Parameterized Complexity of Simultaneous Planarity

Master Thesis of

Matthias Pfretzschner

At the Department of Informatics and Mathematics
Chair of Theoretical Computer Science

UNIVERSITÄT PASSAU

Reviewers:	Prof. Dr. Ignaz Rutter
	Prof. Dr. Dirk Sudholt
Advisor:	Simon Fink, M. Sc.

Time Period: 20th May 2022 – 23rd November 2022

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Passau, November 21, 2022

Abstract

Given two graphs $G^{①}$ and $G^{②}$ that share some vertices and edges forming a shared graph $G = G^{①} \cap G^{②}$, the problem Simultaneous Embedding With Fixed Edges (SEFE) asks whether there exist planar drawings of $G^{①}$ and $G^{②}$ that coincide on G. While the generalization of SEFE to k input graphs is $\mathcal{NP}$-complete, SEFE is still an open problem for only two input graphs. In this work, we explore the parameterized complexity of SEFE. We start by developing FPT-algorithms for SEFE parameterized by the vertex cover number and feedback edge set number, respectively, of the union graph $G^{\cup} = G^{①} \cup G^{②}$. Subsequently, we show that SEFE is FPT parameterized by the vertex cover number plus the number of degree-1 vertices of the shared graph.

For instances with a connected shared graph G, SEFE has recently been solved in quadratic time using a reduction to the problem Synchronized Planarity. We show how the resulting instance can be augmented to also handle the case where the shared graph is disconnected. Unfortunately, we cannot solve this augmented Synchronized Planarity problem in polynomial time in the general case. However, for restricted cases, we derive an FPT-algorithm for SEFE parameterized by the number of connected components and the maximum degree of the shared graph. Finally, we use a similar augmented Synchronized Planarity instance to solve SEFE in quadratic time, if both input graphs $G^{①}$ and $G^{②}$ are biconnected and have maximum degree 4. This includes cases where the shared graph has maximum degree 4, while existing algorithms only solve instances where the shared graph has maximum degree 3.

Deutsche Zusammenfassung

Seien $G^{①}$ und $G^{②}$ zwei Graphen mit einem gemeinsamen Graph $G = G^{①} \cap G^{②}$. Das Problem Simultaneous Embedding With Fixed Edges (SEFE) stellt die Frage, ob es zwei planare Zeichnungen von $G^{①}$ und $G^{②}$ gibt, die auf G übereinstimmen. Während die Variante von SEFE mit drei oder mehr Graphen $\mathcal{NP}$-vollständig ist, bleibt SEFE weiterhin ein offenes Problem für nur zwei Eingabegraphen. In dieser Arbeit erforschen wir die parametrisierte Komplexität von SEFE. Als Erstes entwickeln wir FPT-Algorithmen für SEFE parametrisiert nach der "Vertex Cover Number" und der "Feedback Edge Set Number" des Vereinigungsgraphen $G^{\cup} = G^{①} \cup G^{②}$. Anschließend zeigen wir, dass SEFE FPT parametrisiert nach der "Vertex Cover Number" plus der Anzahl von Grad-1 Knoten des gemeinsamen Graphen ist.

Für Instanzen mit zusammenhängendem gemeinsamen Graph G wurde SEFE kürzlich mit quadratischer Laufzeit mithilfe einer Reduktion auf das Problem Synchronized Planarity gelöst. Wir zeigen, wie die resultierende Instanz erweitert werden kann, um auch den Fall abzudecken, in dem G nicht zusammenhängend ist. Leider können wir diese erweiterte Instanz im Allgemeinen nicht in Polynomialzeit lösen. Für eingeschränkte Fälle entwickeln wir allerdings einen FPT-Algorithmus für SEFE parametrisiert nach der Anzahl der Zusammenhangskomponenten und dem Maximalgrad des gemeinsamen Graphen G. Zuletzt verwendend wir noch eine ähnliche erweiterte Synchronized Planarity Instanz, um SEFE in quadratischer Zeit zu lösen, wenn beide Eingabegraphen $G^{①}$ und $G^{②}$ zweifach zusammenhängend sind und Maximalgrad 4 haben. Das schließt auch Fälle ein, in denen der gemeinsame Graph G Maximalgrad 4 hat, während bereits existierende Algorithmen nur Instanzen lösen, in denen der gemeinsame Graph Maximalgrad 3 hat.

Contents

1. Introduction

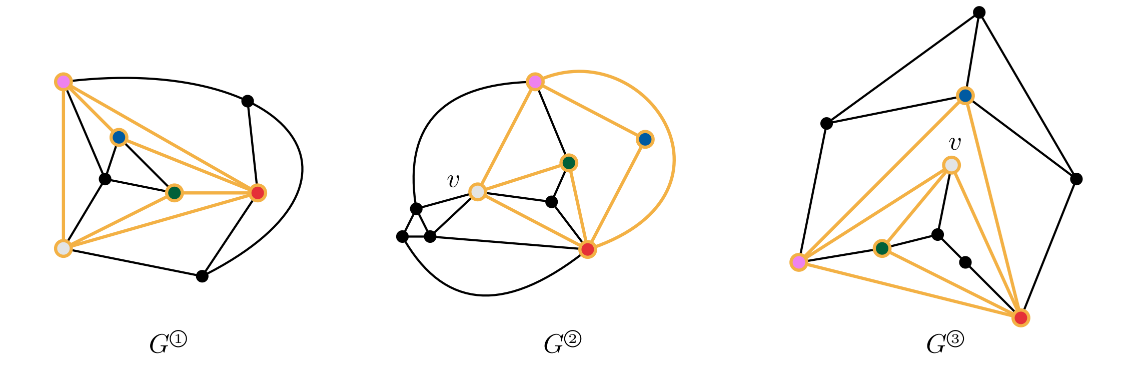
Let $G^{\textcircled{1}} = (V^{\textcircled{1}}, E^{\textcircled{1}})$ and $G^{\textcircled{2}} = (V^{\textcircled{2}}, E^{\textcircled{2}})$ denote two graphs with a common *shared graph* $G = G^{\textcircled{1}} \cap G^{\textcircled{2}} = (V^{\textcircled{1}} \cap V^{\textcircled{2}}, E^{\textcircled{1}} \cap E^{\textcircled{2}})$. The problem Simultaneous Embedding With Fixed Edges (SEFE) asks, whether there exist planar drawings $\Gamma^{\textcircled{1}}$ and $\Gamma^{\textcircled{2}}$ of $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$, respectively, such that $\Gamma^{\textcircled{1}}$ and $\Gamma^{\textcircled{2}}$ induce the same drawing of the shared graph G. The problem definition can be naturally generalized to the problem k-SEFE with $k \geq 2$ input graphs, where each pair of graphs has a shared graph that must be drawn identically. In the restricted *sunflower case*, every pair of input graphs has the same intersection.

One of the main applications for the SEFE problem is dynamic graph drawing. Given a graph that changes over time, a visualization of k individual snapshots of the graph should aesthetically display the changes between successive snapshots. To this end, it is helpful to draw the part of the graph that remains unchanged between snapshots consistently. Figure 1.1 gives an example illustrating this for $k = 3$ snapshots, matching the problem k-SEFE for $k = 3$ in the sunflower case. In Figure 1.1a, the shared graph has a different layout in each of the three drawings, which makes it difficult to distinguish the common features of the drawings, although the shared graph is highlighted in each drawing. In contrast, Figure 1.1b shows drawings of the same three graphs, but the vertices and edges of the shared graph have the same position in each drawing. Such a visualization notably simplifies recognizing the similarities between the “snapshots“. For more than two input graphs, k-SEFE has been proven to be $\mathcal{NP}$-complete [GJP$^+$06], even in the sunflower case [Sch13]. For two input graphs, however, SEFE remains an open problem.

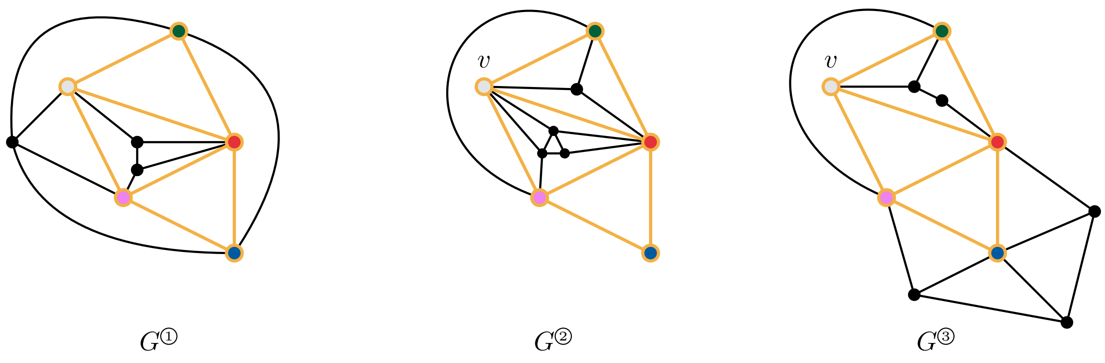
Aside from the aforementioned applications in dynamic graph drawing, SEFE takes a central role in the spectrum of planarity variants. In addition to the standard planarity problem, which asks whether a given graph can be drawn crossing-free, many other famous planarity variants are polynomial-time reducible to SEFE. This includes the problems Partially Embedded Planarity, Partially PQ-Constrained Planarity, (Radial) Level Planarity, Clustered Planarity [Sch13], and Partitioned T-Coherent 2-Page Book Embedding [ABF$^+$12]. A polynomial-time algorithm that solves SEFE would consequently yield a unified planarity test for many well-known planarity variants [Rut20].

1.1 Related Work

Jünger and Schulz [JS09] showed that an instance of SEFE with input graphs $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$ is a yes-instance if and only if there exists a pair of embeddings of $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$ that



(a)



(b)

Figure 1.1: (a) Three planar graphs $G^{①}$, $G^{②}$, and $G^{③}$ with a shared graph $G = G^{①} \cap G^{②} \cap G^{③}$. The edges of G are drawn in orange, and each vertex of G has a unique color. (b) Planar drawings of the same three graphs, but the subgraph G is drawn the same way in all three drawings.

is *compatible*, i.e., the two embeddings must induce the same embedding for the shared graph G. For this to be the case, the pair of embeddings must satisfy the following two requirements [Rut20]. First, the cyclic ordering of the edges around every vertex of G must be identical in both embeddings. Second, for every pair C and C' of connected components in G, the face of C that C' is embedded in must be the same in both embeddings. We call the former property *consistent edge orderings* and the latter property *consistent relative positions*. For example, the edges of G around the vertex v in $G^{②}$ and $G^{③}$ induce different cyclic orders in Figure 1.1a, the shown embeddings therefore do not have consistent edge orderings. In Figure 1.1b, however, the embeddings of $G^{②}$ and $G^{③}$ have consistent edge orderings for the shared graph G. Figure 1.2 shows an example of two embeddings with consistent edge orderings but inconsistent relative positions.

Most approaches for solving SEFE use this constraint-based characterization by Jünger and Schulz [JS09] and test whether the input graphs admit compatible embeddings. However, all existing solutions for SEFE require additional restrictions on the input graphs. Interestingly, there are several solutions that solve SEFE in cases where only consistent edge orderings or only consistent relative positions between the embeddings must be ensured. If the shared graph G is connected, there are no relative positions to consider, because then G only consists of a single connected component. Haeupler et al.[HJL13] solved SEFE in linear time in the even more restricted case where the shared graph G is biconnected using a PQ-tree-based planarity test. Angelini et al.[ABF$^+$12] achieve the same result using a different approach, where they analyze the constraints the two input graphs impose on the embedding of nodes in the SPQR-tree of the shared graph G. Bläsius and Rutter [BR16]

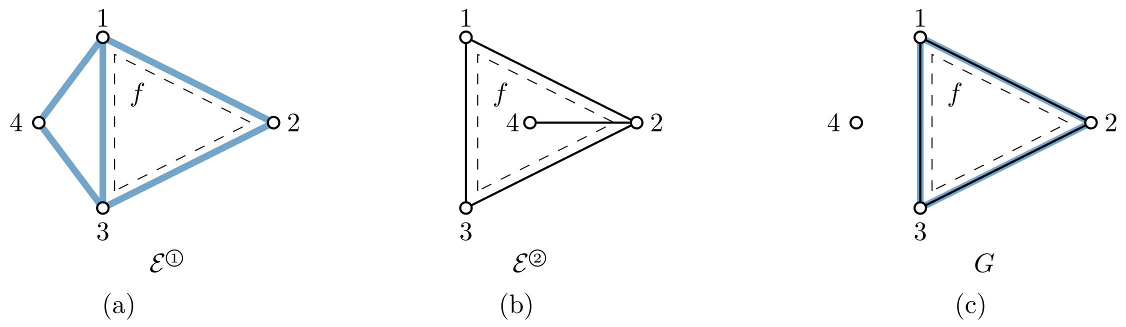


Figure 1.2: Planar embeddings $\mathcal{E}^{①}$ (a) and $\mathcal{E}^{②}$ (b) of $G^{①}$ and $G^{②}$, respectively. Because the shared graph G (c) has maximum degree 2, the edge orderings between embeddings of $G^{①}$ and $G^{②}$ are trivially consistent. But because vertex 4 is placed inside face f of G (shown with dashed lines) in $\mathcal{E}^{②}$, but outside of f in $\mathcal{E}^{①}$, the two embeddings do not have consistent relative positions.

later developed a quadratic-time algorithm for SEFE if both input graphs $G^{①}$ and $G^{②}$ are biconnected and the shared graph G is connected, using a reduction to restricted instances of the problem Simultaneous PQ-Ordering. Very recently, Fulek and Tóth [FT20] achieved a major breakthrough by solving SEFE in polynomial time if the shared graph is connected. This way, they essentially solve the problem of assuring consistent edge orderings between the two input graphs. Bläsius et al. [BFR20] later improved the running time and solved SEFE in quadratic time if the shared graph is connected, using a reduction to the new problem Synchronized Planarity.
On the other hand, there are also algorithms that can ensure consistent relative positions if the edge orderings are trivially consistent. Bläsius and Rutter [BR15] gave a linear-time algorithm that solves SEFE if each connected component of G is a cycle and a quadratic-time algorithm for the case where each connected component of G has a fixed embedding.

While there are algorithms that can handle consistent edge orderings and consistent relative positions individually, unrestricted SEFE instances require both of them to be considered at the same time. Combining these two requirements significantly complicates the problem. Schaefer [Sch13] introduced an algorithm that uses a completely different approach and was the first to handle cases where the shared graph consists of multiple connected components that do not have a trivial embedding. Instead of focusing on compatible embeddings, he uses Hanani-Tutte style theorems to characterize yes-instances of SEFE via crossing numbers in drawings of the *union graph* $G^{\cup} = G^{①} \cup G^{②}$. His algorithm handles consistent edge orderings and relative positions implicitly and solves SEFE in polynomial time if every connected component of the shared graph is biconnected or has maximum degree 3.

Bläsius et al. [BKR18] introduced the first constraint-based algorithm that could handle edge orderings and relative positions at the same time. Their algorithm runs in cubic time and solves SEFE instances where each connected component of the shared graph is biconnected or has maximum degree 3. Since, in this case, the components of the shared graph do not necessarily have a fixed embedding, handling relative positions is quite involved, because the faces of the shared graph depend on the particular embedding. To solve this issue, Bläsius et al. [BKR18] express relative positions with respect to cycles in a cycle basis of the shared graph. They show that, in this restricted case, the important embedding choices in the two input graphs affecting the shared graph are essentially binary decisions and can be combined with the relative positions using a linear system of equations.

There are also other interesting variants of SEFE. The problem Simultaneous Geometric Embedding (SGE) is a more restrictive version of SEFE that additionally requires both drawings to have straight lines. This additional restriction makes the problem $\mathcal{NP}$-hard, even for two graphs [EGJ+07]. The variant Simultaneous Embedding (SE) only requires all vertices of the shared graph to have the same position in both drawings, the drawings of the shared edges may differ. Since a planar graph always admits a planar drawing, even if the position of all vertices is fixed [PW01], any pair of planar graphs is a yes-instance of SE. Instead of developing testing algorithms for SE, the research therefore focuses on drawings with small area and edges with few bends. We refer to the survey by Bläsius et al. [BKR13] for an overview of results for the problems SGE and SE. For a thorough history of developments for the SEFE problem, see the recent survey by Rutter [Rut20].

1.2 Contribution and Outline

This work consists of two parts. On the one hand, we explore the parameterized complexity landscape of SEFE. On the other hand, we try to combine constraint-based solution approaches for SEFE that solve consistent edge orderings and consistent relative positions individually in order to solve SEFE in more general cases.

In Chapter 3, we start by developing a kernelization algorithm for SEFE parameterized by the vertex cover number of the union graph $G^{\cup}$, thus showing that SEFE is FPT parameterized by said parameter. Similarly, we obtain a linear kernel for SEFE parameterized by the feedback edge set number of the union graph $G^{\cup}$ in Chapter 4. In Chapter 5, we show that SEFE is FPT parameterized by the treedepth of the union graph, under the additional restriction that both input graphs must be connected in every subtree of the corresponding treedepth decomposition. In Chapter 6, we show that SEFE is FPT parameterized by the vertex cover number plus the number of degree-1 vertices of the shared graph G.
In the subsequent chapters, we try to extend existing constraint-based approaches in order to solve SEFE in more general cases. Our main focus lies on the algorithm very recently introduced by Bläsius et al. [BFR20], which solves SEFE in quadratic time if the shared graph is connected, by reducing the instance to the new problem Synchronized Planarity. Roughly speaking, Synchronized Planarity takes as input a graph G and a set P of disjoint pairs of vertices of G and asks whether there exists a planar embedding of G such that each pair of vertices in P has the same order of incident edges under some predefined bijection. While the reduction to Synchronized Planarity can ensure consistent edge orderings and can thus solve any instance of SEFE where the shared graph is connected, it generally fails in the presence of multiple connected components, because it cannot ensure consistent relative positions. We ask the natural question whether the resulting Synchronized Planarity instance can be augmented with additional constraints that enforce consistent relative positions.
In Chapter 7, we first characterize the embeddings of the input graphs that satisfy consistent relative positions. Subsequently, we augment the Synchronized Planarity instance to ensure that only those embeddings are admissible. We adjust the Synchronized Planarity algorithm by Bläsius et al. [BFR20] so it can also handle these additional constraints. This way, we develop an FPT-algorithm that solves SEFE parameterized by the number of connected components and the maximum degree of the shared graph. While this algorithm only works in restricted cases, it gives interesting insights on the interplay between relative positions and Synchronized Planarity.
In Chapter 8, we use a very similar approach. We place pairs of triples in the Synchronized Planarity instance that additionally ensure consistent relative positions with respect to cycles in a cycle basis of the shared graph. We thus integrate the cycle-basis approach of Bläsius et al. [BKR18] into the Synchronized Planarity-reduction by Bläsius et al. [BFR20]. This leads to a quadratic-time algorithm that solves SEFE in the case where both exclusive graphs are biconnected and have maximum degree 4.

2. Preliminaries

Graphs and Connectivity. Let $G = (V, E)$ be a simple graph. A graph $H = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. The subgraph H is an *induced subgraph* of G, if E' contains all edges of E connecting two vertices of V'. The graph G is *connected*, if, for every pair $u, v \in V$, there exists a path between u and v in G, otherwise G is *disconnected*. A set $S \subseteq V$ of vertices is a *separating k-set* of G, if $|S| = k$ and the graph $G - S$ obtained after removing S from G is disconnected. The *split components* of a separating k-set S are the maximal subgraphs of G that are not disconnected by removing S from G. A separating 1-set is also called a *cutvertex* and a separating 2-set is also called a *separating pair*. We say that G is *biconnected*, if it contains no cutvertex and *triconnected* if it contains no separating pair. A maximal induced subgraph of G that is connected (respectively biconnected, triconnected) is called a *connected component* (respectively *biconnected component*, *triconnected component*) of G. A biconnected component is also called a *block*. Every vertex of G that is not a cutvertex is located in a unique block of G and called a *block vertex*.
The *contraction* of an edge $e = \{u, v\}$ in G is an operation that first removes e from G and subsequently merges the two vertices u and v. A graph H is a *minor* of G, if H can be obtained from G by a sequence of vertex deletions, edge deletions, and edge contractions.

Circular Orderings. For a ground set X, a *circular ordering* (or *cyclic ordering*) σ of X arranges the elements of X in a clockwise order around a circle. In this way, the cyclic ordering σ defines an asymmetric and transitive ternary relation $<_\sigma$, where $x_1 <_\sigma x_2 <_\sigma x_3$ for $\{x_1, x_2, x_3\} \subseteq X$ indicates that x_2 appears after x_1 and before x_3 in the clockwise ordering around the aforementioned circle. Two circular orderings on the same ground set are equivalent if they induce the same ternary relation. For a circular ordering $\sigma = \langle x_1, \dots, x_n \rangle$, we let $\overline{\sigma} = \langle x_n, \dots, x_1 \rangle$ denote its reversal.
For a set $A \subseteq X$, we let $\sigma[A]$ denote the circular ordering of A obtained from σ by removing all elements of $X \setminus A$ from σ. We say that a cyclic ordering of a subset $A \subseteq X$ is a *partial ordering* of X. An ordering σ of X *satisfies the partial ordering* σ' for a subset $A \subseteq X$ if $\sigma[A] = \sigma'$. We say that a set A is *consecutive* in σ, if for every $a_1, a_2 \in A$ and for every $x_1, x_2 \in X \setminus A$, $x_1 <_\sigma a_1 <_\sigma x_2$ holds if and only if $x_1 <_\sigma a_2 <_\sigma x_2$ holds. For a consecutive set A and an element $c \notin A$, we let $\sigma[A \to c]$ denote the circular ordering of $(X \setminus A) \cup \{c\}$ obtained from σ by replacing A with c.

Drawings, Embeddings, and Planarity. A *drawing* Γ of G maps every vertex $v \in V$ to a coordinate $\Gamma(v) \in \mathbb{R}^2$ and every edge $\{u, v\} \in E$ to a simple arc between the coordinates $\Gamma(u)$ and $\Gamma(v)$. A drawing is a *planar drawing*, if for any pair of edges in E, the corresponding

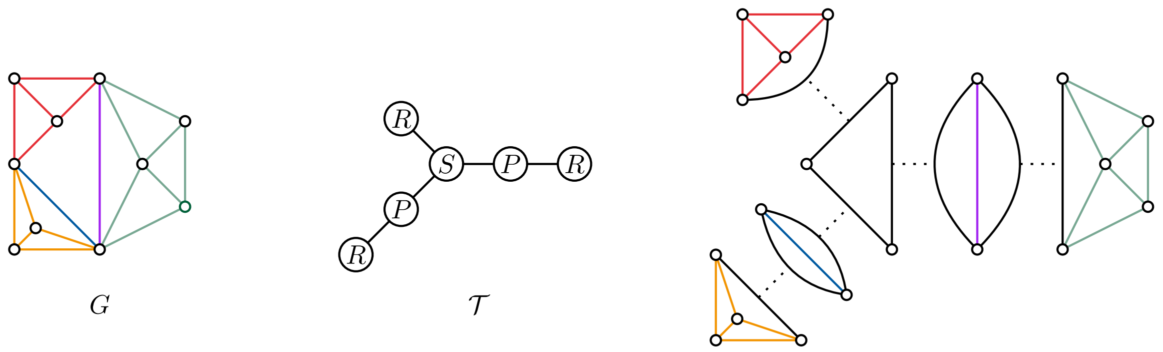


Figure 2.1: A biconnected planar graph G with its corresponding SPQR-tree $\mathcal{T}$. As a simplification, the Q-nodes of $\mathcal{T}$ are omitted. Illustrated on the right are the skeletons of the respective nodes of $\mathcal{T}$. The virtual edges of the skeletons are colored black, all other edges are marked with the color they are also marked with in G. Every pair of twin edges is connected via a dashed line.

arcs of Γ do not intersect at interior points. The graph G is a *planar graph*, if there exists a planar drawing of G. By Kuratowski's Theorem [Kur30], a graph is planar if and only if it does not contain the complete graph K_5 or the complete bipartite graph $K_{3,3}$ as a minor. Every drawing Γ of G uniquely defines *rotation system* of G, a circular order for the edges around each vertex in G. In this sense, drawings define an equivalence relation, where two drawings Γ_1 and Γ_2 are equivalent if and only if Γ_1 and Γ_2 induce the same rotation system. The equivalence classes of this equivalence relation are called *(combinatorial) embeddings*. The edges of a planar embedding $\mathcal{E}$ partition the plane into several regions, called the *faces* of $\mathcal{E}$. For a subgraph H of G, $\mathcal{E}[H]$ denotes the restriction of $\mathcal{E}$ to H. With $\mathcal{E}(v)$ we denote the circular order of the edges incident to a vertex $v \in V$ induced by the embedding $\mathcal{E}$.

SPQR-Trees. A pair $\{u, v\} \in V^2$ is a *split pair* of G, if $\{u, v\}$ is either a separating pair or a pair of adjacent vertices. An *SPQR-tree* [BT96] $\mathcal{T}$ is a rooted tree that decomposes a biconnected graph G along its split pairs; see Figure 2.1 for an example. It consists of the four node types S, P, Q, and R, where the Q-nodes constitute the leaves of $\mathcal{T}$ and correspond bijectively to the edges of G. Every node μ of $\mathcal{T}$ is associated with a biconnected multigraph skel(μ), called the *skeleton* of μ. The edges of skel(μ) are called *virtual edges*. The virtual edges of μ correspond bijectively to the neighbors of μ in $\mathcal{T}$, i.e., if μ contains k virtual edges, then μ also has degree k in $\mathcal{T}$. For two adjacent nodes of $\mathcal{T}$, this bijective mapping defines pairs of virtual edges, called *twin edges*, that are associated with each other. After rooting $\mathcal{T}$ at an arbitrary node, the virtual edge of μ corresponding to the parent of μ in $\mathcal{T}$ is called the *parent edge*.
The *pertinent graph* pert(μ) of a node μ of $\mathcal{T}$ is recursively defined as follows. If μ is a Q-node, then pert(μ) consists of the edge of G that μ corresponds to. Otherwise, μ is an inner node of $\mathcal{T}$, and pert(μ) is obtained by replacing every virtual edge of skel(μ) with the pertinent graph of the child it corresponds to and by subsequently removing the parent edge in skel(μ).
For a virtual edge ε of skel(μ), let μ' be the neighbor of μ that ε corresponds to. The *expansion graph* exp(ε) of ε in skel(μ) is defined as the pertinent graph of the neighbor μ' after rooting $\mathcal{T}$ at μ. For a virtual edge $\varepsilon = uv$ of μ, we say that a vertex x of G is *contained* in ε, if $x \notin \{u, v\}$ and x is contained in the expansion graph exp(ε) of μ.
Given a graph G, the SPQR-tree $\mathcal{T}$ can be obtained by recursively splitting the graph along its split pairs. Initially, $\mathcal{T}$ consists of a single node μ with skel$(\mu) = G$. For a split pair $\{u, v\}$ in skel(μ), let $S_1, \dots, S_l$ denote the split components of $\{u, v\}$. Note that $S_1 \cup \dots \cup S_l = \text{skel}(\mu)$ and $S_i \cap S_j = \{u, v\}$ for $1 \leq i, j \leq l$. If $l = 2$, split μ into two adjacent nodes μ_1 and μ_2, where $skel(\mu) = S_1 + uv$ and $skel(\mu_2) = S_2 + uv$. The virtual edges uv

appearing in $skel(\mu_1)$ and $skel(\mu_2)$ are the corresponding twin edges. If $l > 2$, create a new node μ' whose skeleton consists of the two vertices u and v and l parallel virtual edges $\varepsilon_1, \dots, \varepsilon_l$. For $1 \leq i \leq l$, create a new node μ_i with $\text{skel}(\mu_i) = S_i + uv$ adjacent to μ' and associate the virtual edge uv in $\text{skel}(\mu_i)$ with its twin ε_i of $\text{skel}(\mu')$.
Exhaustively applying this procedure eventually yields the SPQR-tree $\mathcal{T}$ of G, where each inner node of $\mathcal{T}$ is either an S-node, a P-node, or an R-node. The skeleton $\text{skel}(\mu)$ is a simple cycle if μ is an S-node ("*series*"), several (at least 3) parallel edges if μ is a P-node ("*parallel*"), or a triconnected graph if μ is an R-node ("*rigid*"). Fixing a planar embedding of G completely fixes the embedding of the skeletons of all nodes in $\mathcal{T}$ and conversely, choosing a planar embedding for every skeleton in $\mathcal{T}$ uniquely defines a planar embedding of G. In this sense, the SPQR-tree of G breaks down the possible embedding choices of G into simpler embedding choices for the individual skeletons of μ. Since the skeleton $\text{skel}(\mu)$ of an R-node μ is triconnected, the embedding of $\text{skel}(\mu)$ is fixed up to mirroring, thus an R-node only leaves a binary embedding decision. For a P-node μ, the k parallel virtual edges in $\text{skel}(\mu)$ can be ordered arbitrarily, leaving $(k-1)!$ possible embeddings of $\text{skel}(\mu)$ after fixing a virtual edge on the outer face of $\text{skel}(\mu)$. Since the skeleton of an S-node is a simple cycle, its embedding is completely fixed. Because the skeletons of Q-nodes also have a fixed embedding, all embedding choices of a biconnected graph G therefore break down to embedding choices at the P-nodes and R-nodes of the SPQR-tree of G. The SPQR-tree of G can be computed in linear time [GM00]. For a P-node μ of $\mathcal{T}$, we call the two vertices u and v in $\text{skel}(\mu)$ the *poles* of μ.

PQ-trees and PC-trees. The *PQ-tree* is a data structure introduced by Booth and Lueker [BL76] that represents circular orderings of a ground set X, subject to specific consecutivity constraints. While a PQ-tree is rooted, there also exists an unrooted variant called *PC-tree* [SH99]. PQ-trees and PC-trees are equivalent [Hsu01] and can thus be used interchangeably.
The leaves $L(T)$ of a PC-tree T correspond bijectively to the elements in X and its inner nodes $I(T)$ are partitioned into *P-nodes* and *C-nodes* (respectively *Q-nodes* for PQ-trees). The edges incident to P-nodes may be ordered arbitrarily, while for C-nodes, this order is fixed up to reversal. The set of circular orders of $L(T)$ that can be obtained by such reorderings in T is called $\sigma(T)$. A circular order σ of $L(T)$ *satisfies* T if $\sigma \in \sigma(T)$. A PC-tree T is *trivial*, if it consists of a single P-node incident to all leaves of T. Note that, in this case, $\sigma(T)$ contains all distinct circular permutations of $L(T)$.
The circular orders that satisfy T can also be described as a set of *restrictions* (or *consecutivity contraints*) $R(T) \subseteq \mathcal{P}(L(T))$. By this definition, a circular order σ satisfies T if all $R \in R(T)$ are consecutive in σ. A set $A \subseteq L(T)$ is *consecutive* in T if $A \in R(T)$. For a consecutive set $A \in R(T)$, we define the *pertinent subtree* $\text{pert}(T, A)$ recursively as follows. Every leaf in A belongs to $\text{pert}(T, A)$ and an inner node of T belongs to $\text{pert}(T, A)$ if all of its neighbors, except for one, belong to $\text{pert}(T, A)$.
We now describe the four PC-tree operations we will use later on. See Figure 2.2 for an example illustrating each operation.

`Update` Let T be a PC-tree and let $A \subseteq L(T)$ be a set of its leaves. The operation `Update` produces a new PC-tree T' that ensures that A is consecutive in T', thus $\sigma(T') = \{\sigma \in \sigma(T) \mid A \text{ is consecutive in } \sigma\}$ and $R(T') = R(T) \cup \{A\}$. If $\sigma(T') = \emptyset$, then the operation produces a trivial *null tree*. Hsu and McConnell [HM03, HM04] showed that the procedure `Update` can be implemented to run in amortized linear time in the size of A.

`Intersect` Let T_1, T_2 denote two PC-trees with $L(T_1) = L(T_2)$. The operation `Intersect` produces a new PC-tree T' that contains the constraints of T_1 and T_2, thus $R(T') = R(T_1) \cup R(T_2)$ and $\sigma(T') = \sigma(T_1) \cap \sigma(T_2)$. Booth [Boo75] gave a linear-time algorithm for this operation.

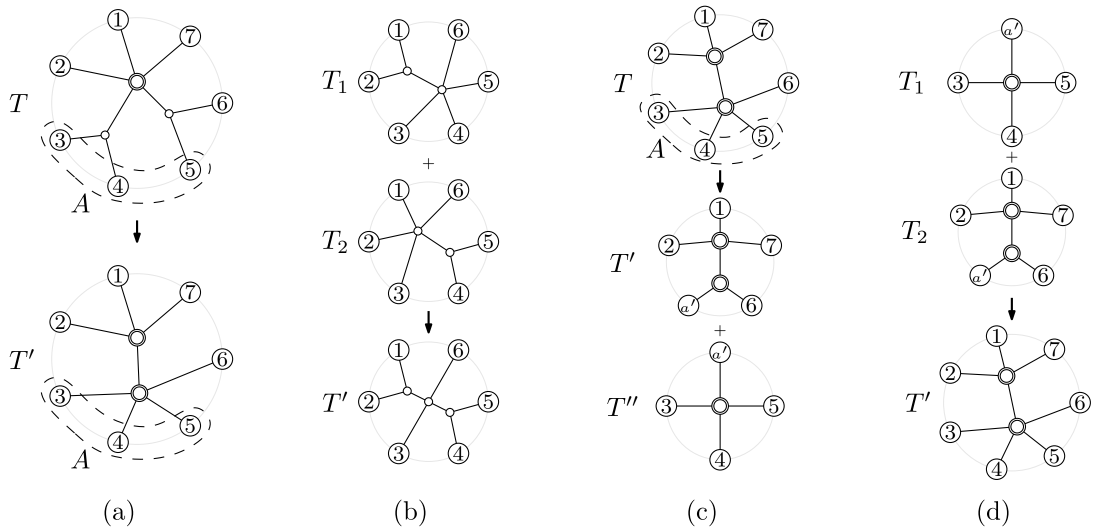


Figure 2.2: The PC-tree operations `Update` (a), `Intersect` (b), `Split` (c), and `Merge` (d). The P-nodes of each PC-tree are drawn as small circles, the C-nodes as big double circles.

`Split` Let T denote a PC-tree and let $A \subseteq L(T)$ denote a consecutive set of its leaves. The procedure `Split` produces a new PC-tree T' by replacing the pertinent subtree $\text{pert}(T, A)$ with a single leaf a'. The operation also creates another PC-tree T'' by replacing the pertinent subtree $\text{pert}(T, L(T) \setminus A)$ with the leaf a'. It holds that $\sigma(T') = \{\sigma[A \to a'] \mid \sigma \in \sigma(T)\}$ and $\sigma(T'') = \{\sigma[(L(T) \setminus A) \to a'] \mid \sigma \in \sigma(T)\}$.

`Merge` Let T_1, T_2 denote two PC-Trees sharing exactly one leaf l. Let x_1 and x_2 be the two neighbors of l in T_1 and T_2, respectively. The operation `Merge` creates a new PC-Tree T' by first removing l from T_1 and T_2 and subsequently adding the edge x_1x_2 between T_1 and T_2.

Embedding Trees and Partial Constraints. Let v be a vertex of a planar biconnected graph G and let $E(v)$ denote the set of its incident edges. Let further $\mathcal{K}_v$ denote the set of circular orderings σ of $E(v)$ such that there exists a planar embedding $\mathcal{E}$ of G with $\sigma = \mathcal{E}(v)$. The circular orderings of $E(v)$ that are induced by some planar embedding of G can be efficiently represented by a PQ-tree T_v [BL76] with $L(T_v) = E(v)$ and $\sigma(T) = \mathcal{K}_v$. We call T_v the *embedding tree* of v in G. If an embedding $\mathcal{E}$ induces an order $\mathcal{E}(v) \notin \sigma(T_v)$, $\mathcal{E}$ cannot be planar. Every Q-node of T_v originates from an R-node of the SPQR-tree $\mathcal{T}$ of G and every P-node of T_v stems from a P-node of $\mathcal{T}$ [BR16]. Note that choosing a circular ordering $\sigma \in \sigma(T_v)$ independently at every vertex v of G does not necessarily lead to a planar embedding of G. Instead, Q-nodes of different embedding trees stemming from the same R-node μ of $\mathcal{T}$ must be flipped consistently with the rotation of μ. Similarly, two P-nodes stemming from the same P-node μ of $\mathcal{T}$ must be ordered compatibly with each other under the bijective mapping induced by μ. If both poles of a P-node μ have a trivial embedding tree, we refer to μ as a *(trivial) bond*.
In addition to embedding trees, we will also sometimes require a vertex v to satisfy additional ordering constraints for subsets of $E(v)$ in any embedding of G. Such a constraint is called a *partial constraint* and is represented by a PQ-tree P_v with $L(P_v) \subseteq E(V)$. An embedding $\mathcal{E}$ *satisfies* the partial constraint P_v if $\mathcal{E}(v)[L(P_v)] \in \sigma(P_v)$.

Cycle bases. For an undirected graph G, let $\mathcal{C}$ denote the set of all cycles in G. For two cycles $C_1, C_2 \in \mathcal{C}$, define the *sum* $C_1 \oplus C_2$ as the exclusive disjunction of the edges in C_1 and C_2, i.e., an edge is present in $C_1 \oplus C_2$ if and only if it is present in exactly one of the cycles C_1 and C_2. In combination with an analogous scalar multiplication $\odot$, $(\mathcal{C}, \oplus, \odot)$

forms a vector space over the field $\mathbb{F}_2$. We call a basis of this vector space a *cycle basis* of G.

SEFE. Let $G^{\textcircled{1}} = (V^{\textcircled{1}}, E^{\textcircled{1}})$ and $G^{\textcircled{2}} = (V^{\textcircled{2}}, E^{\textcircled{2}})$ denote two graphs with a common *shared graph* $G = G^{\textcircled{1}} \cap G^{\textcircled{2}} = (V^{\textcircled{1}} \cap V^{\textcircled{2}}, E^{\textcircled{1}} \cap E^{\textcircled{2}})$. The problem SIMULTANEOUS EMBEDDING WITH FIXED EDGES (SEFE) asks, whether there exist planar drawings $\Gamma^{\textcircled{1}}$ and $\Gamma^{\textcircled{2}}$ of $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$, respectively, such that $\Gamma^{\textcircled{1}}$ and $\Gamma^{\textcircled{2}}$ induce the same drawing on the shared graph G [Rut20]. We refer to such a pair of drawings as a *simultaneous drawing* and to the corresponding pair of embeddings as a *simultaneous embedding.* We usually refer to the two input graphs $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$ as the *exclusive graphs.* The graph $G^{\cup} = G^{\textcircled{1}} \cup G^{\textcircled{2}} = (V^{\textcircled{1}} \cup V^{\textcircled{2}}, E^{\textcircled{1}} \cup E^{\textcircled{2}})$ is called the *union graph.*
For brevity, we describe instances of SEFE using the union graph $G^{\cup}$. To this end, every edge and vertex of $G^{\cup}$ is marked either ①-*exclusive* (respectively ②-*exclusive*), if it is only contained in $G^{\textcircled{1}}$ (respectively in $G^{\textcircled{2}}$), or *shared*, if it is contained in both $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$ (i.e., it is contained in the shared graph G). For a vertex v of the shared graph G, we let $v^{\textcircled{1}}$ and $v^{\textcircled{2}}$ denote the corresponding vertices in $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$, respectively. We refer to connected components of the shared graph G as *shared components.*
We use a consistent drawing style to illustrate SEFE instances in our figures. Edges and vertices of $G^{\textcircled{1}}$ are thick and blue, edges and vertices of $G^{\textcircled{2}}$ are thin and black. Shared edges are drawn as overlapping blue and black lines, shared vertices are filled white; see Figure 2.3a for an example.
Jünger and Schulz [JS09] showed that an instance of SEFE admits a simultaneous embedding if and only if there exists a pair of embeddings of $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$ that is *compatible*, i.e., the two embeddings must induce the same embedding for the shared graph G. For this to be the case, the pair of embeddings must satisfy the following two requirements [Rut20]. First, the cyclic ordering of the edges around every vertex of G must be identical in both embeddings. Second, for every pair C and C' of connected components in G, the face of C that C' is embedded in must be the same in both embeddings. We call the former property *consistent edge orderings* and the latter property *consistent relative positions.*
Note that any SEFE instance with a non-planar exclusive graph is a trivial no-instance. Since the planarity of a graph can be tested in linear time [HT74], we thus assume that both exclusive graphs are planar. Similarly, Bläsius et al. [BKR18] gave a linear-time preprocessing step that ensures that both exclusive graphs are connected. Therefore, we assume that both exclusive graphs of our SEFE instance (and consequently also the union graph) are connected.

Link graphs. Let μ denote a P-node in a block B of the shared graph G with poles u and v. We adopt some notation from Bläsius et al. [BKR18]. We say that two virtual edges ε_1 and ε_2 of μ are ①-*linked*, if there exists a path (called ①-*link*) between vertices of ε_1 and ε_2 in $G^{\textcircled{1}}$ that is vertex-disjoint from B (except for the endpoints of the path in ε_1 and ε_2). Two virtual edges are *union-linked*, if there exists an analogous path (called *union-link*) in the union graph $G^{\cup}$. Let the ①-*link graph* $L_\mu^{\textcircled{1}}$ of μ be the graph that contains the virtual edges of μ as nodes, and two nodes are adjacent if and only if the corresponding virtual edges are ①-linked. We analogously define the *union-link graph* $L_\mu^{\cup}$ of μ. The graphs $L_\mu^{\textcircled{1}}$ and $L_\mu^{\textcircled{2}}$ are subgraphs of $L_\mu^{\cup}$, however, two virtual edges may be union-linked but not ①-linked or ②-linked and thus $L_\mu^{\cup}$ is not the union of $L_\mu^{\textcircled{1}}$ and $L_\mu^{\textcircled{2}}$. We call the union $L_\mu^{\textcircled{1}} \cup L_\mu^{\textcircled{2}}$ the *exclusive-link graph* of μ.

Connected SEFE and Synchronized Planarity. The problem CONNECTED SEFE is the restriction of the problem SEFE to instances where the shared graph G is connected. Bläsius et al. [BFR20] showed that CONNECTED SEFE can be solved in time $\mathcal{O}(n^2)$ using a reduction to the problem SYNCHRONIZED PLANARITY. The problem SYNCHRONIZED

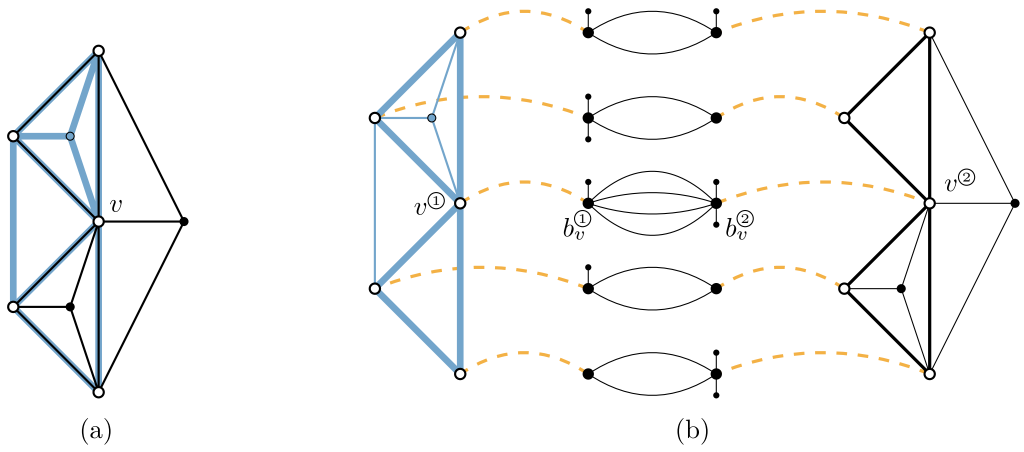


Figure 2.3: A linear-time reduction from an instance of SEFE with a connected shared graph (a) to an equivalent instance of Synchronized Planarity (b).

Planarity takes as input a tuple $(G,\ \mathcal{P},\ \mathcal{Q},\ \psi)$, where $G = (P \cup Q, E)$ is a multigraph consisting of *P-vertices* P and *Q-vertices* Q. The set $\mathcal{P}$ contains *pipes*, i.e., triples $p = (u, v, \varphi_{uv})$, where u and v are P-vertices of the same degree and φ_{uv} is a bijection between their incident edges. An embedding $\mathcal{E}$ of G satisfies pipe p if the edges incident to u and v have opposite rotations under φ_{uv} in $\mathcal{E}$, i.e., $\varphi_{uv}(\mathcal{E}(u)) = \overline{\mathcal{E}(v)}$. Each P-vertex is restricted to appear in at most one pipe. The mapping ψ allots a specific rotation to each Q-vertex. The set $\mathcal{Q}$ is a partition of the vertices in Q, where each cell defines a *Q-constraint*. An embedding $\mathcal{E}$ of G satisfies a Q-constraint $X \in \mathcal{Q}$, if it holds that either $\mathcal{E}(v) = \psi(v)$ for all $v \in X$ or $\mathcal{E}(v) = \overline{\psi(v)}$ for all $v \in X$. The tuple $(G,\ \mathcal{P},\ \mathcal{Q},\ \psi)$ is a yes-instance of Synchronized Planarity if and only if G admits an embedding where all pipes and Q-constraints are satisfied.

Using Synchronized Planarity, the problem Connected SEFE can be solved as follows [BFR20]. Given an instance of SEFE with input graphs $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$, add two new vertices $b_v^{\textcircled{1}}$ and $b_v^{\textcircled{2}}$ for each shared vertex v appearing in the exclusive graphs as $v^{\textcircled{1}}$ and $v^{\textcircled{2}}$, respectively. For each shared edge incident to v, add a parallel edge between $b_v^{\textcircled{1}}$ and $b_v^{\textcircled{2}}$, creating a bond between $b_v^{\textcircled{1}}$ and $b_v^{\textcircled{2}}$ where the parallel edges correspond to the shared edges incident to v. Create two pipes $(v^{\textcircled{1}}, b_v^{\textcircled{1}}, \varphi_1)$ and $(v^{\textcircled{2}}, b_v^{\textcircled{2}}, \varphi_2)$, where φ_1 and φ_2 map the parallel edges to their corresponding edges in G. Additionally, insert degree-1 vertices incident to $v^{\textcircled{1}}$ and $v^{\textcircled{2}}$ representing the exclusive edges incident to $v^{\textcircled{1}}$ and $v^{\textcircled{2}}$, respectively; see Figure 2.3 for an example. Synchronized Planarity will then determine whether G_1 and G_2 can be embedded such that the shared edges around each vertex have opposite rotation. Mirroring the embedding of either G_1 or G_2 then yields the desired SEFE. The problem Synchronized Planarity can therefore be used to ensure consistent edge orderings between the two exclusive graphs.

However, this algorithm cannot handle consistent relative positions of connected components and therefore only works if the shared graph is connected. Figure 2.4 shows an instance where the reduction described above fails. While the two graphs shown in Figure 2.4a clearly do not admit a SEFE, the corresponding instance of Synchronized Planarity is a yes-instance and accepts the embedding shown in Figure 2.4b. The problem is that Synchronized Planarity only synchronizes the rotation of shared edges, but does not ensure consistent relative positions of connected components of the shared graph. In Figure 2.4b, the vertex x is located in different faces of the shared graph in the two embeddings, which is not valid for the original SEFE instance. The reduction to the problem Synchronized Planarity can therefore not be used to also ensure consistent relative positions between the two exclusive graphs.

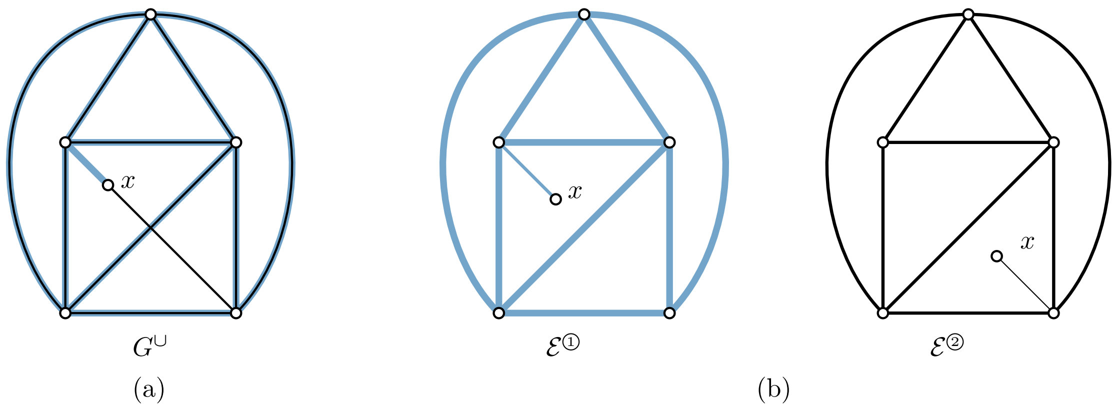


Figure 2.4: (a) A no-instance of SEFE with a disconnected shared graph (b) Possible embeddings $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$ that satisfy the Synchronized Planarity instance obtained from the reduction from SEFE. Note that the shared vertex x is embedded into different faces of the shared graph in the two embeddings, thus the reduction does not work if the shared graph is disconnected.

Parameterized Complexity. A *parameterized problem* is a language in $\Sigma^* \times \mathbb{R}$, where Σ is a finite alphabet and $k \in \mathbb{R}$ is the *parameter* of the problem. A parameterized problem L is *fixed-parameter-tractable* (FPT) if an algorithm can solve it in time $O(n^c \cdot f(k))$, where n is the input size, c is a constant, and f is a computable function.
One of the main methods for finding FPT-algorithms is called *kernelization*. In this approach, one defines *safe reduction rules*, that is, transformations of the initial input that result in a smaller instance that is equivalent with respect to the problem. A reduced instance obtained by exhaustively applying reduction rules is called a *kernel* of the problem. If applying the reduction rules takes polynomial time in the input size and the size of the kernel only depends on the parameter k, one can often brute-force a solution for the kernel and immediately obtain an FPT-algorithm.
Another technique we will use utilizes *bounded search trees* to obtain an FPT-algorithm. When faced with a decision with multiple possible choices (e.g., picking an order for a set of edges), one creates multiple subproblems, called *branches*. Each branch corresponds to one possible decision and in the corresponding subproblem, that decision is assumed to be necessary, thus the original problem has a solution if and only if one of the subproblems has a solution. Repeatedly applying this step essentially yields a search tree for the problem. If the size of this search tree is bounded by a function in k and every step takes polynomial time, this branching technique yields an FPT-algorithm [CFK+15].

3. Parameterization by the Vertex Cover Number of the Union Graph

For a graph $G = (V, E)$, a *vertex cover* is a subset of its vertices $V' \subseteq V$ such that every edge $e \in E$ is incident to a vertex in V'. The *vertex cover number* of G is the size of a minimum vertex cover of G. As our first parameterization, we consider SEFE parameterized by the vertex cover number k of the union graph $G^{\cup}$. We use a similar approach as Bhore et al. [BGMN20] in their parameterization of the problem Book Thickness. Let C be a minimum vertex cover of size k of $G^{\cup}$ and let N denote the partition of the vertices in $V \setminus C$ according to their neighborhood in C. Note that we also consider the type of the edges connecting a vertex to C, i.e., two vertices connected to the same vertex in C via different types of exclusive edges belong to different sets in the partition. Also observe that each vertex in $V \setminus C$ must have all of its neighbors contained in C. Since a vertex in $V \setminus C$ can be connected to a vertex in C either via a common edge, via one of two exclusive edges, or via no edge, we have $|N| \leq 4^k$. Therefore, bounding the size of each set in N by a function in k will yield a kernel for this problem.

To this end, we first consider a set $U \in N$, where each $v \in U$ has degree at most 1. Since vertices of degree 0 can be embedded at an arbitrary position and a vertex of degree 1 can simply be embedded directly next to its neighbor, we get the following trivial reduction rule.

Reduction Rule 1
If $G^{\cup}$ contains a vertex v of degree 0 or 1, reduce the instance to $(G^{\cup} - v, k)$.

Subsequently, every vertex of $G^{\cup}$ has degree at least 2. We now define a reduction rule that limits the number of vertices in each set $U \in N$ to three.

Reduction Rule 2
If there exists a set $U \in N$ with $|U| > 3$, pick an arbitrary vertex $v \in U$ and reduce the instance to $(G^{\cup} - v, k)$

To show the safeness of Reduction Rule 2 we use the following two lemmas.

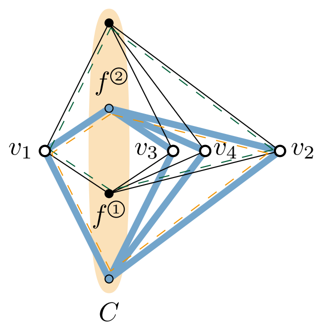


Figure 3.1: A SEFE of four degree-4 vertices with identical neighborhood in the vertex cover C. The colored dashed edges mark the faces $f^{①}$ and $f^{②}$ of $G^{①} - \{v_3, v_4\}$ and $G^{②} - \{v_3, v_4\}$, respectively, where v_3 is embedded in.

Lemma 3.1. *If there exists a set $U \in N$ with $|U| \geq 3$, then every vertex $v \in U$ has degree at most 2 in both exclusive graphs.*

Proof. Recall that all vertices in U have the same neighborhood in the vertex cover C. Therefore, if the vertices in U have degree at least 3 in one of the two exclusive graphs, we get a subgraph isomorphic to $K_{3,3}$ in this exclusive graph, because $|U| \geq 3$. □

Lemma 3.2. *Let $C \subset V(G^{\cup})$ be a subset of the vertices in $G^{\cup}$. Let further v_1, v_2, v_3, and v_4 denote four vertices of $V(G^{\cup}) \setminus C$ such that all four vertices have identical neighborhood and all these neighbors are contained in C. Then $G^{\cup}$ admits a SEFE if and only if $G^{\cup} - v_4$ admits a SEFE.*

Proof. A simultaneous embedding $\mathcal{E}^{\cup}$ of $G^{\cup}$ immediately yields a simultaneous embedding $\mathcal{E}^{\cup} - v_4$ of $G^{\cup} - v_4$.

Conversely, assume that $G^{\cup} - v_4$ admits a simultaneous embedding. By Reduction Rule 1 we only have to consider the case where $v_1, \ldots, v_4$ have degree at least 2. The statement clearly holds if $v_1, \ldots, v_4$ have degree 2, as they can simply be placed directly next to one another, since they have no other neighbors than the two contained in C. If the four vertices have degree at least 3, first consider the graph $G^{\cup} - v_4$. Since $G^{\cup} - v_4$ admits a SEFE, there must be a corresponding drawing Γ of $G^{\cup} - v_4$ where both exclusive graphs are planar. Let $f^{①}$ and $f^{②}$ denote the faces of $G^{①} - \{v_3, v_4\}$ and $G^{②} - \{v_3, v_4\}$, respectively, where the vertex v_3 is placed in Γ; see Figure 3.1. Since v_4 has the identical neighborhood as v_3 (and since all these neighbors are contained in C), placing v_4 in the faces $f^{①}$ and $f^{②}$ together with v_3 ensures that the edges incident to v_4 do not cross corresponding exclusive edges of $G^{\cup} - \{v_3, v_4\}$. By Lemma 3.1, v_3 and v_4 are both incident to at most two exclusive edges of each type, thus v_4 can be placed next to v_3 such that no exclusive edges of the same type incident to v_3 and v_4 cross and we get a SEFE of $G^{\cup}$. □

Also note that, because the four vertices have degree at most 4 in the union graph $G^{\cup}$ by Lemma 3.1, the statement of Lemma 3.2 can also be verified by enumerating all possible configurations. The safeness of Reduction Rule 2 follows directly from Lemma 3.2, because the vertex cover C satisfies the necessary requirements.

After exhaustively applying these two Reduction Rules, every $U \in N$ contains at most three vertices. Since $|N| = 4^k$, we get $|V \setminus C| \in \mathcal{O}(4^k)$. Because $|C| = k$, we therefore obtain a kernel of size $\mathcal{O}(4^k)$.

Theorem 3.3. *SEFE admits a kernel of size $\mathcal{O}(4^k)$, where k is the size of a minimum vertex cover of the union graph $G^\cup$.*

Combined with the fact that a minimum vertex cover of size k can be computed in time $O(1.2738^k + kn)$ [CKX10], Theorem 3.3 yields an FPT-algorithm for SEFE parameterized by the vertex cover number of the union graph $G^\cup$.

4. Parameterization by the Feedback Edge Set Number of the Union Graph

For a graph $G = (V, E)$, a *feedback edge set* is a subset $F' \subseteq E$ of its edges such that removing the edges in F from G makes G acyclic, i.e., the graph $G' = (V, E \setminus F)$ is a forest. The *feedback edge set number* ψ of G is the size of a minimum feedback edge set of G.

In this section, we consider SEFE parameterized by the feedback edge set number ψ of the union graph $G^{\cup}$. The reduction rules and arguments we use are very similar to those used by Binucci et al [BGL+22] in their parameterization of the problem StoryPlan. Furthermore, Bläsius and Rutter [BR15] showed that any SEFE instance can be reduced in linear time to an equivalent instance where the input graphs $G^{①}$ and $G^{②}$ are connected, hence we can assume the union graph $G^{\cup}$ to be connected.

As already mentioned in Chapter 3, we can safely remove any degree-1 vertex from the union graph, leading to the following trivial reduction rule.

Reduction Rule 1
If $G^{\cup}$ contains a vertex v of degree 1, reduce the instance to $(G^{\cup} - v, \psi)$.

Let a *k-chain* of $G^{\cup}$ denote a path consisting of $k + 2$ vertices, where each of its k inner vertices has degree 2. The following two reduction rules limit the size of each k-chain of $G^{\cup}$.

Reduction Rule 2
If $G^{\cup}$ contains a k-chain c with $k \geq 2$ and c contains a shared edge e, contract e and reduce the instance to $(G^{\cup\prime}, \psi)$.

Proof of Safeness. If $G^{\cup}$ admits a SEFE, let Γ denote a corresponding drawing. Since e is a shared edge, no other edge crosses e in Γ. Therefore, contracting e in Γ introduces no new crossings and we get a SEFE drawing Γ' of $G^{\cup\prime}$.

Conversely, let Γ' denote a SEFE drawing of $G^{\cup\prime}$. Let v denote the vertex in $G^{\cup\prime}$ obtained by contracting edge e and let u denote one of its two neighbors in the remaining $(k-1)$-chain c'. Starting with the drawing Γ', subdivide the edge uv, creating a new vertex v', and

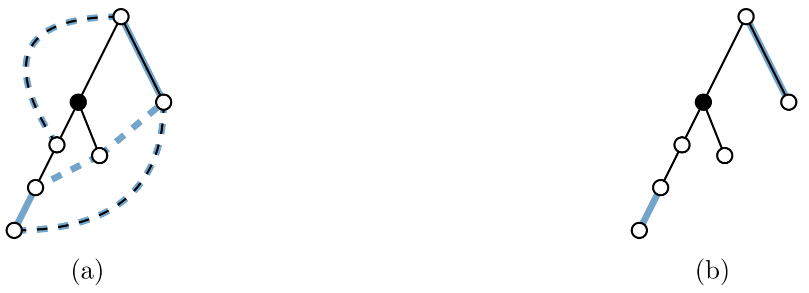


Figure 4.1: (a) An instance $G^{\cup}$ of SEFE where no reduction rules can be applied. The dashed edges belong to a minimum feedback edge set F of $G^{\cup}$. (b) The tree H obtained after removing F from H. Observe that, if two degree-2 vertices are adjacent in H, then one of them must be incident to an edge of F in $G^{\cup}$, otherwise Reduction Rule 2 or Reduction Rule 3 could still be applied in $G^{\cup}$.

then turn the edge $v'v$ into a shared edge. Observe that the resulting graph is isomorphic to $G^{\cup}$ and the edge $v'v$ corresponds to e. Because the edge $v'v$ can be drawn arbitrarily short, no edge crosses $v'v$ and therefore we have a SEFE drawing Γ of $G^{\cup}$.

If e is contained in a minimum feedback edge set F, replace e with any other edge of c. Observe that this yields a minimum feedback edge set of the graph $G^{\cup'}$, thus the parameter ψ does not change. □

Reduction Rule 3
If $G^{\cup}$ contains a k-chain c with $k \geq 1$, let e denote the first (or the last) edge of c. If both e and its successor (respectively its predecessor) in c are ①-exclusive edges, then contract e and reduce the instance to $(G^{\cup'}, \psi)$.

Proof of Safeness. Let f denote the predecessor (or the successor) of e in c such that e and f have the same type. If $G^{\cup}$ admits a simultaneous embedding, all edges crossing e can simply be rerouted through f after contracting e, thus $G^{\cup'}$ also admits a SEFE. Conversely, if $G^{\cup'}$ admits a SEFE, simply subdivide edge f to obtain a simultaneous embedding of $G^{\cup}$.

The parameter ψ does not change, for the same reason mentioned in the proof of Reduction Rule 2. □

After exhaustively applying Reduction Rule 1, every k-chain with $k \geq 1$ only contains exclusive edges. Recall that we can assume that both exclusive graphs are connected. Therefore, we have at most one path of exclusive edges in a k-chain for each of the two exclusive graphs and thus, after exhaustively applying Reduction Rule 2, we have that $k \leq 1$ for every k-chain in $G^{\cup}$. Using this observation, we get the following result.

Theorem 4.1. *SEFE admits a kernel of size $15\psi - 5$, where ψ is the feedback edge set number of the union graph $G^{\cup}$.*

Proof. Let $G^{\cup'}$ denote the SEFE instance obtained by exhaustively applying Reduction Rule 1, Reduction Rule 2, and Reduction Rule 3. Let F denote a minimum feedback edge set of $G^{\cup'}$. Because the reduction rules do not affect the parameter, we have $|F| = \psi$. Let H be the graph obtained by removing the edges in F from $G^{\cup'}$; see Figure 4.1 for an example. Because F is a minimum feedback edge set and $G^{\cup'}$ is connected, H is a tree.

For any leaf l of H, observe that l must be incident to an edge from the feedback edge set in $G^{\cup'}$, because otherwise we could still apply Reduction Rule 1 in $G^{\cup'}$. Because an edge of F can be incident to two such leaves in $G^{\cup'}$, it follows that H overall has at most 2ψ leaves. Since H is a tree, it contains at most $2\psi - 2$ vertices of degree 3 or more. It therefore remains to bound the number of degree-2 vertices. Consider a k-chain c of H with $k \geq 2$. Because for every k'-chain in $G^{\cup'}$ it is $k' \leq 1$, there must be an edge of F incident to one of the degree-2 vertices in c in $G^{\cup'}$. More specifically, there must be at least one edge of F incident to every other degree-2 vertex in c, hence at least $\lfloor \frac{k}{2} \rfloor$ degree-2 vertices in c must be incident to edges of F. It is therefore not hard to see that the number of degree-2 vertices of H is also linear in the number of its leaves. In the following, we develop a more precise upper bound for the size of the kernel.

Let L denote the set of leaves of H and let N_3 denote the vertices of degree at least 3 in H. As argued above, it is $|L| \leq 2\psi$ and $|N_3| \leq |L| - 2 \leq 2\psi - 2$. Let further N_2^1 denote the degree-2 vertices of H that are contained in 1-chains of H and let N_2^k denote the degree-2 vertices of H contained in k-chains with $k \geq 2$. Note that any vertex of N_2^1 can only be adjacent to vertices of L and N_3. For this reason, $|N_2^1|$ is bounded by the number of edges a tree with $|L| + |N_3|$ nodes can have, thus $|N_2^1| \leq |L| + |N_3| - 1 \leq 4\psi - 3$. As argued above, at least $\lfloor \frac{k}{2} \rfloor$ vertices of a k-chain with $k \geq 2$ must be incident to an edge of F in $G^{\cup'}$. Therefore, one endpoint of an edge in F can "pay" for up to three vertices in N_2^k and thus $|N_2^k| \leq 6\psi$. Since $|V(H)| = |L| + |N_3| + |N_2^1| + |N_2^k|$, we get the upper bound $|V(H)| \leq 14\psi - 5$ for the number of vertices in H. Because $|V(G^{\cup'})| = |V(H)| + |F|$, we finally obtain an upper bound of $|V(G^{\cup'})| \leq 15\psi - 5$ for the size of our kernel. □

Since a minimum feedback edge set of an undirected graph can be computed in linear time by computing an arbitrary spanning tree, Theorem 4.1 yields an FPT algorithm for SEFE parameterized by the feedback edge set number ψ of the union graph $G^{\cup}$.

4.1 Remarks About the Feedback Vertex Set Number

We now want to briefly argue how our FPT-algorithm for SEFE parameterized by the feedback edge set number of the union graph is affected when we use a feedback vertex set instead of a feedback edge set. Similarly to a feedback edge set, a *feedback vertex set* is a subset S of vertices of $G^{\cup}$ such that removing S from $G^{\cup}$ makes the graph acyclic. The *feedback vertex set number* ϕ of $G^{\cup}$ is the size of a minimum feedback vertex set of $G^{\cup}$. Note that, given a feedback edge set F, a feedback vertex set can be obtained from F by picking one endpoint of every edge in F. The feedback edge set number ψ is therefore an upper bound for the feedback vertex set number ϕ.

Unfortunately, our algorithm (in its current form) does not work for feedback vertex sets. While our Reduction Rules 1–3 can also be used to simplify the instance in this case, some of the arguments used in the proof of Theorem 4.1 no longer apply. Recall that we found our kernel by bounding the size of the tree H obtained after removing the minimum feedback edge set F from the reduced union graph. This was possible because a single edge from F only has at most two endpoints in the tree H. For a vertex from a feedback vertex set, however, the number of incident edges, and thus the number of neighbors in H, can be linear in the size of $G^{\cup}$. Therefore, more refined reduction rules are necessary to obtain a kernel for SEFE parameterized by the feedback vertex set number of the union graph. We leave this as an open problem for future work.

5. Parameterization by the Treedepth of the Union Graph

For a connected graph $G = (V, E)$, a *treedepth decomposition* is an arrangement of the vertices in V in a rooted tree $\mathcal{T} = (V, E')$ such that, for all edges in E, one endpoint is the descendant in $\mathcal{T}$ of the other endpoint. The *treedepth* of G is the minimum height of a treedepth decomposition of G.

In this section, we consider SEFE parameterized by the treedepth d of the union graph $G^{\cup}$. For this purpose, we build on ideas by Bannister et al. [BCE18] from their kernelization for 1-Planarity to bound the number of children for each node in $\mathcal{T}$. As in the previous chapter, we assume that both exclusive graphs (and therefore the union graph) are connected, which is safe due to the linear-time preprocessing step introduced by Bläsius and Rutter [BR15].

Let $G^{\cup} = (V, E)$ and let $\mathcal{T} = (V, E')$ denote a treedepth decomposition of height d for $G^{\cup}$. For a vertex $v \in V$, let A_v be the set of ancestors of v in $\mathcal{T}$, including v itself (i.e., A_v is the path from v to the root of $\mathcal{T}$), and let S_v denote the set of all subtrees rooted at a child of v in $\mathcal{T}$. For a subtree $T \in S_v$, let $T^{①}$ denote the subgraph of $G^{①}$ corresponding to T, and let v_T denote the vertex obtained by contracting T into a single vertex.
Similar to Chapter 3, we create a partition N_v by grouping the subtrees in S_v according to their neighborhood in A_v in $G^{\cup}$. In this case, we define that two subtrees $T_1, T_2 \in S_v$ have the same neighborhood in A_v if the corresponding contracted vertices v_{T_1} and v_{T_2} have the same neighborhood in A_v. Once again, we also consider the specific type of each edge connecting the contracted vertices v_{T_1} and v_{T_2} to A_v when creating the partition. Since a vertex v_T corresponding to a subtree T can be connected to a vertex $u \in A_v$ either via a shared edge, via one of two exclusive edges, or via no edge, we get $|N_v| \leq 4^d$ using $|A_v| \leq d$. Therefore, bounding the size of each set in N_v also bounds the number of children of each vertex v in $\mathcal{T}$ and thus also the total size of $\mathcal{T}$.
We also require that, for every subtree T in S_v, the two exclusive graphs $T^{①}$ and $T^{②}$ are connected. This restriction will be helpful, because it allows us to use the following variations of Lemma 3.1 and Lemma 3.2.

Lemma 5.1. *If there exists a set $U \in N_v$ with $|U| \geq 3$, then for every subtree $T \in U$, the two exclusive graphs $T^{①}$ and $T^{②}$ each have at most two neighbors in A_v. Consequently, T has at most four neighbors in A_v.*

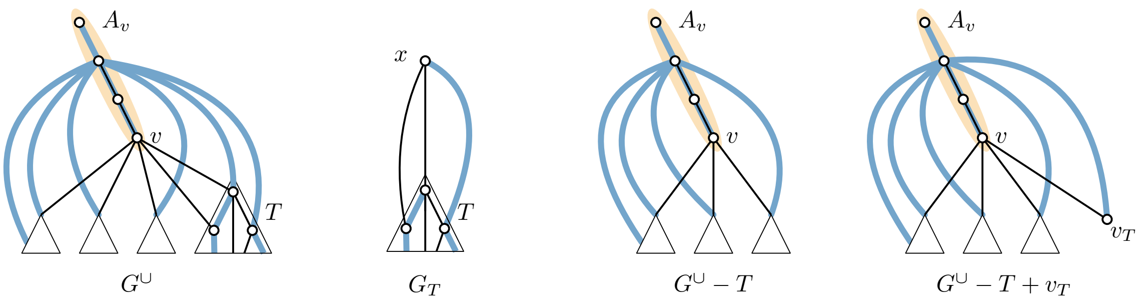


Figure 5.1: Reduction Rule 1: Subgraph T is removed from the instance after verifying that G_T admits a SEFE.

Proof. Let T_a, T_b, and T_c denote three distinct subtrees in U. Assume, for the sake of contradiction, that one of the two exclusive graphs of T_a, say $T_a^{①}$, has more than two neighbors in A_v. Then the same also holds for $T_b^{①}$ and $T_c^{①}$, because all subtrees in U have identical neighborhood in A_v. But because the graphs $T_a^{①}$, $T_b^{①}$, and $T_c^{①}$ are connected and have the same three neighbors in A_v, this yields a subgraph of $G^{①}$ that is homeomorphic to $K_{3,3}$. Since $G^{①}$ is planar, this is a contradiction. □

Lemma 5.2. *Let $U \in N_v$ with $|U| \geq 3$ be a set of subtrees with identical neighborhood in A_v. Let $G^\cup + u$ denote the graph obtained from $G^\cup$ by adding a single vertex u that has the same neighborhood in A_v as the subtrees in U. Then $G^\cup$ admits a SEFE if and only if $G^\cup + u$ admits a SEFE.*

Proof. If $G^\cup + u$ admits a simultaneous embedding, simply removing u yields a simultaneous embedding of $G^\cup$.

Conversely, assume that $G^\cup$ admits a simultaneous embedding and let Γ denote a corresponding drawing. Let T_1, T_2, and T_3 denote three distinct subtrees in U. Let $\Gamma[T_i]$ denote the restriction of Γ to the subgraph T_i. For $i \in \{1, 2, 3\}$, we contract the subgraph T_i into a single vertex v_{T_i} and call the resulting drawing Γ'. Note that this contraction does not introduce crossings in the exclusive graphs, because we require $T^{①}$ to be connected. By Lemma 5.1, each vertex v_{T_i} has degree at most two in both exclusive graphs. Therefore, we can apply the proof of Lemma 3.2 verbatim to obtain a SEFE drawing Γ'_u that additionally contains the vertex u. Note that this construction does not alter the order of incident edges at each vertex v_{T_i}, thus the order of edges incident to v_{T_i} is the same in Γ' and Γ'_u. We can therefore replace v_{T_i} with the drawing $\Gamma[T_i]$ of the original subgraph T_i (for $i \in \{1, 2, 3\}$) and we obtain a SEFE drawing of $G^\cup + u$. □

Let $U \in N_v$ with $|U| > 3$ be a set of subtrees with identical neighborhood in A_v. We now use Lemma 5.1 and Lemma 5.2 to derive a reduction rule that allows us the extract a tree of U and to test whether it admits a SEFE independently. For a subtree $T \in U$, let G_T denote the graph obtained from $G^\cup[V(T) \cup A_v]$ by contracting all vertices in A_v into a single vertex x; see Figure 5.1 for an example.

Reduction Rule 1
Let $U \in N_v$ with $|U| > 3$ be a set of subtrees with identical neighborhood in A_v. Pick an arbitrary subtree $T \in U$ and test recursively whether G_T admits a SEFE. If G_T does not admit a SEFE, reduce to a trivial no-instance. Otherwise, remove T from $G^\cup$ and reduce the instance to $(G^\cup - T, d)$.

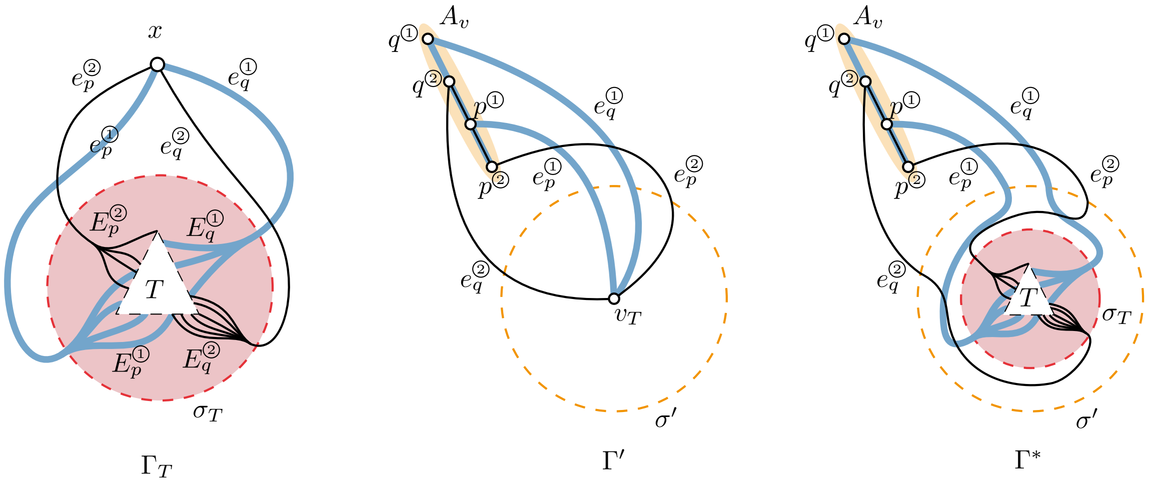


Figure 5.2: Proof of Reduction Rule 1: The drawing of T in Γ_T can be used to replace vertex v_T in Γ' to obtain a drawing Γ^* of $G^{\cup}$. This is possible, because the circular orders σ_T and σ' for the bundles induced by Γ_T and Γ', respectively, are always equivalent, thus the edges connecting T to A_v can be drawn crossing-free.

Proof of Safeness. We first need to show that, if $G^{\cup}$ admits a SEFE, then G_T and $G^{\cup} - T$ (see Figure 5.1) also admit a SEFE. If $G^{\cup}$ admits a simultaneous embedding, then simply removing T immediately yields a simultaneous embedding of $G^{\cup} - T$. Let Γ denote a SEFE drawing of $G^{\cup}$ and let $\Gamma[V(T) \cup A_v]$ denote its restriction to the subgraph induced by the vertices in T and A_v. It is not immediately clear that contracting A_v into a single vertex in the drawing $\Gamma[V(T) \cup A_v]$ yields a SEFE drawing of G_T. This is because contracting A_v into a single vertex could theoretically introduce crossings, if the vertices of A_v are distributed among several faces of T in Γ. We now show that this cannot happen. Assume that there are distinct vertices $q, r \in A_v$ in $G^{\cup}$ connected to $T^{①}$ such that q and r are placed in different faces of $T^{①}$ in Γ. Since we have $|U| \geq 3$, there is another subtree $S \in U$ with identical neighborhoods as T, i.e., $S^{①}$ is connected to q and r. But since we require $S^{①}$ to be connected and q, r lie in different faces of $T^{①}$ in Γ, this means that Γ contains a crossings in $G^{①}$, a contradiction. Therefore, all vertices of A_v that T is connected to lie in the same face of $T^{①}$ and $T^{②}$, respectively, and thus, contracting A_v into a single vertex in the drawing $\Gamma[V(T) \cup A_v]$ does not introduce crossings. We therefore obtain a SEFE drawing of G_T.

Conversely, we now need to show that simultaneous embeddings of G_T and $G^{\cup} - T$ can be combined to obtain a simultaneous embedding of $G^{\cup}$. Let Γ and Γ_T denote SEFE drawings of $G^{\cup} - T$ and G_T, respectively. Let $G^{\cup} - T + v_T$ be the graph obtained from $G^{\cup} - T$ by adding a new vertex v_T that has the same neighborhood in A_v as T and the other subtrees in U; see Figure 5.1. Since we have $|U| > 3$, $U \setminus \{T\}$ still contains at least three subtrees. Therefore, we can use Lemma 5.2 on the graph $G^{\cup} - T$ to obtain a SEFE drawing Γ' of $G^{\cup} - T + v_T$.
We now want to show that we can combine Γ' and Γ_T to a SEFE drawing of $G^{\cup}$. Essentially, we want to replace the vertex v_T in Γ' with the drawing of T obtained from Γ_T; see Figure 5.2 for an illustration. Recall that $T^{①}$ and $T^{②}$ each have at most two neighbors in A_v in $G^{\cup}$ by Lemma 5.1. Without loss of generality, we assume that both have exactly two such neighbors. Let $p^{①}$ and $q^{①}$ denote the two neighbors of $T^{①}$ in A_v. We call the set of edges connecting $T^{①}$ to $p^{①}$ (respectively $q^{①}$) a *bundle* and denote it by $E_p^{①}$ (respectively $E_q^{①}$). Note that the edges of each of the two bundles $E_p^{①}$ and $E_q^{①}$ must appear consecutively around the outer face of $T^{①}$ in Γ_T, otherwise we find a crossing between edges of $E_p^{①}$ and $E_q^{①}$. Therefore, we can draw the edges of each bundle arbitrarily close together and we

subsequently treat the bundle $E_p^{①}$ (respectively $E_q^{①}$) as a single edge $e_p^{①}$ (respectively $e_q^{①}$); see Figure 5.2. Note that the edges incident to the vertex v_T in $G^{\cup} - T + v_T$ correspond bijectively to the bundles incident to T in Γ_T, since v_T and T have the same neighborhood in A_v. We therefore also denote the edges incident to v_T in Γ' by $e_p^{①}$ and $e_q^{①}$ according to this bijection; see Figure 5.2. The drawing Γ_T defines a cyclic order $\sigma_T^{①}$ for the edges $e_p^{①}$ and $e_q^{①}$ around T and the drawing Γ' defines a cyclic order $\sigma'^{①}$ for the edges $e_p^{①}$ and $e_q^{①}$ around v_T. Since we want to replace v_T with the drawing of T obtained from Γ_T, we thus need to show that $\sigma_T^{①}$ and $\sigma'^{①}$ are equivalent. If this is the case, the replacement can be performed without introducing any crossings; see Figure 5.2 for an example. But since $\sigma_T^{①}$ and $\sigma'^{①}$ both consist of only two elements, they are always equivalent and thus the edges connecting T to A_v can be drawn crossing-free. We therefore obtain a SEFE drawing Γ^* of $G^{\cup}$.

We note that this proof relies heavily on Lemma 5.1 restricting the number of bundles per exclusive graph to at most two. Without this restriction, σ_T and σ' could be two distinct cyclic orders, thus the replacement could not always be performed without introducing crossings.

It remains to show that Reduction Rule 1 can be applied in FPT-time. We do this using a simple inductive argument. As the inductive hypothesis, we state that we can solve SEFE in FPT time if $G^{\cup}$ has at most n vertices. As the base case, observe that SEFE can be solved in constant time for graphs of constant size by brute-forcing all pairs of embeddings. Now assume that $G^{\cup}$ contains $n + 1$ vertices. Then $G^{\cup} - T$ and G_T both contain at most n vertices and we can thus test whether each of them admits a simultaneous embedding in FPT time by the inductive hypothesis. As shown above, we can combine simultaneous embeddings of $G^{\cup} - T$ and G_T to obtain a simultaneous embedding of $G^{\cup}$, which concludes the inductive step. □

After exhaustively applying Reduction Rule 1 for every $v \in V$, every set $U \in N_v$ contains at most three subtrees. Since $|N_v| \leq 4^d$, it follows that every vertex has $\mathcal{O}(4^d)$ children in $\mathcal{T}$. Because $\mathcal{T}$ has height d, we therefore get the following result.

Theorem 5.3. *SEFE admits a kernel of size $\mathcal{O}((4^d)^d) = \mathcal{O}(4^{d^2})$, where d is the treedepth of the union graph $G^{\cup}$, if every subtree of the corresponding treedepth decomposition is connected in both exclusive graphs.*

As argued before, we remark that Lemma 5.1 and Lemma 5.2 rely on the requirement that every subtree of the treedepth decomposition is connected in both exclusive graphs. Without this restriction, the proof of Reduction Rule 1 is not correct.

6. Parameterization by the Vertex Cover Number of the Shared Graph

After developing FPT algorithms with various parameters of the union graph $G^\cup$ in the previous chapters, we now consider parameters of the shared graph G. In this case, finding safe reduction rules becomes significantly more involved. For example, isolated vertices or vertices of degree 1 of the union graph can be safely removed, because they can be positioned in any drawing without introducing crossings. In the context of the shared graph, even isolated vertices may hold important information due to their connectivity in the exclusive graphs. The example shown in Figure 2.4 is a no-instance of SEFE, but if the isolated vertex x is removed from the graph, the graph becomes a trivial yes-instance. Similarly, it is also not trivial to reduce degree-1 vertices of the shared graph in the general case. For example, consider a degree-1 vertex v of a shared component C such that v is adjacent to a vertex u of C. It is clear that all other neighbors of v in $G^\cup$ must be embedded in the same face of C. However, if we contract v into u, then this is not necessarily the case anymore, thus degree-1 vertices cannot be contracted in the general case; see Figure 6.1 for an example.

First consider parameterizations for l-Sunflower SEFE with $l \geq 3$. Angelini et al. [ALN15] showed that l-Sunflower SEFE is $\mathcal{NP}$-complete for $l \geq 3$, even if the shared graph is a tree and all input graphs are biconnected. Since a tree has treewidth 1, we immediately get the following result.

Corollary 6.1. *The problem l-Sunflower SEFE with $l \geq 3$ is FPT with respect to the treewidth of the shared graph if and only if $\mathcal{P} = \mathcal{NP}$.*

Since the treewidth of a graph is at most as big as its pathwidth, the same statement also holds for parameterizations using the pathwidth of the shared graph. Angelini et al. gave an even stronger result for l-Sunflower SEFE with $l \geq 3$ in their $\mathcal{NP}$-completeness proof for the problem Partitioned l-Page Book Embedding (PBE-l)[ALN15, Theorem 4]. In this proof they give a polynomial-time reduction from the $\mathcal{NP}$-complete problem Betweenness to l-Sunflower SEFE, where the shared graph is a star. Since a star has a vertex cover of size 1, we get the following result regarding parameterizations using the vertex cover number of the shared graph.

Corollary 6.2. *The problem l-Sunflower SEFE with $l \geq 3$ is FPT with respect to the vertex cover number of the shared graph if and only if $\mathcal{P} = \mathcal{NP}$.*

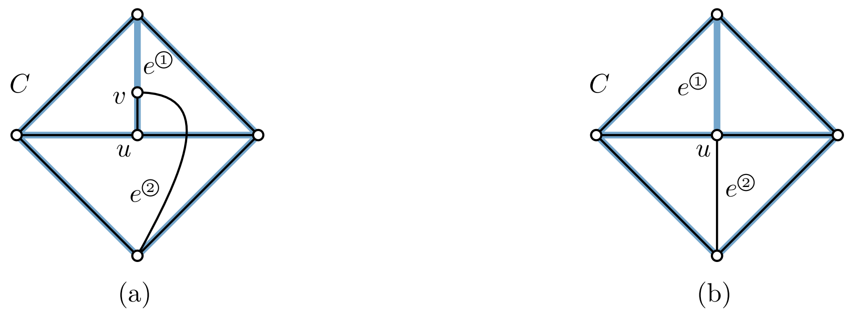


Figure 6.1: (a) A graph $G^{\cup}$ that is a no-instance of SEFE, because the two exclusive edges $e^{①}$ and $e^{②}$ cannot be embedded into the same face of C. (b) After contracting the vertex v into u, $e^{①}$ and $e^{②}$ must no longer be embedded into the same face of C and the graph becomes a yes-instance.

This result implies that any parameterization of 2-SEFE by the vertex cover number of the shared graph must explicitly exploit the existence of only two exclusive graphs. Since it is not clear how such an FPT-algorithm for SEFE would look like, we employ an additional parameter that bounds the number of degree-1 vertices in the shared graph.

For the shared graph G, let d denote the number of degree-1 vertices in $V(G)$ and let C denote a minimum vertex cover of G with size k. Our goal is to enumerate all suitable embeddings of each connected component of the shared graph using the parameters d and k. Subsequently, we will use the algorithm by Bläsius and Rutter [BR15] that tests in quadratic time whether $G^{\cup}$ admits a SEFE if every connected component of G has a fixed embedding. To this end, we start by bounding the number of vertices of degree at least 3 in G using the following lemma.

Lemma 6.3. *Let N_3 denote the number of vertices of G with degree at least 3. It holds that $|N_3| \leq 3k$, where k is the vertex cover number of G.*

Proof. Let C denote a minimum vertex cover of G with $|C| = k$. The number of vertices of degree at least 3 in C is therefore also at most k, i.e., $|N_3 \cap C| \leq k$. Observe that all neighbors of a vertex in $V(G) \setminus C$ must be contained in C, because otherwise, we immediately get an edge that is not covered by C. Thus every vertex in $N_3 \cap (V(G) \setminus C)$ has at least three neighbors in C. Therefore, we can use planarity properties derived from Euler's Formula to infer that $|N_3 \cap (V(G) \setminus C)| \leq 2|C| - 4 = 2k - 4$ [FLSZ19, Lemma 13.3]. Together we get that

$$|N_3| = |N_3 \cap C| + |N_3 \cap (V(G) \setminus C)| \leq k + (2k - 4) = 3k - 4 \leq 3k.$$

□

Since the number of degree-1 vertices of G is d and the number of vertices with degree at least 3 is at most $3k - 4$ by Lemma 6.3, only the number of isolated vertices and degree-2 vertices remains unbounded. Instead of trying to limit the number of these vertices to obtain a kernel for SEFE, we show that degree-2 vertices only allow very limited embedding choices for their two incident vertices. Since isolated vertices have a fixed rotation choice and the number of all other vertices is bounded, this allows us to brute-force all suitable embeddings of each shared component in FPT time and to subsequently determine whether $G^{\cup}$ admits a SEFE with the fixed embeddings using the algorithm by Bläsius and Rutter [BR15]. In other words, we use the bounded search tree technique where the branches of the search tree correspond to embedding decisions in the shared graph. Our algorithm will

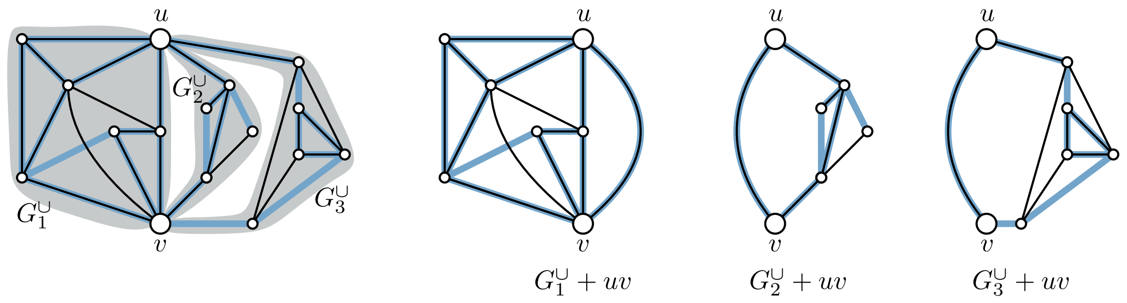


Figure 6.2: An example illustrating Reduction Rule 1. The three union split components $G_1^\cup$, $G_2^\cup$, and $G_3^\cup$ with respect to the split pair $\{u, v\}$ can be decomposed into the independent SEFE instances $G_1^\cup + uv$, $G_2^\cup + uv$, and $G_3^\cup + uv$, because $G_1^\cup$ and $G_2^\cup$ each contain a shared path between u and v [BKR18].

first branch for all possible embeddings of the blocks in G, and subsequently branch for all possible configurations of blocks around cutvertices. This means that our search tree has constant constant depth and we therefore only have to ensure that the number of branches in each step is bounded by a function in $k + d$.

6.1 Embedding the Blocks

We start by enumerating all suitable embeddings of the individual blocks in the shared graph G. Let $U \subseteq (V(G) \setminus C)$ denote a set of degree-2 vertices not contained in C with $|U| \geq 3$, where all vertices in U have the same two neighbors u and v. Observe that u and v must both be contained in C and that $\{u, v\}$ is a separating pair of the shared graph with at least $|U| \geq 3$ split components. Let μ denote the corresponding P-node with at least $|U|$ virtual edges. We call such a P-node of G corresponding to U *two-parallel*. Since the virtual edges of μ can be ordered arbitrarily with respect to G and since we currently have no bound for $|U|$, we cannot afford to enumerate all possible embeddings of μ. Instead, we use a preprocessing step given by Bläsius et al. [BKR18], which helps us reduce the number of possible embedding choices in the P-node μ. This preprocessing step additionally uses the connectivity information from the union graph $G^\cup$ to limit the number of possible embeddings of P-nodes in G. We state this preprocessing step in the following reduction rule; see Figure 6.2 for an example.

Reduction Rule 1 [BKR18, Lemma 4]
Let $\{u, v\}$ be a separating pair of the union graph $G^\cup$ with split components $G_1^\cup, \ldots, G_l^\cup$, such that at least two split components contain a shared path between u and v. Reduce the instance to l independent instances $G_1^\cup + uv, \ldots, G_l^\cup + uv$ of SEFE.

Let $\{u, v\}$ be a separating pair of the shared graph G and let μ be the corresponding P-node. Observe that exhaustively applying Reduction Rule 1 implies that the union-link graph $L_\mu^\cup$ (see Chapter 2 and Figure 6.4a) of μ is connected, because otherwise, $\{u, v\}$ is also a separating pair of the union graph, and we could still apply Reduction Rule 1. Note that two virtual edges ε_1 and ε_2 that are adjacent in $L_\mu^\cup$ must also be adjacent in any simultaneous embedding, because otherwise, the union-link between ε_1 and ε_2 crosses a shared path in the virtual edge of μ embedded between ε_1 and ε_2 [BKR18, Lemma 2]. A node of degree 3 or higher in $L_\mu^\cup$ thus implies that $G^\cup$ is a no-instance. Because $L_\mu^\cup$ must additionally be connected by Reduction Rule 1, we therefore get the following reduction rule.

Reduction Rule 2 [BKR18, Lemma 3 + Lemma 4]
Let μ be a P-node of the shared graph G. If the union-link graph $L_\mu^\cup$ of μ is not a cycle or a path, reduce to a trivial no-instance.

Consider a P-node μ of G. After exhaustively applying Reduction Rules 1 and 2, the union link graph $L_\mu^\cup$ of μ must be either a path or a cycle. Since two virtual edges of μ that are adjacent in $L_\mu^\cup$ must be adjacent in any simultaneous embedding of $G^\cup$ [BKR18, Lemma 2], we get the following corollary.

Corollary 6.4. *In an irreducible SEFE instance $G^\cup$, the cyclic order of virtual edges in any P-node of any block in G is fixed up to reversal.*

Corollary 6.4 is very helpful, because it essentially states that, as for R-nodes, there are only two possible embedding choices for every P-node of G. Thus it only remains to bound the number of P-nodes and R-nodes in G, which we do with the following lemma.

Lemma 6.5. *The number of all P-nodes and R-nodes in G is in $\mathcal{O}(k^2)$.*

Proof. First consider a block B of G with its corresponding SPQR-tree $\mathcal{T}$. We let $\mathrm{PR}(\mathcal{T})$ denote the number of P-nodes and R-nodes in $\mathcal{T}$. Since B does not contain any degree-1 vertices and since we have bounded the number of vertices of degree at least 3 in G linearly in k (Lemma 6.3), only the degree-2 vertices of B that are not contained in the vertex cover C remain unbounded.
Let B' denote the biconnected graph obtained by contracting every degree-2 vertex in B into one of its neighbors and let $\mathcal{T}'$ denote the SQPR-tree of B'. We keep the parallel edges that may emerge after such contractions, thus B' is a multi-graph. Because we keep the parallel edges, note that $\mathcal{T}$ and $\mathcal{T}'$ contain the same number of P-nodes and R-nodes, i.e., $\mathrm{PR}(\mathcal{T}) = \mathrm{PR}(\mathcal{T}')$. Since a biconnected graph does not contain degree-1 vertices and since we have just eliminated all degree-2 vertices, B' contains $\mathcal{O}(k)$ vertices by Lemma 6.3. The number of bundles of parallel edges in B' is therefore bounded by the number of edges a planar graph with $\mathcal{O}(k)$ vertices can have, which is also $\mathcal{O}(k)$.
Let B^* denote the biconnected graph obtained from B' by replacing bundles of parallel edges with a single edge and let $\mathcal{T}^*$ denote the SPQR-tree of B^*. Since B' contains $\mathcal{O}(k)$ bundles of parallel edges, removing parallel edges can only eliminate $\mathcal{O}(k)$ P-nodes. Observe that removing parallel edges does not affect R-nodes, thus $\mathrm{PR}(\mathcal{T}') - \mathrm{PR}(\mathcal{T}^*) \in \mathcal{O}(k)$, and therefore $\mathrm{PR}(\mathcal{T}) - \mathrm{PR}(\mathcal{T}^*) \in \mathcal{O}(k)$. Since B^* contains $\mathcal{O}(k)$ vertices and no parallel edges, planarity ensures that it also contains $\mathcal{O}(k)$ edges. Note that the size of an SPQR-tree is linear in the number of its Q-nodes and thus linear in the number of edges of the corresponding block. Therefore, $\mathcal{T}^*$ contains $\mathcal{O}(k)$ nodes and thus $\mathrm{PR}(\mathcal{T}^*) \in \mathcal{O}(k)$. As argued above, the difference $\mathrm{PR}(\mathcal{T}) - \mathrm{PR}(\mathcal{T}^*)$ is also in $\mathcal{O}(k)$, thus $\mathrm{PR}(\mathcal{T}) \in \mathcal{O}(k)$ and we have shown that each block contains at most $\mathcal{O}(k)$ P-nodes and R-nodes.

Observe that a block of G that contains a P-node or R-node must contain at least two vertices of the vertex cover C. Since two blocks share at most one vertex, the number of blocks containing P-nodes or R-nodes is therefore in $\mathcal{O}(k)$. Since the number of all P-nodes and Q-nodes in a single block is also in $\mathcal{O}(k)$, the statement of the lemma follows. □

By Corollary 6.4, every P-node in G has a fixed embedding up to mirroring. In combination with Lemma 6.5, we can therefore enumerate all embeddings of all blocks in G in FPT-time, as there are at most $2^{\mathcal{O}(k^2)}$ of such embeddings.

Corollary 6.6. $2^{\mathcal{O}(k^2)}$ *branches are sufficient to fix the embedding of every block in G.*

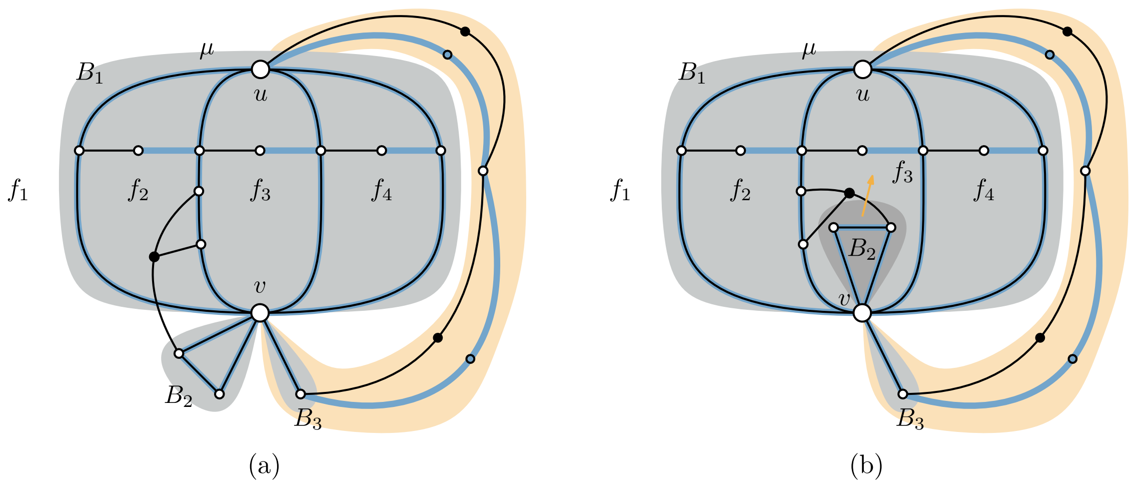


Figure 6.3: (a) Three blocks B_1, B_2, and B_3 sharing the same cutvertex v. By Reduction Rule 3, all blocks containing v must be connected in $G^\cup$ via a path that is vertex-disjoint from v. B_2 can only be embedded in the faces f_2 and f_3 of the μ, thus it is a binary block with respect to B_1. The block B_3 is connected to the other pole u of the P-node μ and can be embedded in all faces of μ, thus it is a mutable block with respect to B_1. Note that B_3 is contained in a separate split component (highlighted in orange) with respect to the separating pair $\{u, v\}$ in the union graph, because B_3 is a mutable block. (b) After assigning block B_2 to the face f_3 of μ, f_3 becomes occupied.

6.2 Nesting Blocks around Cutvertices

It remains to fix the order of incident edges at all cutvertices of G. Since this order is fixed for cutvertices of degree 2 and the total number of vertices of degree at least 3 in G is at most $3k$ by Lemma 6.3, the number of relevant cutvertices is also at most $3k$. Because we have already fixed the embedding of all blocks in G by Corollary 6.6, fixing the order of incident edges at a cutvertex v boils down to enumerating all suitable nestings of the blocks incident to v. We use another preprocessing step given by Bläsius et al. [BKR18], which allows us to assume that v is not a cutvertex in the union graph $G^\cup$. Thus all pairs of blocks incident to a cutvertex v are connected in $G^\cup$ via a path that is vertex-disjoint from v. We state this preprocessing step in the following reduction rule.

Reduction Rule 3 [BKR18, Lemma 1]
Let v be a cutvertex of the union graph $G^\cup$ with split components $G_1^\cup, \dots, G_l^\cup$. Reduce the instance to l independent instances $G_1^\cup, \dots, G_l^\cup$.

From now on, when we refer to faces of a block B of G, we implicitly refer to the faces of the fixed embedding $\mathcal{E}$ of B. For a cutvertex v of G, let B_1 and B_2 denote two blocks of G containing v. Since we currently have no bound for the number of faces of B_1 that contain v, we cannot simply successively try to embed B_2 into every face of B_1. However, by Reduction Rule 3, B_2 must be connected to B_1 in $G^\cup$ via a path that is vertex-disjoint from v. If there exist two such paths with endpoints u and w in B_1, then B_2 must be embedded in a face of B_1 that contains the vertices v, u and w, thus the decision is at most binary. Now consider the case where B_2 is only connected to the vertices v and u of B_1 in $G^\cup$. If v and u are contained in at most two faces of B_1, the embedding decision is again at most binary. In these two cases we say that B_2 is a *binary block* with respect to B_1; see Figure 6.3a for an example. If v and u are contained in more than two faces of B_1, then v

and u must be the poles of a P-node μ of block B_1 in G, and we cannot necessarily bound the number of virtual edges in μ by a function in k and d, since μ may be two-parallel. In this case, we say that B_2 is a *mutable block* with respect to B_1. Note that $\{u, v\}$ is a separating pair in $G^\cup$ if B_2 is a mutable block, because B_2 is only connected to u and v in $G^\cup$. This also means that B_2 and B_1 (and thus μ) are contained in different split components with respect to $\{u, v\}$ in $G^\cup$; see Figure 6.3a for an example.

If B_2 is a mutable block with respect to B_1, let $S^\cup$ denote the split component of the separating pair $\{u, v\}$ in $G^\cup$ that B_2 is contained in and let S, $S^{①}$, and $S^{②}$ denote the corresponding subgraphs of the shared graph and the exclusive graphs. Note that $S^\cup$ cannot contain a shared path between u and v, because otherwise, we could still apply Reduction Rule 1. We say that $S^\cup$ is *exclusive connected*, if the poles u and v are connected by paths in both $S^{①}$ and $S^{②}$. It is *①-connected*, if u and v are connected in $S^{①}$ but not in $S^{②}$, and analogously *②-connected*, if u and v are connected in $S^{②}$ but not in $S^{①}$. If neither of these three categories applies, then $S^\cup$ is *union connected*; see Figure 6.4b for an example.
We now compare the connectivity of $S^\cup$ with the connectivity of the faces of μ in $G^\cup$. We say that $S^\cup$ is *compatible* with a face f between two virtual edges ε_1 and ε_2 of μ, if both of the following conditions apply (see also Figure 6.4b for an example):

- Either $S^\cup$ is not ①-connected or ε_1 and ε_2 are not connected in the ①-link graph $L_\mu^{①}$, and
- Either $S^\cup$ is not ②-connected or ε_1 and ε_2 are not connected in the ②-link graph $L_\mu^{②}$.

Note that a split component $S^\cup$ that is not compatible with a face f of μ can never be embedded in f [BKR18, Proof of Lemma 5]. This is because, due to the incompatibility, there is an exclusive path in $S^\cup$ that would cross the exclusive link between the two corresponding adjacent virtual edges of μ. If $S^\cup$ is not compatible with any face of μ, the corresponding SEFE instance is therefore a no-instance. We say that the P-node μ is *impossible*, if there exists a split component $S^\cup$ that is not compatible with any face of μ. We therefore obtain the following reduction rule.

Reduction Rule 4 [BKR18, Lemma 5]
If $G^\cup$ contains an impossible P-node, reduce to a trivial no-instance.

Therefore, we can now assume that $G^\cup$ contains no impossible P-nodes.

6.2.1 Embedding Split Components Into Compatible Faces

As the next step, we show how the notion of compatibility allows us to limit the number of faces of a P-node each split component of the shared graph (and therefore each mutable block) can be embedded in. Let u and v denote the poles of a P-node μ of G such that $\{u, v\}$ is a separating pair of the union graph $G^\cup$. Let $G_\mu^\cup$ denote the split component that the P-node μ is contained in[1] and let $S_1^\cup, \dots, S_l^\cup$ denote the other split components. Bläsius et al. showed that, if none of the exclusive connected split components of $S_1^\cup, \dots, S_l^\cup$ contains a shared edge incident to one of the poles u or v of μ, then $S_1^\cup, \dots, S_l^\cup$ can be tested for a SEFE independently and can be removed from the graph [BKR18, Lemma 6]. They show that, with this restriction, the simultaneous embeddings of $S_1^\cup, \dots, S_l^\cup$ can always be combined with a simultaneous embedding of $G_\mu^\cup$, because each split component $S_i^\cup$ can be placed in an arbitrary face of μ it is compatible with without introducing any crossings. Because each split component is compatible with some face of μ by Reduction Rule 4,

[1]Note that μ must be contained in a single split component by Reduction Rule 1.

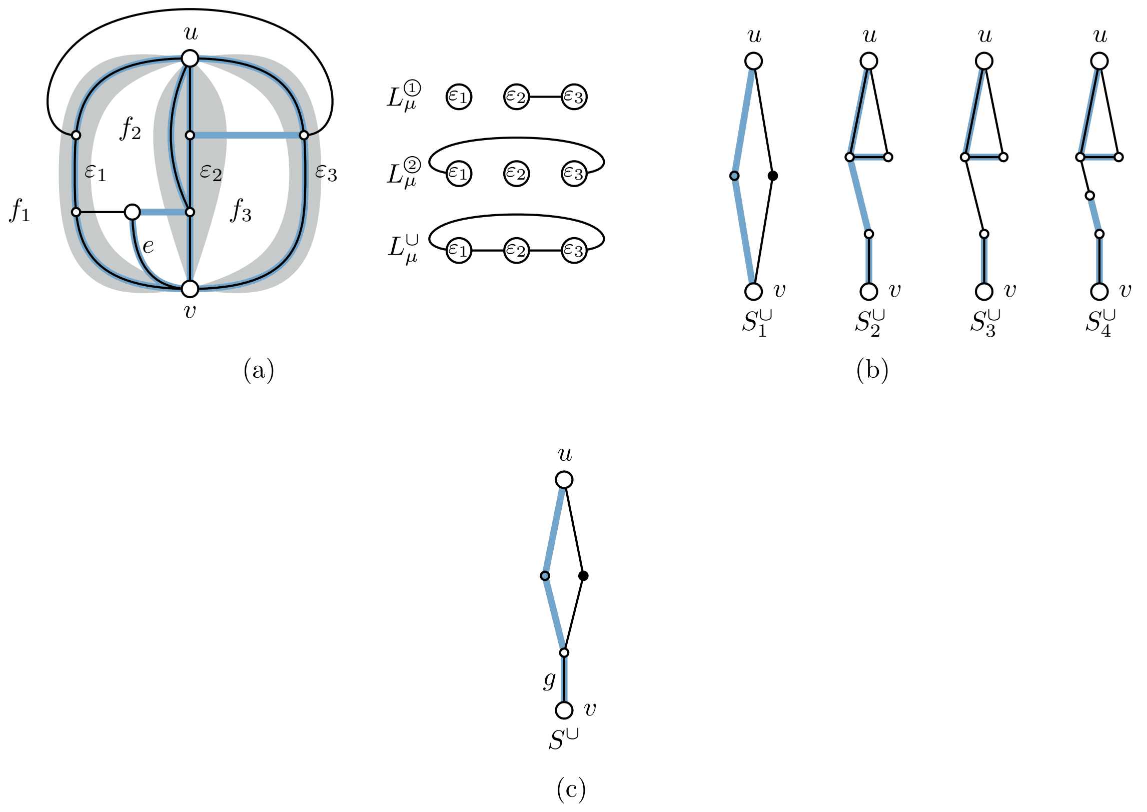


Figure 6.4: (a) A P-node μ of the shared graph with three virtual edges ε_1, ε_2, and ε_3 along with its corresponding link graphs $L^{\textcircled{1}}_\mu$, $L^{\textcircled{1}}_\mu$, and $L^{\cup}_\mu$. (b) Four split components with respect to the separating pair $\{u, v\}$ in $G^{\cup}$. The four components $S^{\cup}_1$, $S^{\cup}_2$, $S^{\cup}_3$, and $S^{\cup}_4$ are exclusive connected, $\textcircled{1}$-connected, $\textcircled{2}$-connected, and union connected, respectively. $S^{\cup}_1$ is compatible with f_2, $S^{\cup}_2$ is compatible with f_1 and f_2, $S^{\cup}_3$ is compatible with f_2 and f_3, and $S^{\cup}_4$ is compatible with all faces of μ. (c) Although the exclusive connected split component $S^{\cup}$ is compatible with the face f_2 (ε_1 and ε_2 are not linked in $L^{\textcircled{1}}_\mu$ or $L^{\textcircled{2}}_\mu$), $S^{\cup}$ cannot be embedded into f_2, because the two shared edges e and g are in conflict.

the split components can therefore be decomposed into independent instances [BKR18, Lemma 6]. Unfortunately, this does not work in general, if one of the split components in $S_1^\cup, \dots, S_l^\cup$ is exclusive connected and contains a shared edge incident to one of the poles u or v. It is possible that such a split component $S^\cup$ cannot be embedded into a face f of μ, even if $S^\cup$ is compatible with f; see Figure 6.4c for an example. Since an exclusive connected split component may contain mutable blocks of the shared graph incident to u or v that we need to embed into a face of μ (e.g., block B_3 in Figure 6.3a), we cannot exclude this case. Instead, we will adapt Lemma 6 from the paper by Bläsius et al. [BKR18] to fit our specific application.

We start by giving a high-level overview of our strategy. Let B_1 denote the block of the shared graph that the P-node μ is contained in. First, we assign a face of B_1 to each block that is a binary block with respect to B_1. If a face f of μ subsequently contains such a block incident to u or v, we say that f is *occupied*; see Figure 6.3b. For each remaining mutable block B_2 with respect to B_1 that can be embedded into any face of μ, recall that there exists a different split component $S^\cup$ of the separating pair $\{u, v\}$ in $G^\cup$ that B_2 is contained in (see Figure 6.3a for an example). The link graphs of μ will give us a set $\mathcal{F}_{S^\cup}$ of compatible faces of μ where $S^\cup$ (and therefore B_2) can potentially be embedded in. We will show that we can adapt Lemma 6 in the paper by Bläsius et al. [BKR18], such that we can simply remove $S^\cup$ from the instance, if $\mathcal{F}_{S^\cup}$ contains a face that is not occupied. Finally, all remaining embedding choices with respect to the P-node μ only concern faces of μ that are occupied. Since we can bound the number of occupied faces and the number of blocks in G, we will finally show that we can brute-force all remaining embedding choices of the shared graph.

As the first step, we now fix the position of all binary blocks. For every cutvertex v in G and for each pair B_1, B_2 of blocks containing v, we create two new branches if B_2 is a binary block with respect to B_1. The two branches correspond to the (at most) two faces of B_1 that B_2 can be embedded into. Since we have at most $3k$ cutvertices with degree at least 3 in G and since we have at most $k + d$ blocks in G that contain v[2], we create at most $\mathcal{O}((2^{(k+d)^2})^{3k}) = \mathcal{O}(2^{(k+d)^2 \cdot 3k})$ branches to assign the binary blocks to all possible faces.

Now consider two blocks B_1 and B_2 both containing cutvertex v, such that B_2 is a mutable block with respect to B_1. Recall that there exists a P-node μ with poles v and u in B_1, such that B_2 is contained in a different split component $S^\cup$ of the separating pair $\{u, v\}$ in the union graph $G^\cup$ (see Figure 6.3a for an example). For a face f of μ, recall that we say that f is occupied, if there exists a binary block incident to u or v that is assigned to the face f; see Figure 6.3b for an example. Note that we need to be cautious around an occupied face f, since f may contain additional shared edges incident to u and v, thus the problematic case shown in Figure 6.4c can occur and we have no guarantee that we can embed $S^\cup$ into f, even if $S^\cup$ is compatible with f. If, however, $S^\cup$ itself admits a SEFE and is compatible with a face f of μ that is not occupied, we now show that we can always embed $S^\cup$ (and all other split components of μ that are compatible with f) into f. Therefore, the following reduction rule allows us to decompose $S^\cup$ into an independent SEFE instance.

Reduction Rule 5 [Derived from [BKR18, Lemma 6]]
Let f denote an unoccupied face of a non-impossible P-node μ of the shared graph G whose poles u and v are a separating pair of the union graph $G^\cup$. Let $G_\mu^\cup$ denote the split component with respect to the separating pair $\{u, v\}$ that μ is contained in[3] and

[2]The number of blocks containing v is exactly the number of split components of v. Observe that each split component of v must contain at least one additional degree-1 vertex or one additional vertex of the vertex cover C. Therefore, there are at most $k + d$ blocks containing v.

let $S_1^\cup, \dots, S_t^\cup$ be the split components that are compatible with f. Reduce the instance to the independent instances $G_\mu^\cup$ and $S_i^\cup + uv$ for $i = 1, \dots, t$, where $S_i^\cup + uv$ is the split component $S_i^\cup$ together with the shared edge uv.

Proof of Safeness. The proof of safeness for this reduction rule is almost identical to the proof of Lemma 6 in the paper by Bläsius et al [BKR18]. We only have to additionally show that the fact that f is unoccupied allows us to also consider exclusive connected split components that contain a shared edge incident to u or v.

If $G^\cup$ admits a SEFE, then the corresponding simultaneous embedding can be easily decomposed into simultaneous embeddings of the split components $G_\mu^\cup$ and $S_i^\cup + uv$ for $i = 1, \dots, t$.

Conversely, assume that $G_\mu^\cup$ and $S_i^\cup + uv$ (for $i = 1, \dots, t$) admit simultaneous embeddings. We first glue the simultaneous embeddings of $S_1^\cup + uv, \dots, S_t^\cup + uv$ together in an arbitrary order to obtain a simultaneous embedding of $(S_1^\cup \cup \dots \cup S_t^\cup) + uv$; see Figure 6.5 for an example. This is possible, because these graphs only share the vertices u and v and since each $S_i^\cup + uv$ contains the shared edge uv, we may assume that u and v are on the outer face of each $S_i^\cup$. For this reason, we can add $S_i^\cup$ to the outer face of $(S_1^\cup \cup \dots \cup S_{i-1}^\cup) + uv$ to retain a simultaneous embedding. We denote the resulting graph by $H^\cup := S_1^\cup \cup \dots \cup S_t^\cup$ and the version of $H^\cup$ that additionally contains the shared edge uv by $H^\cup + uv$.

First assume that there exists an $i \in \{1, \dots, t\}$ such that $S_i^\cup$ is exclusive connected, thus $H^\cup$ is also exclusive connected. Note that, in contrast to Lemma 6 in the paper by Bläsius et al. [BKR18], we explicitly allow $S_i^\cup$ (and therefore also $H^\cup$) to contain shared edges incident to u and v. We can do this, as shown in the following, because the face f of μ in $G_\mu^\cup$ is not occupied by prerequisite of this reduction, thus f does not contain any shared edges incident to u or v in $G_\mu^\cup$ and therefore the problematic case shown in Figure 6.4c can not occur.
Since $S_i^\cup$ is exclusive connected and (by prerequisite of this reduction) compatible with face f, the two virtual edges of μ corresponding to f cannot be ①- or ②-linked. This means that $G_\mu^{①}$ contains face $f^{①}$ that is incident to both u and v and $G_\mu^{②}$ contains face $f^{②}$ that is also incident to both u and v. Since $H^\cup + uv$ contains the shared edge uv, we may assume that u and v are positioned on the outer face of the simultaneous embedding of $H^\cup$. We embed all vertices of $H^{①}$ in the face $f^{①}$ of $G_\mu^{①}$ and all vertices of $H^{②}$ in the face $f^{②}$ of $G_\mu^{②}$ to obtain embeddings of $G_\mu^{①} \cup H^{①}$ and $G_\mu^{②} \cup H^{②}$, respectively; see Figure 6.5. Recall that the face f is unoccupied and thus does not contain any shared edges incident to u or v in $G_\mu^\cup$. Therefore, the edge orderings at vertex u are still consistent, since all edges of $H^{①}$ and $H^{②}$ incident to u are embedded between the two shared edges belonging to the facial cycle of f. For the same reason, the edge orderings at v are also consistent. Because we did not change the embeddings of G_μ or $H^\cup$, the edge orderings for all other vertices are also consistent. Additionally, since all shared components of $H^\cup$ now belong to the face f of the shared graph, and all shared components of $G_\mu^\cup$ lie on the outer face of $H^\cup$, the relative positions are also still consistent. Therefore, we obtain a simultaneous embedding of $G^\cup = G_\mu^\cup \cup H^\cup$.

Now consider the case where neither of the split components $S_1^\cup \cup \dots \cup S_t^\cup$ is exclusive connected. Then each $S_i^\cup$ is either only ①-connected, only ②-connected, or union connected. For these three cases, the proofs in Lemma 6 in the paper by Bläsius et al. [BKR18] can be applied verbatim to our use case, because the proofs for these three cases also allow for shared edges in $S_i^\cup$ that are incident to u or v. Therefore, we also obtain a simultaneous embedding of $G^\cup$ if neither of the $S_i^\cup$ is exclusive connected. □

[3]Note that μ must be contained in a single split component by Reduction Rule 1.

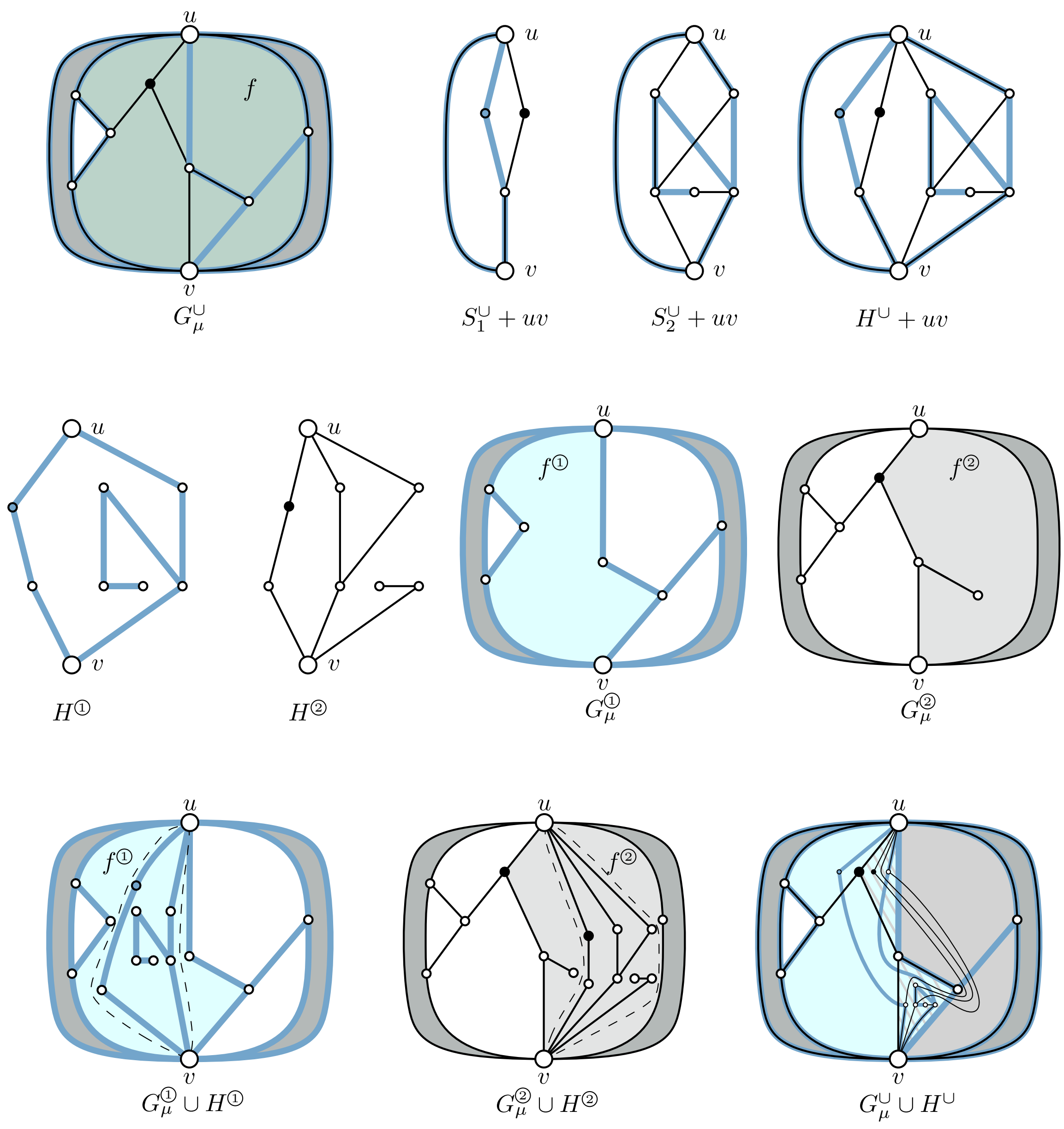


Figure 6.5: An example illustrating how exclusive connected split components can be embedded in an unoccupied face f of G_μ in Reduction Rule 5. First, simultaneous embeddings of $S^{\cup}_{1} + uv$ and $S^{\cup}_{2} + uv$ are glued together to obtain a simultaneous embedding of $H^{\cup} + uv$. Subsequently, the embeddings $H^{①}$ and $H^{②}$ are placed into faces $f^{①}$ and $f^{②}$ of $G^{①}_{\mu}$ and $G^{②}_{\mu}$, respectively. The resulting pair of embeddings of $G^{①} \cup H^{①}$ and $G^{②} \cup H^{②}$ is a simultaneous embedding of $G^{\cup}_{\mu} \cup H^{\cup} = G^{\cup}$.

After exhaustively applying Reduction Rule 5, consider once again a cutvertex v of G and two blocks B_1 and B_2 containing v, such that B_2 is a mutable block with respect to B_1. Recall that this means that B_1 contains a P-node μ with poles v and u, such that B_2 is contained in a split component $S^{\cup}$ of the separating pair $\{u, v\}$ in $G^{\cup}$. Since we cannot bound the number of faces in μ, we need to further restrict the faces of μ that $S^{\cup}$ can be embedded in. Let $\mathcal{F}_{S^{\cup}}$ denote the faces of μ that $S^{\cup}$ is compatible with. By Reduction Rule 4, $\mathcal{F}_{S^{\cup}}$ is not empty. Additionally, $\mathcal{F}_{S^{\cup}}$ only contains occupied faces of μ, because otherwise, we would have removed $S^{\cup}$ with Reduction Rule 5. But since a face f of μ is only occupied if we assigned a binary block incident to u or v to f, the number of occupied faces in f is bounded by the number of blocks incident to u and v. As argued before, there can be at most $k+d$ blocks incident to a single vertex, thus $|\mathcal{F}_{S^\cup}| \leq 2(k+d)$. We create a new branch for every face $f \in \mathcal{F}_{S^\cup}$ and assign the mutable block B_2 to f in this branch. Thus, for every pair B_1, B_2 of blocks containing the same cutvertex v we need at most $2(k+d)$ branches to assign B_2 to every admissible face of B_1. Since we have to do this for each of the $\mathcal{O}((k+d)^2)$ pairs of blocks incident to a single cutvertex and since we have at most $3k$ cutvertices of degree at least 3, we need $\mathcal{O}(((2(k+d))^{(k+d)^2})^{3k}) = \mathcal{O}((2(k+d))^{(k+d)^2 \cdot 3k})$ branches to fix the nesting of the blocks at each cutvertex of G. Note that this bound also includes the branches we created to assign all binary blocks to their respective faces.

Corollary 6.7. *$\mathcal{O}((2(k+d))^{(k+d)^2 \cdot 3k})$ branches are sufficient to fix the nesting of the blocks around every cutvertex of G.*

6.3 Ordering Blocks around Cutvertices

We have now successfully fixed the nesting of the blocks at each cutvertex of G. Note, however, that there may be multiple blocks assigned to the same face f of another block at a cutvertex v of G. In this case, we still need to determine the order of these blocks in the face f. For this reason, we simply enumerate all possible orders of the blocks incident to every cutvertex of G with degree at least 3. As argued before, a single vertex is contained in at most $k+d$ blocks, hence there are at most $(k+d)!$ orderings for the blocks around a single cutvertex. Since we have to do this at each of the at most $3k$ cutvertices of degree at least 3 (Lemma 6.3), this creates an additional $\mathcal{O}(((k+d)!)^{3k})$ branches. Because every block of G has a fixed embedding, we have now completely fixed the order of edges around each cutvertex of G.

Corollary 6.8. *$\mathcal{O}(((k+d)!)^{3k})$ branches are sufficient to fix the order of the blocks around every cutvertex of G.*

6.4 Putting Things Together

Finally, every connected component of the shared graph G now has a fixed embedding in every branch, thus we can use the algorithm by Bläsius and Rutter [BR15] to determine whether $G^\cup$ allows a simultaneous embedding with the given embeddings. Recall that our algorithm first fixes the embedding of each block in G and subsequently nests and orders all blocks around cutvertices of G. By combining Corollaries 6.6, 6.7, and 6.8, we need a total of $\mathcal{O}(2^{\mathcal{O}(k^2)} \cdot (2(k+d))^{(k+d)^2 \cdot 3k} \cdot ((k+d)!)^{3k})$ branches to enumerate all admissible embeddings of all connected components in G. Note that not all branches lead to a valid embedding of all connected components in G (e.g., different nesting decisions can contradict one another). If this is the case, we reduce to a trivial no-instance in the corresponding branch.

Theorem 6.9. *SEFE is FPT parameterized by the number of degree-1 vertices d and the vertex cover number k of the shared graph and can be solved in time*

$$\mathcal{O}(2^{\mathcal{O}(k^2)} \cdot (2(k+d))^{(k+d)^2 \cdot 3k} \cdot ((k+d)!)^{3k} \cdot \mathrm{poly}(n)).$$

7. Parameterization by the Number of Connected Components and the Maximum Degree of the Shared Graph

In this chapter, we develop an FPT-algorithm for SEFE parameterized by the number of shared components k and the maximum degree Δ of the Shared graph. In order to obtain this parameterization, we extend the quadratic-time algorithm for Connected SEFE by Bläsius et al. [BFR20] with additional constraints that also ensure consistent relative positions between the two exclusive graphs. Their algorithm uses the linear-time reduction to the problem Synchronized Planarity described in Chapter 2 (see Figure 2.3). In order to solve the resulting Synchronized Planarity instance, Bläsius et al. [BFR20] transform the graph into an equivalent instance without any pipes using different operations, depending on whether the P-vertices matched by a pipe are cut-vertices or block-vertices, respectively. They show that, after exhaustively applying said operations, the resulting instance contains only Q-constraints but no pipes. These Q-constraints, together with the natural embedding restrictions represented by an SPQR-Tree of the graph, can be expressed as an instance of 2-SAT. Because applying the operations takes time in $\mathcal{O}(m^2)$ and the resulting 2-SAT formula has size $\mathcal{O}(m)$ and can be solved in linear time, Synchronized Planarity can be solved in time $\mathcal{O}(m^2)$.

However, this algorithm cannot handle consistent relative positions of connected components and therefore only works if the shared graph is connected. Figure 2.4 shows an instance where the reduction described above fails. While the two graphs shown in Figure 2.4a clearly do not admit a SEFE, the corresponding instance of Synchronized Planarity is a yes-instance and accepts the embedding shown in Figure 2.4b. The problem is that Synchronized Planarity only synchronizes the rotation of shared edges, but does not ensure consistent relative positions of connected components of the shared graph. In Figure 2.4b, the vertex x is located in different faces of the shared graph in the two embeddings, which is not valid for the original SEFE instance. While the reduction to Synchronized Planarity ensures consistent edge orderings between the two exclusive graphs, it evidently does not ensure consistent relative positions.

Recall from Chapter 2 that a partial constraint for a vertex v is a PQ-tree that constrains the admissible cyclic orderings of a subset of edges incident to v. We will show that we can characterize the embeddings that satisfy consistent relative positions using a set of partial constraints for the vertices in both exclusive graphs. Subsequently, we will encode

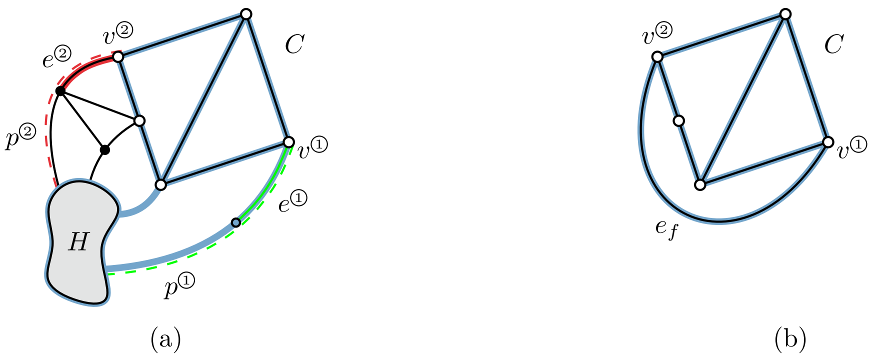


Figure 7.1: (a) A graph $G^{\cup}$ with a pairs $f = (v^{\textcircled{1}}, v^{\textcircled{2}})$ of fixpoints fixing the relative position of shared component H with respect to shared component C. The fixation edges $e^{\textcircled{1}}$ and $e^{\textcircled{2}}$ are marked green and red, respectively. The fixation paths $p^{\textcircled{1}}$ and $p^{\textcircled{2}}$ are illustrated as dashes paths. (b) The corresponding auxiliary graph G'_f containing the dummy edge $v^{\textcircled{1}}v^{\textcircled{2}}$.

these partial constraints into the Synchronized Planarity instance obtained from the reduction from SEFE. The resulting instance represents exactly the simultaneous embeddings of our initial SEFE instance. However, we will still have to modify the operations of the Synchronized Planarity algorithm by Bläsius et al. [BFR20] to also handle the additional partial constraints. Because this is a difficult task in general, we will further restrict our initial SEFE instance and we will finally obtain the desired FPT-algorithm.

7.1 Computing the Partial Constraints

We start by determining the partial constraints that ensure consistent relative positions. Let C and H be two connected components of the shared graph. Because we can assume that $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$ are both connected [BR15], there must be at least one path connecting C and H in $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$, respectively. For $i \in \{1, 2\}$, pick a vertex $v^{\textcircled{i}}$ in C, such that $v^{\textcircled{i}}$ is connected to H via a path $p^{\textcircled{i}}$ in $G^{\textcircled{i}}$ containing no vertices from C. We call the pair $f = (v^{\textcircled{1}}, v^{\textcircled{2}})$ of vertices the *fixpoints* of H with respect to C; see Figure 7.1a for an example. We also say that f *directly fixes* the position of H with respect to C. We refer to the path $p^{\textcircled{i}}$ as the *fixation path* of f in $G^{\textcircled{i}}$. The edge $e^{\textcircled{i}}$ of $p^{\textcircled{i}}$ that is incident to $v^{\textcircled{i}}$ is called the *fixation edge* of fixpoint $v^{\textcircled{i}}$. We let $\mathcal{F}$ denote the set of fixpoint pairs obtained after repeating this for every pair of shared components in G. Note that $|\mathcal{F}| \leq k^2$, since G contains k connected components. With the following lemma, we show that ensuring that each pair of fixation edges is embedded into the same face of G is sufficient to guarantee consistent relative positions for the SEFE instance.

Lemma 7.1. *Let $\mathcal{E}^{\cup} = (\mathcal{E}^{\textcircled{1}}, \mathcal{E}^{\textcircled{2}})$ denote planar embeddings of $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$ with consistent edge orderings. Then $\mathcal{E}^{\cup}$ is a simultaneous embedding if and only if the two corresponding fixation edges are embedded in the same face of G for every pair of fixpoints in $\mathcal{F}$.*

Proof. Let C and H denote two shared components and let $(v^{\textcircled{1}}, v^{\textcircled{2}}) \in \mathcal{F}$ denote the fixpoints of H with respect to C. It is clear that, for $i \in \{1, 2\}$, the fixation path $e^{\textcircled{i}}$ of $v^{\textcircled{i}}$ must be entirely contained in a single face h of C in $\mathcal{E}^{\textcircled{i}}$, because otherwise, $e^{\textcircled{i}}$ crosses the boundary of h. Since this boundary consists of shared edges, this would contradict the planarity of $\mathcal{E}^{\textcircled{i}}$, thus $e^{\textcircled{i}}$ is entirely contained in a single face h of C. Since $e^{\textcircled{i}}$ also contains a vertex of the shared component H, H must also be fully contained in the face h of C in $\mathcal{E}^{\textcircled{i}}$. Consequently, the relative position of H with respect to C is consistent if and only if the two fixation paths are contained in the same face of C, which concludes the proof. □

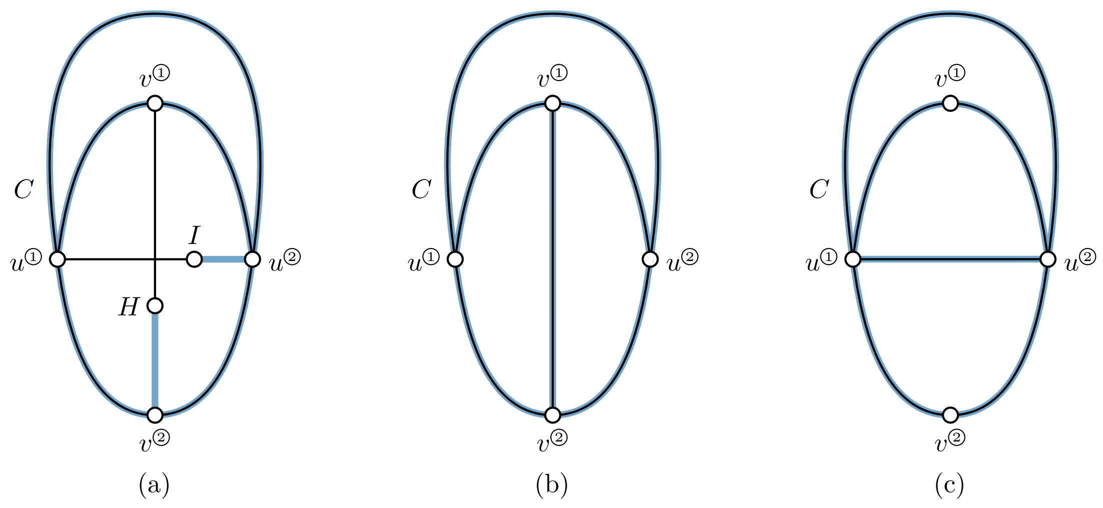


Figure 7.2: (a) A graph $G^{\cup}$ with two pairs $f = (v^{①}, v^{②})$ and $g = (u^{①}, u^{②})$ of fixpoint vertices fixing the relative positions of H and I with respect to C. (b) The auxiliary graph G'_f containing the dummy edge $v^{①}v^{②}$. (c) The auxiliary graph G'_g containing the dummy edge $u^{①}u^{②}$.

Lemma 7.1 states that, in any simultaneous embedding of $G^{\cup}$, both fixation edges of a pair $f \in \mathcal{F}$ of fixpoints must be embedded in the same face of G. In order to determine the embeddings of G that fulfill this requirement, we now construct a set of auxiliary graphs from which we will derive the necessary partial constraints for our Synchronized Planarity instance.

To this end, let $f = (v^{①}, v^{②}) \in \mathcal{F}$ be a pair of fixpoints of $G^{\cup}$ such that f directly fixes the position of a shared component H with respect to another shared component C. We obtain the auxiliary graph G'_f from the shared graph G by adding the edge $v^{①}v^{②}$ to G; see Figure 7.1b. We refer to $v^{①}v^{②}$ as the *dummy edge* corresponding to f in G'_f. This shared edge represents a path p_f between $v^{①}$ and $v^{②}$ in $G^{\cup}$ that consists of the fixation paths $p^{①}$ and $p^{②}$, and vertices of the shared component H. No shared edge of C may cross an edge of p_f in $G^{\cup}$ and, similarly, no shared edge of C may cross the dummy edge $v^{①}v^{②}$ in G'_f. We call this path p_f the *representation path* of f. Therefore, if G'_f is not planar, we can immediately reduce to a trivial no-instance, because in any embedding of $G^{\cup}$ with consistent edge orderings, there is a shared edge of G that crosses p_f. Since every pair of fixation edges must be embedded in the same face of G in any simultaneous embedding of $G^{\cup}$ (Lemma 7.1), the auxiliary graph G'_f describes exactly the admissible embeddings of G where the fixation edges of f can be embedded in a common face. Essentially, we want to ensure that the shared graph in $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$ only takes embeddings that are "allowed" by the auxiliary graph G'_f. We will additionally ensure that the fixation edges of f are embedded in the same position in $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$ as the dummy edge $v^{①}v^{②}$ in the corresponding embedding of G'_f. Since the dummy edge lies within a single face of the shared graph G, the two fixation edges of f consequently lie in that same face of G. By Lemma 7.1, ensuring this for every pair of fixpoints in $\mathcal{F}$ yields exactly the simultaneous embeddings of $G^{\cup}$. Note that we create a separate auxiliary graph for each pair of fixpoints; see Figure 7.2.

We now want to derive partial constraints for the vertices of $G^{\cup}$ from the auxiliary graph G'_f that ensure that we only admit embeddings of the shared graph where both fixation edges are embedded in the same face. To this end, let B denote a block of the auxiliary graph G'_f. Observe that B may consist of several blocks of the shared graph G; see Figure 7.3 for an example. Using the SPQR-tree representation of G'_f, determine the embedding tree $T_{f,v}$

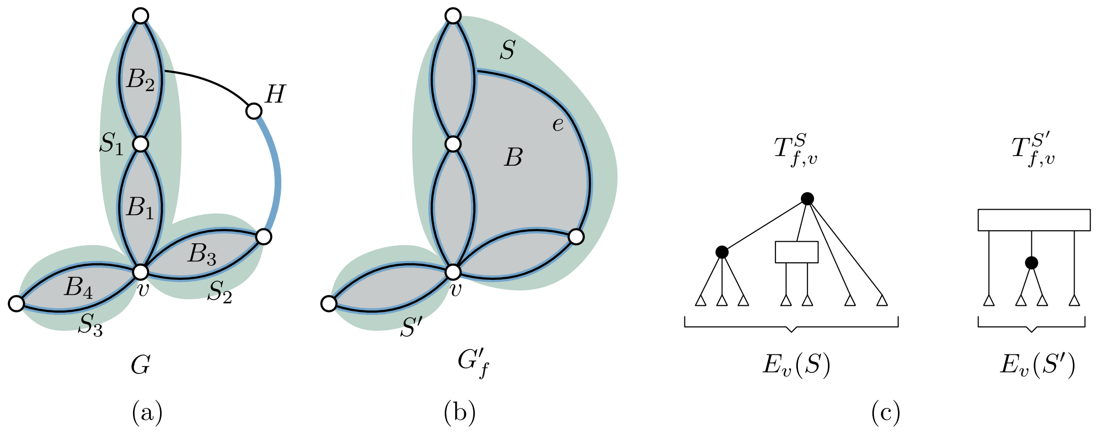


Figure 7.3: (a) The shared graph G consisting of blocks $B_1, \dots, B_4$. There are three split components S_1, S_2, and S_3 of G incident to cutvertex v. Additionally, the two fixation paths corresponding to shared component H are shown. (b) The auxiliary graph G'_f containing the dummy edge e corresponding to shared component H. The blocks B_1, B_2, and B_3 fall together into a single block B in G'_f and the split components S_1 and S_2 fall together into a single split component S incident to v in G'_f. (c) Possible face constraint trees $T^S_{f,v}$ and $T^{S'}_{f,v}$ obtained from the SPQR-trees of G'_f. The tree $T^S_{f,v}$ constrains the edges $E_v(S)$ of split component S incident to v in G'_f (Analogously for $T^{S'}_{f,v}$).

for every vertex v in B. If v is contained in multiple blocks of G'_f, we get an embedding tree $T^S_{f,v}$ for every split component S incident to v in G'_f. Since G'_f only contains a single additional edge compared to G, S consists of at most two split components incident to v in the shared graph G. We call $T^S_{f,v}$ the *face constraint tree* of S at v for the pair f of fixpoints; see Figure 7.3c for an example. If a face constraint tree of v contains a dummy edge of G'_f as one of its leaves l, we subsequently identify l with the corresponding fixation edge incident to v.

Unfortunately, we cannot just pick orderings for each face constraint tree independently. In order to obtain a valid embedding of G'_f, every pair of P-nodes stemming from the same P-node of the SPQR-tree $\mathcal{T}$ of G'_f and all Q-nodes stemming from the same R-node of $\mathcal{T}$ must be ordered consistently. For a Q-node q, let $\psi(q)$ be a binary variable denoting the rotation of q. For a set Q of Q-nodes stemming from the same R-node of $\mathcal{T}$, we require that $\psi(q_1) = \psi(q_2)$ holds for every $q_1, q_2 \in Q$ in any embedding of G'_f. For a P-node p of a face constraint tree, let σ_p denote an order of its children in an embedding of G'_f. For two P-nodes p_1 and p_2 stemming from the same P-node μ of $\mathcal{T}$, we require that $\sigma_{p_1} = \delta_{p_2 p_1}(\sigma_{p_2})$ holds in any embedding of G'_f, where $\delta_{p_2 p_1}$ maps the children of p_2 to the children of p_1 according to the virtual edges in μ. We call these constraints the *synchronization constraints* between the face constraint trees.

Recall that a single split component incident to a vertex v in the auxiliary graph G'_f may consist of up to two split components incident to v in the shared graph G. For this reason, we need to be extra cautious at cutvertices of the shared graph. Let v denote a cutvertex of G and let S_1, S_2, and S_3 denote three split components of G incident to v such that S_1 and S_2 are connected as a single split component S of G'_f (i.e., S_1 and S_2 belong to the same block of G'_f). For a given embedding of the shared graph, even if the edges in S satisfy the face constraint tree $T^S_{f,v}$, S_1 and S_2 could still be embedded in different faces of S_3; see Figure 7.4 for an example. This means that the face constraint trees alone are not sufficient to ensure that every relative position corresponds to a face of G if a single pair of fixpoints is contained in two blocks of G. We solve this issue using additional partial constraints.

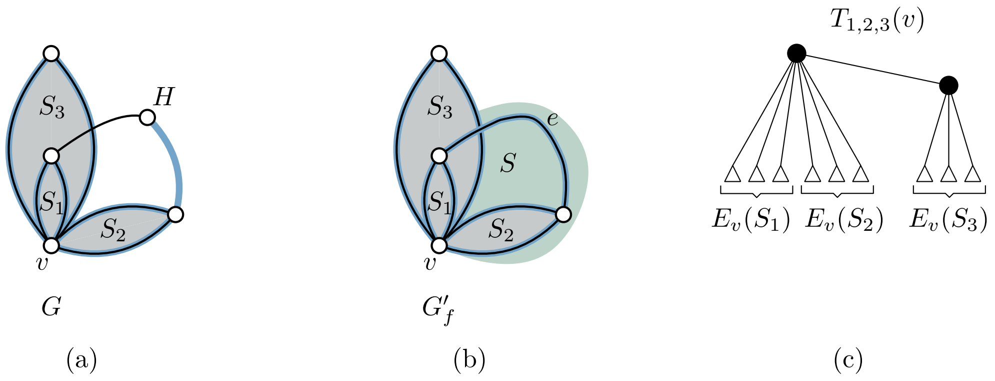


Figure 7.4: (a) Three split components S_1, S_2, and S_3 of a cutvertex v in the shared graph G. Because there are fixation paths connecting S_1 and S_2 to H, S_1 and S_2 must be embedded in the same face of S_3. (b) In the corresponding auxiliary graph G'_f, the dummy edge e connects S_1 and S_2 into a new split component S. However, since S_3 and S are still disconnected in G'_f, no face constraint tree forbids S_1 and S_2 to be embedded into different faces of S_3. (c) The pairwise consecutivity tree $T_{1,2,3}(v)$ ensuring that the edges of S_1 and S_2 are consecutive with respect to the edges in S_3.

Let $S_1, \dots, S_l$ denote the split components of G incident to v and let $E_v(S_i)$ denote the set of edges incident to v in G belonging to split component S_i. Let S_i, S_j be a pair of split components with $1 \leq i < j \leq l$ such that $E_v(S_i)$ and $E_v(S_j)$ belong to the same split component of G'_f. For every split component S_k with $1 \leq k \leq l$, let $T_{i,j,k}(v)$ denote the PQ-tree with leaves $L(T_{i,j,k}(v)) = E_v(S_i) \cup E_v(S_j) \cup E_v(S_k)$ that ensures that the edges in $E_v(S_i) \cup E_v(S_j)$ are consecutive. We call $T_{i,j,k}(v)$ the *pairwise consecutivity tree* of (S_i, S_j) and S_k; see Figure 7.4c for an example. Observe that these partial constraints ensure that S_i and S_j are always embedded in the same face of S_k. Also note that these trees are similar to the pairwise consecutivity trees introduced by Bläsius et al. [BKR18].

Lemma 7.2. *An instance $G^\cup$ of SEFE admits a simultaneous embedding $\mathcal{E}^\cup = (\mathcal{E}^{①}, \mathcal{E}^{②})$ if and only if $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$ have consistent edge orderings and $\mathcal{E}^\cup$ satisfies all face constraint trees, all synchronization constraints, and all pairwise consecutivity trees.*

Proof. If $\mathcal{E}^\cup = (\mathcal{E}^{①}, \mathcal{E}^{②})$ is a simultaneous embedding $G^\cup$, then it is clear that both embeddings must have consistent edge orderings. Consider two split components S_1 and S_2 of a cutvertex of the shared graph such that there exists a pair $f = (v^{①}, v^{②})$ of fixpoints with $v^{①}$ and $v^{②}$ contained in S_1 and S_2, respectively. Then clearly, S_1 and S_2 must be embedded in the same face of the shared graph, because otherwise, a shared edge would cross the representation path of f. Since the pairwise consecutivity trees only ensure that S_1 and S_2 are embedded in the same face of the shared graph, they are therefore satisfied for $\mathcal{E}^\cup$. Now take the embedding $\mathcal{E}^\cup$ of $G^\cup$, replace the representation path p_f corresponding to f with a single dummy edge, and subsequently remove all exclusive edges. The result is a planar embedding $\mathcal{E}$ of G'_f. Since the face constraint trees and synchronization constraints are obtained directly from the embedding representation of G'_f, they are therefore all satisfied in $\mathcal{E}^\cup$.

Conversely, let $\mathcal{E}^\cup = (\mathcal{E}^{①}, \mathcal{E}^{②})$ be an embedding of $G^\cup$ with consistent edge orderings and such that $\mathcal{E}^\cup$ satisfies all pairwise consecutivity constraints, all face constraint trees, and all synchronization constraints. Since $\mathcal{E}^\cup$ has consistent edge orderings, it only remains to show that for any pair C, H of shared components of G, H is embedded in the same face of C in $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$. Let $f = (v^{①}, v^{②})$ denote the pair of fixpoints in $G^\cup$ that directly fixes the position of H with respect to C. Then we have an auxiliary graph G'_f with an additional dummy edge that corresponds to the representation path p_f. First assume that both vertices of f are contained in the same block of the shared graph. Because the face constraint trees and the synchronization constraints are satisfied in $\mathcal{E}^{\cup}$, and no shared edge may cross the dummy edge in any embedding of G'_f, no shared edge crosses the representation path p_f in $\mathcal{E}^{\cup}$. If the two vertices of f are contained in two different blocks of G, then the pairwise consecutivity trees additionally ensure that no shared edge of another block may cross the representation path p_f. Overall, if all these constraints are satisfied, no shared edge may cross p_f in $\mathcal{E}^{\cup}$ and thus H is contained in the same face of C in $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$. □

Observe that we now have a set of face constraint trees for each shared vertex v of G and, if v is a cutvertex in G, additionally a set of pairwise consecutivity trees. From now on, we group these constraints together in a set $\mathcal{R}_v$ and refer to them as the *partial constraints* of v.

7.2 Reduction to Synchronized Planarity

Recall that finding a simultaneous embedding of $G^{\cup}$ requires finding an embedding with consistent edge orderings and consistent relative positions. By Lemma 7.2, ensuring that every shared vertex of $G^{\cup}$ satisfies the partial constraints together with the synchronization constraints guarantees consistent relative positions. The problem SYNCHRONIZED PLANARITY is suitable to ensure that all shared edges are ordered consistently in both exclusive graphs [BFR20, Theorem 16]. Therefore, our goal is to augment the SYNCHRONIZED PLANARITY instance with our partial constraints in order to represent exactly the simultaneous embeddings of $G^{\cup}$.

Given our initial instance $G^{\cup}$ of SEFE with exclusive graphs $G^{①}$ and $G^{②}$, we now reduce our instance to an instance $\mathcal{I}^* = (H, \mathcal{P}, \mathcal{Q}, \varphi)$ of SYNCHRONIZED PLANARITY using the reduction by Bläsius et al. [BFR20]. Add two new vertices $b_v^{①}$ and $b_v^{②}$ for each shared vertex v appearing in the exclusive graphs as $v^{①}$ and $v^{②}$, respectively. For each shared edge incident to v, add a parallel edge between $b_v^{①}$ and $b_v^{②}$, creating a bond μ between $b_v^{①}$ and $b_v^{②}$ where the parallel edges correspond to the shared edges incident to v. Additionally, insert degree-1 vertices incident to $b_v^{①}$ and $b_v^{②}$ representing the exclusive edges incident to $v^{①}$ and $v^{②}$, respectively; see Figure 2.3 for an example. Create two pipes $(v^{①}, b_v^{①}, \varphi_1)$ and $(v^{②}, b_v^{②}, \varphi_2)$, where φ_1 and φ_2 map the edges incident to $v^{①}$ and $v^{②}$ to their corresponding edges of μ. Recall that SYNCHRONIZED PLANARITY only ensures consistent edge orderings and not consistent relative positions for our SEFE instance, i.e., $G^{\cup}$ and $\mathcal{I}^*$ are not yet equivalent. We still have to restrict the admissible edge orderings at each vertex in $\mathcal{I}^*$ to ensure that they conform with the partial constraints and synchronization constraints obtained in Section 7.1.

To this end, let $\rho = (v^{①}, b_v^{①}, \varphi)$ be an arbitrary pipe of $\mathcal{I}^*$. Observe that $v^{①}$ and $b_v^{①}$ correspond to the same shared vertex v of the SEFE instance $G^{\cup}$. We now augment ρ with the partial constraints $\mathcal{R}_v$, turning ρ into a *constrained pipe* $\rho' = (v^{①}, b_v^{①}, \varphi, \mathcal{R}_v)$. We say that an embedding satisfies the constrained pipe ρ' if it satisfies the pipe ρ and all partial constraints in $\mathcal{R}_v$. Additionally, all synchronization constraints between partial constraints must be satisfied. The resulting instance is $\mathcal{I}_{\text{init}} = (H, \mathcal{P}', \mathcal{Q}, \varphi)$ with $H = H^{①} + H^{②} + X$, where $H^{①}$ and $H^{②}$ are the subgraphs of H corresponding to the exclusive graphs $G^{①}$ and $G^{②}$ of $G^{\cup}$ and X contains the bonds between $H^{①}$ and $H^{②}$. Since the standard SYNCHRONIZED PLANARITY reduction ensures consistent edge orderings, the instance $\mathcal{I}_{\text{init}}$ is equivalent to the instance $G^{\cup}$ of SEFE by Lemma 7.2.

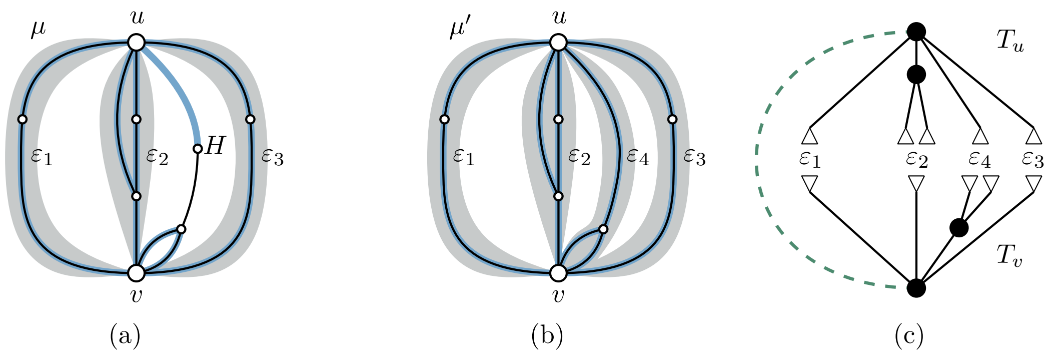


Figure 7.5: (a) A P-node μ of the shared graph G consisting of three virtual edges. (b) The corresponding P-node μ' of the corresponding auxiliary graph after adding the dummy edge corresponding to the fixation edges of shared component H. This results in an additional virtual edge ε_4. (c) Since the structure of G itself does not ensure that ε_4 has the same position incident to both poles u and v, the P-nodes of the corresponding face constraint trees T_u and T_v require a synchronization constraint.

Since synchronization constraints are not necessarily local, they create complicated dependencies between the pipes of our initial Synchronized Planarity instance $\mathcal{I}_{\text{init}}$. While synchronization constraints between Q-nodes can simply be handled by synchronizing the binary rotation decision of the Q-nodes, synchronization constraints between P-nodes are more difficult. Therefore, we seek to eliminate this issue using bounded search trees. Recall that $\mathcal{F}$ denotes the set of fixpoint pairs of $G^\cup$, with $|\mathcal{F}| \leq k^2$. For every pair $f = (v^{①}, v^{②})$ of fixpoints in $\mathcal{F}$, we enumerate all possible orderings of shared edges and fixation edges around $v^{①}$ and $v^{②}$, creating a new branch for each of the $\mathcal{O}((\Delta + k)!)$ orderings at each vertex. Regarding the synchronization constraints between inner nodes of partial constraints derived from an auxiliary graph, observe that these synchronization constraints are only necessary if the orderings of the corresponding nodes are not already synchronized due to the structure of the shared graph G itself. Recall that we only add a single edge to G in every auxiliary graph. This is equivalent to an application of the operation `insertEdge` in the paper by Di Battista and Tammassia [BT96]. They thoroughly describe the possible changes this operation can make to the SPQR-trees of the graph. Essentially, we only have to add an additional synchronization constraint between P-nodes, if `insertEdge` either created a new P-node in the SPQR-tree, or an existing P-node gets an additional virtual edge; see Figure 7.5 for an example. By Battista and Tammassia [BT96], there can only be one such P-node per operation, thus we only need to additionally fix the edge ordering at the two poles of this P-node. Let u and v denote the poles of such a P-node. As we did with the fixpoints, we again create a new branch for every possible ordering of shared edges incident to u and v. Let $\mathcal{S}$ denote the set of all vertices that now have a fixed ordering of their incident edges. Since we have at most $2k^2$ fixpoints and k^2 auxiliary graphs in total, it is $|\mathcal{S}| \leq 4k^2$ and we therefore get $\mathcal{O}(((\Delta + k)!)^{4k^2})$ branches in total. For each vertex v, for which we have now "guessed" an ordering ϕ, we fix the edges incident to v according to ϕ in every auxiliary graph and we update the partial constraints we derived from the auxiliary graphs accordingly. Whenever the enumerated orderings at different vertices are incompatible, we reduce to a trivial no-instance in that branch. Observe that we now eliminated all synchronization constraints between P-nodes of the partial constraints in the instance $\mathcal{I}_{\text{init}}$.

To simplify the partial constraints in the instance $\mathcal{I}_{\text{init}}$ even further, we now eliminate all partial constraints that contain fixation edges. Let $v \in \mathcal{S}$ denote a vertex with a fixed ordering ϕ of its incident shared edges and fixation edges and let $v^{①}$ and $v^{②}$ denote the

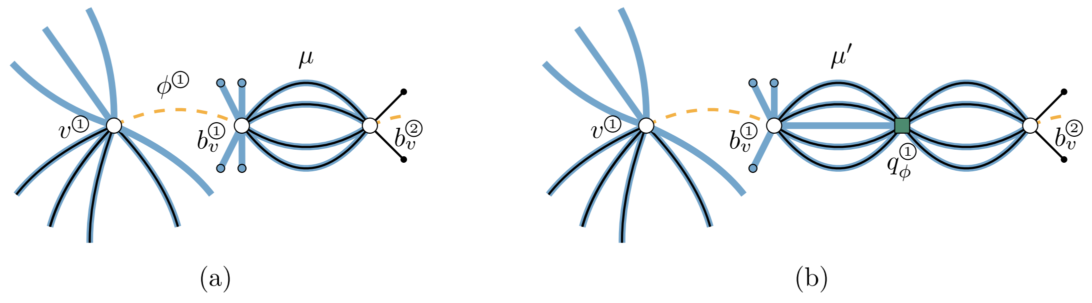


Figure 7.6: (a) A pipe connecting two vertices $v^{\textcircled{1}}$ and $b_v^{\textcircled{1}}$ with a fixed ordering $\phi^{\textcircled{1}}$ for all shared edges and fixation edges. (b) The resulting equivalent instance of Synchronized Planarity after applying the operation `PropagatePartial`. The Q-vertex $q_\phi^{\textcircled{1}}$ ensure that the edges constrained by $\phi^{\textcircled{1}}$ appear in the order defined by $\phi^{\textcircled{1}}$ around the vertex $b_v^{\textcircled{1}}$ (and consequently also around the vertex $v^{\textcircled{1}}$).

two occurrences of v in $H^{\textcircled{1}}$ and $H^{\textcircled{2}}$, respectively, of $\mathcal{I}_{\text{init}}$. Let $\rho^{\textcircled{1}}$ denote the pipe between $v^{\textcircled{1}}$ and $b_v^{\textcircled{1}}$, where $b_v^{\textcircled{1}}$ is the corresponding vertex of the bond μ between $b_v^{\textcircled{1}}$ and $b_v^{\textcircled{2}}$; see Figure 7.6a. Let $\phi^{\textcircled{1}}$ denote the restriction of ϕ to only shared edges and $\textcircled{1}$-exclusive edges. We now directly encode the ordering $\phi^{\textcircled{1}}$ into the graph structure. First, order the parallel edges of μ as they appear in $\phi^{\textcircled{1}}$ and subdivide each of the parallel edges. Merge the subdivision vertices into a single Q-vertex $q_\phi^{\textcircled{1}}$ that fixes the order of the shared edges according to $\phi^{\textcircled{1}}$. For every degree-1 vertex x incident to $b_v^{\textcircled{1}}$ that corresponds to a fixation edge, also merge x into $q_\phi^{\textcircled{1}}$ at the position defined by $\phi^{\textcircled{1}}$; see Figure 7.6b for an example. The Q-constraint corresponding to $q_\phi^{\textcircled{1}}$ has a fixed rotation, as defined by $\phi^{\textcircled{1}}$. We subsequently turn the pipe $\rho^{\textcircled{1}}$ into a regular pipe without any partial constraints. We repeat this procedure symmetrically for $\phi^{\textcircled{2}}$ at vertex $v^{\textcircled{2}}$ and call this procedure `PropagatePartial`.

Lemma 7.3. *Applying the operation* `PropagatePartial` *to an instance $\mathcal{I}$ of* Synchronized Planarity *yields an equivalent instance $\mathcal{I}'$.*

Proof. We show that applying the construction to $v^{\textcircled{1}}$ is correct, the correctness of the whole operation then follows from symmetry. Let $\rho = (v^{\textcircled{1}}, b_v^{\textcircled{1}}, \varphi, \mathcal{R})$ denote the pipe matching $v^{\textcircled{1}}$ and $b_v^{\textcircled{1}}$. Note that we can assume that $\mathcal{R}$ only contains the ordering $\phi^{\textcircled{1}}$ as a partial constraint, since $\phi^{\textcircled{1}}$ completely fixes the order of all shared edges and fixation edges around $v^{\textcircled{1}}$. Let $\mathcal{E}$ be a valid embedding of $\mathcal{I}$ such that $\mathcal{E}(v^{\textcircled{1}}) = \varphi(\overline{\mathcal{E}(b_v^{\textcircled{1}})})$ satisfies the partial ordering $\phi^{\textcircled{1}}$. To obtain an embedding $\mathcal{E}$' of $\mathcal{I}$', insert the Q-vertex $q_\phi^{\textcircled{1}}$ into the bond μ as described above. Since $\overline{\mathcal{E}(b_v^{\textcircled{1}})}$ satisfies $\phi^{\textcircled{1}}$ and since all other edges incident to $b_v^{\textcircled{1}}$ that are not constrained by $\phi^{\textcircled{1}}$ belong to degree-1 dummy vertices, this insertion does not introduce any crossings.

Conversely, let $\mathcal{E}'$ denote an embedding of $\mathcal{I}'$. Since the order of shared edges and fixation edges incident to $b_v^{\textcircled{1}}$ is fixed by $q_\phi^{\textcircled{1}}$, $\mathcal{E}'(v^{\textcircled{1}}) = \varphi(\overline{\mathcal{E}'(b_v^{\textcircled{1}})})$ satisfies the partial ordering $\phi^{\textcircled{1}}$. Revert the insertion of the vertex $q_\phi^{\textcircled{1}}$, i.e., first “unmerge“ $q_\phi^{\textcircled{1}}$ and subsequently contract the subdivision vertices of the shared parallel edges in μ. This does not change the order of edges incident to $b_v^{\textcircled{1}}$ and thus we obtain the desired embedding of $\mathcal{I}$. □

To summarize, recall that we have completely fixed the order of shared edges and fixation edges around all fixpoints and around all vertices that would require synchronization constraints between P-nodes of their corresponding partial constraints. The set of these vertices is denoted by $\mathcal{S}$. We then used the operation `PropagatePartial` to encode these fixed orderings directly into the graph structure, thus the vertices in $\mathcal{S}$ have no partial constraints in the resulting Synchronized Planarity instance. This is helpful for two reasons. First, we no longer have partial constraints that additionally constrain fixation edges, because all fixpoints are contained in $\mathcal{S}$. Therefore, all remaining partial constraints only constrain shared edges. Second, we no longer have to worry about synchronization constraints between P-nodes, all remaining synchronization constraints only concern Q-nodes. These synchronization constraints will be no problem, because we can simply encode them into the Q-constraints of our Synchronized Planarity instance.

7.3 Invariants

As the next step, we still have to show how the operations of the Synchronized Planarity algorithm can be adjusted to also handle constrained pipes. For arbitrary Synchronized Planarity instances with arbitrary constrained pipes, this is very difficult. Especially the operation `EncapsulateAndJoin` poses a challenge, because the partial constraints essentially restrict the admissible cuts of the resulting bipartition, which is difficult to model using pipes and Q-constraints. However, the Synchronized Planarity instances we can obtain from the reduction from SEFE are very restricted. To restrict the possible cases even more, we assume that both exclusive graphs are biconnected. This way, the only cutvertices of our initial Synchronized Planarity instance $\mathcal{I}_{\text{init}}$ are the vertices adjacent to degree-1 dummy vertices. Additionally, we require that every pair of fixpoints in $\mathcal{F}$ is *block-local*, i.e., they are contained in the same block of the shared graph G. With this additional restriction, every face constraint tree restricts exactly one split component of the shared graph incident to a vertex. Additionally, the pairwise consecutivity trees become superfluous.

While these restrictions significantly limit the possible cases that can occur in our initial Synchronized Planarity instance $\mathcal{I}_{\text{init}}$, we also need to ensure that this also remains the case in the intermediate instances obtained after applying an operation of the Synchronized Planarity algorithm. To this end, we use this chapter to state several invariants that restrict the possible structures of our Synchronized Planarity instance. In the next chapter, we will then show that these invariants are sufficient to modify the Synchronized Planarity operations to also handle the additional partial constraints. This will allow us to solve the initial Synchronized Planarity instance $\mathcal{I}_{\text{init}}$ and consequently the equivalent SEFE instance $G^{\cup}$.

To have access to more information about our Synchronized Planarity instance, we retain a mapping $o : V(\mathcal{I}) \to V(G^{\cup})$ from every vertex v of an instance $\mathcal{I}$ to a vertex $o(v)$ of the initial SEFE instance $G^{\cup}$. In the instance $\mathcal{I}_{\text{init}}$, every vertex $u^{①}$ of $H^{①}$ is mapped to the vertex u of $G^{\cup}$ it originates from and for every pipe matching vertices $u^{①}$ and $b_u^{①}$, we set $o(b_u^{①}) = o(u^{①}) = u$. Let v denote a vertex of the Synchronized Planarity instance $\mathcal{I}$ and let $o(v)$ denote the corresponding vertex of the SEFE instance $G^{\cup}$. We additionally define a mapping from the shared edges incident to $o(v)$ to the edges incident to v. For an edge e incident to $o(v)$, we let $r_v(e)$ denote the corresponding edge incident to v. This defines a mapping $r : V(\mathcal{I}) \times E(G) \to E(\mathcal{I})$. In the initial Synchronized Planarity instance $\mathcal{I}_{\text{init}}$, we simply map the shared edges incident to $o(v)$ in $G^{\cup}$ to the corresponding edges incident to v in $\mathcal{I}_{\text{init}}$, thus r_v is an injection in $\mathcal{I}_{\text{init}}$; see Figure 7.7. However, r_v is not necessarily injective after the operations `PropagatePQ` and `EncapsulateAndJoin` introduce new vertices. After these operations, r_v can map several shared edges incident to $o(v)$ to a single edge incident to v. Additionally, there may be shared edges incident to $o(v)$ that have no corresponding edge incident to v. In this case, we write $r_v(e) = \perp$

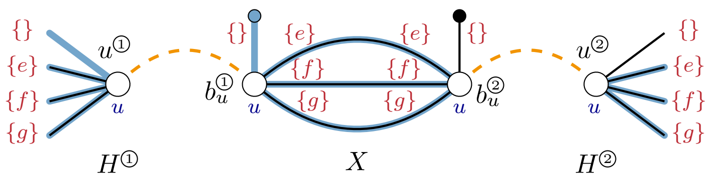


Figure 7.7: The mappings o and r in the Synchronized Planarity instance $\mathcal{I}_{\text{init}}$. Each vertex x is annotated in blue with its corresponding vertex $o(x)$ of the instance $G^{\cup}$. For each edge h incident to a vertex x, the red set represents $r_x^{-1}(h)$, i.e., the set of shared edges incident to $o(x)$ that h corresponds to.

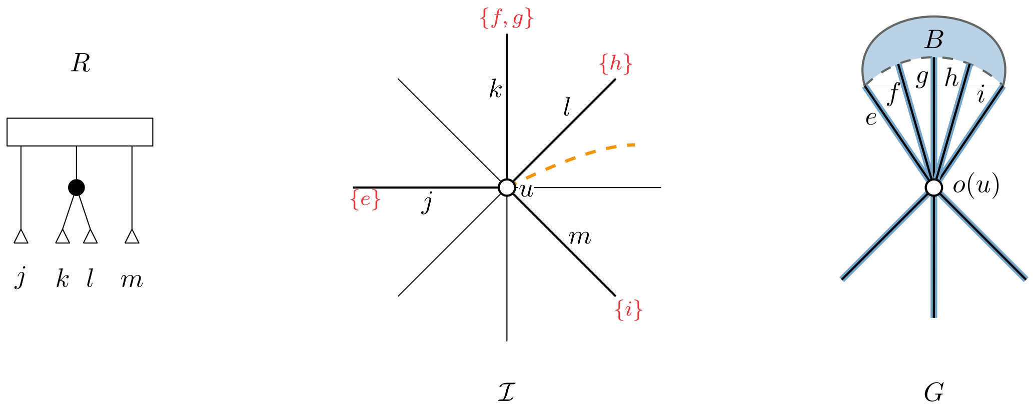


Figure 7.8: An illustration of Invariant 7.1. The partial constraint R restricts exactly the edges $\{j, k, l, m\}$ incident to vertex u in $\mathcal{I}$ that correspond to the edges of the shared block B incident to $o(u)$ in G.

to denote that the shared edge e incident to $o(v)$ has no corresponding edge incident to v. We will define how the mappings o and r are updated when we describe the individual operations in Section 7.4.

Roughly speaking, the mapping r_u gives us an idea which edges incident to u correspond to the shared edges incident to $o(u)$ in G. This mapping therefore also tells us which edges incident to u correspond to edges of a single block incident to $o(u)$ in G. This allows our invariants to make statements about the blocks of G containing $o(u)$ and about the corresponding edges incident to u in the Synchronized Planarity instance $\mathcal{I}$. From now on, if we say *a shared block B around $o(u)$ in G*, we mean a block B of the shared graph G that contains $o(u)$. Additionally, B implicitly refers to the set of edges incident to $o(u)$ that belong to the block B in G. For a set E of edges incident to $o(u)$, we define $r_u(E) = \{r_u(e) \mid e \in E\}$. For a shared block B around $o(u)$ in G, the set $r_u(B)$ is therefore now well-defined.

Our first invariant states that every partial constraint in the Synchronized Planarity instance refers to a block in the shared graph G; see Figure 7.8 for an illustration. This will later allow us to make statements about the structure of our partial constraints using the mapping r. Recall that a partial constraint R is a PQ-tree, thus $L(R)$ denotes the set of leaves of R.

Invariant 7.1. *Let $\rho = (u, v, \varphi_{uv}, \mathcal{R})$ be a pipe of $\mathcal{I}$. For every partial constraint $R \in \mathcal{R}$ there exists a shared block B around $o(u)$ in G, such that R constrains exactly the edges incident to u that correspond to the block B, i.e., $L(R) = r_u(B)$.*

Since we require that every pair of fixpoints is block-local and since we have eliminated all partial constraints containing fixation edges, we have already shown before that every

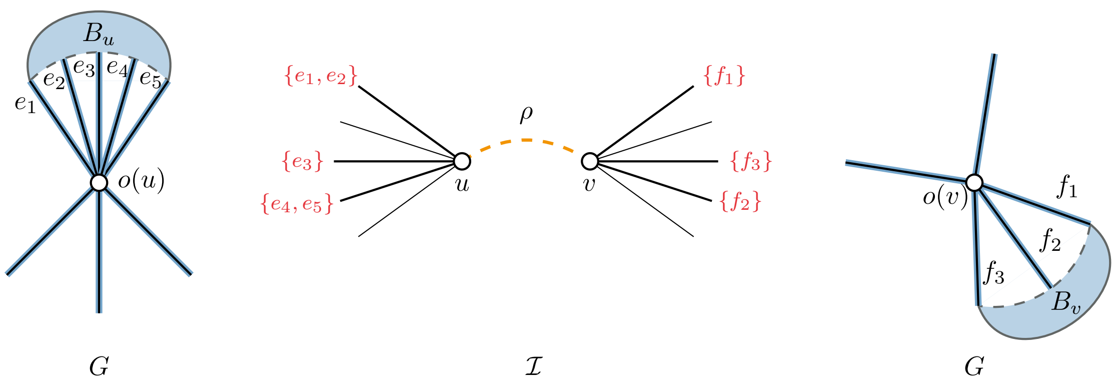


Figure 7.9: An illustration of Invariant 7.2. The edges incident to u in $\mathcal{I}$ that correspond to the shared block B_u are bijectively mapped to the edges incident to v that correspond to B_v by the pipe ρ.

partial constraint restricts the edges of a single block in the shared graph. The construction of the Synchronized Planarity instance $\mathcal{I}_{\text{init}}$ and our choice of the mapping functions o and r in $\mathcal{I}_{\text{init}}$ therefore ensure that Invariant 7.1 holds in $\mathcal{I}_{\text{init}}$.

Our next invariant states that, for two vertices u and v matched by a pipe ρ, the shared blocks around $o(u)$ and $o(v)$ come in pairs whose corresponding edges incident to u and v are matched bijectively by ρ; see Figure 7.9 for an illustration.

Invariant 7.2. *Let $\rho = (u, v, \varphi_{uv}, \mathcal{R})$ be a pipe of $\mathcal{I}$ and let B_u denote a shared block around $o(u)$ in G with $|r_u(B_u)| > 1$. Then there exists a shared block B_v around $o(v)$ such that $r_v(B_v) = \varphi_{uv}(r_u(B_u))$*

We define $\varphi_{uv}(\perp) = \perp$, thus the invariant is also well-defined if edges incident to $o(u)$ are mapped to $\perp$. Recall that every pipe in the initial Synchronized Planarity instance $\mathcal{I}_{\text{init}}$ matches vertices $v^{\textcircled{1}}$ and $b_v^{\textcircled{1}}$ both corresponding to the same vertex v of the shared graph, thus we defined $o(v^{\textcircled{1}}) = o(b_v^{\textcircled{1}}) = v$. Since the pipe ρ matches the edges incident to $v^{\textcircled{1}}$ and $b_v^{\textcircled{1}}$ that correspond to the same edge of the SEFE instance $G^{\cup}$, Invariant 7.2 holds in $\mathcal{I}_{\text{init}}$.

The next invariant basically makes the same statement as the previous invariant, but for poles of a P-node μ in the Synchronized Planarity instance; see Figure 7.10 for an illustration. While we will only need this invariant for bonds, it is easier to show that the invariant remains intact throughout the algorithm if we state it for general P-nodes. Together with the previous invariant, this invariant will allow us to ensure that Invariant 7.1 remains intact after an application of the transitive case of the operation `SimplifyMatching`.

Invariant 7.3. *Let u and v be the poles of a P-node μ in $\mathcal{I}$ and let B_u denote a shared block around $o(u)$ in G such that edges of $r_u(B_u)$ are contained in at least two virtual edges of μ. Then there exists a shared block B_v around $o(v)$ such that $r_v(B_v) = \delta_{uv}(r_u(B_u))$, where δ_{uv} maps the edges incident to u to the edges incident to v according to the virtual edges in μ.*

In the initial Synchronized Planarity instance $\mathcal{I}_{\text{init}}$, a P-node μ can either appear in one of the subgraphs $H^{\textcircled{1}}$ and $H^{\textcircled{2}}$ corresponding to $G^{\textcircled{1}}$ and $G^{\textcircled{2}}$, or in the subgraph X that synchronizes the corresponding shared edges between $H^{\textcircled{1}}$ and $H^{\textcircled{2}}$. In the latter case, Invariant 7.3 holds, because both poles of μ correspond to the same vertex of the

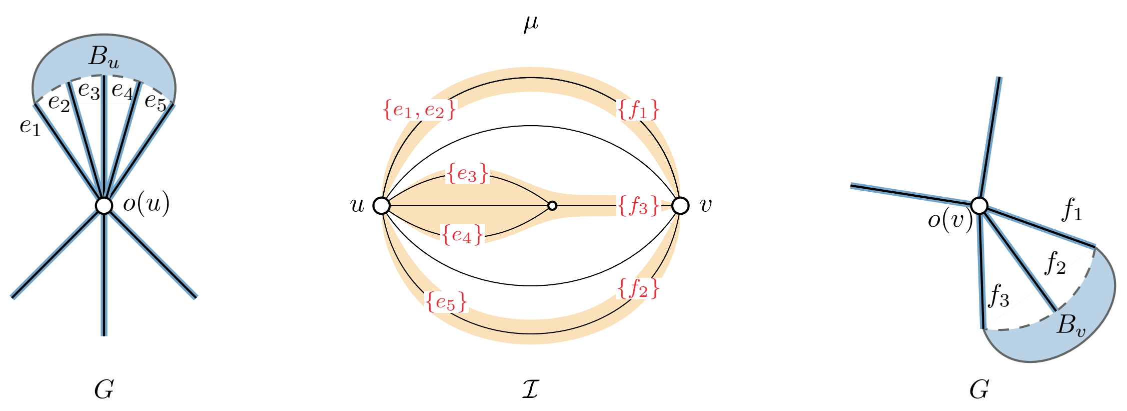


Figure 7.10: An illustration of Invariant 7.3. The edges incident to u in $\mathcal{I}$ that correspond to the shared block B_u are contained in the same three virtual edges of μ as the edges incident to v that correspond to the shared block B_v.

shared graph and the parallel edges of μ are exactly the corresponding shared edges. Now consider the former case. Without loss of generality, assume μ is contained in $H^{①}$ with poles $u^{①}$ and $v^{①}$. Now let B denote a shared block around u in G such that edges of B incident to $u^{①}$ are contained exactly in the set τ of virtual edges in μ with $|\tau| > 1$. Then the edges of B incident to $v^{①}$ must also be contained exactly in the virtual edges of τ, because otherwise, $u^{①}$ or $v^{①}$ would split B. Let B_u denote the edges of B incident to $u^{①}$ and let B_v denote the edges of B incident to $v^{①}$. Since the mappings $r_{u^{①}}$ and $r_{v^{①}}$ map the shared edges incident to u and v in G to their corresponding versions incident to $u^{①}$ and $v^{①}$, B_u and B_v therefore satisfy Invariant 7.3.

The next invariant states that the connectivity of the shared edges in the Synchronized Planarity instance does not decrease compared to the shared graph G, which will be helpful for the operation `EncapsulateAndJoin`.

Invariant 7.4. *Let u be an arbitrary vertex of $\mathcal{I}$ and let B denote a shared block around $o(u)$ in G. Then either $r_u(B) = \perp$, or all edges of $r_u(B)$ belong to a single block in $\mathcal{I}$.*

If u is contained in $H^{①}$ or $H^{②}$ then all edges incident to u that belong to a single block of the shared graph must also belong to a single block $H^{①}$, because the connectivity in the exclusive graphs only increases. If u is contained in the subgraph X, then all edges corresponding to shared edges are parallel edges in the corresponding bond and thus also belong to a single block. Invariant 7.4 therefore holds in $\mathcal{I}_{\text{init}}$.

The next invariant ensures that all cutvertices of the Synchronized Planarity instance are rather simple, which also helps us handle the operation `EncapsulateAndJoin`.

Invariant 7.5. *Let v be an arbitrary vertex of $\mathcal{I}$. If v is a cutvertex, then all split components of v, except for at most one, consist of a single degree-1 vertex.*

Since we require both exclusive graphs of our SEFE instance to be biconnected, $H^{①}$ and $H^{②}$ both contain no cutvertices. For a vertex v of the subgraph X, recall that all edges incident to x that correspond to shared edges are parallel edges in the corresponding bond, all other edges are incident to degree-1 vertices. Therefore, Invariant 7.5 holds in $\mathcal{I}_{\text{init}}$.

For a vertex u of the Synchronized Planarity instance, the last invariant makes the following statement. If we remove all partial constraints from our Synchronized Planarity instance, then any planar embedding of the resulting instance must also induce

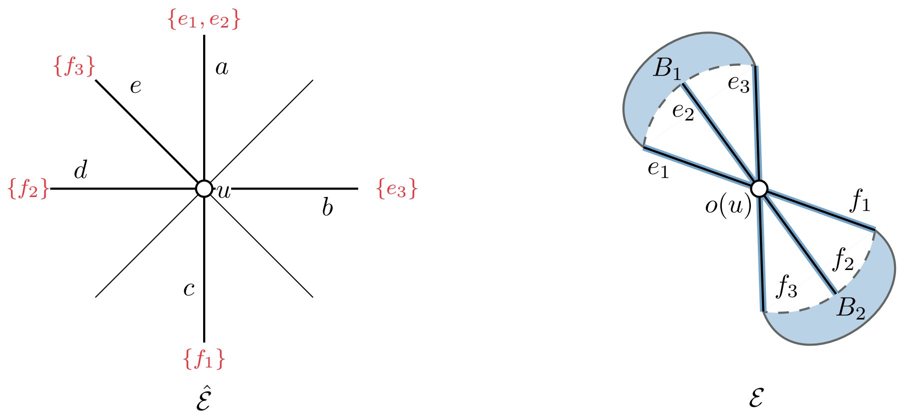


Figure 7.11: An illustration of Invariant 7.6. The shown embedding $\hat{\mathcal{E}}$ of the Synchronized Planarity instance $\hat{\mathcal{I}}$ induces the edge ordering $\hat{\mathcal{E}}(u)[O_u] = \langle a, b, c, d, e\rangle$ around u. The embedding $\mathcal{E}$ of the shared graph G induces the edge ordering $\mathcal{E}(o(u)) = \langle e_1, e_2, e_3, f_1, f_2, f_3\rangle$ around $o(u)$. Note that $r_u(\langle e_1, e_2, e_3, f_1, f_2, f_3\rangle) = \langle a, b, c, d, e\rangle$. If $\hat{\mathcal{I}}$ allows the edges b and c to be swapped in $\hat{\mathcal{E}}$, then the corresponding edges e_3 and f_1 must also be swapped in $\mathcal{E}$. This would yield a non-planar embedding of G, because edges of the blocks B_1 and B_2 would alternate. By Invariant 7.6, swapping b and c in $\hat{\mathcal{E}}$ therefore yields an embedding that does not satisfy $\hat{\mathcal{I}}$.

a valid edge ordering for the shared edges around $o(u)$ in G, subject to the mapping r_u. This invariant is particularly powerful in combination with the previous invariants that refer to shared blocks around $o(u)$ in G. If the vertex u allows an arbitrary order of its incident edges, this invariant will let us draw conclusions for the blocks around $o(u)$ in G, because edges of different blocks cannot alternate arbitrarily in a planar embedding. Together with Invariant 7.1, this will later allow us to restrict the number of partial constraints in a single pipe, which helps us solve case i of the operation `SimplifyMatching`. See Figure 7.11 for an example illustrating this invariant.

Invariant 7.6. *Let u be an arbitrary vertex of $\mathcal{I}$ and let $\hat{\mathcal{I}}$ denote the Synchronized Planarity instance obtained by removing all partial constraints from all pipes in $\mathcal{I}$. Let O denote the set of shared edges incident to $o(u)$ in G and let $O_u = r_u(O)$ denote the set of the corresponding edges incident to u in $\hat{\mathcal{I}}$. For any embedding $\hat{\mathcal{E}}$ of $\hat{\mathcal{I}}$ there also exists a planar embedding $\mathcal{E}$ of the shared graph G with $\hat{\mathcal{E}}(u)[O_u] = r_u(\mathcal{E}(o(u)))$.*

Since the standard Synchronized Planarity reduction ensures consistent edge orderings, any planar embedding of $\hat{\mathcal{I}}_{\text{init}}$ induces a planar embedding of the shared graph G. Since the mapping r maps all shared edges around vertices of G to the corresponding shared edges around the vertices in $\hat{\mathcal{I}}_{\text{init}}$, Invariant 7.6 must therefore hold in $\mathcal{I}_{\text{init}}$.

Note that Invariant 7.1 and Invariant 7.4 together imply that every partial constraint only constrains edges of a single block of $\mathcal{I}$. We state this in the following corollary.

Corollary 7.4. *Let $\rho = (u, v, \varphi_{uv}, \mathcal{R})$ be a pipe of $\mathcal{I}$ and let $R \in \mathcal{R}$ be a partial constraint. Then all edges incident to u that are constrained by R belong to a single block.*

7.4 Operations for Constrained Pipes

As the next step, we modify the Synchronized Planarity operations `PropagatePQ`, `EncapsulateAndJoin`, and `SimplifyMatching` for constrained pipes. Because this is a difficult task in general, we rely on the invariants we defined in Section 7.3 to restrict the possible cases that can arise. We also show that the invariants we defined remain intact after applying one of these operations.

7.4.1 Constrained PropagatePQ

In order to handle constrained pipes with the operation `PropagatePQ`, we need to break down partial constraints to the individual inner nodes of the embedding tree of a vertex. Given a PC-tree T and an additional PC-tree P with $L(P) \subseteq L(T)$ defining a partial constraint over $L(T)$, we want to compute a PC-tree N_x for every x in the set $I(T)$ of inner nodes of T, such that N_x defines a partial constraint for the edges incident to x in T. In other words, we want to break down the partial constraint P to partial constraints for each individual inner node of T. In order to simplify the notation, we root each PC-tree at an arbitrary inner node and denote by T_x the subtree of T rooted at x. Let T' denote the tree obtained by projecting T to the leaves in P, i.e., $T' := \texttt{Project}(T, L(P))$. As the first step, we obtain $S := \texttt{Intersect}(T', P)$ by intersecting T' and P, thus the resulting PC-tree S contains all restrictions of both T' and P; see Figure 7.12 for an example. Additionally, if a C-node q' of T' and a C-node q'' of P coincide in a C-node q in S, we synchronize their rotation using equations $\psi(q) = \psi(q') = \psi(q'')$. We call these equations *rotation constraints*. Let r denote the root of T and let $c(r)$ denote the set of children of r. For each child $c \in c(r)$, the set $A(c) := L(T_c) \cap L(P)$ is the set of leaves of the subtree T_c that are part of the partial constraint P (Note that $A(c)$ might be empty). Because $A(c)$ is consecutive in T', $A(c)$ is also consecutive in the intersection S and we can apply the operation $(K_c, S') := \texttt{Split}(S, A(c))$. If a C-node q from S is split into C-nodes q' in K_c and q'' in S', we set $\psi(q) = \psi(q') = \psi(q'')$ to ensure that q, q', and q'' are flipped in compatible ways. Let N_r denote the PC-tree obtained by repeatedly applying this split operation to the resulting PC-tree S' for each $c \in c(r)$, thus N_r is the desired broken-down partial constraint for the root r of T. Subsequently, recursively executing this procedure for K_c for each non-leaf $c \in c(r)$ eventually yields the desired partial constraint N_x for every inner node $x \in I(T)$. Figure 7.12 gives an example for this procedure.

Lemma 7.5. *Let ϕ denote an order of the leaves of T that satisfies T. Then ϕ satisfies the partial constraint P if and only if ϕ satisfies the divided partial constraint N_x for all inner nodes $x \in I(T)$ together with their rotation constraints.*

Proof. Let ϕ denote an order of the leaves of T that satisfies both T and P. Then clearly, $\phi[L(P)]$ satisfies $S = \texttt{Intersect}(\texttt{Project}(T, L(P)), P)$. Let r be the root of T with children $c_1, \dots, c_j$ and let N_r be the corresponding partial constraint for r. Because N_r is obtained using a sequence of `Split` operations on S, it holds, by the definition of the `Split` operation, that $\phi[A(c_1) \to a_1, \cdots, A(c_j) \to a_j]$ satisfies N_r. Since, for any $x \in I(T)$, N_x can also be obtained by rooting T at x, the same holds for N_x for all $x \in I(T)$. Furthermore, because the rotation constraints only synchronize the rotation of C-nodes that stem from the same C-node in S and since ϕ must satisfy that C-node, $\phi[A(c_1) \to a_1, \cdots, A(c_j) \to a_j]$ satisfies all divided partial constraints N_x for all $x \in I(T)$ together with the rotation constraints between their C-nodes.

Conversely, let ϕ be an order that satisfies N_x for all $x \in I(T)$ together with their rotation constraints. In a leaf-to-root BFS-order of T, compute $N'_x := \texttt{Merge}(N_x, N'_{c_1}, \cdots, N'_{c_j})$, where $c_1, \dots, c_j$ are the children of x in T. For root r of T, modify N'_r as follows. For any two adjacent C-nodes q_1 and q_2 in N'_r with rotation constraints $\psi(q_1) = \psi(q_2)$, merge q_1 and q_2 accordingly. Denote the resulting PC-tree by $\hat{S}$. Because ϕ respects the rotation constraints between the partial constraints, ϕ satisfies $\hat{S}$. The only difference between S and $\hat{S}$ is that a P-node of S may be split into adjacent P-nodes in $\hat{S}$. This means that the

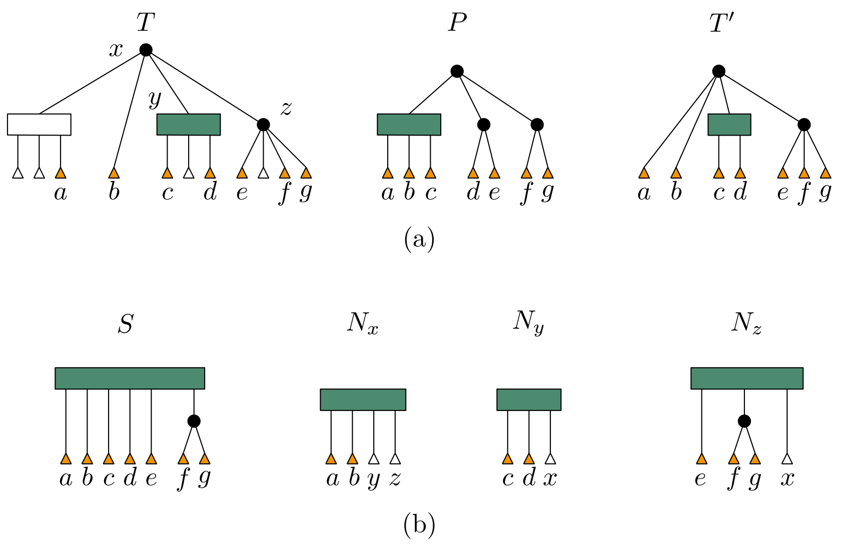


Figure 7.12: (a) An embedding tree T together with a partial constraint P and the projection $T' = \texttt{Project}(T, L(P))$ of T to the leaves in P. (b) The resulting intersection $S = \texttt{Intersect}(T', P)$ of T' and P, together with the divided partial constraints N_x, N_y, and N_z corresponding to the inner nodes x, y, and z in T. The rotation of all green C-nodes is synchronized using rotation constraints.

restrictions of S are a subset of the restrictions of $\hat{S}$ and since ϕ satisfies $\hat{S}$, ϕ therefore also satisfies S and thus ϕ also satisfies the partial constraint P. □

This powerful tool allows us to break down a partial constraint to the individual inner nodes of an embedding tree. Using this, we modify the operation `PropagatePQ` for constrained pipes as follows; see Figure 7.13. Let $\rho = (u, v, \varphi_{uv}, \mathcal{R})$ be a constrained pipe where u is a blockvertex with a non-trivial embedding tree T_u. First, we apply the standard version of `PropagatePQ`, replacing u (respectively v) with the embedding tree T_u (respectively the mirrored version T'_u of T_u). For each Q-node appearing as q in T_u and as q' in T'_u we add a Q-constraint between q and q'. Next, for each partial constraint $P_i \in \mathcal{R} = \{P_1, \dots, P_j\}$, we break down P_i to the inner nodes $I(T_u)$ of T_u using the algorithm described above, resulting in a new partial constraint N^i_x for each inner node $x \in I(T_u)$. If this operation fails, we reduce to a trivial no-instance. Repeating this for every partial constraint $P_i \in \mathcal{R} = \{P_1, \dots, P_j\}$ yields a new set $\mathcal{R}_x = \{N^1_x, \dots, N^j_x\}$ for each inner node $x \in I(T_u)$. For every P-node appearing as x in T_u and as x' in T'_u, we therefore create the new pipe $\rho_x = (x, x', \varphi_{xx'}, \mathcal{R}_x)$, where $\varphi_{xx'}$ naturally maps the edges incident to x in T_u to the corresponding edges incident to x' in T'_u. We call this operation `Constrained PropagatePQ`.

We now describe how the mappings o and r are updated after the operation `Constrained PropagatePQ`. For each inner node x of the embedding tree T_u we used to replace vertex u, we set $o(x) = o(u)$. For every shared edge e incident to $o(u)$, we set $r_x(e) = e'$, where e' is the edge incident to x whose subtree in T_u contains the edge $r_u(e)$ as a leaf; see Figure 7.14 for an example. The updated mappings for the other matched vertex v are analogous.

Lemma 7.6. *Applying the operation `Constrained PropagatePQ` operation to an instance $\mathcal{I}$ yields an equivalent instance $\mathcal{I}'$ where all invariants remain intact.*

Proof. The correctness of `Constrained PropagatePQ` follows directly from the correctness of `PropagatePQ` [BFR20, Lemma 5] in combination with Lemma 7.5.

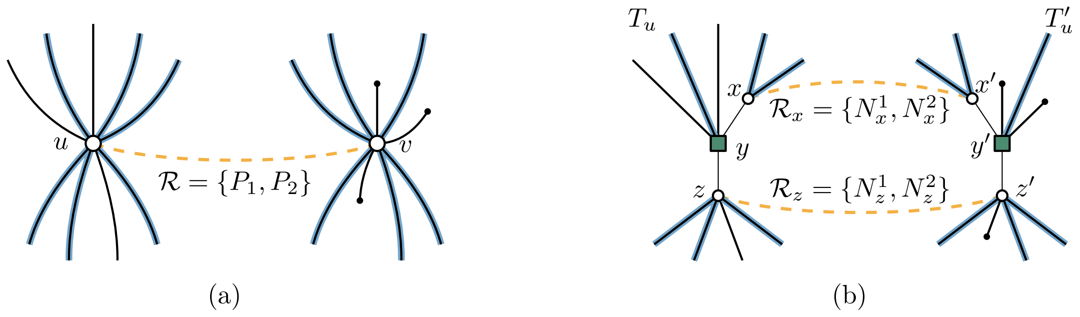

u
v
$\mathcal{R} = \{P_1, P_2\}$
(a)
T_u
T'_u
x
x'
y
y'
z
z'
$\mathcal{R}_x = \{N_x^1, N_x^2\}$
$\mathcal{R}_z = \{N_z^1, N_z^2\}$
(b)

Figure 7.13: A constrained pipe matching a block vertex u and a cutvertex v with partial constraints $\mathcal{R} = \{P_1, P_2\}$ (a). Note that the edge ordering around v is mirrored with respect to u. The resulting equivalent instance (b) obtained after applying the operation `Constrained PropagatePQ`, which replaces u and v with the embedding tree T_u and its mirrored version T'_u, respectively. The partial constraints in $\mathcal{R}$ are broken down to the individual nodes of the embedding tree and used as constraints for the newly added pipes between P-nodes of T_u and T'_u.

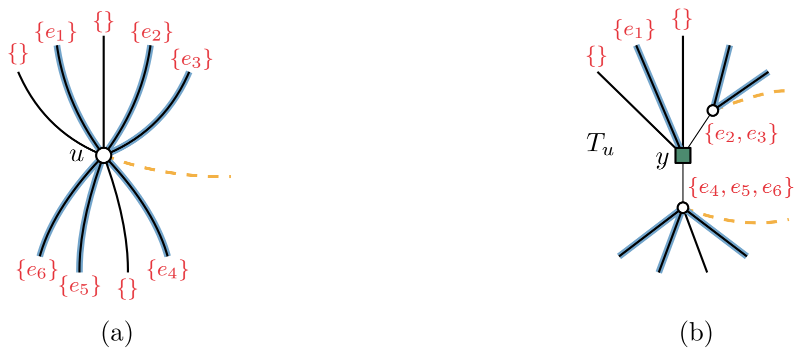


Figure 7.14: (a) A matched vertex u. The red sets indicate the shared edges incident to $o(u)$ in G that each edge incident to u corresponds to. (b) After replacing u with the embedding tree T_u, the edges incident to the inner node y are annotated with all edges contained in the corresponding subtree.

We first show that Invariant 7.1 holds for the newly created vertices, which all correspond to inner nodes of the embedding tree T_u. Essentially, we have to show that our updated mapping r is consistent with the way we break down the partial constraints to the inner nodes of T_u. Let R denote a partial constraint of $\mathcal{R}$ and let x denote an inner node of T_u. Let B denote the shared block around $o(u)$ in G that R corresponds to by Invariant 7.1, hence $L(R) = r_u(B)$. Our updated mapping ensures that the set $r_x(B)$ contains exactly the edges incident to x whose subtrees in T_u contain edges of $r_u(B) = L(R)$ as leaves. By construction, the new partial constraint N_x derived from R contains as leaves exactly the edges incident to x whose subtrees in T_u contain an edge of $L(R) = r_u(B)$ as a leaf. Since $L(R) = r_u(B)$, it therefore follows that $L(N_x) = r_x(B)$ and thus Invariant 7.1 remains intact.

Since all new pipes match vertices corresponding to the same inner node of T_u, the mapping r is updated symmetrically for both endpoints of these pipes. Since Invariant 7.2 holds for ρ in $\mathcal{I}$, it therefore also holds for all newly created pipes in $\mathcal{I}'$.

Now we want to show that Invariant 7.3 holds for all P-nodes in $\mathcal{I}'$. Let x be an inner node of T_u such that x is the pole of a P-node μ' with twin pole y in $\mathcal{I}'$ after replacing u with T_u. Since `Constrained PropagatePQ` only increases the connectivity of the graph, this means that u was also already the pole of a P-node μ with twin pole y in $\mathcal{I}$. Every virtual edge of μ' thus consists of one or more virtual edges of μ. Since we defined the mapping r_x for the new pole accordingly, and since Invariant 7.3 holds for μ in $\mathcal{I}$, Invariant 7.3 therefore also holds for μ' in $\mathcal{I}'$.

Because `Constrained PropagatePQ` only increases the connectivity of the graph, it is easy to see that Invariant 7.4 and Invariant 7.5 also remain intact.

Finally, we still need to show that Invariant 7.6 remains intact. Let $\hat{\mathcal{I}}$ denote the Synchronized Planarity instance obtained from $\mathcal{I}$ by ignoring all partial constraints of $\mathcal{I}$. For the vertex u of pipe ρ, let O denote the set of shared edges incident to $o(u)$ in G, and let $O_u = r_u(O)$ denote the set of the corresponding edges incident to u in $\hat{\mathcal{I}}$. Let $\hat{\mathcal{E}}$ denote a planar embedding of $\hat{\mathcal{I}}$ and let $\hat{\phi} := \hat{\mathcal{E}}(u)[O_u]$ be the ordering of the edges in O_u induced by $\hat{\mathcal{E}}$. By Invariant 7.6, there exists an embedding $\mathcal{E}$ of G with $\phi := \mathcal{E}(o(u))$ such that $r_u(\phi) = \hat{\phi}$. Let $\hat{\mathcal{I}}'$ denote the reduced instance, again ignoring all partial constraints, and let x denote an inner node of the embedding tree T_u we used to replace vertex u. Since the operation `PropagatePQ` is correct [BFR20, Lemma 5], any embedding $\hat{\mathcal{E}}'$ of $\hat{\mathcal{I}}'$ can be obtained from $\hat{\mathcal{E}}$ by replacing vertex u with a suitable embedding of the tree T_u in $\hat{\mathcal{E}}$. Let x denote an inner node of T_u and let $O_x = r_x(O)$ denote the edges incident to x that correspond to the shared edges incident to $o(x) = o(u)$. Since the mapping r_x groups the edges of O_u according to the subtrees of T_u incident to x (see the definition above) and since $o(x) = o(u)$, it is $\hat{\mathcal{E}}'(x)[O_x] = r_x(\mathcal{E}(o(x))) = r_x(\mathcal{E}(o(u))) = r_x(\phi)$. Therefore, Invariant 7.6 holds for all newly created vertices. □

7.4.2 Constrained EncapsulateAndJoin

Now we handle constrained pipes $\rho = (u, v, \varphi_{uv}, \mathcal{R})$ between two cutvertices u and v using a modified version of `EncapsulateAndJoin` called `Constrained EncapsulateAndJoin`. First apply the operation `Encapsulate` to u (respectively to v). By Invariant 7.5, u has at most one split component S_u that consists of more than one edge, all other split components are degree-1 vertices. As a simplification, we do not split the degree-1 vertices incident to u. The operation `Encapsulate` thus only creates two new matched vertices w_{S_u} and w'_{S_u} corresponding to the split component S_u; see Figure 7.15. By Corollary 7.4, all partial constraints of $\mathcal{R}$ in the pipe ρ that are non-trivial (i.e., they constrain at least three edges) only constrain edges belonging to S_u. We can therefore move all partial constraints in $\mathcal{R}$

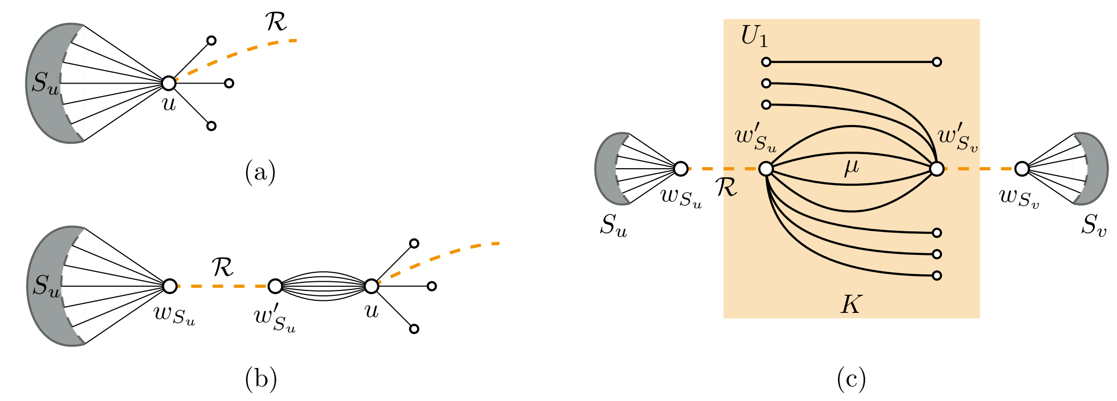


Figure 7.15: (a) A cutvertex u matched via a constrained pipe with partial constraints $\mathcal{R}$. By Invariant 7.5, u has at most one split component S_u that consists of more than one edge, all other split components are degree-1 vertices. (b) The result of encapsulating u. As a simplification, the degree-1 vertices are not split. The partial constraints in $\mathcal{R}$ are moved to the new pipe matching w'_{S_u} and w_{S_u}, which is possible by Corollary 7.4. (c) An example illustrating the resulting bipartition K after joining u with its partner v.

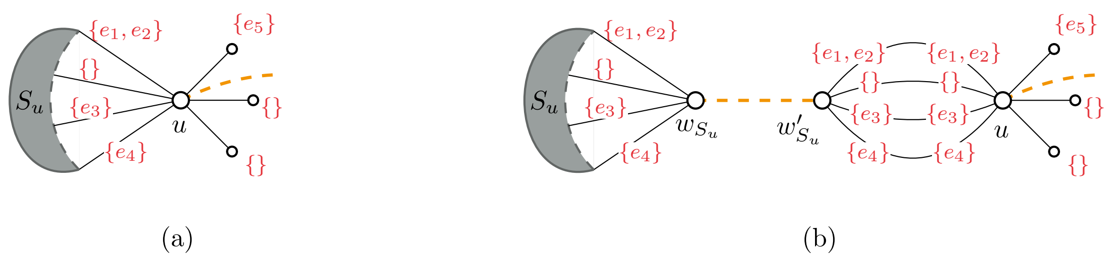


Figure 7.16: (a) A matched cutvertex u with a single split component S_u consisting of more than one edge. The edges incident to u are annotated with the shared edges of $o(u)$ they correspond to. (b) The mappings $r_{w_{S_u}}$ and $r_{w'_{S_u}}$ for the new vertices w_{S_u} and w'_{S_u} after applying the operation `Encapsulate` to u. Note that $r_{w_{S_u}}(e_5) = r_{w'_{S_u}}(e_5) = \perp$, because the edge incident to u corresponding to e_5 does not belong to the split component S_u.

to the new pipe matching w'_{S_u} and w_{S_u} and subsequently, the pipe ρ no longer contains partial constraints. We can therefore apply the operation `Join` to the pipe ρ as in the standard operation.

We update the mappings o and r after the operation `Encapsulate` as follows. We set $o(w_{S_u}) = o(w'_{S_u}) = o(u)$. Let δ denote the natural mapping of the edges of S_u incident to u to the edges incident to w_{S_u}. For every edge e incident to $o(u)$, we set $r_{w_{S_u}}(e) = \delta(r_u(e))$, if $r_u(e)$ is contained in S_u, and $r_{w_{S_u}}(e) = \perp$ otherwise. For the vertex w'_{S_u} that is matched with w_{S_u} via a pipe with mapping φ, we set $r_{w'_{S_u}}(e) = \varphi(r_{w_{S_u}}(e))$; see Figure 7.16 for an example. Since the operation `Join` does not create new vertices, o and r do not have to be updated afterwards.

Lemma 7.7. *Applying the operation `Constrained EncapsulateAndJoin` to an instance $\mathcal{I}$ yields an equivalent instance $\mathcal{I}'$ where all invariants remain intact.*

Proof. We have already shown above that it is correct to move the partial constraint in $\mathcal{R}$ to the new pipe matching w'_{S_u} and w_{S_u} after the operation `Encapsulate`. Subsequently, the pipe ρ no longer contains partial constraints, thus the correctness of

Constrained EncapsulateAndJoin follows from the correctness of the standard operation EncapsulateAndJoin [BFR20, Lemma 4].

It therefore only remains to show that the invariants remain intact. First consider the operation Encapsulate. Since we only encapsulate the split component S_u incident to u, it is clear that Invariant 7.5 remains intact. Recall that we have $o(w_{S_u}) = o(w'_{S_u}) = o(u)$ and that the mappings $r_{w_{S_u}}$ and $r_{w'_{S_u}}$ map the edges incident to $o(u)$ in G to the edges incident to w_{S_u} and w'_{S_u} the same way the mapping r_u maps them to u. The only difference is that, for any edge e incident to $o(u)$ where $r_u(e)$ does not belong to the split component S_u, it is $r_{w_{S_u}}(e) = r_{w'_{S_u}}(e) = \perp$. The new mappings are therefore basically copies of the mapping r_u, only the shared edges that do not "belong" to S_u are mapped to $\perp$. Invariant 7.4 therefore remains intact and Invariant 7.2 holds for the new pipe matching w'_{S_u} and w_{S_u}. For the same reason, Invariant 7.3 also holds for the new bond with poles w_{S_u} and u[1]. Because Invariant 7.1 was correct for ρ in $\mathcal{I}$, it also holds for the pipe matching w'_{S_u} and w_{S_u} in $\mathcal{I}'$. Observe that in any embedding of $\mathcal{I}'$ the edges incident w_{S_u} and w'_{S_u} have the same (possibly reversed) cyclic order as the corresponding edges incident to u. Because Invariant 7.6 holds for u in $\mathcal{I}$, and since, as argued above, the mappings $r_{w_{S_u}}$ and $r_{w'_{S_u}}$ are well-defined, Invariant 7.6 therefore also holds for w_{S_u} and w'_{S_u}.

Now consider the subsequent application of the operation Join. It is clear that Invariant 7.5 still holds. Since we do not alter the mappings o and r in this step, Invariant 7.4 and Invariant 7.6 also remain intact. Because no new pipes are created, Invariant 7.1 and Invariant 7.2 still hold. Note that Join might create new P-nodes in the resulting bipartition K in the form of a trivial bond μ with poles w'_{S_u} and w'_{S_v}; see Figure 7.15 for an example. Note that the operation Join ensures that the bond μ matches the between w'_{S_u} and w'_{S_v} the same way as the pipe ρ matches the corresponding edges between u and v in $\mathcal{I}$. Since Invariant 7.2 holds for pipe ρ in $\mathcal{I}$, Invariant 7.3 therefore holds for the bond μ in $\mathcal{I}'$ □

Now the only operation that remains to be adjusted for constrained pipes is SimplifyMatching. For standard pipes, the three cases of SimplifyMatching are quite straightforward and are therefore grouped together. In the presence of constrained pipes, these cases become significantly more involved. For this reason, we split the three cases into different operations and show their correctness separately.

7.4.3 Transitive Constrained SimplifyMatching

Let $\rho = (u, u', \varphi_{uu'}, \mathcal{R}_1)$ be a pipe of the Synchronized Planarity instance $\mathcal{I}$, where one of the matched vertices, say u, has a trivial embedding tree. This means that u is a pole of a bond μ; let v denote the other pole of μ. We first consider case iii of SimplifyMatching, where v is part of a constrained pipe $\rho' = (v, v', \varphi_{vv'}, \mathcal{R}_2)$ with $v' \neq u$; see Figure 7.17c. In this case, we remove ρ and ρ' and create a new pipe $\rho^* = (u', v', \varphi_{u'v'}, \mathcal{R}_1 \cup \mathcal{R}_2)$, where $\varphi_{u'v'} := \varphi_{vv'} \circ \delta_{uv} \circ \varphi_{u'u}$ and where δ_{uv} bijectively maps the edges of u to the edges of v according to the bond μ. We call this operation Transitive Constrained SimplifyMatching.

Lemma 7.8. *Applying the operation Transitive Constrained SimplifyMatching to an instance $\mathcal{I}$ yields an equivalent instance $\mathcal{I}'$ where all invariants remain intact.*

Proof. Let $\mathcal{E}$ denote an embedding of the original instance $\mathcal{I}$ that satisfies ρ and ρ', i.e., $\mathcal{E}(u) = \overline{\varphi_{u'u}(\mathcal{E}(u'))}$ and $\mathcal{E}(v') = \overline{\varphi_{vv'}(\mathcal{E}(v))}$. Additionally, $\mathcal{E}(u)$ satisfies the partial

[1]Recall that we defined $\delta(\perp) = \perp$ for any mapping δ, thus the statements of the invariants are also well-defined after setting $r_{w_{S_u}}(e) = r_{w'_{S_u}}(e) = \perp$ for some edges incident to $o(u)$.

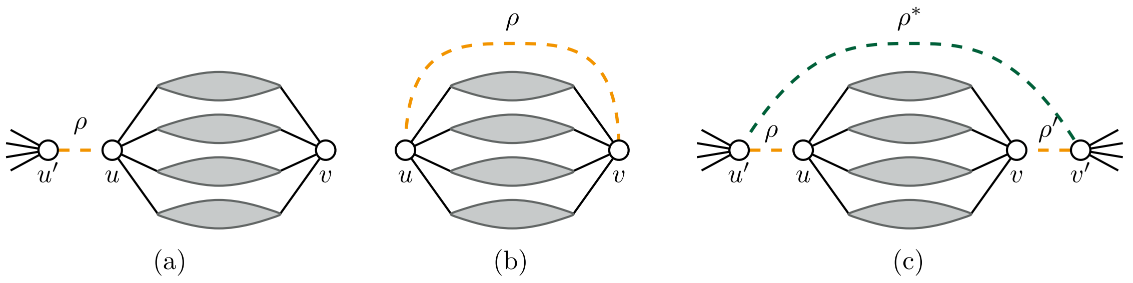


Figure 7.17: The three different cases of the operation `SimplifyMatching`.

constraints in $\mathcal{R}_1$ and $\mathcal{E}(v')$ satisfies $\mathcal{R}_2$. Because the embedding is planar, it is $\mathcal{E}(v) = \overline{\delta_{uv}(\mathcal{E}(u))}$ and therefore [BFR20, Lemma 6]

$$\begin{aligned}\mathcal{E}(v') &= \overline{\varphi_{vv'}(\overline{\delta_{uv}(\mathcal{E}(u))})} = \varphi_{vv'}(\delta_{uv}(\mathcal{E}(u))) \\ &= \varphi_{vv'}(\delta_{uv}(\overline{\varphi_{u'u}(\mathcal{E}(u'))})) = \overline{\varphi_{vv'} \circ \delta_{uv} \circ \varphi_{u'u}(\mathcal{E}(u'))} = \overline{\varphi_{u'v'}(\mathcal{E}(u'))}.\end{aligned}$$

Therefore, $\mathcal{E}(v')$ satisfies ρ^* with partial constraints $\mathcal{R}_1 \cup \mathcal{R}_2$ and thus $\mathcal{E}$ is also an embedding of the reduced instance $\mathcal{I}'$.

To obtain an embedding $\mathcal{E}$ of the original instance from an embedding $\mathcal{E}'$ of the reduced instance, set $\mathcal{E}(u) = \overline{\varphi_{u'u}(\mathcal{E}(u'))}$ and $\mathcal{E}(v) = \overline{\varphi_{v'v}(\mathcal{E}(v'))}$. Since both $\mathcal{E}(u')$ and $\mathcal{E}(v')$ satisfy the partial constraints $\mathcal{R}_1 \cup \mathcal{R}_2$, both pipes ρ and ρ' are satisfied in $\mathcal{E}$. Additionally, $\mathcal{E}$ is planar [BFR20, Lemma 6].

It remains to show that the operation retains all invariants. Since `Transitive Constrained SimplifyMatching` does not alter the graph structure or the mappings r and o, Invariants 7.3–7.6 are not affected.

To show that Invariant 7.2 holds for the new pipe ρ^* in $\mathcal{I}'$, first note that Invariant 7.2 holds for the old pipes ρ and ρ' in the original instance $\mathcal{I}$, and that Invariant 7.3 holds for μ in $\mathcal{I}$. Let $B_{u'}$ denote a shared block around $o(u')$ in the shared graph G. We can use these invariants successively for ρ, μ, and ρ', to find a block $B_{v'}$ around $o(v')$ in G with $r_{v'}(B_{v'}) = \varphi_{vv'} \circ \delta_{uv} \circ \varphi_{u'u}(r_{u'}(B_{u'})) = \varphi_{u'v'}(r_{u'}(B_{u'}))$. Therefore, Invariant 7.2 holds in $\mathcal{I}'$.

It only remains to show that Invariant 7.1 remains intact. Let $R \in \mathcal{R}_1 \cup \mathcal{R}_2$ be a partial constraint of the new pipe ρ^*. First consider the case where R originates from the pipe ρ matching u' and u, i.e., $R \in \mathcal{R}_1$. By Invariant 7.1, there exists a block $B_{u'}$ around $o(u')$ in the shared graph G, such that $L(R) = r_{u'}(B_{u'})$[2], thus we are immediately done. Now consider the case where $R \in \mathcal{R}_2$. Then we can use Invariant 7.2 on the pipe ρ' to find a block $B_{v'}$ around $o(v')$ such that $L(R) = r_{v'}(B_{v'})$. Since we have already shown above that Invariant 7.2 holds for the pipe ρ^* in $\mathcal{I}'$, we can use Invariant 7.2 in ρ^* to find a block $B_{u'}$ around $o(u')$ with $r_{u'}(B_{u'}) = \varphi^{-1}_{u'v'}(r_{v'}(B_{v'}))$. Hence the partial constraint R constrains exactly the edges of $B_{u'}$ incident to u' and thus Invariant 7.2 remains intact. □

7.4.4 Trivial Constrained SimplifyMatching

As the next operation, we modify case i of the standard operation `SimplifyMatching`. In this case, we have a pipe $\rho = (u, u', \varphi_{uu'}, \mathcal{R})$ and u is a pole of a trivial bond μ with twin pole v, where v is unmatched; see Figure 7.17a. In the standard operation, we can simply remove ρ from the instance, because the bond μ can always mirror the edge orderings enforced by u' [BFR20, Lemma 6]. But in our case, we also have to ensure that these edge orderings enforced by u' are compatible with the partial constraints $\mathcal{R}$ of ρ.

[2]Note that pipes are bidirectional, hence we can also use the invariant in the reverse direction.

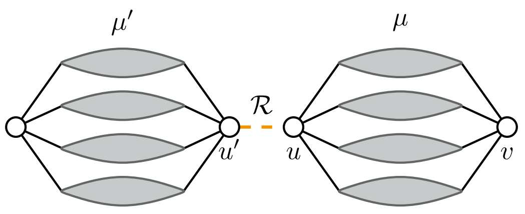


Figure 7.18: After eliminating cutvertices and if no other operation can be applied, every pipe satisfying the prerequisites of the operation `Trivial Constrained SimplifyMatching` matches the poles of two trivial bonds μ and μ'. Since the other poles of μ and μ' must be unmatched, only the partial constraints in $\mathcal{R}$ restrict the order of virtual edges in μ and μ'.

First consider the case where u' is a cutvertex. By Invariant 7.5, there is at most one split component S incident to u' that contains more than one edge. Since every partial constraint only restricts the edges of a single split component by Corollary 7.4, all non-trivial partial constraints restrict edges in S. Therefore, all other split components are not contained in any partial constraints and we remove them from u' and we split off their corresponding virtual edges in μ.

Now we assume that no other operation can be applied to any pipe in the Synchronized Planarity instance $\mathcal{I}$. Therefore, we now know that u' is also the pole of a different trivial bond μ', because if u' had a non-trivial embedding tree, we could still apply `Constrained PropagatePQ`. Additionally, the twin pole of u' in μ' is unmatched, because otherwise, we could still apply `Transitive Constrained SimplifyMatching` on μ'. The bonds μ and μ' therefore have the structure shown in Figure 7.18.

Since μ and μ' themselves carry no restrictions on the possible orderings of their virtual edges, we only need to verify that the partial constraints in $\mathcal{R}$ are compatible, i.e., they do not contradict one another. For arbitrary partial constraints, this is a very difficult problem, however, our invariants heavily restrict the possible cases. In fact, we will show that $\mathcal{R}$ can only contain partial constraints restricting the same leaf set, which makes the problem trivial. To show this, we will use Invariant 7.6. Recall that this invariant essentially states that any planar embedding of the instance $\hat{\mathcal{I}}$ obtained by removing all partial constraints from $\mathcal{I}$ must also induce a valid edge ordering for the shared edges around $o(u)$ in G, subject to the mapping r_u.

As an example illustrating this, consider again the situation shown in Figure 7.11. As stated in the caption, swapping the edges b and c around u would mean that the corresponding embedding of the shared graph G cannot be planar, because swapping the corresponding edges e_3 and f_1 around $o(u)$ would mean that edges of the blocks B_1 and B_2 alternate. We used this observation, in combination with Invariant 7.6, to conclude that $\hat{\mathcal{I}}$ does not allow the edges b and c incident to u to be swapped. However, in the context of `Trivial Constrained SimplifyMatching`, any ordering of the edges incident to u leads to a valid embedding of $\hat{\mathcal{I}}$ (see Figure 7.18), since $\hat{\mathcal{I}}$ contains no partial constraints. This means that the blocks around $o(u)$ cannot have the structure shown in Figure 7.11, because otherwise, Invariant 7.6 would be violated.

Lemma 7.9. *There is at most one shared block B around $o(u)$ in G such that $|r_u(B)| \geq 3$.*

Proof. Assume, for the sake of contradiction, that there are two distinct blocks B_a and B_b around $o(u)$ in G such that $|r_u(B_a)| \geq 3$ and $|r_u(B_b)| \geq 3$. Let $\hat{\mathcal{I}}$ denote the Synchronized Planarity instance obtained by removing all partial constraints from $\mathcal{I}$. Observe that, after removing the partial constraints in $\mathcal{R}$ from the pipe ρ, any cyclic ordering of the edges incident to u can be extended to a planar embedding of $\hat{\mathcal{I}}$; see Figure 7.18. We will use this fact, combined with Invariant 7.6, to find a contradiction.

Let $E(u)$ denote the set of edges incident to u, let O denote the shared edges incident to $o(u)$ in G and let $O_u = r_u(O) \subseteq E(u)$ denote the corresponding edges incident to u in $\hat{\mathcal{I}}$. Recall that $r_u(B_a) \subseteq E(u)$ and $r_u(B_b) \subseteq E(u)$. Also recall that $r_u(B_a)$ and $r_u(B_b)$ are not necessarily disjoint, because the mapping r_u may map several shared edges incident to $o(u)$ to the same edge incident to u. Let $\{a_1, a_2, a_3\} \subseteq r_u(B_a)$ and let $\{b_1, b_2, b_3\} \subseteq r_u(B_b)$. Pick an arbitrary planar embedding $\hat{\mathcal{E}}$ of $\hat{\mathcal{I}}$[3], where the cyclic ordering $\sigma = \hat{\mathcal{E}}(u)$ satisfies the constraint $\{a_1, b_1\} <_\sigma \{a_2, b_2\} <_\sigma \{a_3, b_3\}$. Note that we always find an ordering satisfying this constraint (possibly after renaming the elements in $\{b_1, b_2, b_3\}$), even if $r_u(B_a) = r_u(B_b)$. Now consider an order σ' of the shared edges incident to $o(u)$, such that $r_u(\sigma') = \sigma[S_u]$. Then $\{r_u^{-1}(a_1), r_u^{-1}(b_1)\} <_{\sigma'} \{r_u^{-1}(a_2), r_u^{-1}(b_2)\} <_{\sigma'} \{r_u^{-1}(a_3), r_u^{-1}(b_3)\}$ must hold for σ'. But this means that edges of B_a and B_b must alternate, which means that no planar embedding of G can induce the edge ordering σ' around $o(v)$, since B_a and B_b are distinct blocks. Hence we have a planar embedding $\hat{\mathcal{E}}$ of $\hat{\mathcal{I}}$, but there cannot exist a planar embedding $\mathcal{E}$ of the shared graph G with $\mathcal{E}(o(u)) = \hat{\mathcal{E}}(u)[O_u]$. This is a contradiction to Invariant 7.6. □

Now consider a partial constraint $R \in \mathcal{R}$. Note that any partial constraint with at most two leaves is always trivially satisfied and can therefore be removed from $\mathcal{R}$. Thus, for all remaining partial constraints in $\mathcal{R}$, it is $|L(R)| \geq 3$. Recall that Invariant 7.1 ensures that, for every partial constraint $R \in \mathcal{R}$, there exists a shared block B around $o(u)$ in the shared graph G, such that $L(R) = r_u(B)$. Since $|L(R)| = |r_u(B)| \geq 3$, this block B is unique by Lemma 7.9. This means that for every pair R_1, R_2 of partial constraints in $\mathcal{R}$, it is $L(R_1) = L(R_2) = r_u(B)$, hence all partial constraints in $\mathcal{R}$ have the same set of leaves. Since the intersection of two PQ-trees with the same leaf set is well-defined, we can now intersect all partial constraints in $\mathcal{R}$ and obtain a single partial constraint R'. If this operation fails, no ordering of the virtual edges in μ and μ' satisfies the partial constraints in $\mathcal{R}$, thus we reduce to a trivial no-instance. Since the virtual edges in μ and μ' can be ordered arbitrarily, we can simply pick any ordering that satisfies the partial constraint R'. Since $\mathcal{R}$ contains no other partial constraints, the constrained pipe ρ can therefore always be satisfied and we can remove it as in the standard operation. We call this operation `Trivial Constrained SimplifyMatching`.

7.4.5 Toroidal Constrained SimplifyMatching

The last remaining operation is case ii of the standard operation `SimplifyMatching`. In this case, we have a pipe $\rho = (u, v, \varphi_{uv}, \mathcal{R})$, where u and v are the poles of a trivial bond μ; see Figure 7.17b. For $x \in \{u, v\}$, let δ_x denote the bijection between the edges incident to x and the virtual edges of bond μ and let $\delta_{uv} = \delta_u^{-1} \circ \delta_v$ be the corresponding bijection between the edges incident to u and v, respectively. Let further $\pi = \varphi_{uv} \circ \delta_{uv}$ be the permutation of the virtual edges in μ defined by φ. As in the standard version of the operation, we first determine whether all cycles in π have the same length and we reduce to a trivial no-instance if this is not the case, because then μ has no planar embedding that satisfies ρ [BFR20]. The proof of the standard operation [BFR20, Lemma 6] uses the fact that all cycles of permutation π have the same length if and only if π is *order-preserving* with respect to some cyclic ordering O of the virtual edges in μ [BR16, Lemma 2.2], i.e., it is $\pi(O) = (O)$. If this is the case, an unconstrained pipe can simply be removed, because O immediately yields a planar embedding of μ that satisfies ρ. If ρ is a constrained pipe,

[3]As argued before, if $\hat{\mathcal{I}}$ is a yes-instance, then any ordering σ of $E(u)$ is induced by some planar embedding of $\hat{\mathcal{I}}$; see Figure 7.18.

however, O does not necessarily satisfy the partial constraints defined in the set $\mathcal{R}$ of ρ. Therefore, we need to determine whether there is an ordering O such that O satisfies all partial constraints in $\mathcal{R}$ and such that π is order-preserving with respect to O.

Corollary 7.10. *Let L be a set and let $\pi : L \to L$ be a permutation where all cycles in π have length m. Then π is order-preserving with respect to a cyclic order $O = l_1 \dots l_n$ of L with $n = |L|$ if and only if $l_i <_O l_j <_O l_k$ implies $\pi(l_i) <_O \pi(l_j) <_O \pi(l_k)$ for all $i, j, k \in \{1, \dots, n\}$*

We call the requirement from Corollary 7.10 the *order-preservation constraints* of π. To determine whether the trivial bond μ admits a planar embedding that satisfies pipe ρ, we have to find an ordering of its virtual edges that satisfies the partial constraints $\mathcal{R}$ of ρ and the order-preservation constraints of π. However, we also have to consider that there might be Q-nodes in partial constraints of $\mathcal{R}$ that are part of a synchronization constraint with Q-nodes of partial constraints in other pipes of the Synchronized Planarity instance. We therefore cannot simply combine the order-preservation constraints of π with the partial constraints of $\mathcal{R}$ and solve them independently from the remaining instance. Recall from the operation `Trivial Constrained SimplifyMatching` that we used the fact that the virtual edges of μ can be ordered arbitrarily to infer that all partial constraints must constrain the same set of leaves using Invariant 7.1 and Invariant 7.6. We were able to do this, because the virtual edges of μ can be ordered arbitrarily if we ignore the partial constraints in the pipes, thus Invariant 7.6 significantly restricts the possibilities. For the operation `Trivial Constrained SimplifyMatching`, this does not work as easily. Even if we ignore the partial constraints in ρ, the pipe ρ itself restricts the possible orderings of the virtual edges in μ via the permutation π, thus we can not as easily conclude that all partial constraints in $\mathcal{R}$ constrain the same set of edges.

Due to these considerations, we currently cannot solve this case in general. However, we believe that the following always holds.

Conjecture 7.11. *If $\mathcal{R}$ contains a partial constraint R with $|L(R)| \geq 3$ and all cycles of the permutation π have the same length, then all cycles of the permutation π have length 1.*

If Conjecture 7.11 holds, then the operation `Toroidal Constrained SimplifyMatching` is very simple. If not all cycles of π have the same length, then we can reduce to a trivial no-instance as described above. If the pipe ρ contains no partial constraints, or only trivial partial constraints with less than three leaves, we can proceed as in the standard operation. Otherwise, if $\mathcal{R}$ contains a partial constraint restricting at least three edges, then all cycles of π must have length 1 by Conjecture 7.11. If all cycles of the permutation π have length 1, then the permutation π is the identity, thus we can handle this case the same way as the operation `Trivial Constrained SimplifyMatching`.

We believe that Conjecture 7.11 holds for the following reason. The different operations most likely cannot create the structure consisting of a trivial bond μ and a pipe matching its poles out of thin air. Instead, the permutation π must have already been hidden in the instance previously, as a cyclic sequence $v_0, \dots, v_l, v_0$ of vertices, where l is an odd integer. The vertices v_i and v_{i+1} are matched via a pipe for even i and, for odd i, v_i and $v_{(i+1)\%l}$ belong to the same connected component. We believe that, if one of the pipes contains a non-trivial partial constraint, there exists a permutation π defined by the mappings of the pipes in combination with disjoint paths through the connected components, such that at least one cycle of π has length 1. If this can be stated as an invariant, then Conjecture 7.11 holds for the operation `Toroidal Constrained SimplifyMatching`, which makes the operation simple as described above. Since the partial constraints correspond to blocks of the shared graph and since corresponding shared edges are matched consistently via pipes, we believe that it is possible to show that this invariant holds in $\mathcal{I}_{\text{init}}$ and that it also remains intact throughout the algorithm.

7.5 Solving the Reduced Instance

Observe that our modified Synchronized Planarity operations make the same structural changes to the graph as the original operations, they only differ in the way they treat the partial constraints. Therefore, the analysis of the original Synchronized Planarity algorithm still applies and we obtain an equivalent pipe-free instance in polynomial time [BFR20]. Since a pipe-free instance also contains no partial constraints, the reduced instance can be solved using a 2-Sat instance obtained from the Q-constraints of the instance, as in the original algorithm [BFR20]. We remark that the additional partial constraints of our SEFE instance are implicitly encoded into this 2-Sat instance, because we added additional Q-constraints between Q-vertices when breaking down partial constraints in the operation `Constrained PropagatePQ`. Since we have shown in Section 7.4 that all our modified operations are correct, we finally get the following result.

Theorem 7.12. *The problem Simultaneous Embedding With Fixed Edges is FPT parameterized by the number k of connected components of the shared graph and the maximum degree Δ of the shared graph and can be solved in time $\mathcal{O}(((k+\Delta)!)^{4k^2} \cdot \text{poly}(n))$, if the following conditions apply:*

1. *both exclusive graphs are biconnected,*
2. *every pair of fixpoints is block-local, and*
3. *Conjecture 7.11 holds for the operation `Toroidal Constrained SimplifyMatching`.*

8. Polynomial-Time Algorithm for SEFE of two Biconnected Graphs with Maximum Degree 4

In this section, we construct a polynomial-time algorithm to solve SEFE when the two input graphs $G^{①}$ and $G^{②}$ are both biconnected and have maximum degree at most 4. Observe that this implies that the shared graph G also has maximum degree at most 4 and that the union graph $G^{\cup}$ has maximum degree at most 8. This extends previous results from algorithms by Schaefer [Sch13] and by Bläsius et al. [BKR18], which solve SEFE for non-trivial connected components of the shared graph if all these components have maximum degree 3.

In order to obtain this polynomial-time algorithm, we once again augment the Synchronized Planarity instance obtained from the linear-time reduction from SEFE [BFR20] (see Chapter 2 and Figure 2.3) with additional constraints enforcing consistent relative positions, similar to our approach in Chapter 7. This time, instead of deriving necessary and sufficient partial constraints from a set of auxiliary graphs, we aim to ensure consistent relative positions by incorporating the approach introduced by Bläsius et al. [BKR18] into Synchronized Planarity. Since the shared graph has no fixed embedding, relative positions of connected components of the shared graph cannot be expressed with respect to faces of the shared graph. Instead, Bläsius et al. [BKR18] express the position of a component D with respect to another component H by fixating the position of D relative to all cycles of a cycle basis $\mathcal{B}$ of H. For this purpose, assign a direction to each cycle $C \in \mathcal{B}$ and let the binary variable $\text{pos}_C(D)$ denote whether component D lies to the left or to the right of the edges of the directed cycle C. Bläsius et al. showed that, for any embedding $\mathcal{E}$ of H, assigning a value $\text{pos}_C(D)$ for each $C \in \mathcal{B}$ already uniquely defines the position of D in a face in $\mathcal{E}$ [BKR18, Theorem 8], if such a face exists.

Bläsius et al. [BKR18] use relative positions with respect to a cycle basis to ensure that every shared component is embedded in the same face of the shared graph in $G^{①}$ and $G^{②}$. To this end, they examine the SPQR-tree representation of both exclusive graphs and derive necessary and sufficient (in-)equations between the binary variables determining the relative positions. Since the embedding of R-nodes is fixed up to reversal, the relative positions can be simply synchronized with a binary variable representing the flip of the R-node. For P-nodes, they add equations that ensure that all shared components contained in a single virtual edge have the same relative position. Since they additionally need to ensure that both exclusive graphs have consistent edge orderings, they also use a system of equations to synchronize the possible embedding decisions for both graphs. Since the admissible orderings of P-nodes and cutvertices cannot be expressed using just equations on binary variables, they restrict themselves to the case where all P-nodes and cutvertices have shared degree at most 3. With this restriction, all relevant embedding choices become binary decisions, which allows them to solve SEFE in cubic time. However, their algorithm in general only works on graphs where the shared graph has maximum degree 3 [BKR18, Theorem 9].

We now develop an algorithm that allows the shared graph to have maximum degree 4, if both exclusive graphs are biconnected and have maximum degree 4. Similarly to Bläsius et al. [BKR18], we also use relative positions with respect to cycles of a cycle basis to ensure consistent relative positions. However, we use Synchronized Planarity to ensure consistent edge orderings of both exclusive graphs. For this purpose, let $\mathcal{B}$ denote a cycle basis of the shared graph G. Let $C \in \mathcal{B}$ be a cycle of the cycle basis and let K denote a shared component of G. We may assume, without loss of generality, that $\mathcal{B}$ only consists of simple cycles [BKR18]. For $i \in \{1, 2\}$, pick an arbitrary path $p^{①}$ in $G^{①}$ that is vertex-disjoint from C (except for its endpoint) and that connects C to K in $G^{①}$; see Figure 8.1a for an example. Since both exclusive graphs are connected, such a path always exists. Let $v^{①}$ denote the endpoint of path $p^{①}$ on C and let $f^{①}$ denote the edge of $p^{①}$ incident to $v^{①}$. Let further $e_1^{①}$ and $e_2^{①}$ denote the two shared edges of C incident to $v^{①}$. We create a triple $t^{①} = (e_1^{①}, f^{①}, e_2^{①})$ and we use the binary variable $\mathrm{ord}(t^{①})$ to express the *rotation* of $t^{①}$, i.e., whether the edge $f^{①}$ (and therefore also the shared component K) is embedded between $e_1^{①}$ and $e_2^{①}$ in clockwise cyclic order or not. We call these variables the *rotation variables*. The equation $\mathrm{ord}(t^{①}) = \mathrm{ord}(t^{②})$ is clearly necessary to ensure that the shared component K is embedded on the same side of cycle C in both exclusive graphs. We call these equations the *triple equations* and we say that $t^{①}$ is the *partner triple* of $t^{②}$. To compute all triples of a single cycle C, start a DFS at C in both exclusive graphs, putting all vertices of C on the stack. When extending a path, only consider vertices that are not contained in C. If the DFS first reaches a vertex of a new shared component K, we have found the desired path $p^{①}$ connecting C to K in $G^{①}$. This way, we can compute all triples of the cycle C in linear time and since the cycle basis $\mathcal{B}$ contains $\mathcal{O}(n)$ cycles [BKR18], we can compute all triples and the corresponding triple equations in time $\mathcal{O}(n^2)$.

Since the triples determine all relative positions of a shared component H with respect to cycles of the cycle basis $\mathcal{B}$, assigning a value to each triple uniquely determines a face of the shared graph where H is embedded, if such a face exists [BKR18, Theorem 8]. The triple equations additionally ensure that this relative position is identical for both exclusive graphs. Therefore, it remains to determine whether there exist embeddings of the exclusive graphs with consistent edge orderings such that the triple equations are satisfied.

Since the problem Synchronized Planarity can determine the set of embeddings of $G^{①}$ and $G^{②}$ with consistent edge orderings using a system of equations on the rotation of Q-nodes, our goal is to encode the triple equations into this system of equations. We start with the Synchronized Planarity instance $(H = (H^{①} \cup H^{②} \cup B, E), \mathcal{P}, \mathcal{Q}, \psi)$ obtained after the reduction from SEFE, where $H^{①}$ and $H^{②}$ denote the subgraphs of H corresponding to the two exclusive graphs, and B contains the bonds synchronizing the shared edges between $H^{①}$ and $H^{②}$. Recall that the vertices and edges of $H^{①}$ and $H^{②}$ correspond bijectively to the vertices and edges in $G^{①}$ and $G^{②}$. We place every triple located at $G^{①}$ and $G^{②}$ at its corresponding position in $H^{①}$ or $H^{②}$; see Figure 8.1b. We let $\mathcal{T}$ denote the set of these triples and we let ξ denotes the set of triple equations between the triples in $\mathcal{T}$. For a triple t in $\mathcal{T}$, recall that the binary rotation variable $\mathrm{ord}(t)$ indicates the rotation in which the edges of t appear in the cyclic order around the corresponding vertex of H. Observe that an embedding of H fixes a rotation for each triple in $\mathcal{T}$ and thus induces an assignment

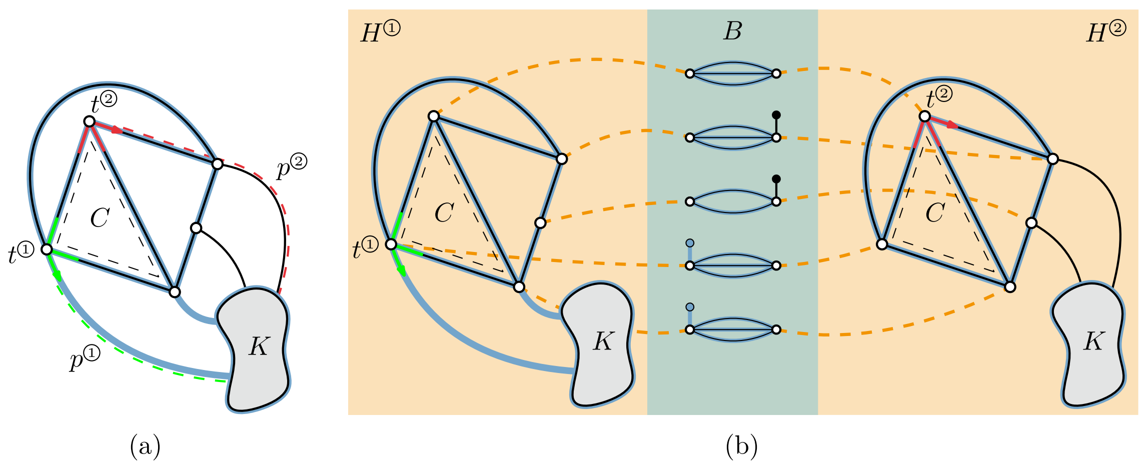


Figure 8.1: (a) A cycle C of the cycle basis of the shared graph. To ensure that the shared component K lies on the same side of C in both exclusive graphs, the triples $t^{①}$ and $t^{②}$ must be ordered consistently. The corresponding paths $p^{①}$ and $p^{②}$ are marked as dashed colored edges. (b) An equivalent instance of TRIPLE-SYNCPLAN.

for these binary variables. We therefore say that an embedding of H *satisfies the triple equations* ξ, if the corresponding assignment for the rotation of all triples satisfies the triple equations. In this way, we define an instance $\mathcal{I}_{\text{init}} = (H = (H^{①} \cup H^{②} \cup B, E), \mathcal{P}, \mathcal{Q}, \psi, \mathcal{T}, \xi)$ of the the new augmented problem TRIPLE-SYNCPLAN. A planar embedding $\mathcal{E}$ of H satisfies the TRIPLE-SYNCPLAN instance $\mathcal{I}_{\text{init}}$ if it satisfies all pipes, all Q-constraints, and additionally the triple equations in ξ.

Since the reduction from SEFE to SYNCHRONIZED PLANARITY takes linear time [BFR20] and since we can compute all triples in $\mathcal{T}$ in quadratic time as argued before, the TRIPLE-SYNCPLAN instance $\mathcal{I}_{\text{init}}$ can be obtained in quadratic time. First, we will now show that the SEFE instance $G^{\cup}$ and the TRIPLE-SYNCPLAN instance $\mathcal{I}_{\text{init}}$ are equivalent. Subsequently, we will show how the operations of the SYNCHRONIZED PLANARITY algorithm can be modified to solve the TRIPLE-SYNCPLAN instance $\mathcal{I}_{\text{init}}$.

Lemma 8.1. *An instance $G^{\cup}$ of SEFE admits a simultaneous embedding $\mathcal{E}^{\cup} = (\mathcal{E}^{①}, \mathcal{E}^{②})$ if and only if the corresponding TRIPLE-SYNCPLAN instance $\mathcal{I}_{\text{init}}$ admits a planar embedding.*

Proof. Since a simultaneous embedding of $G^{\cup}$ guarantees consistent edge orderings between $G^{①}$ and $G^{②}$, the pipes and Q-constraints of $\mathcal{I}_{\text{init}}$ are also satisfied. Since, as argued before, the additional triple equations are necessary, they must also be satisfied and thus $\mathcal{I}_{\text{init}}$ admits a planar embedding.

Conversely, let $(\mathcal{E}^{①}, \mathcal{E}^{②})$ be embeddings of $H^{①}$ and $H^{②}$ (and therefore of $G^{①}$ and $G^{②}$) that satisfy the TRIPLE-SYNCPLAN instance. Since the SYNCHRONIZED PLANARITY reduction ensures that the shared edges are ordered consistently in $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$ [BFR20], it suffices to show that every connected component of the shared graph is embedded in the same face of G in $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$. Let H denote a connected component of the shared graph and let $f^{①}$ and $f^{②}$ denote the faces of G in $\mathcal{E}^{①}$ and $\mathcal{E}^{②}$, respectively, that H is embedded in. Since, for each cycle C in $\mathcal{B}$ and for every shared component H in G, we have a triple in each exclusive graph, we get two relative position variables $\text{pos}_C^{①}(H)$ and $\text{pos}_C^{②}(H)$. However, our triple equations ensure that $\text{pos}_C^{①}(H) = \text{pos}_C^{②}(H)$ and since the face containing H is determined by the relative position variables [BFR20, Theorem 8], it must hold that $f^{①} = f^{②}$. □

Note that Lemma 8.1 does not rely on the restriction that both exclusive graphs of the SEFE instance are biconnected and have maximum degree 4. However, our algorithm will only be able to solve the resulting Triple-SyncPlan instance $\mathcal{I}_{\text{init}}$, if it stems from a SEFE instance $G^{\cup}$ where these restrictions hold.

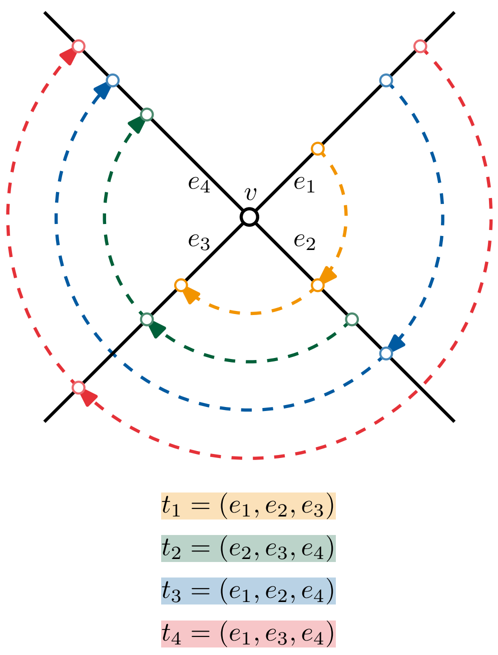
8.1 Consistent Triple Assignments

Consider an assignment ϕ that assigns a boolean value to each rotation variable $\text{ord}(t)$ for all $t \in \mathcal{T}$ and that additionally satisfies all triple equations in ξ. Unfortunately, although ϕ satisfies the triple equations, there is no guarantee that the triples at a single vertex do not contradict each other; see Figure 8.2. This is a problem for us, because we will be confronted with the situation, where Synchronized Planarity lets us choose an arbitrary order for the edges incident to a vertex v of degree 4 independently from the remaining instance. However, since there may be multiple triples located at v and since these triples can communicate with other parts of the instance via triple equations, we cannot pick an arbitrary assignment for the triples located at v. In this case, we need to find an assignment for the triples at v that satisfies the triple equations and does not produce contradictions between the triples located at v. As explained in Figure 8.2, the latter constraint can (in general) not be formulated as a 2-Sat instance and thus also not as a set of equations. Since our final goal will be to encode our triple equations into the 2-Sat instance that is used to solve the regular Synchronized Planarity problem, this poses a problem.
If an assignment ϕ for the triples does not produce a contradiction between triples located at a single vertex v, we say that ϕ is *consistent* for v. If every assignment satisfying the triple equations is consistent for v, we say that v itself is *consistent*. In order to be able to encode our triple equations into the 2-Sat formula solving Synchronized Planarity, we need to ensure that every vertex v is consistent. To this end, we will show that, for a vertex v that is not consistent in $\mathcal{I}_{\text{init}}$, the union graph $G^{\cup}$ always yields additional necessary equations between triples located at v. We will encode these equations into the triple equations in ξ and subsequently, v will be consistent.

To simplify our proofs, we assume that all possible triples containing two edges of a cycle in $\mathcal{B}$ are present at v, even if some of these triples are not part of a triple equation. Therefore, a triple (e_1, e_2, e_3) is present at v if and only if there exists a cycle C in the cycle basis $\mathcal{B}$ such that e_1 and e_3 are contained in C. Additionally, if we have two triples at v constraining the same three edges, we can synchronize them using an (in-)equality and subsequently remove one of the triples from v. Therefore, we end up with at most distinct $\binom{4}{3} = 4$ triples at every vertex. Note that this way, we cannot have vertices that contain exactly three triples. For vertices with at most two distinct triples, these triples cannot contradict each other. Therefore, any vertex that contains less than four triples is automatically consistent and we only have to consider vertices with exactly four triples.

Corollary 8.2. *Any vertex with less than four distinct triples is consistent.*

Now consider a vertex v that is not consistent, i.e, v contains four distinct triples. If we can find one necessary equation between two distinct triples located at v, all valid assignments for the triples at v can be formulated via equations. This can be verified using the truth table in Figure 8.2. For example, if $\text{ord}(t_1) \neq \text{ord}(t_2)$ is necessary, an assignment for the triples at v is valid if and only if additionally $\text{ord}(t_3) \neq \text{ord}(t_4)$ and $\text{ord}(t_1) = \text{ord}(t_3)$ holds. From now on, we therefore immediately infer that v is consistent, if we find such a necessary equation between two of its triples. Implicitly, we add the corresponding equations to the triple equations in ξ to ensure that v is consistent. This yields the following corollary.



ord(t_1)	ord(t_2)	ord(t_3)	ord(t_4)	valid
1	1	1	1	1
1	1	1	0	0
1	1	0	1	0
1	1	0	0	1
1	0	1	1	0
1	0	1	0	1
1	0	0	1	0
1	0	0	0	0
0	1	1	1	0
0	1	1	0	0
0	1	0	1	1
0	1	0	0	0
0	0	1	1	1
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Figure 8.2: A vertex v with four incident edges, such that the cyclic order of each subset of three edges is constrained by a triple. The binary variable $\text{ord}(t_i)$ represents whether the three edges belonging to t_i appear clockwise around v in the shown order or its reverse. Not every assignment for the triples is valid, e.g., $\text{ord}(t_1) = \text{ord}(t_2) = \text{ord}(t_3) = 1$ and $\text{ord}(t_4) = 0$ does not induce an order of the edges incident to v, since the triples contradict each other. Because exactly six of the $2^4 = 16$ possible assignments are valid, the valid assignments cannot be expressed as a 2-Sat formula, and therefore also not by a set of equations. If, however, there is an additional constraint that two triples must be (un-)equal, the remaining valid assignments can be formulated as a 2-Sat formula.

Corollary 8.3. *If there exists a necessary equation between two distinct triples located at a vertex v, then v is consistent.*

With the following lemma, we will show that we always find such a necessary equation at vertices with non-trivial embedding tree, which implies that the vertex is consistent.

Lemma 8.4. *If a vertex v of degree at most 4 has a non-trivial embedding tree, then v is consistent.*

Proof. By Corollary 8.2, we only have to consider the case where v has degree exactly 4 and contains exactly four triples. Let T_v denote the embedding tree of v. Since T_v is non-trivial, there exists a consecutivity constraint $R \in R(T)$ with $|R| = 2$ (note that $|R| \in \{0, 1, 3, 4\}$ would be a trivial constraint). Without loss of generality, assume $R = \{e_1, e_2\}$, i.e., e_1 and e_2 must be consecutive. Since we only have to consider the case where v contains exactly four distinct triples, there must be two triples t_1 and t_2 constraining edges $E(t_1) = \{e_1, e_2, e_3\}$ and $E(t_2) = \{e_1, e_2, e_4\}$, respectively. But since e_1 and e_2 must be consecutive, either $\text{ord}(t_1) = \text{ord}(t_2)$ or $\text{ord}(t_1) \neq \text{ord}(t_2)$ is necessary, depending on the default rotation of t_1 and t_2. For example, for $t_1 = (e_1, e_2, e_3)$ and $t_2 = (e_4, e_2, e_1)$, only assignments with $\text{ord}(t_1) \neq \text{ord}(t_2)$ yield cyclic orders where e_1 and e_2 are consecutive, thus $\text{ord}(t_1) \neq \text{ord}(t_2)$ is a necessary equation. By Corollary 8.3, v is therefore consistent. □

We now want to modify the Triple-SyncPlan instance $\mathcal{I}_{\text{init}}$ to ensure that each vertex of H is consistent. Recall that the subgraph B of H matching the shared edges of $H^{①}$ and $H^{②}$ does not contain any triples, thus all vertices of B are consistent. Since $G^{①}$ and $G^{②}$ are both biconnected, the two subgraphs $H^{①}$ and $H^{②}$ of the Triple-SyncPlan instance $\mathcal{I}_{\text{init}}$ are also biconnected, thus we only have to consider the consistency of block vertices. Since any block vertex with a non-trivial embedding tree is consistent by Lemma 8.4, only vertices of $H^{①}$ or $H^{②}$ with a trivial embedding tree remain. First, we will now show that we always find a necessary equation between triples located at each such vertex v due to the structure of the union graph $G^{\cup}$, hence v becomes consistent by Corollary 8.3. This way, we can show that each vertex of the initial instance $\mathcal{I}_{\text{init}}$ is consistent. Subsequently, we will show how the operations of the Synchronized Planarity algorithm can be modified to yield corresponding operations for our Triple-SyncPlan problem that can additionally handle triples. We will also show that the vertices of our instance remain consistent throughout applications of these operations. For the final reduced instance, we can then combine our triple equations with the 2-Sat instance that solves the regular Synchronized Planarity problem, to obtain a 2-Sat formula that solves our Triple-SyncPlan instance.

8.2 Consistent Vertices in $\mathcal{I}_{\text{init}}$

We now want to show that all vertices of the initial Triple-SyncPlan instance $\mathcal{I}_{\text{init}}$ are consistent. As argued above, it only remains to ensure that block vertices of $H^{①}$ and $H^{②}$ with a trivial embedding tree are consistent to ensure that all vertices in $\mathcal{I}_{\text{init}}$ are consistent. Recall that the vertices and edges in $H^{①}$ and $H^{②}$ of $\mathcal{I}_{\text{init}}$ correspond bijectively to the vertices in $G^{①}$ and $G^{②}$ of the SEFE instance. We therefore argue the consistency for the vertices in the SEFE instance, the consistency of the corresponding vertices in $H^{①}$ and $H^{②}$ then follows immediately. We also only argue the consistency of vertices in $G^{①}$, but all arguments can be applied symmetrically to $G^{②}$.

Consider a vertex $u^{①}$ in $G^{①}$ with a trivial embedding tree, hence $u^{①}$ has degree exactly 4. Then $u^{①}$ is a pole of a trivial bond μ with twin pole $v^{①}$. We will assume as a simplification that the triples located at $u^{①}$ directly refer to the virtual edges of μ. We consider different cases for μ, based on how the cycles of the cycle basis $\mathcal{B}$ appear in the virtual edges of μ. We say that a vertex or edge is *contained* in a virtual edge ε of μ, if the vertex or edge is contained in the expansion graph $\exp(\varepsilon)$ of ε. We say that a virtual edge ε of μ is *cycle-contained*, if there exists a cycle $C \in \mathcal{B}$ such that edges of C are contained in ε and another virtual edge $\varepsilon' \neq \varepsilon$ of μ; see Figure 8.3 for an example. Note that any cycle-contained virtual edge contains a path of shared edges between the poles $u^{①}$ and $v^{①}$. We say that two virtual edges ε_1 and ε_2 of μ are *union-linked*, if ε_1 and ε_2 are connected in $G^{\cup}$ via a path that is vertex-disjoint from the poles of μ and all other virtual edges in μ. If no virtual edge of μ is cycle-contained, then we have no triples located at the poles of μ and we are done. Observe that, if we have a cycle-contained virtual edge in μ, there must also be a second cycle-contained virtual edge in μ. Therefore, we have to consider the three remaining cases, where μ has two, three, or four cycle-contained virtual edges.

8.2.1 Four Cycle-Contained Virtual Edges

First consider the case where μ contains exactly four cycle-contained virtual edges. In this case, every virtual edge contains a shared path between $u^{①}$ and $v^{①}$. If there are two virtual edges in μ that are union-linked, they must therefore be adjacent in any embedding of μ [BFR20, Lemma 2]. We can therefore implicitly restrict the embedding tree of the pole $u^{①}$ of μ accordingly, thus μ is subsequently consistent by Lemma 8.4. If no pair of virtual edges is union-linked, then $\{u^{①}, v^{①}\}$ is a separating pair in the union graph $G^{\cup}$, which we can decompose into independent subinstances [BKR18, Lemma 4] of lower degree.

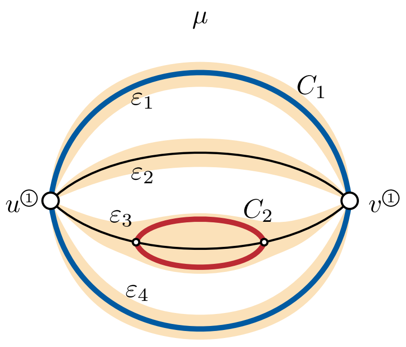


Figure 8.3: A trivial bond μ with four virtual edges. The virtual edges ε_1 and ε_4 are cycle-contained, because edges of the cycle C_1 (blue) are contained in ε_1 and ε_4. Note that ε_3 is not cycle-contained, because the edges of cycle C_2 (red) are only contained in ε_3 and not in any other virtual edge.

8.2.2 Three Cycle-Contained Virtual Edges

Before we consider the case where μ contains three cycle-contained virtual edges, we need some auxiliary tools. We start with the following lemma, which essentially states that triples at the poles of a trivial bond constraining the same virtual edges must be synchronized.

Lemma 8.5. *Let u and v denote two poles of a trivial bond μ of degree 4. Let δ_{uv} denote the bijection from the edges incident to u to the edges incident to v according to the bond μ. For a triple $t_u = (e_1, e_2, e_3)$ located at u and a triple $t_v = (\delta_{uv}(e_1), \delta_{uv}(e_2), \delta_{uv}(e_3))$ located at v, it is $\operatorname{ord}(t_u) \neq \operatorname{ord}(t_v)$ in any embedding of μ.*

Proof. Assume that there exists an embedding $\mathcal{E}$ of μ with $\operatorname{ord}(t_u) = \operatorname{ord}(t_v)$. Then the edges $\{e_1, e_2, e_3\}$ appear in the same clockwise cyclic order around u as the edges $\{\delta_{uv}(e_1), \delta_{uv}(e_2), \delta_{uv}(e_3)\}$ appear around v, which is a contradiction to the fact that u and v are the poles of a trivial bond. □

Using Lemma 8.5, we will now add additional necessary triple equations such that we subsequently know that the triples located at the two poles of a bond are essentially just synchronized copies of each other. Let μ denote a bond in $G^{①}$ with poles $u^{①}$ and $v^{①}$ and let δ_{uv} denote the bijective mapping of the edges incident to $u^{①}$ to the edges incident to $v^{①}$ according to the bond μ. Recall that we assume that a triple $t_u = (e_1, e_2, e_3)$ is present at $u^{①}$ if and only if there exists a cycle C in the cycle basis $\mathcal{B}$ such that e_1 and e_3 are contained in C. But since e_1 and e_3 are contained in two different virtual edges of μ, the edges $\delta_{uv}(e_1)$ and $\delta_{uv}(e_3)$ are also part of the cycle C and thus the triple $t_v = (\delta_{uv}(e_1), \delta_{uv}(e_2), \delta_{uv}(e_3))$ is present at $v^{①}$ and constrains the same three virtual edges as t_u. We add the equation $\operatorname{ord}(t_u) \neq \operatorname{ord}(t_v)$ to our triple equations, which is a necessary equation by Lemma 8.5. This means there now exists a bijective mapping between the triples at $u^{①}$ and the triples at $v^{①}$ and due to the new equations, the triples at $u^{①}$ can never contradict the triples at $v^{①}$ and vice versa. We therefore now say that the bond μ is *triple-mirrored*.

Corollary 8.6. *Every trivial bond of $G^{①}$ and $G^{②}$ is triple-mirrored.*

Now consider the case where exactly three edges of μ are cycle-contained. In this case, a path in $G^{\cup}$ between two cycle-contained virtual edges ε_1 and ε_2 of μ does not immediately lead to a consecutivity constraint for two virtual edges of μ, because the non-cycle-contained virtual edge ε does not necessarily contain a shared path and could possibly still be embedded between ε_1 and ε_2. This leads to instances where we cannot express the valid assignments for the triples located at $u^{①}$ via our triple equations. However, we will show with the following lemma that we can always find an equivalent instance where μ is consistent.

Lemma 8.7. *Let μ denote a trivial bond of the exclusive graph $G^{①}$ with exactly three cycle-contained virtual edges. If the poles of μ are not consistent, an equivalent instance where both poles are consistent can be computed in time $\mathcal{O}(n^2)$.*

Proof. We will assume that μ is part of $G^{①}$, but all arguments can also be applied to the symmetric case where μ is located in $G^{②}$.

If the non-cycle-contained virtual edge of μ is union-linked to a cycle-contained virtual edge of μ, these two virtual edges must be adjacent in any embedding of μ, thus we can once again infer that both poles of μ are consistent using Lemma 8.4. Therefore, we now assume that this is not the case.

Consider the case where no triple of μ has a partner triple in $G^{②}$. Note that this can only happen because we added additional "dummy-triples" earlier. In this case, all triples located at the poles of μ are "dummy-triples" which do not communicate with the remaining instance. We can thus simply remove all triples located at the poles of μ and we end up with an equivalent instance where both poles of μ are consistent. Therefore, we now assume that some triple of μ has a partner triple in $G^{②}$.

Let $u^{①}$ and $v^{①}$ denote the poles of μ and let $t = (\varepsilon_1, \varepsilon_2, \varepsilon_3)$ denote a triple of μ corresponding to a cycle $C \in \mathcal{B}$, where ε_2 is not cycle-contained and ε_1, ε_3 are cycle contained. For the partner triple t' of t in $G^{②}$ with respect to cycle C, let w denote the vertex of the shared graph that t' is located at and consider the position of w in μ. If w is contained in ε_1 (or ε_3), then ε_2 and ε_1 (or ε_2 and ε_3) are union-linked (by the definition of partner triples), which is a contradiction to an earlier assumption. Therefore, w must be one of the poles of μ. Hence, every triple of μ that contains the non-cycle-contained virtual edge ε_2 only communicates with triples located at the vertices $u^{②}$ or $v^{②}$ corresponding to $u^{①}$ and $v^{①}$ in $G^{②}$. Without loss of generality, we assume that t is located at $u^{①}$ in μ.
Let S denote the set of the three shared edges incident to $u^{①}$ that correspond to the cycle-contained virtual edges in μ and let $e^{①}$ denote the edge corresponding to the non-cycle-contained virtual edge ε_2. Let further $t^{①} = (s_1, e^{①}, s_2)$ denote a triple of μ located at $u^{①}$ such that $s_1, s_2 \in S$, i.e., $t^{①}$ constrains $e^{①}$ and two cycle-contained virtual edges in μ. Assume without loss of generality that the partner triple $t^{②}$ of $G^{②}$ that $t^{①}$ communicates with (i.e., $\mathrm{ord}(t^{①}) = \mathrm{ord}(t^{②})$) is located at $u^{②}$, i.e., $t^{②} = (s_1, f^{②}, s_2)$. It is $f^{②} \notin S$, because otherwise, the non-cycle-contained virtual edge ε_2 would be union-linked to a cycle-contained virtual edge by the definition of partner triples, a contradiction. Since $t^{①}$ and $t^{②}$ are partner triples, $e^{①}$ and $f^{②}$ are connected in $G^{\cup}$ via a path that is vertex-disjoint from the poles of μ. This path is also disjoint from all cycle-contained virtual edges of μ, otherwise we again find a union-link for ε_2. Therefore, $\mathrm{ord}((s'_1, e^{①}, s'_2)) = \mathrm{ord}((s'_1, f^{②}, s'_2))$ also holds for all $s'_1, s'_2 \in S$ in any simultaneous embedding of $G^{\cup}$. Since the fourth triple located at $u^{①}$ contains three shared edges, it is inherently synchronized with the corresponding triple located at $u^{②}$ due to consistent edge orderings between the two exclusive graphs. Hence all four triples are synchronized between $u^{①}$ and $u^{②}$ and they therefore induce a bijective mapping φ between the edges incident to $u^{①}$ and $u^{②}$, where $\varphi(e^{①}) = f^{②}$, and $\varphi(s) = s$ for all $s \in S$. For any simultaneous embedding $(\mathcal{E}^{①}, \mathcal{E}^{②})$, it therefore holds $\mathcal{E}^{②}(u^{②}) = \overline{\varphi(\mathcal{E}^{①}(u^{①}))}$. Additionally, $u^{②}$ is consistent if and only if $u^{①}$ is consistent. If $u^{②}$ has a non-trivial embedding tree, it thus immediately follows that $u^{①}$ is consistent by Lemma 8.4. Because μ is triple-mirrored by Corollary 8.6, this also implies that $v^{①}$ is consistent.

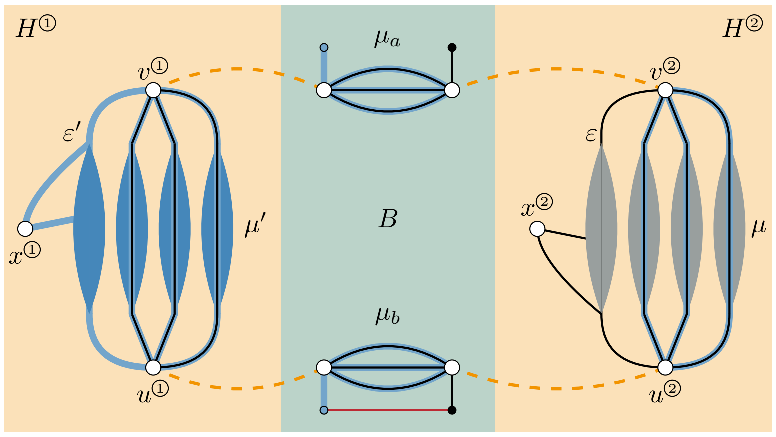


Figure 8.4: Two corresponding bonds μ and μ' of the two exclusive graphs in the initial Synchronized Planarity instance $\mathcal{I}_{\text{init}}$. Both bonds contain three cycle-contained virtual edges, marked with a shared path. Because the non-cycle-contained virtual edges $\mathcal{E}$ and $\mathcal{E}'$ are both connected to the two exclusive versions $x^{①}$ and $x^{②}$ of the same shared vertex x, $\mathcal{E}$ and $\mathcal{E}'$ must be embedded in the same face of the shared graph. The additional edge marked in red ensures that this is always the case.

If $u^{②}$ has a trivial embedding tree it is the pole of a trivial bond μ' in $G^{②}$ with twin pole $v^{②}$, and the bijective mapping φ between the edges incident to $u^{①}$ and $u^{②}$ naturally extends to a bijective mapping $\varphi_{\mu\mu'}$ between the virtual edges of μ and μ'. The non-cycle-contained virtual edge ε in μ must therefore be embedded in the same position in μ as the non-cycle-contained virtual edge $\varepsilon' = \varphi_{\mu\mu'}(\varepsilon)$ in μ'. If ε (and therefore also ε') is attached to one of its poles via a shared edge, then the consistent edge orderings already ensure this constraint. If ε is attached to both of its poles via an exclusive edge, we modify the initial Triple-SyncPlan instance as follows. Instead of matching the exclusive edges incident to $u^{①}$ and $u^{②}$ to dummy degree-1 vertices in B, we create a fourth parallel edge in the corresponding bond; see Figure 8.4. This change only adds the additional constraint that ε and ε' must be embedded consistently, which is a necessary constraint due to the mapping $\varphi_{\mu\mu'}$ induced by the triples in μ and μ' as argued above. Since the triples in μ and μ' only communicate among each other (as shown above), and since μ and μ' are triple-mirrored by Corollary 8.6, these triples only ensure that any embeddings of μ and μ' respect the mapping $\varphi_{\mu\mu'}$. But since the Triple-SyncPlan instance now always guarantees this constraint, we can once again remove all triples from μ and μ' and thus the poles of μ and μ' are now consistent. □

Although the admissible assignments of triples at a trivial bond μ with three cycle-contained edges cannot always be expressed using just triple equations, we can use Lemma 8.7 to obtain an equivalent instance, where both poles of μ are consistent.

8.2.3 Two Cycle-Contained Virtual Edges

Finally, we now consider the last case, where μ contains exactly two cycle-contained virtual edges ε_1 and ε_2; let ε_3 and ε_4 denote the other two virtual edges. Then μ contains at most the two triples $(\varepsilon_1, \varepsilon_3, \varepsilon_2)$ and $(\varepsilon_1, \varepsilon_4, \varepsilon_2)$. By Corollary 8.2, both poles of μ are therefore consistent.

8.2.4 Remaining Cases

We have now shown that all vertices of $G^{①}$ and $G^{②}$ with a trivial embedding tree are consistent. Every vertex with a non-trivial embedding tree is consistent by Lemma 8.4.

Since $G^{①}$ and $G^{②}$ are biconnected, they contain no cutvertices, thus all vertices of $G^{①}$ and $G^{②}$ are consistent. Since the vertices in $G^{①}$ and $G^{②}$ correspond bijectively to the vertices in the subgraphs $H^{①}$ and $H^{②}$ of H in $\mathcal{I}_{\text{init}}$, the same also holds for all vertices in $H^{①}$ and $H^{②}$. Therefore, any assignment for the triples that also satisfies the triple equations does not produce any contradictions between triples located at a single vertex. After applying Lemma 8.7 to $\mathcal{I}_{\text{init}}$, and after adding all new triple equations to ξ in $\mathcal{I}_{\text{init}}$, we get the following corollary.

Corollary 8.8. *In the initial* Triple-SyncPlan *instance $\mathcal{I}_{\text{init}}$, all vertices of the subgraphs $H^{①}$ and $H^{②}$ are consistent.*

Corollary 8.8 states that the triples located at every vertex of the initial Triple-SyncPlan instance behave nicely. As our next goal, we want to modify the operations of the regular Synchronized Planarity algorithm such that they can additionally handle triples. Before that, we will now state invariants that essentially guarantee that triples still behave nicely after each application of such an operation.

8.3 Invariants

Before we develop our invariants, we first state an additional auxiliary lemma, which helps us characterize the interactions between triples located at the two endpoints of a pipe.

Lemma 8.9. *Let $\rho = (u, v, \varphi_{uv})$ denote a pipe in the* Triple-SyncPlan *instance $\mathcal{I}$. For a triple $t_u = (e_1, e_2, e_3)$ located at u and a triple $t_v = (\varphi_{uv}(e_1), \varphi_{uv}(e_2), \varphi_{uv}(e_3))$ located at v, it is $\operatorname{ord}(t_u) \neq \operatorname{ord}(t_v)$ in any embedding of $\mathcal{I}$.*

Proof. Assume that there exists an embedding $\mathcal{E}$ of $\mathcal{I}$ with $\operatorname{ord}(t_u) = \operatorname{ord}(t_v)$. Then $\mathcal{E}(u) \neq \overline{\mathcal{E}(v)}$, thus $\mathcal{E}$ does not satisfy pipe ρ, a contradiction. □

Let u and v denote two vertices in a Triple-SyncPlan instance $\mathcal{I}$ and let δ_{uv} denote a bijective mapping between the edges incident to u and the edges incident to v. We say that the triples located at u and v are *mirrored* with respect to δ_{uv}, if for every triple $t_u = (e_1, e_2, e_3)$ located at u, there is a corresponding triple $t_v = (\delta_{uv}(e_1), \delta_{uv}(e_2), \delta_{uv}(e_3))$ located at v and the triple equations in ξ enforce $\operatorname{ord}(t_v) \neq \operatorname{ord}(t_v)$. Additionally, the same must also hold for the triples located at v with the mapping δ_{uv}^{-1}.
We say that a pipe $\rho = (u, v, \varphi_{uv})$ is *triple-mirrored*, if the triples located at u and v are mirrored with respect to the mapping φ_{uv}; see Figure 8.5a for an example. We say that a trivial bond μ with poles u and v is *triple-mirrored*, if the triples located at u and v are mirrored with respect to the bijective mapping δ_{uv} defined by μ; see Figure 8.5b. Note that this definition of triple-mirrored bonds is consistent with the definition of triple-mirrored bonds of Section 8.2.2.

Recall that all triples of our Triple-SyncPlan instance $\mathcal{I}_{\text{init}}$ are located at vertices in the subgraphs $H^{①}$ and $H^{②}$ of H. We now copy these triples along the pipes into the subgraph B of H; see Figure 8.6. Let $u^{①}$ denote a vertex of $H^{①}$ that is matched with a vertex $b_u^{①}$ of B via a pipe $\rho = (u^{①}, b_u^{①}, \varphi)$. We copy all triples from $u^{①}$ to $b_u^{①}$ via the mapping φ and add an inequality for their rotations to the triple equations in ξ. By Lemma 8.9, this does not alter the set of admissible embeddings of the Triple-SyncPlan instance. Note that subsequently, every pipe of $\mathcal{I}_{\text{init}}$ is triple-mirrored, which we state in the following invariant.

Invariant 8.1. *Every pipe ρ of the* Triple-SyncPlan *instance $\mathcal{I}$ is triple-mirrored.*

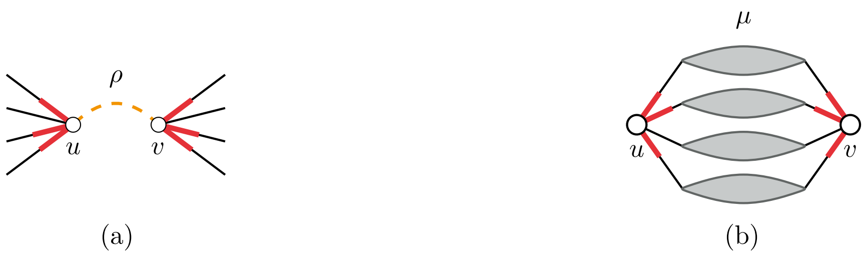


Figure 8.5: Two examples illustrating triples that are mirrored with respect to a pipe ρ (a) and triples that are mirrored with respect to a trivial bond μ (b).

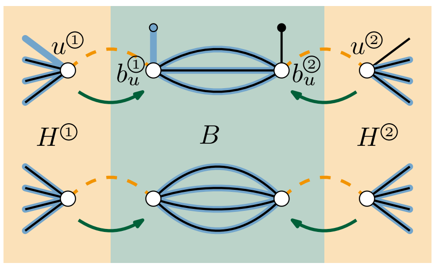


Figure 8.6: The three subgraphs $H^{①}$, $H^{②}$, and B of the initial Triple-SyncPlan instance $\mathcal{I}_{\text{init}}$. The triples from vertices in $H^{①}$ and $H^{②}$ are copied to the vertices in B along the pipes, as indicated by the green arrows. Subsequently, all pipes are triple-mirrored.

We also want to ensure that every trivial bond of our Triple-SyncPlan instance is triple-mirrored. In the reduced instance, this will help us find an embedding for such bonds that is compatible with the triple equations.

Invariant 8.2. *Every trivial bond of degree 4 in the Triple-SyncPlan instance $\mathcal{I}$ is triple-mirrored.*

Lemma 8.10. *Invariant 8.2 holds in $\mathcal{I}_{\text{init}}$.*

Proof. By Corollary 8.6, every bond of $G^{①}$ and $G^{②}$, and therefore also every bond of $H^{①}$ and $H^{②}$ is triple-mirrored. It only remains to show that the same also holds for all bonds in the subgraph B of H in $\mathcal{I}_{\text{init}}$ after copying the triples from $H^{①}$ and $H^{②}$ along pipes into B.

Consider a trivial bond μ of degree 4 in B with poles $b_u^{①}$ and $b_u^{②}$. Recall that the bonds in B synchronize the shared edges of corresponding shared vertices in $H^{①}$ and $H^{②}$. Therefore, there exists a vertex $u^{①}$ in $H^{①}$ and a vertex $u^{②}$ in $H^{②}$ such that $u^{①}$ is matched with $b_u^{①}$ and $u^{②}$ is matched with $b_u^{②}$. Let $\rho_1 = (u^{①}, b_u^{①}, \varphi_1)$ and $\rho_2 = (u^{②}, b_u^{②}, \varphi_2)$ denote the corresponding pipes.
There are two possibilities for the origin of μ. Either μ still has the same structure that it originally had after the reduction from SEFE to Triple-SyncPlan, or we altered μ with a transformation in Lemma 8.7. In the latter case, we removed all triples from $u^{①}$ and $u^{②}$ after the transformation. Therefore, after copying triples from $H^{①}$ and $H^{②}$ along the pipes into B, the poles of μ still contain no triples and therefore μ is triple-mirrored.
It thus only remains to consider the case where μ directly originates from the reduction from SEFE to Triple-SyncPlan. Since μ is a bond of degree 4 in B, the edges incident to $u^{①}$ and $u^{②}$ refer to the same four shared edges of the shared graph G in the SEFE instance. Let δ_μ denote the mapping from the edges incident to $b_u^{①}$ to the edges incident to $b_u^{②}$ according to bond μ and let φ denote the bijective mapping that maps every shared edge

incident to $u^{①}$ to its corresponding version incident to $u^{②}$. Note that $\varphi = \varphi_1 \circ \delta_\mu \circ \varphi_2^{-1}$, i.e., the two pipes ρ_1 and ρ_2, and the bond μ map the shared edges incident to $u^{①}$ to the shared edges incident to $u^{②}$ the same way as φ. Since the pipes ρ_1 and ρ_2 are triple-mirrored by Invariant 8.1, it therefore suffices to show that the triples located at $u^{①}$ and $u^{②}$ are mirrored with respect to φ to also show that μ is triple-mirrored.

Recall that we assume that a triple (e, f, g) is present at a vertex v in $G^{①}$ and $G^{②}$ (and thus also in $H^{①}$ and $H^{②}$) if and only if there exists a cycle C in the cycle basis $\mathcal{B}$ of G that contains the edges e and g. Let $t^{①} = (e_1, e_2, e_3)$ denote a triple located at $u^{①}$ (proceed symmetrically for triples located at $u^{②}$). This means that e_1 and e_3 belong to a cycle C in the cycle basis $\mathcal{B}$ of the shared graph G. But since $\varphi(e_1)$ and $\varphi(e_3)$ refer to the same shared edges as e_1 and e_3, there must also be the triple $t^{②} = (\varphi(e_1), \varphi(e_2), \varphi(e_3))$ located at $u^{②}$. We add the equation $\mathrm{ord}(t^{①}) \neq \mathrm{ord}(t^{②})$ to the triple equations in ξ and subsequently, the triples located at $u^{①}$ and $u^{②}$ are mirrored with respect to φ. Since $\varphi = \varphi_1 \circ \delta_\mu \circ \varphi_2^{-1}$ and since ρ_1 and ρ_2 are triple-mirrored by Invariant 8.1, this implies that μ is also triple-mirrored. □

The next invariant ensures that all vertices of our Triple-SyncPlan instance remain consistent throughout our algorithm. This will ensure that we will be able to encode the constraints imposed by triples into a 2-Sat instance in the reduced instance. By Corollary 8.8, the invariant holds for all vertices of the subgraphs $H^{①}$ and $H^{②}$ corresponding to the exclusive graphs $G^{①}$ and $G^{②}$. Subsequently, we only added triples to the vertices of the remaining subgraph B when we copied the triples of $H^{①}$ and $H^{②}$ along the pipes into B. Since these pipes are triple-mirrored by Invariant 8.1, the consistency of the vertices in B follows from the consistency of their matched vertices in $H^{①}$ or $H^{②}$. Therefore, every vertex of $\mathcal{I}_{\text{init}}$ is consistent, which we state in the following invariant.

Invariant 8.3. *Every vertex of the Triple-SyncPlan instance $\mathcal{I}$ is consistent.*

Since both exclusive graphs have maximum degree 4, the reduction from SEFE to Triple-SyncPlan ensures that all vertices of $\mathcal{I}_{\text{init}}$ also have maximum degree 4. To ensure that this remains the case throughout the algorithm, we state the following invariant.

Invariant 8.4. *Every vertex v of the Triple-SyncPlan instance $\mathcal{I}$ has degree at most 4.*

The next two invariants restrict the structure of cutvertices, which will help us handle the operation `EncapsulateAndJoin`.

Invariant 8.5. *Let v denote a degree-4 cutvertex of the Triple-SyncPlan instance $\mathcal{I}$. Then all split components of v, except for at most one, only contain a single edge incident to v.*

Invariant 8.6. *Let v denote a degree-4 cutvertex of the Triple-SyncPlan instance $\mathcal{I}$ that contains a triple t. Then at least two of the edges constrained by t belong to the same split component of v.*

Lemma 8.11. *Invariant 8.5 and Invariant 8.6 hold in $\mathcal{I}_{\text{init}}$.*

Proof. Since the exclusive graphs $G^{①}$ and $G^{②}$ of the SEFE instance are biconnected, the subgraphs $H^{①}$ and $H^{②}$ of $\mathcal{I}_{\text{init}}$ contain no cutvertices. The subgraph B matching the shared edges between $H^{①}$ and $H^{②}$ therefore only contains cutvertices because there are

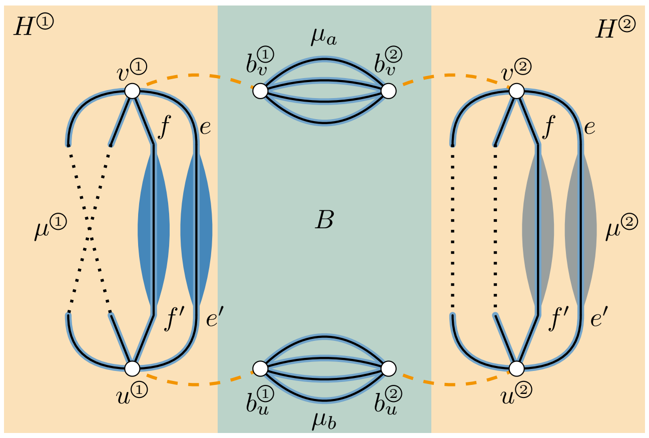


Figure 8.7: A bond-pipe cycle $v^{①}, b_v^{①}, b_v^{②}, v^{②}, u^{②}, b_u^{②}, b_u^{①}, u^{①}, v^{①}$.

degree-1 dummy vertices corresponding to exclusive edges. Because a split component corresponding to such a degree-1 dummy vertex contains only a single edge, Invariant 8.5 holds in $\mathcal{I}_{\text{init}}$.

Now consider a cutvertex $b_u^{①}$ of the subgraph B that contains a triple $t = (e_1, e_2, e_3)$ (the case for $b_u^{②}$ is symmetrical). Let $u^{①}$ denote the vertex of $H^{①}$ that $b_u^{①}$ is matched with via pipe $\rho_1 = (u^{①}, b_u^{①}, \varphi)$. Then t originates from copying the triple $t' = (\varphi^{-1}(e_1), \varphi^{-1}(e_2), \varphi^{-1}(e_3))$ from $u^{①}$ to $b_u^{①}$ (see Figure 8.6). Recall that this means that the edges $\varphi^{-1}(e_1)$ and $\varphi^{-1}(e_3)$ must be contained in a cycle of the shared graph, i.e., $\varphi^{-1}(e_1)$ and $\varphi^{-1}(e_3)$ are both shared edges. By the construction of the subgraph B in the Triple-SyncPlan reduction, e_1 and e_2 are therefore two parallel edges in the subgraph B and thus belong to the same split component incident to $b_u^{①}$. Thus, Invariant 8.6 holds for all cutvertices in B. Since $H^{①}$ and $H^{②}$ contain no cutvertices, Invariant 8.6 therefore holds in $\mathcal{I}_{\text{init}}$. □

The next invariant will help us handle the toroidal case of `SimplifyMatching`. For an odd k, let $v_0, v_1, v_2, \ldots, v_k, v_0$ be a sequence of distinct degree-4 vertices in H, such that, for even i, v_i and $v_{(i+1)}$ are matched via a pipe ρ_i and, for odd i, v_i and $v_{(i+1)\%k}$ are the poles of a trivial bond μ_i. We call such a sequence a *bond-pipe cycle*; see Figure 8.7 for an example. Let δ_i denote the bijective mapping of bond μ_i and let φ_i denote the mapping corresponding to pipe ρ_i. Let $\pi := \delta_k \circ \varphi_{k-1} \circ \ldots \delta_1 \circ \varphi_0$ be the permutation of the edges incident to v_0 defined by the bond-pipe-cycle.

Invariant 8.7. *Let v_0 be a degree-4 vertex in $\mathcal{I}$. Let $v_0, v_1, v_2, \ldots, v_k, v_0$ be a bond-pipe cycle with permutation π. If v_0 contains a triple, then there exists an edge e incident to v_0 such that $\pi(e) = e$.*

Lemma 8.12. *Invariant 8.7 holds in $\mathcal{I}_{\text{init}}$.*

Proof. Without loss of generality, we assume that v_0 is contained in $H^{①}$ of $\mathcal{I}_{\text{init}}$ (otherwise shift the cycle until this holds). Note that, since all pipes of $\mathcal{I}_{\text{init}}$ match vertices of $H^{①}$ to vertices of B, the bond-pipe cycle must consist of one bond $\mu^{①}$ in $H^{①}$, one bond $\mu^{②}$ in $H^{②}$, and two bonds μ_a and μ_b in B; see Figure 8.7 for an example. Recall that a subgraph of B is only a trivial bond of degree 4, if the corresponding vertices in $H^{①}$ and $H^{②}$ both have four incident shared edges[1], thus the bond-pipe cycle must have the structure shown in Figure 8.7.

[1]This is not necessarily the case for bonds that we altered using Lemma 8.7; see bond μ_b in Figure 8.4. However, the other bond (μ_a in Figure 8.4) does not have degree 4 and therefore μ_a and μ_b are not contained in a bond-pipe cycle.

We show that the invariant holds using Figure 8.7 with $v_0 = v^{①}$, i.e., the bond-pipe cycle c is defined by the sequence $v^{①}, b_v^{①}, b_v^{②}, v^{②}, u^{②}, b_u^{②}, b_u^{①}, u^{①}, v^{①}$. Since $v^{①}$ contains a triple, there must be two shared edges e and f incident to $v^{①}$ that are contained in the same cycle C of the cycle basis $\mathcal{B}$ of the shared graph. Since e and f are located in different virtual edges of $\mu^{①}$, there are also two edges e' and f' of C incident to $u^{①}$ such that e and e' (respectively f and f') are connected via a shared path in the corresponding virtual edge[2]. Since C is a shared cycle, the same edges are also incident to $v^{②}$ and $u^{②}$ in $H^{②}$. Because the bonds μ_a and μ_b and their incident pipes directly match the corresponding versions of the same edge, it must therefore hold $\pi(e) = e$ and $\pi(f) = f$ for the permutation π corresponding to the bond-pipe cycle c. Thus Invariant 8.7 holds in $\mathcal{I}_{\text{init}}$. □

Our next invariant states that every cutvertex containing a triple must be either a Q-vertex or matched via a pipe. Because our reduced instance will be pipe-free (as in the standard Synchronized Planarity algorithm), this invariant heavily simplifies the embedding choices of cutvertices containing triples.

Invariant 8.8. *Let v denote a cutvertex of degree 4 in the Triple-SyncPlan instance $\mathcal{I}$ such that v contains a triple. Then v is either a Q-vertex or matched via a pipe.*

Recall that we assume that both exclusive graphs of the SEFE instance are biconnected. Therefore, in the instance $\mathcal{I}_{\text{init}}$, all cutvertices are contained in the subgraph B of H matching the shared edges of $H^{①}$ to the corresponding edges in $H^{②}$. Since all vertices in B are matched via a pipe to a vertex in $H^{①}$ or $H^{②}$, Invariant 8.8 holds in $\mathcal{I}_{\text{init}}$.

The last invariant is essentially an auxiliary invariant guaranteeing that case i of the `SimplifyMatching` operation can never occur in our application. This way, we can ensure that Invariant 8.8 remains intact throughout our algorithm.

Invariant 8.9. *Let μ denote a trivial bond of degree 4 in the Triple-SyncPlan instance $\mathcal{I}$ such that one of the poles in μ contains a triple. Then either both poles of μ are part of a pipe or neither of them.*

Lemma 8.13. *Invariant 8.9 holds in $\mathcal{I}_{\text{init}}$.*

Proof. Note that the poles of a trivial bond μ of degree 4 in B are matched with vertices in $H^{①}$ and $H^{②}$, respectively, thus the invariant holds for all bonds in B.

Now consider a trivial bond μ of degree 4 contained in $H^{①}$ and let $u^{①}$ and $v^{①}$ denote its poles. Now assume that the pole $u^{①}$ is matched via a pipe, but $v^{①}$ is not. Due to the reduction from SEFE to Triple-SyncPlan, this means that $u^{①}$ is a shared vertex in $G^{①}$, and $v^{①}$ is an exclusive vertex. Note that any cycle of $G^{①}$ containing $u^{①}$ also contains $v^{①}$. Since $v^{①}$ is an exclusive vertex, neither $u^{①}$ nor $v^{①}$ can be contained in a cycle of the shared graph. But since vertices of $G^{①}$ and $G^{②}$ only contain triples if they are contained in a cycle of the shared graph, $u^{①}$ and $v^{①}$ both contain no triples. □

Like Bläsius et al. [BFR20], we temporarily replace Q-vertices with wheels of corresponding degree when we compute their embedding trees. This implies that no Q-vertex can be the pole of a trivial bond.

[2]Note that we make no assumptions how the other two virtual edges of $\mu^{①}$ and $\mu^{②}$ look like internally. As shown in Figure 8.7, these two virtual edges can even be matched differently in $\mu^{①}$ and $\mu^{②}$.

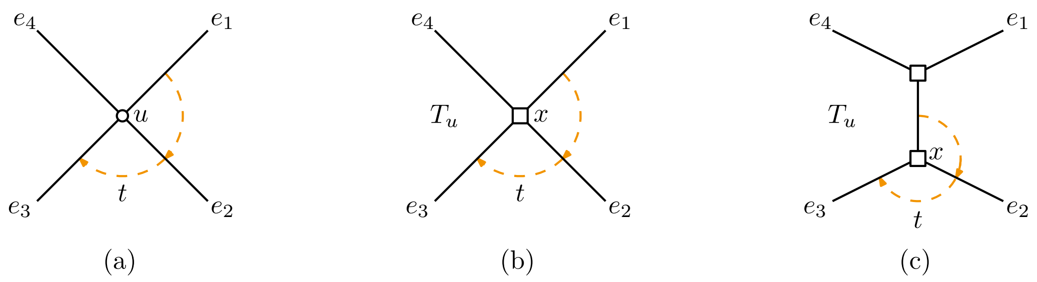


Figure 8.8: (a) A triple t constraining the edges e_1, e_2, and e_3 incident to u. (b)-(c) The new position of the triple t if the PQ-tree T_u used to replace u consists of a single inner Q-node, or two inner Q-nodes.

8.4 Modified Synchronized Planarity Operations

We now show how the different operations of the SYNCHRONIZED PLANARITY algorithm can be adapted to additionally handle the triples in the TRIPLE-SYNCPLAN problem. We also show that all invariants remain intact after every application of such an operation.

8.4.1 PropagatePQ

First consider an application of the operation `PropagatePQ` to the instance $\mathcal{I}$, i.e., we have a pipe $\rho = (u, v, \varphi_{uv})$ where u is a blockvertex with a non-trivial embedding tree. In this case, `PropagatePQ` replaces the vertex u with its embedding tree T_u, and vertex v with the mirrored version T'_u of T_u [BFR20]. In the general case, the corresponding P-nodes of T_u and T'_u are subsequently matched with a pipe and corresponding Q-nodes are synchronized using a Q-constraint. However, by Invariant 8.4, u has degree at most 4 and thus the non-trivial embedding tree T_u only contains Q-nodes. Therefore, an application of `PropagatePQ` completely eliminates the pipe ρ. Given a triple $t = (e_1, e_2, e_3)$ at vertex u (the case for v is symmetrical), there exists a unique inner node x in T_u, such that three neighbors of x each contain exactly one of the three edges of t in their subtree. Place the triple t at the corresponding position at node x and observe that t still represents the binary decision whether e_2 is embedded between e_1 and e_3 or not; see Figure 8.8.

Because `PropagatePQ` only introduces Q-vertices and thus no new pipes and bonds, Invariants 8.1 and 8.2 remain intact. Since these Q-vertices have a non-trivial embedding tree, we can use Lemma 8.4 to infer that all new vertices are consistent, thus Invariant 8.3 remains intact. All newly introduced vertices have degree at most 4, thus Invariant 8.4 remains intact. If the embedding tree T_u has more than one inner node, then all new vertices have degree 3 and Invariant 8.5 and Invariant 8.6 remain intact. Otherwise, T'_u consists of a single inner node x. But since Invariant 8.5 and Invariant 8.6 hold in $\mathcal{I}$ for vertex u, they still hold after replacing u with x. If u (and therefore also v) belongs to a bond-pipe cycle in $\mathcal{I}$, then `PropagatePQ` breaks this cycle, since the newly created vertices are not matched via pipes or poles of a trivial bond. For the same reason, no new bond-pipe cycles can be created by this operation. Since all other bond-pipe cycles are unaffected, Invariant 8.7 also remains intact. Since `PropagatePQ` only creates Q-vertices, Invariant 8.9 and Invariant 8.8 also remain intact.

8.4.2 EncapsulateAndJoin

Now consider an application of the operation `EncapsulateAndJoin`, i.e., we have a pipe $\rho = (u, w, \varphi_{uw})$, where both u and w are cutvertices. For every split component S incident to u or w, let v_S denote the vertex of the resulting instance $\mathcal{I}'$ corresponding to S. If a triple t located at u or w is contained in a single split component S, we move t to

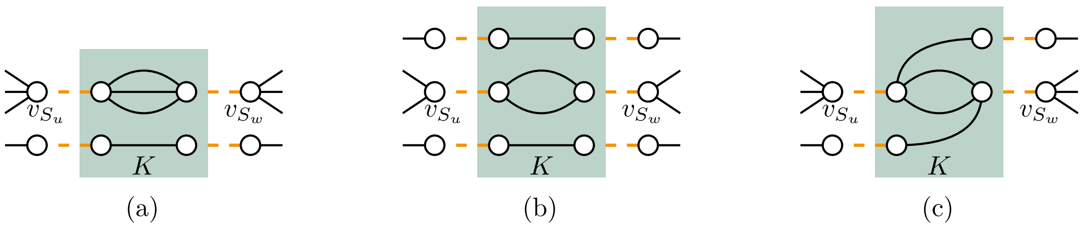


Figure 8.9: The three different cases that can occur for the bipartite graph K resulting from an operation of `EncapsulateAndJoin`, if both exclusive graphs are biconnected and have maximum degree 4, and one of the matched vertices contains a triple. Note that all pipes are of degree at most 3 and could thus be replaced with Q-constraints.

the corresponding position at v_S. Any other triple is removed from the graph, but the corresponding rotation variable remains part of the triple equations.

Lemma 8.14. *Applying the operation* `EncapsulateAndJoin` *to an instance $\mathcal{I}$ of* Triple-SyncPlan *yields an equivalent instance $\mathcal{I}'$.*

Proof. If u and w do not contain triples, the correctness follows immediately from the correctness of the standard operation [BFR20, Lemma 4]. Since u and w are matched in $\mathcal{I}$, they have mirrored triples by Invariant 8.1 and we thus now assume that both of them contain triples. By Invariant 8.5, u and w each have at most one non-trivial split component (i.e., a split component consisting of more than one edge incident to the cutvertex). Since u and w both contain triples and at least two edges of each triple must be contained in the same split component by Invariant 8.6, there are only three possibilities for the structure of the bipartition K in the resulting instance $\mathcal{I}'$; see Figure 8.9.

Let $\mathcal{E}$ denote an embedding of $\mathcal{I}$ and let ψ denote the corresponding truth assignment for all triples in T of $\mathcal{I}$ satisfying all triple equations. Then an embedding $\mathcal{E}'$ of $\mathcal{I}'$ can be obtained by encapsulating each split component incident to u and subsequently joining u and w via the satisfied pipe (u, w, φ_{uw}) [BFR20, Lemma 4]. Let S_u and S_w denote the non-trivial split component incident to u and w, respectively. Therefore, any triple of $\mathcal{I}$ that is completely contained in S_u (resp. S_w) is still satisfied at the new vertex v_{S_u} (resp. v_{S_w}). Since we removed all other triples from the instance, $\mathcal{E}'$ is an embedding of $\mathcal{I}'$ that satisfies ψ.

Conversely, let $\mathcal{E}'$ denote an embedding of $\mathcal{I}'$ with triple assignment ψ that satisfies all triple equations. In the following, we argue for triples located at u in $\mathcal{I}$, but all arguments can also be applied to triples located at w. We obtain an embedding $\mathcal{E}$ of $\mathcal{I}$ via a cut of the bipartition K in $\mathcal{E}'$ [BFR20, Lemma 4]. Clearly, any triple located at vertex S_u in $\mathcal{E}'$ is therefore also satisfied at vertex u in $\mathcal{E}$. Now consider a triple $t = (e_1, e_2, e_3)$ located at u in $\mathcal{I}$ such that e_1 and e_3 belong to S_u, but e_2 belongs to a different trivial split component, thus t is not present in $\mathcal{E}'$ (but ψ still assigns a rotation to t). Since the pipe matching u and w is triple-mirrored by Invariant 8.1, there is a triple $t' = (\varphi_{uw}(e_1), \varphi_{uw}(e_2), \varphi_{uw}(e_3))$ located at w in $\mathcal{I}$ with $\operatorname{ord}(t) \neq \operatorname{ord}(t')$. First consider the case where t' is completely contained in split component S_w of w in $\mathcal{I}$ and is thus located at vertex v_{S_w} in $\mathcal{I}'$. Note that this can only occur in the case shown in Figure 8.9c. In this case, the triple t' already restricts the embedding (and therefore the cut) of the bipartite graph K such that t is also satisfied in $\mathcal{I}$. Now consider the case where t' is also not present in $\mathcal{I}'$, thus $\varphi_{uw}(e_1)$ and $\varphi_{uw}(e_3)$ belong to the split component S_w at w in $\mathcal{I}$ and $\varphi_{uw}(e_2)$ is a trivial split component. Thus e_2 and $\varphi_{uw}(e_2)$ are both a trivial split component incident to u and w, respectively. Note that this can only occur in the two cases shown in Figure 8.9a and

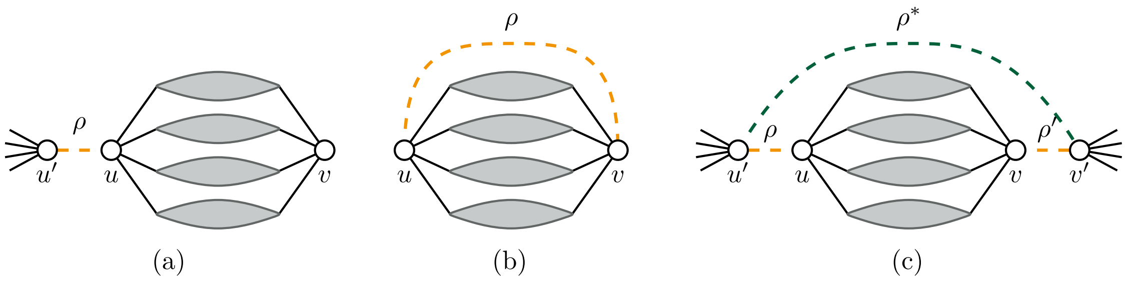


Figure 8.10: The three different cases of the operation `SimplifyMatching`.

Figure 8.9b. In this case, only the triples located at u and w containing the edges e_2 or $\varphi_{uw}(e_2)$ constrain the position of e_2 and $\varphi_{uw}(e_2)$ in $\mathcal{I}$. In the bipartition K of $\mathcal{I}'$, the two vertices corresponding to e_2 and $\varphi_{uw}(e_2)$ are only connected to each other and can thus appear in any position in a cut of K. Thus, we start with an embedding $\mathcal{E}'$ of $\mathcal{I}'$ and first obtain an embedding $\mathcal{E}$ of $\mathcal{I}$ via a cut of the bipartition K as described above. Subsequently, move e_2 and $\varphi_{uw}(e_2)$ consistently around u and w, respectively, such that the triples at u and w are satisfied with assignment ψ. Since u and w are consistent in $\mathcal{I}$ by Invariant 8.3 and triple-mirrored by Invariant 8.1,and since ψ satisfies the triple equations, such a position always exists. We therefore obtain an embedding of $\mathcal{I}$ satisfying all triple equations.

It remains to show that the invariants remain intact. As shown in Figure 8.9, all newly created vertices have degree at most 3 and can therefore be matched via Q-constraints, thus Invariant 8.1 and Invariant 8.2 remain intact. For the same reason, each newly created vertex has a non-trivial embedding tree, thus we can use Lemma 8.4 to infer that Invariant 8.3 remains intact. Since all newly created vertices have degree at most 3, Invariants 8.5, 8.6, 8.4, 8.9, and 8.8 are also not affected. If u (and therefore also w) belongs to a bond-pipe cycle in $\mathcal{I}$, then `EncapsulateAndJoin` breaks this cycle, since all newly created vertices have degree at most 3. For the same reason, no new bond-pipe cycles can be created by this operation. Since all other bond-pipe cycles are unaffected, Invariant 8.7 also remains intact. □

8.4.3 SimplifyMatching

Now consider an application of `SimplifyMatching` case i, i.e., we have a pipe $\rho = (u, u', \varphi_{uu'})$, where u is part of a trivial bond μ with twin pole v and exactly four virtual edges and v is unmatched (see Figure 8.10a). By Invariant 8.9, this case cannot occur in our application. We remark that the fact that this case cannot occur is the reason why Invariant 8.8 always remains intact.

Next, consider an application of `SimplifyMatching` case ii, i.e., there is a pipe $\rho = (u, v, \varphi_{uv})$, such that u and v are the poles of a trivial bond μ (see Figure 8.10b). If the permutation $\pi = \delta_{uv} \circ \varphi_{uv}^{-1}$ defined by μ and ρ has cycles of different length, we reduce to a trivial no-instance as in the original operation [BFR20]. Since μ and ρ form a bond-pipe cycle with permutation π, by Invariant 8.7, there exists an edge e incident to u such that $\pi(e) = e$, i.e., this particular cycle has length 1 and consequently all cycles of π have length 1. Therefore, the permutation π is the identity and thus the pipe ρ is redundant and can be removed.
Note that removing ρ breaks the bond-pipe cycle formed by μ and ρ, all other cycles are unaffected, thus Invariant 8.7 remains intact. Before the operation, both poles of μ were part of a pipe, afterwards, neither of them are matched, thus Invariant 8.9 remains intact. Since the operation only removes pipe ρ and does not otherwise alter the instance, all other invariants also remain intact.

Finally, consider an application of the operation `SimplifyMatching` case iii (the transitive case), i.e., we have two pipes $\rho = (u', u, \varphi_{u'u})$ and $\rho' = (v, v', \varphi_{vv'})$ such that u and v are the poles of a trivial bond μ (see Figure 8.10c). The operation `SimplifyMatching` removes the pipes ρ and ρ' and adds a new pipe $\rho^* = (u', v', \varphi_{vv'} \circ \delta_{uv} \circ \varphi_{u'u})$, where δ_{uv} bijectively maps the edges incident to u to the edges incident to v according to the bond μ [BFR20]. Note that ρ, ρ', and μ are triple-mirrored by Invariants 8.1 and 8.2, thus the triples located at u, u', v and v' are synchronized copies of one another. Therefore, the operation is also correct in the presence of triples.
Since ρ, ρ', and μ are triple-mirrored, the triples located at u' and v' are also mirrored with respect to the bijection $\varphi_{vv'} \circ \delta_{uv} \circ \varphi_{u'u}$. Therefore, the pipe ρ^* is triple-mirrored in the reduced instance. Since the operation does not create any other new pipes, Invariant 8.1 remains intact. Before the operation, both poles of μ were part of a pipe, afterwards, neither of them are matched, thus Invariant 8.9 remains intact. Note that replacing ρ and ρ' with ρ^* can only shorten a bond-pipe cycle, however, the corresponding permutation π remains the same, thus Invariant 8.7 still holds. Other than that, the operation only removes the pipes ρ and ρ' and does not otherwise alter the instance, thus all other invariants still hold.

8.5 Solving the Reduced Instance

We have shown in Section 8.4 how the operations of the standard Synchronized Planarity algorithm can be modified to additionally handle triples and that all invariants from Section 8.3 remain intact. Observe that all of our operations only differ from their original variants in the way they treat triples, the analysis of the original Synchronized Planarity algorithm still holds and we obtain an equivalent instance $\mathcal{I}'$ containing no pipes in time $\mathcal{O}(n^2)$[3] [BFR20, Theorem 12].

In the standard Synchronized Planarity algorithm, an embedding of the reduced pipe-free instance $\mathcal{I}'$ can be computed as follows [BFR20]. First, every Q-vertex is replaced with a wheel of the respective degree, which yields a graph H'. Subsequently, each Q-vertex is contained in a single R-node of a biconnected component of H'. Since $\mathcal{I}'$ no longer contains pipes, it suffices to find an embedding of H' where the Q-constraints between Q-vertices are satisfied. Finding such an embedding is simple, because every Q-vertex is contained in a single R-node, and the embedding of an R-node is a binary decision. Simply introduce a boolean variable representing the rotation of each Q-vertex and another boolean variable representing the rotation of each R-node. Subsequently, the variables of Q-vertices belonging to the same Q-constraint can be synchronized via equations. Additionally, the rotation of each Q-vertex can be synchronized with the rotation of the unique R-node it is contained in using an equation between the corresponding boolean variables. This yields a 2-Sat formula that can be solved in linear time. Every solution for this 2-Sat formula yields an embedding of all R-nodes in H', such that all Q-constraints are satisfied. Subsequently, choosing an arbitrary embedding for each P-node in H' and picking an arbitrary nesting for the split components incident to each cutvertex yields an embedding that is a valid solution for $\mathcal{I}'$ [BFR20]. Conversely, if the 2-Sat instance has no solution, then $\mathcal{I}'$ is a no-instance.

Unfortunately, triples heavily complicate the reduced instance in general. Since some of the triples may be contained in P-nodes or contained in multiple split components incident to a cutvertex, it is not that simple to represent all admissible solutions via a 2-Sat formula. This is because the embedding choices at P-nodes and cutvertices are usually not binary decisions and even the admissible assignments for triples located at a single vertex can, in general, not be formulated as a 2-Sat instance (see Figure 8.2). This is where our invariants come into play. In Section 8.3, we stated several invariants and showed that

[3]Since both exclusive graphs have maximum degree 4, it is $m \in \mathcal{O}(n)$.

they hold in the initial Triple-SyncPlan instance $\mathcal{I}_{\text{init}}$. Note that these invariants rely heavily on the restriction that both exclusive graphs of our initial SEFE instance are biconnected and have maximum degree 4. Subsequently, we showed in Section 8.4 that these invariants also retain their validity after applying one of the operations `PropagatePQ`, `EncapsulateAndJoin`, or `SimplifyMatching`. As argued above, exhaustively applying these operations yields a pipe-free instance $\mathcal{I}'$ of Triple-SyncPlan. Additionally, all invariants still hold in $\mathcal{I}'$. In the following, we show how these invariants can be used to solve the reduced instance $\mathcal{I}'$.

As in the standard Synchronized Planarity algorithm, we first build a 2-Sat instance that synchronizes the rotation of each Q-vertex with the embedding of the R-node that it is contained in. Additionally, we encode the Q-constraints into this 2-Sat formula [BFR20]. Recall that we have a boolean variable $\text{ord}(t)$ representing the rotation of every triple t. We add these boolean variables and all our triple equations to the 2-Sat instance. Note that these equations can be converted to a 2-Sat formula the same way as the Q-constraints. If a triple t is fully contained within an R-node μ, we synchronize $\text{ord}(t)$ with the corresponding boolean variable representing the rotation of μ. Note that we can convert any degree-3 vertex of $\mathcal{I}'$ into a Q-vertex [BFR20, Lemma 3]. Therefore, only embedding choices regarding P-nodes and cutvertices of degree 4 remain, since all vertices of $\mathcal{I}'$ have degree at most 4 by Invariant 8.4.

We first consider a cutvertex v of degree 4 that contains a triple. By Invariant 8.8, v must be either a Q-vertex or matched via a pipe. Since the reduced instance $\mathcal{I}'$ is pipe-free, v must therefore be a Q-vertex and thus only represents a binary embedding decision. We can therefore simply synchronize the rotation of triples located at v with the Q-constraint corresponding to v in our 2-Sat instance. Subsequently, all embedding choices regarding such cutvertices and their triples are encoded into our 2-Sat formula.

Now it only remains to show that, for any solution of our 2-Sat instance, we also find an embedding for every P-node that satisfies the triples located at its poles. We prove this in the following lemma.

Lemma 8.15. *Let μ be a P-node of degree 4 in $\mathcal{I}'$. Let ψ be an assignment for the triples in $\mathcal{I}'$ that satisfies the triple equations. Then there exists a planar embedding of μ that satisfies the triples located at the poles of μ with assignment ψ.*

Proof. Let u and v denote the poles of μ. By Invariant 8.3, the vertex u is consistent. This means that there exists a cyclic ordering σ of the edges incident to u that satisfies the triples located at u with assignment ψ, since ψ satisfies the triple equations. To create an embedding $\mathcal{E}_\mu$ of μ, pick $\mathcal{E}_\mu(u) = \sigma$ and $\mathcal{E}_\mu(v) = \overline{\delta_{uv}(\mathcal{E}_\mu(u))} = \overline{\delta_{uv}(\sigma)}$. It is clear that $\mathcal{E}_\mu$ is a planar embedding of μ. Because σ satisfies the triples located at u with assignment ψ, the same also holds for $\mathcal{E}_\mu(u) = \sigma$. It remains to show that the same also holds for $\mathcal{E}_\mu(v)$. By Invariant 8.2, the bond μ is triple-mirrored, i.e., the triples located at v are copies of the triples located at u and vice versa. By the definition of triple-mirrored, for each triple $t = (e_1, e_2, e_3)$ located at u and its corresponding copy $t' = (\delta_{uv}(e_1), \delta_{uv}(e_2), \delta_{uv}(e_3))$ located at v, the equation $\text{ord}(t) \neq \text{ord}(t')$ is part of the triple equations. Because $\mathcal{E}_\mu(u)$ satisfies the triples located at u with assignment ψ, the order $\mathcal{E}_\mu(v) = \overline{\delta_{uv}(\mathcal{E}_\mu(u))}$ therefore also satisfies the triples located at v. Thus the embedding $\mathcal{E}_\mu$ is a planar embedding of μ that satisfies the triples at the poles of μ. □

Since we have encoded the triple equations into our 2-Sat instance, any solution of the 2-Sat instance can therefore be extended to a planar embedding for each P-node in $\mathcal{I}'$ by Lemma 8.15. Since the same also holds for all R-nodes in $\mathcal{I}'$ [BFR20], and since all cutvertices containing triples are Q-vertices, any solution for the 2-Sat instance yields a planar embedding that satisfies all Q-constraints and all triple equations of $\mathcal{I}'$. If the 2-Sat instance is unsatisfiable, we report an invalid instance. Our 2-Sat formula has size in $\mathcal{O}(n^2)$ and can be solved in linear time. Finally, we get the following result.

Theorem 8.16. *Simultaneous Embedding with Fixed Edges can be solved in time $\mathcal{O}(n^2)$ if both exclusive graphs are biconnected and have maximum degree 4.*

8.6 Remarks About Cutvertices and Vertices of Higher Degree

Since our algorithm is restricted to instances where both exclusive graphs are biconnected and have maximum degree 4, the question arises whether the algorithm can be adjusted to also handle cutvertices or vertices of higher degree. If the exclusive graphs contain cutvertices, we additionally have to ensure that these cutvertices are consistent in the initial instance $\mathcal{I}_{\text{init}}$ in Section 8.2. Possibly, this can once again be done by finding necessary equations between two distinct triples located at such a cutvertex, by examining the structure of the union graph. Adjusting the invariants in Section 8.3 to also handle cutvertices in the exclusive graphs should be straightforward.
When considering vertices of degree 5, we once again have to ensure their consistency in the initial instance $\mathcal{I}_{\text{init}}$. Unfortunately, a non-trivial embedding tree with five leaves does not necessarily only consist of Q-nodes as its inner nodes, thus Lemma 8.4 does not hold in general in this case. For the modified Synchronized Planarity operations in Section 8.4, a more refined analysis is necessary, because the number of possible cases that can occur increases with vertices of degree 5.

9. Conclusion

In this work, we first showed that SEFE is FPT parameterized by the vertex cover number of the union graph (Chapter 3) and the feedback edge set number of the union graph (Chapter 4), respectively. While our parameterization by the treedepth of the union graph (Chapter 5) only works in restricted cases, it features a promising approach and reveals problems that must be solved before advancing to less restrictive parameters in the FPT-landscape. We remark that these parameterizations can be straightforwardly extended to k-SEFE with $k \geq 3$.

In chapter Chapter 6, we started analyzing parameters of the shared graph. It becomes immediately clear that parameterizations of the shared graph are significantly more involved, because any reduction rule affecting the shared graph must consider all its possible configurations in the union graph. This makes kernelization approaches difficult. Instead, we combined the parameters vertex cover number and number of degree-1 vertices to essentially enumerate all suitable embeddings of the shared components. Subsequently, we used the algorithm by Bläsius and Rutter [BR15] that solves SEFE in quadratic time, if each shared component has a fixed embedding, and we finally obtained an FPT-algorithm. Our algorithm also extends to the sunflower case of k-SEFE with $k \geq 3$. Interestingly, the problem remains far from trivial, even when using a combination of these two very restrictive parameters.

As the next step, our goal was to combine techniques for ensuring consistent relative positions with the recently developed Synchronized Planarity reduction that ensures consistent edge orderings. In Chapter 7, we first characterized the embeddings of the exclusive graphs that satisfy consistent relative positions. We derived partial constraints, a set of PQ-trees we used to annotate the pipes of the Synchronized Planarity instance, that additionally constrain the set of admissible edge orderings. The resulting instance describes exactly the simultaneous embeddings of the original SEFE instance, but the operations of the Synchronized Planarity algorithm must also be adjusted to handle the additional partial constraints. This is particularly challenging for the operation `EncapsulateAndJoin` handling pipes matching two cutvertices, because the partial constraints essentially restrict the admissible cuts of the resulting bipartition. For this reason, we needed additional restrictions, but finally developed an FPT algorithm for SEFE parameterized by the number of connected components and the maximum degree of the shared graph, if both exclusive graphs are biconnected, each pair of fixpoints is block-local, and Conjecture 7.11 holds for the operation `Toroidal Constrained SimplifyMatching`.

Finally, we used a very similar approach in Chapter 8 to solve SEFE if both exclusive graphs are biconnected and have maximum degree 4. We placed triples in both exclusive graphs of the Synchronized Planarity instance that ensure that, for every cycle C in a cycle basis of the shared graph, every shared component lies on the same side of C in both exclusive graphs. As Bläsius et al. [BKR18] have shown, ensuring consistent relative positions with respect to cycles of a cycle basis ensures consistent relative positions in all possible embeddings. We modified the operations of the Synchronized Planarity algorithm to additionally handle the triples and in this way obtained a quadratic time-algorithm that solves SEFE if both exclusive graphs are biconnected and have maximum degree 4. We remark that the Synchronized Planarity reduction only works for two input graphs, our algorithm therefore does not extend to k-SEFE with $k \geq 3$.

While we developed FPT-algorithms for SEFE parameterized by several parameters, we only argued why it is difficult to proceed with less restrictive parameters. Finding more parameterizations or actual hardness results with respect to the W-hierarchy [CFK$^+$15] remains an open problem for future work. In Chapter 6, we showed that the vertex cover number of the shared graph alone is sufficient to enumerate all suitable embeddings of every block in the shared graph G in FPT time. It should be possible to extend our observations to completely fix the embedding of every shared component, except for the position of degree-1 vertices. This basically already fixes all faces of the shared graph G and subsequently, it only remains to determine the degree-1 vertices that can be embedded together into the same face. While this is not very difficult if every face induces a simple cycle, the general case is more challenging. However, solving this could lead to an FPT-algorithm for SEFE parameterized by the vertex cover number of the shared graph alone. Since our parameterization uses the vertex cover number plus the number of degree-1 vertices of the shared graph, it is also an interesting question whether SEFE is still FPT parameterized by the treewidth plus the number of degree-1 vertices of the shared graph.
Although the algorithm from Chapter 7, where we augmented the Synchronized Planarity problem with partial constraints, only works under rather strong and impractical restrictions, we believe it is a very promising approach that provides interesting insights on the interplay between relative positions and Synchronized Planarity. Finding stronger invariants for the augmented Synchronized Planarity instance could lead to an algorithm that solves SEFE in more general cases than currently known algorithms. To this end, it could be helpful to better understand how the intermediate instances of the Synchronized Planarity algorithm relate to the original SEFE instance.
The same also applies to the quadratic-time algorithm that solves SEFE if both exclusive graphs are biconnected and have maximum degree 4 (Chapter 8). It is quite realistic that the algorithm can be adjusted to also handle cutvertices and/or exclusive graphs with maximum degree 5. We leave these considerations as open problems for future work.

Bibliography

[ABF$^+$12] Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph. *J. Discrete Algorithms*, 14:150–172, 2012.

[ALN15] Patrizio Angelini, Giordano Da Lozzo, and Daniel Neuwirth. Advancements on SEFE and partitioned book embedding problems. *Theor. Comput. Sci.*, 575:71–89, 2015.

[BCE18] Michael J. Bannister, Sergio Cabello, and David Eppstein. Parameterized complexity of 1-planarity. *J. Graph Algorithms Appl.*, 22(1):23–49, 2018.

[BFR20] Thomas Bläsius, Simon D. Fink, and Ignaz Rutter. Synchronized planarity with applications to constrained planarity problems. *CoRR*, abs/2007.15362, 2020.

[BGL$^+$22] Carla Binucci, Emilio Di Giacomo, William J. Lenhart, Giuseppe Liotta, Fabrizio Montecchiani, Martin Nöllenburg, and Antonios Symvonis. On the complexity of the Storyplan Problem. *CoRR*, abs/2209.00453, 2022.

[BGMN20] Sujoy Bhore, Robert Ganian, Fabrizio Montecchiani, and Martin Nöllenburg. Parameterized algorithms for book embedding problems. *J. Graph Algorithms Appl.*, 24(4):603–620, 2020.

[BKR13] Thomas Bläsius, Stephen G. Kobourov, and Ignaz Rutter. Simultaneous embedding of planar graphs. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 349–381. Chapman and Hall/CRC, 2013.

[BKR18] Thomas Bläsius, Annette Karrer, and Ignaz Rutter. Simultaneous embedding: Edge orderings, relative positions, cutvertices. *Algorithmica*, 80(4):1214–1277, 2018.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976.

[Boo75] Kellogg S. Booth. *PQ-tree algorithms.* PhD thesis, University of California, Berkeley, 1975.

[BR15] Thomas Bläsius and Ignaz Rutter. Disconnectivity and relative positions in simultaneous embeddings. *Comput. Geom.*, 48(6):459–478, 2015.

[BR16] Thomas Bläsius and Ignaz Rutter. Simultaneous PQ-ordering with applications to constrained embedding problems. *ACM Trans. Algorithms*, 12(2):16:1–16:46, 2016.

[BT96] Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996.

[CFK⁺15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

[CKX10] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010.

[EGJ⁺07] Alejandro Estrella-Balderrama, Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz. Simultaneous geometric graph embeddings. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing, 15th International Symposium, GD 2007*, volume 4875 of *Lecture Notes in Computer Science*, pages 280–290. Springer, 2007.

[FLSZ19] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing.* Cambridge University Press, 2019.

[FT20] Radoslav Fulek and Csaba D. Tóth. Atomic embeddability, clustered planarity, and thickenability. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 2876–2895. SIAM, 2020.

[GJP⁺06] Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz. Simultaneous graph embeddings with fixed edges. In Fedor V. Fomin, editor, *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006*, volume 4271 of *Lecture Notes in Computer Science*, pages 325–335. Springer, 2006.

[GM00] Carsten Gutwenger and Petra Mutzel. A linear time implementation of spqr-trees. In Joe Marks, editor, *Graph Drawing, 8th International Symposium, GD 2000*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2000.

[HJL13] Bernhard Haeupler, Krishnam Raju Jampani, and Anna Lubiw. Testing simultaneous planarity when the common graph is 2-connected. *J. Graph Algorithms Appl.*, 17(3):147–171, 2013.

[HM03] Wen-Lian Hsu and Ross M. McConnell. PC trees and circular-ones arrangements. *Theor. Comput. Sci.*, 296(1):99–116, 2003.

[HM04] Wen-Lian Hsu and Ross M. McConnell. PQ trees, PC trees, and planar graphs. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications.* Chapman and Hall/CRC, 2004.

[Hsu01] Wen-Lian Hsu. PC-trees vs. PQ-trees. In Jie Wang, editor, *Computing and Combinatorics, 7th Annual International Conference, COCOON 2001*, volume 2108 of *Lecture Notes in Computer Science*, pages 207–217. Springer, 2001.

[HT74] John E. Hopcroft and Robert E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[JS09] Michael Jünger and Michael Schulz. Intersection graphs in simultaneous embedding with fixed edges. *J. Graph Algorithms Appl.*, 13(2):205–218, 2009.

[Kur30] Casimir Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta mathematicae*, 15(1):271–283, 1930.

[PW01] János Pach and Rephael Wenger. Embedding planar graphs at fixed vertex locations. *Graphs Comb.*, 17(4):717–728, 2001.

[Rut20] Ignaz Rutter. Simultaneous embedding. In Seok-Hee Hong and Takeshi Tokuyama, editors, *Beyond Planar Graphs, Communications of NII Shonan Meetings*, pages 237–265. Springer, 2020.

[Sch13] Marcus Schaefer. Toward a theory of planarity: Hanani-tutte and planarity variants. *J. Graph Algorithms Appl.*, 17(4):367–440, 2013.

[SH99] Wei-Kuan Shih and Wen-Lian Hsu. A new planarity test. *Theor. Comput. Sci.*, 223(1-2):179–191, 1999.