree at least 3.

For a vertex of a planar biconnected graph, all rotations induced by planar embeddings can efficiently be represented by a PQ-tree [9]. This PQ-tree is also called the *embedding tree* of the respective node. In the context of Synchronized Planarity, we assume that the embedding tree of a vertex does not allow rotations that would result in a Q-vertex v having any other rotation than its default ordering $\psi(v)$ or its reverse $\overline{\psi(v)}$. To ensure this, we can subdivide each edge incident to v and connect each pair of two of the new nodes if the edges they subdivide are consecutive in the cyclic order $\psi(v)$ [17]. Note that this generates a k-wheel with center v and that there are exactly two planar rotations of the center of a wheel, which are the reverse of each other. We always generate the embedding trees based on the graph where each Q-vertex in G is temporarily replaced with its respective wheel.

Connected Components. A separating k-set is a set of k vertices whose removal increases the number of connected components. Separating 1-sets are called *cut-vertices*, while separating 2-sets are called *separation pairs*. A connected graph is *biconnected* if it does not have a cut-vertex. A biconnected graph is *triconnected* if it does not have a separation pair. Maximal biconnected subgraphs are called *blocks*. A vertex that is not a cut-vertex and thus resides within an unique block is called *block-vertex*.

Hopcroft and Tarjan [20] define a graph decomposition into triconnected components, also called *SPQR-tree* [4], where the components come in three shapes: *bonds* consist of two *pole* vertices connected by multiple parallel edges, *polygons* consist of a simple cycle, and *rigids*, whose embeddings are unique up to reflection. Each edge of these components is either *real*, representing a single edge of the original graph, or *virtual*, representing a subgraph.

Every planar embedding of a biconnected planar graph can be obtained from an arbitrary planar embedding by flipping its rigids and reordering the parallel edges in its bonds [20]. The decomposition can be computed in linear time [18] and can be used to compute the embedding trees in linear time [6, Section 2.5].

Splits and Joins of Graphs and Embeddings. Let $G = (V, E)$ be a graph. We call a partition $C = (X, Y)$ of V into two disjoint cells a *cut* of G. The edges $E(C)$ that have their endpoints in different cells are called *cut edges*. The *split* of G at $C = (X, Y)$ is the disjoint union of the two graphs obtained by contracting X and Y to a single vertex x and y, respectively (keeping possible multi-edges); see Figure 1. Note that the edges incident to x and y are exactly the cut edges, yielding a natural bijection φ_{xy} between them. Conversely, given two graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ and vertices $x \in V_1$, $y \in V_2$ together with a bijection φ_{xy} between their incident edges, their *join* along φ_{xy} is the graph $G = (V, E)$, where $V = V_1 \cup V_2 \setminus \{x, y\}$ and E contains all edges of $E_1 \cup E_2$ that are not incident to x or y, and for each edge $e = ux$ incident to x, E contains an edge uv, where v is the endpoint of $\varphi_{xy}(e)$ distinct from y; see Figure 1. Observe that split and join are inverse operations.

We say that a planar embedding $\mathcal{E}$ of a graph G *respects* a cut $C = (X, Y)$ if and only if for a topological planar drawing Γ of G with embedding $\mathcal{E}$ there exists a closed curve γ such that (i) γ separates X from Y, (ii) γ crosses each edge in $E(C)$ in exactly one point, and (iii) γ does not cross any edge in $E \setminus E(C)$; see Figure 1. We say that γ *represents* C *in* Γ.

If $\mathcal{E}$ respects C, a split at C preserves $\mathcal{E}$ as follows. Let G_1 and G_2 be the graphs resulting from splitting G at C and let $x \in V_1$ and $y \in V_2$ such that φ_{xy} identifies their incident edges. Let Γ be a topological planar drawing with embedding $\mathcal{E}$ and let γ be a curve in Γ that represents C in Γ. We obtain planar drawings Γ_1 and Γ_2 of G_1 and G_2 by contracting to a single point the side of γ that contains V_2 and V_1, respectively. We denote by $\mathcal{E}_1$ and $\mathcal{E}_2$ the corresponding combinatorial embeddings of G_1 and G_2. Note that by construction for

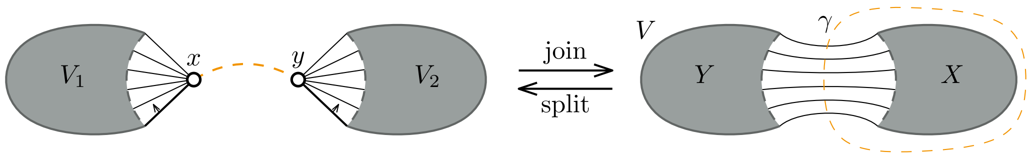


Figure 1 Joining and splitting two graphs at $x \in V_1$ and $y \in V_2$. The bijection φ_{xy} between their incident edges is shown as follows: the two bold edges at the bottom are mapped to each other. The other edges are mapped according to their order following the arrow upwards (i.e. clockwise for x and counter-clockwise for y).

each vertex of $V_1 \setminus \{x\}$ the rotations in $\mathcal{E}$ and $\mathcal{E}_1$ coincide, and the same holds for vertices of $V_2 \setminus \{y\}$ in $\mathcal{E}$ and $\mathcal{E}_2$. Moreover, the rotations $\mathcal{E}_1(x)$ and $\mathcal{E}_2(y)$ are determined by the order in which the edges of $E(C)$ cross γ, and therefore they are oppositely oriented, that is, $\varphi_{xy}(\mathcal{E}_1(x)) = \overline{\mathcal{E}_2(y)}$. We call embeddings $\mathcal{E}_1$ and $\mathcal{E}_2$ with this property *compatible with* φ_{xy}.

Conversely, we can join arbitrary embeddings $\mathcal{E}_1$ of G_1 and $\mathcal{E}_2$ of G_2 that are compatible with φ_{xy} by assuming that x and y lie on the outer face, removing x and y from the embeddings, and connecting the resulting half-edges according to φ_{xy}. The result is a planar embedding $\mathcal{E}$ where for each vertex $v \in V_i \setminus \{x, y\}$ it is $\mathcal{E}(v) = \mathcal{E}_i(v)$ for $i = 1, 2$.

▶ **Lemma 1** (∗). *Let $G = (V, E)$ be a planar graph and let (X, Y) be a cut of G such that X and Y induce connected subgraphs of G. Then every planar embedding of G respects (X, Y).*

▶ **Lemma 2** (∗). *Every planar embedding of a bipartite graph $G = (A \cup B, E)$ respects (A, B).*

3 The Synchronized Planarity Problem

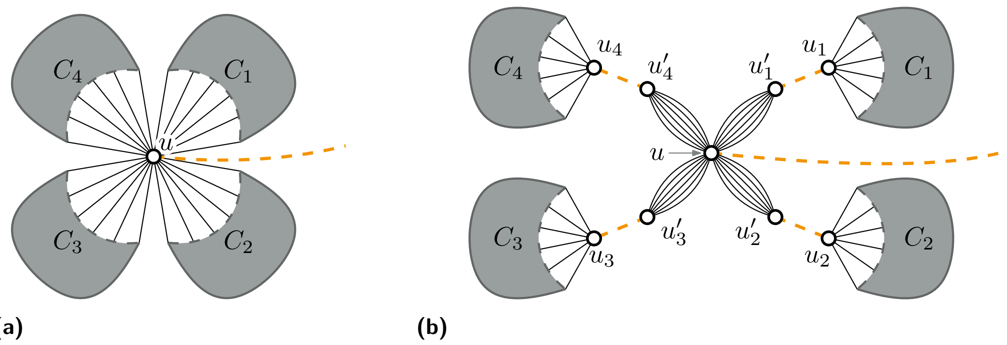
We give an algorithm for solving Synchronized Planarity for graphs with n vertices and m edges in $O(m^2)$ time. Without loss of generality, we assume that G has no isolated vertices and thus $m \in \Omega(n)$. Furthermore, we assume the input graph G to be planar.

3.1 High-Level Algorithm

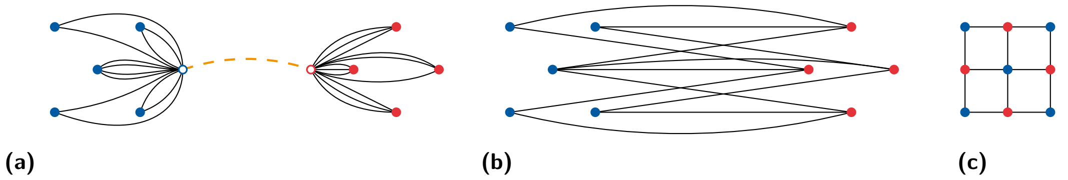
Our approach hinges on three main ingredients. The first are the three operations `EncapsulateAndJoin`, `PropagatePQ`, and `SimplifyMatching`, each of which can be applied to pipes that satisfy certain conditions. If an operation is applicable, it produces an equivalent instance I' of Synchronized Planarity in linear time. Secondly we show that if none of the operations is applicable, then I has no pipes, and we give a simple linear-time algorithm for computing a valid embedding in this case. The third ingredient is a non-negative potential function ϕ for instances of Synchronized Planarity. We show that it is upper-bounded by $2m$, and that each of the three operations decreases it by at least 1.

Our algorithm is therefore extremely simple; namely, while the instance still has a pipe, apply one of the operations to decrease the potential. Since the potential function is initially bounded by $2m$, at most $2m$ operations are applied, each taking $O(m)$ time. We will show that the resulting instance without pipes has size $O(m^2)$ and can be solved in linear time, thus the total running time is $O(m^2)$.

Conversion of small-degree P-vertices. The main difficulty in Synchronized Planarity stems from matched P-vertices. However, P-vertices of degree up to 3 behave like Q-vertices in the sense that their rotations are unique up to reversal. Throughout this paper, we



■ **Figure 2** A matched cut-vertex (a) and the result of encapsulating it (b).



■ **Figure 3** Two (encapsulated) matched cut-vertices (a). Depending on the mapping φ, any bipartite graph can result from joining them. For example, the graph (b) can result, which is isomorphic to the square grid graph shown in (c).

implicitly assume that P-vertices of degree less than 4 are converted into Q-vertices, also converting a pipe of degree less than 4 into a Q-constraint; see the full version for additional details. We therefore assume without loss of generality that P-vertices, and in particular pipes, have degree at least 4.

3.2 The EncapsulateAndJoin Operation

The purpose of the `EncapsulateAndJoin` operation is to communicate embedding restrictions between two matched cut-vertices in two steps: First we encapsulate the cut-vertices into their own independent star components, also disconnecting their incident blocks from each other. In the second step, we join the stars. Figures 2 and 3 show an example.

For an instance $I = (G, \mathcal{P}, \mathcal{Q}, \psi)$ of Synchronized Planarity, let $\rho = (u, v, \varphi_{uv})$ be a pipe matching two cut-vertices u, v of two (not necessarily distinct) connected components C_u, C_v of G. Operation `EncapsulateAndJoin` (ρ, I) can be applied resulting in an instance $I' = (G', \mathcal{P}', \mathcal{Q}', \psi')$ using the following two steps. We first preprocess both cut-vertices to *encapsulate* them into their own separate star components. Let $C_1, \ldots, C_k$ be the connected components of $C_u - u$. We split C_u along the cuts $(V(C_i), V \setminus V(C_i))$ for $i = 1, \ldots, k$. We denote the vertices resulting from the split along $(V(C_i), V \setminus V(C_i))$ as u_i and u'_i, where u_i results from contracting $V \setminus V(C_i)$ and u'_i results from contracting $V(C_i)$. Note that, after all splits, u is the center of a star C'_u whose ray vertices are the u'_i. We add the pipes $(u_i, u'_i, \varphi_{u_i u'_i})$ for $i = 1, \ldots, k$; see Figure 2. The same procedure is also applied to v, resulting in an intermediate instance I^*. In the second step, we *join* the connected components C'_u and C'_v at u and v along the mapping φ_{uv} of ρ into a component C_{uv}. We also remove the pipe ρ from I^*; all other parts of the instance remain unchanged. Figure 3 shows a possible result of joining two stars.

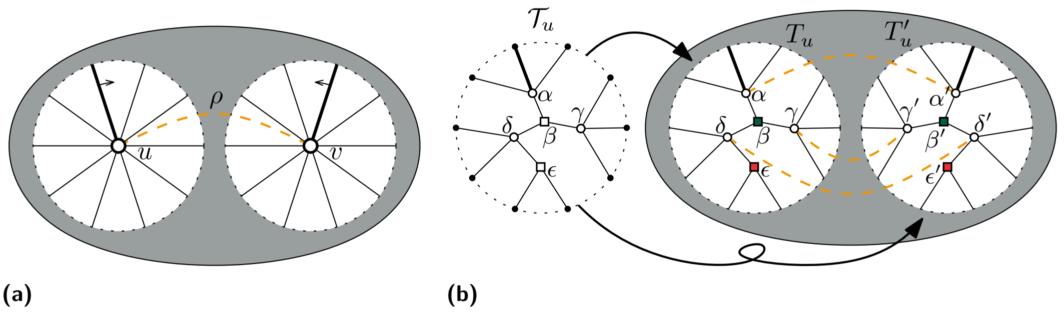


Figure 4 A block-vertex u matched with vertex v (a); the bijection φ_{uv} maps the bold edge of u to the bold edge of v, the remaining edges are mapped according to their order, clockwise around u and counter-clockwise around v. The result of applying `PropagatePQ` (u, I) (b). Note that the second inserted tree T'_u is mirrored with respect to T_u. Q-vertices and -nodes are drawn as squares while P-vertices and -nodes are drawn as disks.

▶ **Lemma 3.** *Applying* `EncapsulateAndJoin` *to a pipe ρ yields an equivalent instance in $O(\deg(\rho))$ time.*

Proof. By Lemma 1, a valid embedding $\mathcal{E}$ of an instance I respects each of the cuts $(V(C_i), V \setminus V(C_i))$ for $i = 1, \ldots, k$, yielding a planar embedding $\mathcal{E}^*$ of I^*. By construction, it is $\mathcal{E}^*(u_i) = \overline{\mathcal{E}^*(u'_i)}$ for $i = 1, \ldots, k$, that is, each new pipe $(u_i, u'_i, \varphi_{u_i u'_i})$ is satisfied and $\mathcal{E}^*$ is a valid embedding of I^*. Conversely, if $\mathcal{E}^*$ is a valid embedding of I^*, we can join u_i with u'_i for $i = 1, \ldots, k$ to obtain a valid planar embedding $\mathcal{E}$ of I, as the pipe $(u_i, u'_i, \varphi_{u_i u'_i})$ ensures that $\mathcal{E}^*$ is compatible with $\varphi_{u_i u'_i}$. The same applies to C_v.

If $\mathcal{E}^*$ is a valid embedding for I^*, it satisfies the pipe (u, v, φ_{uv}) and we can join the embedding at u and v via φ_{uv} to obtain a planar embedding $\mathcal{E}'$ of G'. Since the rotations of vertices different from u, v are unaffected, $\mathcal{E}'$ is valid for I'. Conversely, assume that $\mathcal{E}'$ is a valid embedding for I'. Note that joining two stars at their centers yields a bipartite graph consisting of the rays of the former stars. Thus C_{uv} is bipartite, and by Lemma 2 every embedding respects the cut of the bipartition. Thus, we can split $\mathcal{E}'$ and obtain a valid embedding of I^*.

As the operation affects exactly the edges incident to u and v and potentially creates a new structure with size proportional to their number, its running time is linear in the degree of the affected pipe. ◀

Observe that this operation replaces a pipe and two cut-vertices by smaller pipes and smaller cut-vertices, respectively. Through multiple applications of `EncapsulateAndJoin` we can thus step by step decrease the degree of cut-vertex-to-cut-vertex pipes, until there are none left in the instance. Note that `EncapsulateAndJoin` can yield an arbitrary bipartite component. If the component is non-planar, we abort and report a no-instance.

3.3 The PropagatePQ Operation

The operation `PropagatePQ` communicates embedding restrictions of a biconnected component across a pipe. These restrictions are represented by the embedding tree of the matched P-vertex of interest. Both endpoints of the pipe are replaced by copies of this tree. To ensure that both copies are embedded in a compatible way, we synchronize their inner nodes using pipes and Q-constraints; see Figure 4.

For an instance $I = (G, \mathcal{P}, \mathcal{Q}, \psi)$ of Synchronized Planarity, let u be a block-vertex matched by a pipe $\rho = (u, v, \varphi_{uv})$. If the embedding tree $\mathcal{T}_u$ of u is non-trivial (i.e., it not only consists of a single P-node), then the operation `PropagatePQ` (u, I) can be applied, resulting in an instance $I' = (G', \mathcal{P}', \mathcal{Q}', \psi')$ as follows. We turn the PQ-tree $\mathcal{T}_u$ into a tree T_u by interpreting Q-nodes as Q-vertices and P-nodes as P-vertices. To construct G' from G, we replace u with T_u by reconnecting the incident edges of u to the respective leaves of T_u. We also replace v by a second copy T'_u of T_u by reconnecting an edge e incident to v to the leaf of T'_u that corresponds to $\varphi_{vu}(e)$. For a vertex α of T_u we denote the corresponding vertex of T'_u by α'. For an edge $\alpha\beta$ of T_u we define $\varphi_{T_u T'_u}(\alpha\beta) = \alpha'\beta'$. For each Q-vertex α of T_u, we define $\psi'(\alpha)$ according to the rotation of the corresponding Q-node in $\mathcal{T}_u$. For the Q-vertex α' of T'_u, we define $\psi'(\alpha') = \overline{\varphi_{T_u T'_u}(\psi'(\alpha))}$. For all other Q-vertices of I, ψ' coincides with ψ. We define the partition $\mathcal{Q}' = \mathcal{Q} \cup \{\{\alpha, \alpha'\} \mid \alpha \text{ is a Q-vertex of } T_u\}$. For each P-vertex α of T_u, we define a pipe $\rho_\alpha = (\alpha, \alpha', \varphi_{\alpha\alpha'})$ with $\varphi_{\alpha\alpha'}(e) = \varphi_{T_u T'_u}(e)$ for each edge e incident to α. Finally, we define the matching $\mathcal{P}' = (\mathcal{P} \setminus \{\rho\}) \cup \{\rho_\alpha \mid \alpha \text{ is a P-vertex of } T_u\}$.

▶ **Lemma 4** (∗). *Applying* `PropagatePQ` *to a block-vertex u with a non-trivial embedding tree yields an equivalent instance. If the embedding tree $\mathcal{T}_u$ is known, operation* `PropagatePQ` *runs in $O(\deg(u))$ time.*

Note that the tree T'_u inserted instead of v may not be compatible with the rotations of v. In this case, the component becomes non-planar, potentially causing the later generation of an embedding tree to fail. We can then immediately report a no-instance.

Observe that since we assume $\mathcal{T}_u$ to be non-trivial, the degrees of all P-vertices in T_u and T'_u are strictly smaller than the degree of u. Thus, by repeatedly applying `PropagatePQ`, we eventually arrive at an equivalent instance where all matched block-vertices have a trivial embedding tree. Also note that if $\mathcal{T}_u$ consists of a single Q-node, `PropagatePQ` effectively replaces the affected pipe by two Q-vertices in the same partition. The case where $\mathcal{T}_u$ is trivial and thus consists of a single P-node is handled by the next operation.

3.4 The SimplifyMatching Operation

The remaining operation is `SimplifyMatching`, which is used to resolve pipes where one side has no restrictions to be communicated to the other side. This is the case when one of the two matched vertices is a pole of a bond that allows arbitrary rotation. We distinguish three cases: i) bonds where one pole can always mimic the rotation of the other, ii) bonds where the pipe synchronizes one pole with the other (similar to the toroidal instances of Fulek and Tóth [15]), and iii) bonds that link two distinct pipes.

For an instance $I = (G, \mathcal{P}, \mathcal{Q}, \psi)$ of Synchronized Planarity, let u be a block-vertex of G whose embedding tree is trivial and that is matched by a pipe ρ. Then, its embedding is determined by exactly one triconnected component μ, which is a bond.[3] Thus u is the pole of bond μ, and we call the vertex v that is the other pole of μ the *partner* of u. If v is either unmatched or a block-vertex with a trivial embedding tree, the operation `SimplifyMatching` (u, I) can be applied, resulting in an instance $I' = (G', \mathcal{P}', \mathcal{Q}', \psi')$ as follows. Note that, due to the temporary replacement of Q-vertices by wheels when computing the embedding trees, v cannot be a Q-vertex, as that would make the PQ-tree of u contain a Q-node.

(i) If v is an unmatched P-vertex (Figure 5a), I' is obtained from I by removing ρ.

[3] as a second bond would cause another P-node in the embedding tree, a rigid would cause a Q-node and polygons do not affect the embedding trees [6, Section 2.5]

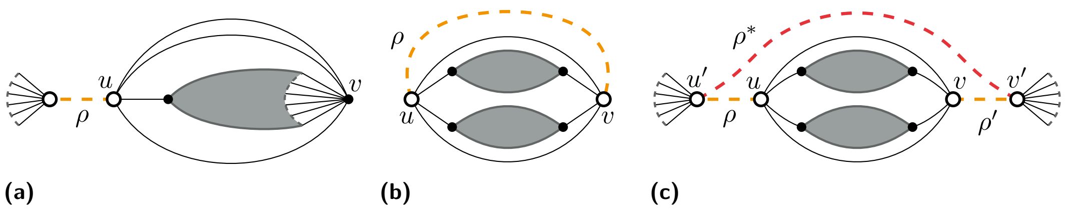


Figure 5 The three cases of the SimplifyMatching operation. In Case i (a) and Case ii (b), the pipe ρ is removed. In Case iii (c) the pipes ρ, ρ' are replaced by pipe $\rho^\star$.

(ii) If ρ matches u with v, it connects the two poles of the bond μ (Figure 5b). Note that the embedding trees of u and v both contain a P-node of the same degree representing μ and the pipe now requires both u and v to have the same degree. Thus, as u has a trivial embedding tree, v also has a trivial embedding tree. The rotation of the vertices is thus exclusively determined by the embedding of the bond and there are bijections δ_u and δ_v between the edges incident to u and v, respectively, and the virtual edges within the bond. We now check that these bijections are compatible with the bijection φ_{uv} given by the pipe. Let $\delta_{vu} = \delta_u^{-1} \circ \delta_v$ be a bijection between the edges incident to v and the edges incident to u, and let $\pi = \varphi_{uv} \circ \delta_{vu}$ be a permutation of the edges incident to v. If all cycles of π have the same length, I' is obtained from I by removing ρ.[4] Otherwise, I is an invalid instance and we set I' to a trivial no-instance.

(iii) If v is matched with a P-vertex $v' \neq u$ via pipe $\rho' = (v, v', \varphi_{vv'})$, let u' be the other endpoint of $\rho = (u, u', \varphi_{uu'})$. We remove ρ and ρ' and add the new pipe $\rho^* = (u', v', \varphi_{u'v'})$ with $\varphi_{u'v'} = \varphi_{vv'} \circ \delta_{uv} \circ \varphi_{u'u}$; see Figure 5c.

▶ **Lemma 5** ($*$). *Applying* `SimplifyMatching` *to a block-vertex u with a trivial embedding tree yields an equivalent instance in $O(\deg(u))$ time.*

3.5 Reduced and Pipe-Free Instances

With our exposition of the fundamental operations complete, we now study how to solve instances where none of those operations can be applied. We call such instances *reduced*.

▶ **Lemma 6.** *An instance is reduced if and only if it contains no pipes.*

Proof. Obviously, a pipe-free instance is reduced. Conversely, consider a reduced instance I. Assume, for the sake of contradiction, that I contains a pipe. We now show that this implies that one of the operations is applicable, that is, I is not reduced.

Assume that I contains no matched cut-vertices and thus all matched vertices are block-vertices. If there is a matched P-vertex with a non-trivial embedding tree, `PropagatePQ` can be applied. Otherwise, all matched P-vertices are block-vertices with trivial embedding trees and `SimplifyMatching` can be applied.

Now let u be a matched cut-vertex of maximum degree that is matched to a vertex v by a pipe ρ. If v is also a cut-vertex, we can apply `EncapsulateAndJoin`. If v is a block-vertex with a non-trivial embedding tree, we can apply `PropagatePQ`. Therefore, v must be a block-vertex with a trivial embedding tree. Now we can apply `SimplifyMatching`, unless the partner pole v' of v is a matched cut-vertex. This is however excluded,

[4] If all cycles of π have the same length, π is *order preserving* and it is $\pi(O) = O$ for any sequence O. See [6, Lemma 2.2] or the proof to the following Lemma 5 in the full version for more details.

since $\deg(u) = \deg(v) < \deg(v')$, contradicting the maximality of $\deg(u)$. The last inequality follows from the fact that $\deg(v) \leq \deg(v')$ already holds in the block of G that contains v and v', but as v' is a cut-vertex, it has at least one neighbor outside that block. ◀

To solve instances without pipes in linear time, note that a planar embedding of such an instance is valid if and only if it satisfies the Q-constraints. As Q-vertices only have a binary choice for their rotation, it is relatively easy to synchronize them via a 2-SAT formula. Linear-time algorithms follow from, e.g., [6], and can also be obtained from techniques similar to those used by Fulek and Tóth [15] for cubic graphs. For the sake of completeness, we present a self-contained solution in the full version.

▶ **Lemma 7** (∗). *An instance of* Synchronized Planarity *without pipes can be solved in $O(m)$ time. A valid embedding can be computed in the same time, if it exists.*

3.6 Finding a Reduced Instance

As mentioned above, we exhaustively apply the operations `EncapsulateAndJoin`, `PropagatePQ`, and `SimplifyMatching`. We claim that this algorithm terminates and yields a reduced instance after a polynomial number of steps. The key idea is that the operations always make progress by either reducing the number of pipes, or by splitting pipes into pipes of smaller degree. This suggests that, eventually, we arrive at an instance without pipes. However, there are two caveats. First, the encapsulation in the first step of `EncapsulateAndJoin` creates new pipes and thus has the potential to undo progress. Second, the smaller pipes resulting from splitting a pipe with `PropagatePQ` might cause further growth of the instance, potentially causing a super-polynomial number of steps.

We resolve both issues by using a more fine-grained measure of progress in the form of a potential function. To overcome the first issue, we show that for each application of `EncapsulateAndJoin`, the progress that is undone in the first step is outweighed by the progress made through the following join in the second step. Similarly, for the second issue, we show that the sum of the parts is no bigger than the whole when splitting pipes.

As P-vertices of degree 3 or less are converted to Q-vertices (see Section 3.1), we use $\deg^*(u) = \deg^*(v) = \deg^*(\rho) = \max\{\deg(x) - 3, 0\}$ to denote the number of incident edges that keep a P-vertex u (and also the other endpoint v of its pipe $\rho = (u, v, \varphi_{uv})$) from becoming converted to a Q-vertex. We also partition the set of all pipes $\mathcal{P}$ into the two cells $\mathcal{P}_{CC}$ and $\mathcal{P}_B = \mathcal{P} \setminus \mathcal{P}_{CC}$, where $\mathcal{P}_{CC}$ contains all pipes where both endpoints are cut-vertices. We define the *potential* of an instance I as $\Phi(I) = \sum_{\rho \in \mathcal{P}_B} \deg^*(\rho) + \sum_{\rho \in \mathcal{P}_{CC}} (2\deg^*(\rho) - 1)$. We show that the operations always decrease this potential.

▶ **Lemma 8** (∗). *For an instance $I = (G, \mathcal{P}, \mathcal{Q}, \psi)$ of* Synchronized Planarity *and an instance $I' = (G', \mathcal{P}', \mathcal{Q}', \psi')$ that results from application of either* `EncapsulateAndJoin`*,* `PropagatePQ` *or* `SimplifyMatching` *to I, the following three properties hold:*

(i) *The potential reduction $\Delta\Phi = \Phi(I) - \Phi(I')$ is at least 1.*
(ii) *The number of nodes added to the graph satisfies $\Delta V = |V(G')| - |V(G)| \leq 2 \cdot \Delta\Phi + 12$.*
(iii) *If the operation replaces a connected component C by one or multiple connected components, then each such component C' satisfies $\Delta E(C) = |E(C')| - |E(C)| \leq 2 \cdot \Delta\Phi$.*

Proof Sketch. We now analyze the effects of `EncapsulateAndJoin` on these three measures. The operations `PropagatePQ` and `SimplifyMatching` are discussed in the full version.

Operation `EncapsulateAndJoin` (ρ, I) in the first step encapsulates both cut-vertices u, v to their own star components. For each block incident to u, this introduces two new vertices that are connected by a new pipe. Let $d_1, \ldots, d_k$ be the degrees of the $k \geq 2$ ray vertices of

u after the encapsulation. As one end of the added pipes is a block-vertex, the potential is increased by $\sum_{i=1}^{k} \max\{d_i - 3, 0\}$. Likewise, the pipes of the k' rays with degrees $d'_1, \dots, d'_{k'}$ around v increase the potential by $\sum_{i=1}^{k'} \max\{d'_i - 3, 0\}$. Using $\deg(\rho) = \sum_{i=1}^{k} d_i = \sum_{i=1}^{k'} d'_i = D \geq 3$ it is $\deg^*(\rho) = \max\{D-3, 0\} = D-3$. In the second step, removing ρ connecting two cut-vertices together with its endpoints reduces the potential by $2\deg^*(\rho) - 1$ and we thus get $\Delta\Phi = 2 \cdot (D-3) - 1 - \sum_{i=1}^{k} \max\{d_i - 3, 0\} - \sum_{i=1}^{k'} \max\{d'_i - 3, 0\}$. In the full version, we show that this yields $\Delta\Phi \geq 1$ as claimed by (i). As the encapsulation generates two vertices for each ray and the join removes two vertices, we have $\Delta V = 2k + 2k' - 2$. We also show that claim (ii) holds as $\Delta V \leq 2 \cdot (\Delta\Phi + 6)$. In the first step of `EncapsulateAndJoin`, two new components with $\deg(u) = \deg(v)$ edges each are added, which are then combined in the second step, yielding a new component with $\sum_{i=1}^{k} d_i$ edges. This is no larger than the components of u or v as required for (iii). ◀

With this lemma, we know that each step decreases the potential by at least 1 without growing the graph too much. As each vertex contributes at most twice its degree, initially $\Phi(I) \leq 2m$. This can then be used to bound the size of instances resulting from applying multiple operations consecutively and finally to bound the time required to find a solution for an instance.

▶ **Theorem 9** (∗). *Synchronized Planarity can be solved in $O(m^2)$ time.*

Proof Sketch. By Lemma 8 the potential function decreases with each applied operation. As initially $\Phi(I) \leq 2m$, a reduced instance I' is reached after $k \leq 2m$ operations. It can be shown that each connected component of I' has $O(m)$ edges, allowing an embedding tree to be computed in $O(m)$ time. Each of the k operations runs in linear time once the PQ-tree it requires is available. In total, it thus takes $O(m^2)$ time to reach a reduced instance. As its size is also in $O(m^2)$, Lemma 7 can be applied to find a solution in $O(m^2)$ time. ◀

4 Conclusion

We have given a quadratic-time algorithm for Synchronized Planarity, which improves the previous $O(n^8)$-time algorithm for the linear-time equivalent problem Atomic Embeddability [15]. Similar to Goldberg and Tarjan's push-relabel algorithm, it relies on few and simple operations that can be applied pretty much in an arbitrary order. This highlights where and how progress is made and thereby clearly exposes key ideas that also underlie the algorithm for Atomic Embeddability. For a direct comparison of the approaches, we refer to the full version.

The applications of Synchronized Planarity include solving Clustered Planarity, Atomic Embeddability, Connected SEFE and Partially PQ-constrained Planarity in quadratic time, thanks to linear-time reductions to Synchronized Planarity for all of them described in the full version. This improves over the previously fastest algorithms via the $O(n^8)$-time algorithm for Atomic Embeddability. In the case of Connected SEFE the reduction used in [15] includes a quadratic blowup and therefore yields an $O(n^{16})$-algorithm. Our direct linear-time reduction leads to a quadratic algorithm.

References

1 Hugo A. Akitaya, Radoslav Fulek, and Csaba D. Tóth. Recognizing weak embeddings of graphs. *ACM Transactions on Algorithms*, 15(4):1–27, 2019. `doi:10.1145/3344549`.

2 Patrizio Angelini and Giordano Da Lozzo. SEFE = c-planarity? *The Computer Journal*, 59(12):1831–1838, 2016. `doi:10.1093/comjnl/bxw035`.

3 Patrizio Angelini and Giordano Da Lozzo. Clustered planarity with pipes. *Algorithmica*, 81(6):2484–2526, 2019. doi:10.1007/s00453-018-00541-w.

4 Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/bf01961541.

5 Thomas Bläsius, Annette Karrer, and Ignaz Rutter. Simultaneous embedding: Edge orderings, relative positions, cutvertices. *Algorithmica*, 80(4):1214–1277, 2017. doi:10.1007/s00453-017-0301-9.

6 Thomas Bläsius and Ignaz Rutter. Simultaneous PQ-ordering with applications to constrained embedding problems. *ACM Trans. Algorithms*, 12(2):16:1–16:46, 2016. doi:10.1145/2738054.

7 Thomas Bläsius, Stephen G. Kobourov, and Ignaz Rutter. Simultaneous embedding of planar graphs. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 11, pages 349–381. CRC Press, Taylor & Francis Group, 2013. arXiv:1204.5853.

8 Thomas Bläsius and Ignaz Rutter. A new perspective on clustered planarity as a combinatorial embedding problem. *Theoretical Computer Science*, 609:306–315, 2016. doi:10.1016/j.tcs.2015.10.011.

9 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. doi:10.1016/s0022-0000(76)80045-1.

10 Johannes Carmesin. Embedding simply connected 2-complexes in 3-space – V. A refined Kuratowski-type characterisation. *CoRR*, 2017. arXiv:1709.04659v3.

11 Pier Francesco Cortese, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Maurizio Pizzonia. C-planarity of c-connected clustered graphs. *Journal of Graph Algorithms and Applications*, 12(2):225–262, 2008. doi:10.7155/jgaa.00165.

12 Pier Francesco Cortese and Maurizio Patrignani. Clustered planarity = flat clustered planarity. In Therese C. Biedl and Andreas Kerren, editors, *Proceedings of the 26th International Symposium on Graph Drawing and Network Visualization (GD'18)*, volume 11282 of *LNCS*, pages 23–38. Springer, 2018. doi:10.1007/978-3-030-04414-5_2.

13 Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. In Paul G. Spirakis, editor, *Proceedings of the 3rd Annual European Symposium on Algorithms (ESA'95)*, volume 979 of *LNCS*, pages 213–226. Springer, 1995. doi:10.1007/3-540-60313-1_145.

14 Radoslav Fulek, Jan Kynčl, Igor Malinović, and Dömötör Pálvölgyi. Clustered planarity testing revisited. *The Electronic Journal of Combinatorics*, 22(4), 2015. doi:10.37236/5002.

15 Radoslav Fulek and Csaba D. Tóth. Atomic embeddability, clustered planarity, and thickenability. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 2876–2895. SIAM, 2020. doi:10.1137/1.9781611975994.175.

16 Carsten Gutwenger, Michael Jünger, Sebastian Leipert, Petra Mutzel, Merijam Percan, and René Weiskircher. Advances in c-planarity testing of clustered graphs. In Stephen G. Kobourov and Michael T. Goodrich, editors, *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, volume 2528 of *LNCS*, pages 220–235. Springer, 2002. doi:10.1007/3-540-36151-0_21.

17 Carsten Gutwenger, Karsten Klein, and Petra Mutzel. Planarity testing and optimal edge insertion with embedding constraints. *Journal of Graph Algorithms and Applications*, 12(1):73–95, 2008. doi:10.7155/jgaa.00160.

18 Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In Joe Marks, editor, *Proceedings of the 8th International Symposium on Graph Drawing (GD'00)*, volume 1984 of *LNCS*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

19 F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.*, 4(3):221–225, 1975. doi:10.1137/0204019.

20 John E. Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973. doi:10.1137/0202012.

21 Wen-Lian Hsu. PC-trees vs. PQ-trees. In Jie Wang, editor, *Proceedings of the 7th Annual International Conference on Computing and Combinatorics (COCOON'01)*, volume 2108 of *LNCS*, pages 207–217. Springer, 2001. doi:10.1007/3-540-44679-6_23.

22 Thomas Lengauer. Hierarchical planarity testing algorithms. *Journal of the ACM*, 36(3):474–509, 1989. doi:10.1145/65950.65952.

23 L. Neuwirth. An algorithm for the construction of 3-manifolds from 2-complexes. *Mathematical Proceedings of the Cambridge Philosophical Society*, 64(3):603–614, 1968. doi:10.1017/S0305004100043279.

24 Marcus Schaefer. Toward a theory of planarity: Hanani-tutte and planarity variants. *Journal of Graph Algorithms and Applications*, 17(4):367–440, 2013. doi:10.7155/jgaa.00298.

25 Wei-Kuan Shih and Wen-Lian Hsu. A new planarity test. *Theoretical Computer Science*, 223(1-2):179–191, 1999. doi:10.1016/s0304-3975(98)00120-0.

26 Roberto Tamassia, editor. *Handbook of Graph Drawing and Visualization.* CRC Press, Taylor & Francis Group, 2014. doi:10.1201/b15385.