Karlsruhe Institute of Technology

Simultaneous PQ-Ordering with Applications to Constrained Embedding Problems

Diploma Thesis of

Thomas Bläsius

At the Department of Informatics
Institute of Theoretical Informatics (ITI)

Reviewer:	Prof. Dr. Dorothea Wagner
Second reviewer:	Prof. Dr. Peter Sanders
Advisor:	Dr. Ignaz Rutter

15. May 2011 – 15. September 2011

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Karlsruhe, den 15. September 2011

Thomas Bläsius

Deutsche Zusammenfassung

Ein PQ-Baum repräsentiert eine Menge von zyklische Ordnungen seiner Blätter, indem die Ordnung der Kanten um einen P-Knoten frei wählbar und um einen Q-Knoten bis auf Umkehrung festgelegt ist. Betrachtet man zwei PQ-Bäume T und T' mit den Blattmengen L beziehungsweise L', sodass jedem Blatt von T' ein Blatt von T zugeordnet ist (zum Beispiel durch die Teilmengenbeziehung $L' \subseteq L$), so induziert jede zyklische Ordnung auf L eine zyklische Ordnung auf L'. Wir bezeichnen T als *Vater* von T' und T' als *Kind* von T. Es stellt sich die Frage, ob eine zyklische Ordnung der Blätter L des Vaters existiert, die von T repräsentiert wird und zusätzlich eine Ordnung der Blätter L' des Kindes induziert, die von T' repräsentiert wird. Für zwei PQ-Bäume lässt sich diese Frage leicht beantworten, doch was passiert, wenn T weitere Kinder oder T' weitere Eltern hat? Diese Frage führt zu dem neuen Problem Simultaneous PQ-Ordering, das wie folgt definiert ist. Gegeben eine Menge an PQ-Bäumen mit Vater-Kind Beziehungen (ein gerichteter azyklische Graph mit PQ-Bäumen als Knoten), können für alle PQ-Bäume simultan zyklische Ordnungen ihrer Blätter gewählt werden, sodass jede Vater-Kind Beziehung berücksichtigt wird?

Wir zeigen, dass Simultaneous PQ-Ordering im allgemeinen $\mathcal{NP}$-schwer ist, für eine Klasse „leichter“ Instanzen jedoch in Polynomialzeit gelöst werden kann. Anschließend wird gezeigt, wie verschiedene Probleme sehr einfach als „leichte“ Instanz von Simultaneous PQ-Ordering formuliert und damit effizient gelöst werden können. Konkret zeigen wir, dass das bedingte Einbettungsproblem Partially PQ-Constrained Planarity für zweifach zusammenhängende Graphen in Linearzeit gelöst werden kann. Des weiteren kann Simultaneous Embedding with Fixed Edges für zweifach zusammenhängende Graphen mit zusammenhängendem Schnitt in quadratischer Zeit gelöst werden. Außerdem verbessern wir die Laufzeit des bisher schnellsten Algorithmus zum Erkennen von simultanen Intervallgraphen von $\mathcal{O}(n^2 \log n)$ auf Linearzeit.

Contents

1. Introduction

Many types of data can be formulated as graphs, such as UML-diagrams in software engineering, evolutionary trees in biology, communication networks or relationships in social networks, to name a few. For a human it is nearly impossible to extract useful information out of such a graph by looking at the pure data. The way a graph is interpreted crucially relies on its visualization. Besides that, creating a small chip with a large number of transistors on it is closely related to the problem of drawing a graph with few bends, few crossings and high resolution (ratio between smallest and largest distances). Thus, drawing graphs and particularly drawing planar graphs is an important filed of research. However, in many applications one needs not only to find a drawing of a given graph but also satisfy additional conditions that are for example specified by a user in an interactive graph drawing system. Such additional conditions lead to constrained embedding problems. Closely related to constrained embeddings are simultaneous embeddings asking for a set of graphs sharing some vertices and edges if they can be drawn simultaneously, such that the common parts are drawn the same in all drawings. This is for example important to compare a dynamic graph at different points in time.

The constrained embedding problem we consider involves PQ-trees. In a PQ-tree every inner node is either a P- or a Q-node and the order of edges around a P-node can be chosen arbitrarily, whereas the order of edges around a Q-node is fixed up to reversal; Figure 1.1a depicts an example. Such a PQ-tree represents a set of possible orders of its leaves. We consider the problem Partially PQ-Constrained Planarity having as input a graph G together with a PQ-tree $T(v)$ for every vertex v with a subset of edges incident to v as leaves. Thus $T(v)$ restricts the possible orders of these edges to the orders that are represented by it. The question is, if G has a planar drawing respecting these restrictions. Furthermore, we consider the problem Simultaneous Embedding with Fixed Edges (SEFE) having two planar graphs $G^{①}$ and $G^{②}$ with a common subgraph G as input, asking whether planar drawings of $G^{①}$ and $G^{②}$ exist, such that the drawing of G is the same in both drawings; see Figure 1.1b for an example. We show how to solve Partially PQ-Constrained Planarity for biconnected graphs. Moreover, we extend the so far known results on Simultaneous Embedding with Fixed Edges by providing a polynomial-time algorithm for the case that both graphs are biconnected and the common graph is connected. To this end, we define the auxiliary problem Simultaneous PQ-Ordering and show how to solve it in polynomial time for "simple" instances, providing a framework to easily achieve the above mentioned results on Partial PQ-Constrained Planarity and Simultaneous Embedding with Fixed Edges. Furthermore, Simultaneous PQ-

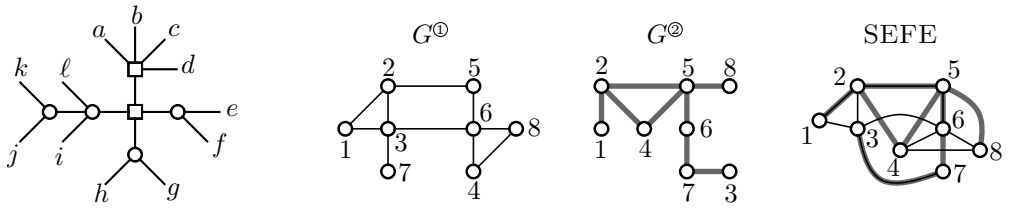

Figure 1.1: (a) A PQ-tree with leaves $\{a, \ldots, \ell\}$ where P- and Q-nodes are depicted as circles and boxes, respectively. For example the degree-5 Q-node at the top enforces the leaves a, b, c, h to occur in this or its reversed order. Furthermore, the two P-nodes on the left enforce the leaves i, j, k, ℓ to appear consecutively. (b) Drawings of two graphs $G^{①}$ and $G^{②}$ on the common node set $\{1, \ldots, 8\}$. Although some of the vertices are drawn to similar positions in both drawings, it is hard to identify the differences and similarities between the two graphs. This is much easier in the SEFE on the right.

Ordering can be used to recognize simultaneous interval graphs in linear time.

Simultaneous PQ-Ordering is defined as follows. Given a PQ-tree T with the leaves L and another PQ-tree T' with leaves $L' \subseteq L$, called a *child* of T, are there orders O and O' of the leaves L and L' represented by the PQ-trees T and T', respectively, such that the order O extends O'? This question is fairly easy to answer, but what happens if T has more than one child or if T' has additional parents? The problem Simultaneous PQ-Ordering asks for a given collection of PQ-trees with child-parent relations whether orders for the PQ-trees can be simultaneously chosen, such that each child-parent relation is satisfied; see Section 3 for a precise definition. We show that Simultaneous PQ-Ordering is $\mathcal{NP}$-complete in general but can be solved efficiently for "simple" instances. As mentioned above, we show how Partially PQ-Constrained Planarity for biconnected graphs, Simultaneous Embedding with Fixed Edges for two biconnected graphs with a connected intersection and the problem of recognizing simultaneous interval graphs can be formulated as "simple" instances of Simultaneous PQ-Ordering.

1.1 Related Work

The quality of a drawing of a graph is hard to measure and of course depends on the specific application. However, there are several generally accepted design goals, such as low number of bends per edge, low number of edge crossings, high resolution (which is equivalent to a small area when drawing on a grid) and good angle resolution (angles between edges incident to a common vertex or angles between crossing edges). Unfortunately, one of the most important design goals, namely crossing minimization, has shown to be $\mathcal{NP}$-hard [GJ83, Bie91] and seems to be really challenging since there are only very few approaches working in practice. Note that it is for example not even known how many crossings are needed in a drawing of the complete graph K_n with n vertices, anyhow, Guy's conjecture [Guy72] stating that $1/4 \cdot \lfloor n/2 \rfloor \cdot \lfloor (n-1)/2 \rfloor \cdot \lfloor (n-2)/2 \rfloor \cdot \lfloor (n-3)/2 \rfloor$ crossings are necessary and sufficient is widely believed. However, testing whether a graph can be drawn without any crossings, that is, testing whether a graph is planar, can be done efficiently. One of the most promising heuristic approaches to crossing minimization starts with a maximal planar subgraph of a given graph and tries to insert the remaining edges optimally [GM04]. The problem of finding the crossing minimal drawing of a planar graph with a single additional edge over all drawings in which the planar graph is drawn crossing free is known as Optimal Edge Insertion and can be done in linear time [GMW01].

Note that inserting a single edge optimally does not mean that the resulting drawing is optimal since an optimal drawing can require that not a single edge is involved in all crossings. This motivates why planar graphs are worth to study them, besides from being interesting on theirselves. One can hope that a result for a problem on a planar graph can be extended somehow to a result for the more challenging problem on general graphs. Note that planar graphs are not only easier to handle for graph drawing problems but also for general graph problems such as maximum cuts or flows [SWK90, BK06]. We start with a short overview about the basic results on planarity.

Planarity

Planarity is a well studied topic. There are several characterizations for planar graphs, such as the well-known theorem of Kuratowski [Kur30] stating that a graph is planar if and only if it does not contain a subdivision of the complete graph K_5 or the complete bipartite graph $K_{3,3}$ as subgraph. Other characterizations are for example in terms of the dimension of partially ordered sets by Schnyder [Sch89] or in terms of a partition of fundamental cycles with respect to a DFS-tree (depth-first search) into clockwise and counter-clockwise oriented cycles by de Fraysseix and Rosenstiehl [dR82, dR85]. The first linear time planarity test is due to Hopcroft and Tarjan [HT74], using the following approach. Consider a cycle in the graph and the components obtained by removing the cycle augmented by the vertices and edges attaching it to the cycle. Then each component can be placed either inside or outside the cycle. Deciding planarity for each of the components and testing if they can be arranged with the cycle in a planar way yields a planarity test for the whole graph. Lempel et al. provided a planarity test iteratively inserting the vertices of the graph in an *st*-order [LEC67]. This can only be done for biconnected graphs, which does not matter since a graph is planar if and only if its biconnected components are planar. The approach by Lempel et al. was improved by Booth and Lueker to run in linear time using PQ-trees [BL76]. Shih and Hsu [SH93] and Boyer and Myrvold [BM99] used a similar approach with a bottom-up order with respect to a DFS-tree instead of an *st*-order. Haeupler and Tarjan provided a common framework for both approaches [HT08]. They start with a completely unembedded graph and add vertices iteratively, such that the unembedded part is always connected, ensuring that the unembedded part can be assumed to lie in the outer face of all embedded components. While inserting vertices, they keep track of the possible embeddings of the embedded parts by representing the possible orders of half embedded edges around every component with a PQ-tree having these edges as leaves. A different approach to planarity testing is the SPQR-tree introduced by Di Battista and Tamassia [DT96a, DT96b]. The SPQR-tree is a decomposition of a biconnected graph into its triconnected components. Furthermore, it represents all planar embeddings of a graph and can be computed in linear time [GM01]. Another linear time planarity testing algorithm is the left-right planarity test [ddR06, Bra09] using the characterization of planar graphs in terms of the orientation of fundamental cycles by de Fraysseix and Rosenstiehl [dR82, dR85].

Fáry showed that every planar graph has a planar drawing such that every line segment is a straight line [Fár48] and there are several algorithms creating straight line drawings of planar graphs with given planar embedding [Tut63, CYN84, CON85]. However, the resulting drawings of these algorithms do not care about the size of the drawing and thus potentially yield exponentially large drawings. De Fraysseix et al. provided an $\mathcal{O}(n \log n)$-time algorithm creating a planar straight-line drawing on a grid of size $\mathcal{O}(n) \times \mathcal{O}(n)$. The running time was improved to linear time by Chrobak and Payne [CP95].

Constrained Planarity

One can conclude that there are several practically usable and fast algorithms creating nice drawings of planar graphs. However, in many applications there are additional requirements. For example a given drawing of a subgraph needs to be extended due to changes in a dynamic graph, the position where an edge enters a node (depicted for example by a rectangle) in a diagram can be given by the application (port-constraints) or the graph needs to be embedded to vertex positions that are (partially) prespecified by the user of an interactive graph drawing system, to name a few. These considerations lead to constrained embedding problems such as Point Set Embedding asking whether a graph has a planar drawing respecting prespecified vertex positions. Pach and Wenger showed that this can be done for every planar graph no matter which vertex positions are given [PW98]. Unfortunately, such a drawing can require linearly many bends per edge. Kaufmann and Wiese proved that two bends per edge are sufficient if only the set of points in the plane is given whereas the mapping of the vertices to these points can be chosen [KW02]. Another constrained embedding problem is Partially Embedded Planarity asking whether a planar drawing of a subgraph can be extended to a planar drawing of the whole graph. Angelini et al. give a linear-time algorithm for testing Partially Embedded Planarity [ADF$^+$10] and Jelínek et al. give a characterization by forbidden substructures similar to Kuratowski's theorem [JKR11].

The problem PQ-Constrained Planarity has as input a planar graph G and a PQ-tree $T(v)$ for every vertex v of G, such that the leaves of $T(v)$ are exactly the edges incident to v. PQ-Constrained Planarity asks whether G has a planar drawing such that the order of incident edges around every vertex v is represented by the PQ-tree $T(v)$. Gutwenger et al. show that PQ-Constrained Planarity can be solved in linear time by simply replacing every vertex with a gadget and testing planarity of the resulting graph [GKM07] (their main result is a solution for Optimal Edge Insertion with these constraints). Furthermore, they show how to deal with PQ-Constrained Planarity if additionally the orientation of some Q-nodes is fixed. In this work, we consider the constrained embedding problem Partially PQ-Constrained Planarity. The difference to PQ-Constrained Planarity is that the tree $T(v)$ may have only a subset of edges incident to v as leaves. Note that Partially PQ-Constrained Planarity is a straight-forward extension of PQ-Constrained Planarity. We show how to solve Partially PQ-Constrained Planarity in linear time if the given graph is biconnected. This solution can be easily extended to also solve the case where the orientation of some Q-nodes is fixed.

Simultaneous Embedding

Closely related to the constrained embedding problems mentioned above are simultaneous embedding problems. Let $G^① = (V^①, E^①)$ and $G^② = (V^②, E^②)$ be two planar graphs with a common subgraph $G = (V, E)$. Simultaneous Embedding (SE for short) ask for planar drawings of $G^①$ and $G^②$ such that the vertices V of the common graph are drawn to the same points in the plane in both drawings. Simultaneous Embedding with Fixed Edges (SEFE) additionally requires the common edges E to be drawn the same in both drawings, whereas for Simultaneous Geometric Embedding (SGE) each edge needs to be drawn as a straight line segment. We also say that $G^①$ and $G^②$ have a SE, SEFE or SGE if there are solutions for these problems. These definitions can be canonically extended to an arbitrary number of graphs, however, SEFE and SGE are challenging enough for two graphs. Besides from being interesting due to the close relation to constrained embedding problems (this relation will be made more precise in a moment), simultaneous embeddings have interesting applications themselves. Consider for example a graph changing over the time. To compare the graph at different points in time for

the purpose of recognizing changes, the parts of the graph remaining need to be drawn the same or at least similar enough to recognize it. Alternatively, the input graphs may represent different relations between the same objects. Note that there are also several practical approaches to simultaneous graph drawing problems [EKLN04, KP05, EBFK09]. The relation to constrained embedding is of the following kind. Fixing the drawing of one of the two graphs reduces SE and SEFE to the constrained embedding problems Point Set Embedding and Partially Embedded Planarity. This shows that every pair of graphs has a SE, since every planar graph can be drawn to arbitrary prespecified vertex positions. Recall that such a drawing can have linearly many bends per edge [PW98], thus it is of interest to find a SE with fewer bends. Erten and Kobourov show that every two graphs can be drawn simultaneously in $\mathcal{O}(n)$ time with at most three bends per edge on a $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ grid ($\mathcal{O}(n^3) \times O(n^3)$ if bends need to be placed on grid points), where n is the number of vertices [EK05]. This result was improved by Di Giacomo and Liotta to at most two bends per edge in general and one bend per edge if $G^①$ and $G^②$ are both outerplanar, both trees or both series-parallel [DL07]. Unfortunately, not every pair of graphs has a SGE and not even a SEFE. Furthermore, the decision problem SEFE is known to be $\mathcal{NP}$-complete for three graphs [GJP+06] and SGE is already $\mathcal{NP}$-hard for two graphs [EBGJ+08]. The complexity of SEFE for two graphs is still open. In most cases, approaches to SEFE and SGE consider special kinds of instances, such as pairs of specified graph classes and there are three types of results. First, that the special instances always have a simultaneous embedding, second, that they do not (by providing a counterexample), and third, testing for such a special instance if it has a simultaneous embedding. To the best of our knowledge no approaches of the third type exist for SGE. Additionally, some of the approaches to SEFE limit the number of bends per edge and the necessary size when drawing on a grid.

Simultaneous Geometric Embedding

Estrella-Balderrama et al. show that SGE is $\mathcal{NP}$-hard even for two graphs. However, there are some additional results on SGE. Erten and Kobourov and Brass et al. give examples for a planar graph and a path not having a SGE [EK05, BCD+07]. Brass et al. furthermore show that a SGE always exists for two caterpillars (graphs being paths after the removal of all degree-1 vertices), k stars, a path and an extended star (collection of stars with an additional special root and paths from the special root to the center of all stars), two paths, and two cycles. On the other hand, Geyer et al. give an example of two trees not having a SGE [GKV09]. This result is further extended by Angelini et al. to the case of a tree and a path [AGKN11]. More precisely, they give an example of a tree of depth 4 and an edge disjoint path not having a SGE. On the other hand they show that every tree of depth 2 has a SGE with every path.

Simultaneous Embedding with Fixed Edges

In the following we give a detailed description of the results on SEFE that are known so far; Figure 1.2 illustrates these results. We start with the instances that are known to always have a SEFE. Erten and Kobourov show that a tree and a path can always be embedded simultaneously [EK05]. They additionally give an algorithm finding a simultaneous embedding in $\mathcal{O}(n)$ time on a grid of size $\mathcal{O}(n) \times \mathcal{O}(n^2)$ such that the edges of $G^①$ and $G^②$ have at most one and zero bends per edge, respectively. Note that a grid of size $\mathcal{O}(n^2) \times \mathcal{O}(n^3)$ is necessary if the bends are required to be drawn on grid points. Di Giacomo and Liotta extend this result to the case of an outerplanar graph and a path with the same grid and bend requirements [DL07]. They extend it further to the case where $G^①$ and $G^②$ are outerplanar and the common graph G is a collection of paths and to the case where $G^①$ is outerplanar and $G^②$ is a cycle. However, in both cases a grid

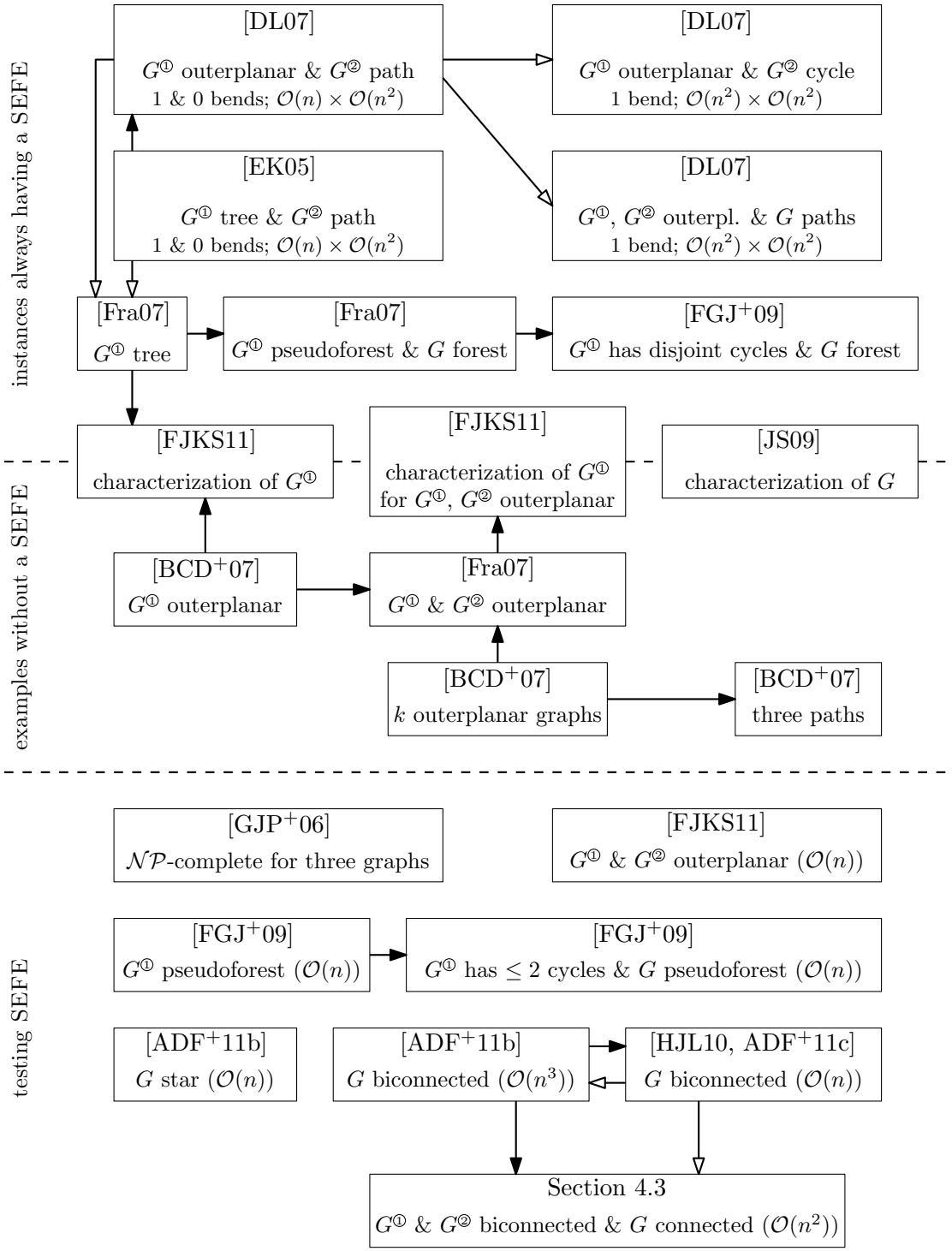

Figure 1.2: Overview over the so far known results on SEFE. Each box represents one result and an arrow highlights that the source-result is extended by the target-result. The arrowheads are empty for the cases in which this is only true, if the number of bends per edge, the consumed grid size or the necessary running time is neglected. Note that transitive arrows are omitted.

of size $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ and up to one bend per edge are required. If the grid and bend requirements are completely neglected, the results considering the pairs tree plus path and outerplanar graph plus path can be extended to the case where one of the two graphs is a tree. Frati shows how a tree $G^{①}$ can be simultaneously embedded with an arbitrary planar graph $G^{②}$ [Fra07]. This algorithm still works if $G^{①}$ contains one additional edge that is not a common edge, yielding the result that every graph with at most one cycle (a pseudoforest) can be embedded simultaneously with every other planar graph if the common graph does not contain this cycle. Fowler et al. extend this result further to the case where $G^{①}$ contains only disjoint cycles and the common graph G does not contain a cycle [FGJ+09].

Aside from instances always having a SEFE, there are also examples that cannot be simultaneously embedded. Brass et al. give examples for k outerplanar graphs, three paths and an outerplanar graph plus a planar graph not having a SEFE [BCD+07]. The results concerning outerplanar graphs can be extended to the case where both graphs are outerplanar [Fra07]. In between the positive and negative results there are some characterizations stating which instances have a SEFE and which do possibly not. Fowler et al. give a characterization of the graphs $G^{①}$ having a SEFE with every other planar graph [FJKS11]. This of course extends all results concerning only $G^{①}$. In particular, the results that a tree can be simultaneously embedded with every other graph whereas an outerplanar graph cannot are extended. This characterization essentially requires that $G^{①}$ must not contain a subgraph homeomorphic to K_3 (a triangle) and an edge not attached to this K_3. The considerations made for this characterization additionally yield a characterization for the outerplanar graphs $G^{①}$ having a simultaneous embedding with every other outerplanar graph $G^{②}$. This of course extends the result that two outerplanar graphs possibly do not have a SEFE. Another characterization, in terms of the common graph, is given by Jünger and Schulz [JS09]. They show that two graphs can be simultaneously embedded if the common graph G has only two embeddings, whereas in all other cases graphs $G^{①}$ and $G^{②}$ with the common graph G not having a SEFE can be constructed. They additionally show that finding a SEFE is equivalent to finding combinatorial embeddings of $G^{①}$ and $G^{②}$ inducing the same combinatorial embedding on the common graph G [JS09, Theorem 4] (note that they use the term topological embedding equivalent to our combinatorial embedding). Note that this is not obvious and not even true for more than two graphs [ADF11a].

Since SEFE has positive and negative instances, it would be nice to have an algorithm deciding for given graphs, whether they can be embedded simultaneously. If more than two graphs are allowed, this problem is known to be $\mathcal{NP}$-complete [GJP+06], whereas the complexity for two graphs is still open. However, there are several results solving SEFE for special cases. For example the characterization of outerplanar graphs having a simultaneous embedding with every other outerplanar graph mentioned above yields a linear time algorithm testing, if two outerplanar graphs have a SEFE [FJKS11]. Fowler at al. showed how to test SEFE, if $G^{①}$ is a pseudoforest, that is, a graph with at most one cycle [FGJ+09]. Note that such an instance has always a SEFE, if this single cycle is not contained in G, as mentioned above. This result can be extended to the case where $G^{①}$ contains up to two cycles, if G does not contain the second cycle, that is, G is a pseudoforest. To achieve this result the following auxiliary problem was solved. Given a planar graph G with a designated cycle C and a partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of the vertices not contained in C, does G admit a planar embedding, such that all vertices in P_i are on the same side of the cycle for every set P_i? Note that this again is a constrained embedding problem, showing that constrained and simultaneous embedding are closely related. Angelini et al. give an algorithm to solve SEFE in linear time if the common graph is a star [ADF+11b]. To this end, they show the equivalence of SEFE for the

case that G is a tree to a constrained version of a 2-Page Book Embedding problem and were able to extend this equivalence to the case where G is connected [ADF+11c]. The problem 2-Page Book Embedding asks for a given graph, whether the vertices can be placed on a straight line (which can be interpreted as the spine of a book) and the edges on the two half planes defined by the line (two pages of the book), such that no two edges cross. Note that this can be seen as an extension of outerplanarity, a graph is outerplanar if and only if only one page is sufficient. 2-Page Book Embedding can be solved in linear time, if additionally the edge partition is fixed [HN09]. Angelini et al. show that SEFE is equivalent to 2-Page Book Embedding with a given edge partition if the common graph G is a star, furthermore, if G is a connected, SEFE is equivalent to a constrained version of 2-Page Book Embedding. Besides this result, Agelini et al. also show how to solve SEFE in $\mathcal{O}(n^3)$ time if G is biconnected [ADF+11b]. They were able to improve this result linear running time [ADF+11c]. Their approach uses the SPQR-tree, choosing an embedding of the common graph (up to a flip) bottom up. A completely different approach is used by Haeupler et al. to solve SEFE in linear time if G is biconnected [HJL10]. Their solution is an extension of the planarity testing algorithm by Haeupler and Tarjan mentioned above [HT08].

In Section 4.3 we show how to solve SEFE for the case that $G^①$ and $G^②$ are biconnected and G is connected. This extends the case where G is biconnected for the following reason. If G is biconnected, then G is completely contained in a single block (maximal biconnected component) of $G^①$ and $G^②$. Thus, even if $G^①$ or $G^②$ are not biconnected, they contain only one block that is of interest, all other blocks can simply be attached to this block. Note that PQ-trees play a central role in the algorithm of Haeupler et al. and in the result we present here. However, we use them in a completely different way. The iterative algorithm by Haeupler et al. uses PQ-trees to keep track of possible edge orders around the parts of the graph that are already embedded. In our approach both graphs $G^①$ and $G^②$ constrain one another with PQ-constraints for the allowed orders of edges around every vertex. This finally results in a large number of PQ-trees describing orders of edges around every vertex and we need to choose orders "fitting" to one another, resulting in the problem Simultaneous PQ-Ordering.

PQ-Trees

Since PQ-Trees play an important role in this work, we given a short overview on what was done about them and what they are used for while their functionality is discussed in Section 2.3. PQ-Trees were originally introduced by Booth and Lueker [BL76]. They were designed to decide if a set L has the Consecutive Ones property with respect to a family $\mathcal{S} = \{S_1, \ldots, S_k\}$ of subsets $S_i \subseteq L$. The set L has this property if a linear order of its elements can be found, such that the elements in each subset $S_i \in \mathcal{S}$ appear consecutively. Booth and Lueker showed how to solve Consecutive Ones in linear time. Furthermore, they showed that all linear orders of the elements in L in which each subset $S_i \in \mathcal{S}$ appears consecutively can be represented by a PQ-tree having the elements in L as leaves. Besides testing planarity in linear time, as mentioned above, they were able to decide in linear time if a given graph is an interval graph. An interval graph is a graph for which each vertex can be represented as an interval of the real numbers, such that there is an edge between two vertices if and only if the corresponding intervals intersect. The relation between interval graphs and the Consecutive Ones property was given by Fulkerson and Gross [FG65]. They showed that a graph G is an interval graph if and only if the maximal cliques of G can be ordered, such that for every vertex v the maximal cliques containing v appear consecutively. Two graphs with common vertices are simultaneous interval graphs, if they have interval representations representing the common vertices by the same intervals. Jampani and Lubiw show that simultaneous interval graphs can be

recognized in $\mathcal{O}(n^2 \log n)$ time [JL10]. We show that it is also possible in linear time.

In the original approach by Booth and Lueker, the PQ-trees were rooted, representing linear orders of its leaves. However, they can also considered to be unrooted representing circular orders. Unrooted PQ-trees are sometimes also called PC-trees [Hsu01, HM01, HM03]. In most cases we will use unrooted PQ-trees representing circular orders while the same results can be achieved for rooted PQ-trees representing linear orders by simply adding a single leaf (see Section 2.3 for further details).

1.2 Outline

This thesis is structured as follows. In Section 2 we define the notation we use and present the known results forming the basis of our work. Section 3 contains the main part of this work. We define the problem Simultaneous PQ-Ordering and show that it is $\mathcal{NP}$-complete in general but efficiently solvable for special instances. In Section 4 we use our results on Simultaneous PQ-Ordering to solve several problems. First, we give a representation of all embeddings of a biconnected planar graph in terms of PQ-trees (Section 4.1). This representation is used to solve Partially PQ-Constrained Planarity for biconnected graphs (Section 4.2) and Simultaneous Embedding with Fixed Edges for the case that both graphs are biconnected and have a connected intersection (Section 4.3). Furthermore, Simultaneous PQ-Ordering can be used to recognize simultaneous interval graphs (Section 4.4). We conclude in Section 5 with some thoughts about open problems.

2. Preliminaries

In this chapter we define the notation and provide some basic tools we use in this work. Section 2.1 deals with graphs and their connectivity, planar graphs and embeddings of planar graphs, directed acyclic graphs and trees. Linear and circular orders and how permutations act on them is considered in Section 2.2. PQ-trees are defined in Section 2.3. Furthermore, the relation between rooted and unrooted PQ-trees is described and operations that can be applied to them are defined. In Section 2.4 we give a short introduction to SPQR-trees, which are used to represent all embeddings of a planar graph. In Section 2.5 we show how PQ- and SPQR-trees are related.

2.1 Graphs, Planar Graphs, DAGs and Trees

An *(undirected) graph* G consists of a set V of *vertices* and a multiset $E \subseteq \{\{u, v\} \mid u, v \in V\}$ of undirected pairwise relations between vertices, called *edges*. The graph G is called simple, if there are no multiple edges, that is, E is a set, and no self loops, that is, $u \neq v$ for every edge $\{u, v\} \in E$. If $u, v \in V$ are connected by an edge $e = \{u, v\}$ they are called *adjacent*, whereas e is said to be *incident* to u and v. A *path* in G is a sequence of adjacent vertices using each edge at most once and the *length* of a path is the number of edges used on the path. A path is *simple* if every vertex occurs at most once. A path of length at least 1 from a vertex to itself is a *cycle* and it is *simple* if every vertex occurs at most once except for the start- and end-vertex occurring twice. The degree of a vertex v, denoted by $\deg(v)$, is the number of edges incident to v. The graph G is *connected* if there is a path from u to v for every pair of vertices $u, v \in V$. A *separating k-set* is a set of k vertices whose removal disconnects G. Separating 1-sets and 2-sets are called *cutvertices* and *separation pairs*. A graph is *biconnected* if it is connected and does not have a cutvertex and it is *triconnected* if it additionally does not have a separation pair. The maximal connected subgraphs (maximal with respect to inclusion) of G are called *connected components* and the maximal biconnected subgraphs are called *blocks*. A subgraph of G that is complete, that is, each pair of vertices is connected by an edge, is called a *clique*. A clique is *maximal* if it is not contained in any other clique. Sometimes we also use the term *node* instead of vertex to emphasize that it represents a larger object.

A *drawing* of a graph G is a mapping of every vertex v to a point (x_v, y_v) in the plane and a mapping of every edge $\{u, v\}$ to a Jordan curve having (x_u, y_u) and (x_v, y_v) as endpoints. A drawing of G is *planar* if edges do not intersect except at common endpoints. The graph G is *planar* if a planar drawing of G exists. Consider G to be a connected planar graph. Every

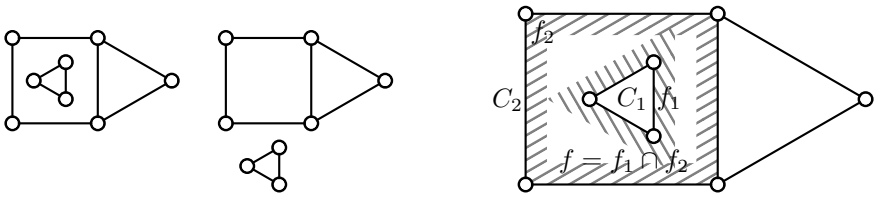

Figure 2.1: (a) Two drawings of a disconnected planar graph with the same order of incident edges around every vertex having different topologies. (b) The component C_1 lies completely inside the face f_2 of C_2 and C_2 lies in the face f_1 of C_1. Considering the whole drawing, they share the common face $f = f_1 \cap f_2$ whereas all other faces of the components remain unchanged.

planar drawing of G splits the plane into several connected regions, called the *faces* of the drawing. Exactly one of these faces, called the *outer face*, is unbounded. The boundary of each face is a cycle in G and two faces in different drawings are said to be the same if they have the same boundary. Additionally, every planar drawing of G induces for every vertex an order of incident edges around it and two drawings inducing the same order for every vertex are called *combinatorially equivalent*. It is clear that two combinatorially equivalent drawings have the same faces, which implies that they have the same topology since G is connected. Note that being combinatorially equivalent is an equivalence relation and the equivalence classes are called *combinatorial embeddings* of G. If G is disconnected, two drawings with the same combinatorial embedding for each connected component can have different topologies, as depicted in Figure 2.1a. To address this problem consider a planar drawing of G consisting of two connected components C_1 and C_2; Figure 2.1b illustrates this case. If we consider the faces of each connected component separately, then C_1 is drawn completely inside one face f_2 of C_2 and C_2 is drawn completely inside one face f_1 of C_1. Thus the drawing of G splits the plane into all the faces of C_1 except for f_1 plus all the faces of C_2 except for f_2 plus the *common face* $f = f_1 \cap f_2$ of the pair $\{C_1, C_2\}$. Note that the drawing of G again has one unbounded face, the outer face. It is clear how to extend this to arbitrary many connected components. Now the combinatorial equivalence can be extended to the case where G has several connected components. Two planar drawings of a possibly disconnected planar graph G are called *combinatorially equivalent*, if each connected component has the same combinatorial embedding and every pair of connected components share the same common face in both drawings. Again, being combinatorially equivalent is an equivalence relation and the equivalence classes are called *combinatorial embeddings*. A combinatorial embedding together with the choice of an outer face is a *planar embedding*. In most cases we do no care about which face is the outer face, thus we mean a combinatorial embedding by simply saying embedding.

Until now, each edge represented an undirected relation between a pair of vertices, but edges can also be directed. Directed edges will be called *arcs* and an arc from the *source* u to the *target* v is denoted by (u, v). A graph with directed edges is called *directed graph*. Paths and cycles are defined as in the undirected case, however, in directed paths and cycles the arcs can only be used from source to target. A directed graph G without any directed cycles is called *directed acyclic graph (DAG)*. Let u and v be vertices of a DAG G such that there exists a directed path from u to v. Then u is called an *ancestor* of v and v a *descendant* of u. If the arc (u, v) is contained in G, then u is a *parent* of v and v is a *child* of u. A vertex v in a DAG G is called *source* (*sink*) if it does not have parents (children). Note that this overloads the term source, but it will be clear from the context which meaning is intended. A *topological ordering* of a DAG G is an ordering of its vertices such that u occurs before v if G contains the arc (u, v). By saying that a

DAG is processed *top down* (*bottom up*) we mean a traversal of its vertices according to a (reversed) topological ordering. Let G be a DAG and let v be a vertex. The *level* of v, denoted by $\text{level}(v)$, is the length of the shortest directed path from a source to v. The *depth* of v, denoted by $\text{depth}(v)$, is the length of the longest directed path from a source to v. Note that the level and the depth have in a sense contrary properties. Let v be a vertex in G and let u be a parent of v. Then the depth of u is strictly smaller than the depth of v whereas the level decreases by at most one: $\text{depth}(u) < \text{depth}(v)$; $\text{level}(u) \geq \text{level}(v) - 1$. By the level and the depth of the DAG G itself we mean the largest level and depth a vertex in G has, respectively.

An *(unrooted) tree* T is a connected graph without any cycles. It is clear that T contains exactly one path between any two vertices. The degree-1 vertices are called *leaves* and the other are *inner vertices*. A tree T together with a special vertex r, called the *root* of T, is a *rooted tree*. A rooted tree can be seen as DAG by directing each edge toward the leaves of the tree. Then the terms top down, bottom up, ancestor, descendant, child, and parent can be defined as for DAGs. Note that a tree with n vertices has $m = n - 1$ edges. However, in general, the ratio between the number of vertices (or edges) and the number of leaves is unbound (consider a tree consisting of a single path). For the special case that T contains no degree-2 vertices we obtain the following lemma.

Lemma 1. *A tree with n_1 leaves and without degree-2 vertices has at most $n_1 - 2$ inner vertices and at most $2n_1 - 3$ edges.*

Proof. Let T be a tree with n_1 leaves and the largest number of edges possible. Then every inner vertex in T has degree 3, because a vertex with four incident edges $e_1, \ldots, e_4$ could be split into two vertices with incident edges e_1, e_2 and e_3, e_4 respectively, plus an additional edge connecting them. Of course T has also the largest possible number of inner vertices for the fixed number of leaves n_1. If we denote the number of vertices of degree 3 by n_3 we have $n = n_1 + n_3$ where n is the total number of vertices. Since T is a tree we have $m = n - 1$ for the number of edges m. Since every edge has two incident vertices we obtain the equation $2m = n_1 + 3n_3$, by counting the incident edges for every vertex. From these three equations we obtain $n_3 = n_1 - 2$ and therefore $m = 2n_1 - 3$. □

2.2 Linear and Circular Orders and Permutations

Let L be a finite set (all sets we consider are finite). A sequence O of all elements in L specifies a relation "$\leq$" on L in the way that $\ell_1 \leq \ell_2$ for $\ell_1 \neq \ell_2 \in L$ if and only if ℓ_2 occurs behind ℓ_1 in O. Such a relation is called *linear order* (or also *total order*) and is identified with the sequence O specifying it. Let O_1 and O_2 be two linear orders on L and let $\ell \in L$ be an arbitrary element. Let further O_i' (for $i = 1, 2$) be the order that is obtained from O_i by concatenating the smallest suffix containing ℓ with the largest prefix not containing ℓ. We call O_1 and O_2 *circularly equivalent* if O_1' and O_2' are the same linear order. Not that this is a equivalence relation not depending on the chosen element ℓ. The equivalence classes are called *circular orders*. For example for $L = \{a, \ldots, e\}$ the orders $O_1 = baedc$ and $O_2 = dcbae$ are circular equivalent and thus define the same circular order since $O_1' = O_2' = aedcb$, if we choose $\ell = a$. In most cases we consider circular orders. Unless stated otherwise, we refer to circular orders by simply writing orders. Note that a linear order can be seen as a graph with vertex set L consisting of a simple directed path, whereas a circular order corresponds to a graph consisting of a simple directed cycle containing L as vertices, see Figure 2.2a for an example. Let L be a set and let O be a circular order of its elements. Let further $S \subseteq L$ be a subset and let O' be the circular order on S that is induced by O. Then O' is a *suborder* of O and O is an *extension* of O'. Note that S does not really need to be a subset of L. Instead it can also be an arbitrary set

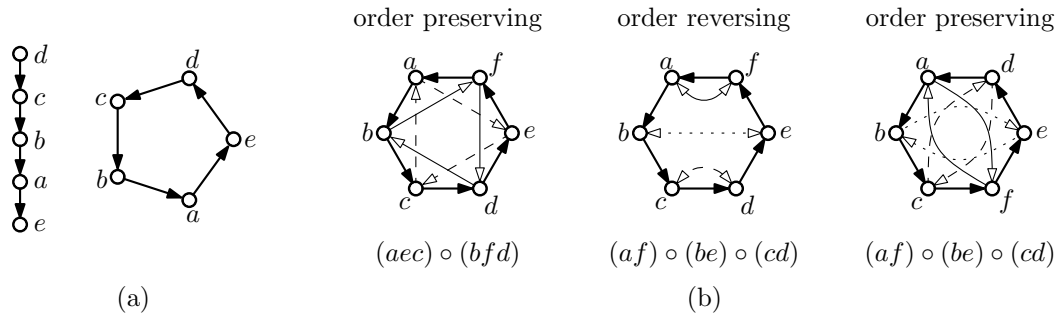

Figure 2.2: (a) The interpretation of the linear and circular order *dcbae* as simple path and simple cycle, respectively. (b) The permutation $\varphi = (aec) \circ (bfd)$ on the left can be seen as as a clockwise rotation by 2 of the circular order *abcdef*, and thus is order preserving, whereas the permutation $\varphi = (af) \circ (be) \circ (cd)$ in the middle is order reversing. However, $\varphi = (af) \circ (be) \circ (cd)$ is not only order reversing but also order preserving (rotation by 3) with respect to the order *abcfed* as shown on the right. The permutations φ are depicted as thin arrows with empty arrowheads and the different permutation cycles are distinct by solid, dashed and dotted lines.

together with an injective map $\varphi : S \to L$. We overload the terms suborder and extension for this case by calling an order O' of S a *suborder* of O and O an *extension* of O' if $\varphi(O')$ is a suborder of O, where $\varphi(O')$ denotes the order obtained from O' by applying φ to each element.

In the following, we consider permutations on the set L and provide some basic properties on how these permutations act on circular orders of L. Let L be a set and let $\varphi : L \to L$ be a permutation. The permutation φ can be decomposed into r disjoint *permutation cycles* $\varphi = (\ell_1 \varphi(\ell_1) \dots \varphi^{k_1}(\ell_1)) \circ \dots \circ (\ell_r \varphi(\ell_r) \dots \varphi^{k_r}(\ell_r))$. We call k_i the *length* of the cycle $(\ell_i \varphi(\ell_i) \dots \varphi^{k_i}(\ell_i))$. Fixpoints for example form a permutation cycle of length 1. We can compute this decomposition by starting with an arbitrary element ℓ and applying φ iteratively until we reach ℓ again. Then we continue with an element not contained in any permutation cycle so far to obtain the next cycle. Now consider a circular order O of the elements in L. The permutation φ is called *order preserving* with respect to O if $\varphi(O) = O$. It is called *order reversing* with respect to O if $\varphi(O)$ is obtained by reversing O. Note that for a fixed order O the order preserving and order reversing permutations are exactly the rotations and reflections of the dihedral group, respectively (the dihedral group is the group of rotations and reflections on a regular k-gon). If we interpret O as a graph as mentioned above, that is, a graph with vertex set L consisting of a simple directed cycle, we obtain that φ is order preserving with respect to O if it is a graph isomorphism on this cycle, whereas the cycle is reversed if φ is order reversing with respect to O; Figure 2.2b depicts this interpretation for an example. We say that φ is *order preserving* or *order reversing* if it is order preserving or order reversing with respect to at least one order O. In this setting the order is not fixed and we want to characterize for a given permutation if it is order preserving or order reversing and additionally we want to find an order that is preserved or reversed, respectively. Note that not fixing the order has for example the effect, that the same permutation φ can be a rotation with respect to one order and a reflection with respect to another, which means that it can be order preserving and order reversing at the same time.

Lemma 2. *A permutation φ on the set L is order preserving if and only if all its permutation cycles have the same length.*

Proof. Assume φ consists of r permutation cycles of length k, let ℓ_i be an element in the

ith permutation cycle. Then φ is order preserving with respect to the following circular order $\ell_1 \dots \ell_r \quad \varphi(\ell_1) \dots \varphi(\ell_r) \quad \dots \quad \varphi^k(\ell_1) \dots \varphi^k(\ell_k)$.

Assume we have a circular order $O = \ell_1 \dots \ell_n$ such that $\varphi(O) = O$. We show that the permutation cycles of two consecutive elements ℓ_i and ℓ_{i+1} have the same size. This claim holds if ℓ_i and ℓ_{i+1} are contained in the same permutation cycle. Assume they are in different permutation cycles with lengths k_i and k_{i+1}, respectively, such that $k_i < k_{i+1}$. Then $\varphi^{k_i}(\ell_{i+1}) \neq \ell_{i+1}$ is not the successor of $\varphi^{k_i}(\ell_i) = \ell_i$ in O. Thus, $\varphi^{k_i}(O)$ cannot be the same circular order O and hence φ is not order preserving, which is a contradiction. □

Lemma 3. *A permutation φ on the set L is order reversing if and only if all its permutation cycles have length 2, except for at most two cycles with length 1.*

Proof. Assume we have $\varphi = (\ell_1 \ell_1') \circ \dots \circ (\ell_r \ell_r')$, $\varphi = (\ell) \circ (\ell_1 \ell_1') \circ \dots \circ (\ell_r \ell_r')$ or $\varphi = (\ell) \circ (\ell') \circ (\ell_1 \ell_1') \circ \dots \circ (\ell_r \ell_r')$. Then φ reverses the orders, $\ell_1 \dots \ell_r \ell_r' \dots \ell_1'$, $\ell_1 \dots \ell_r \ell \ell_r' \dots \ell_1'$ and $\ell_1 \dots \ell_r \ell \ell_r' \dots \ell_1' \ell'$, respectively.

Now assume we have an order O such that φ is order reversing with respect to O, that is it is a reflection in the dihedral group defined by O. Thus, φ^2 is the identity yielding that φ cannot contain a permutation cycle of length greater than 2. Furthermore, a reflection has at most two fixpoints. □

It is clear that the characterizations given in Lemma 2 and Lemma 3 can be easily checked in linear time. Additionally, from the proofs of both lemmas it is clear how to construct an order that is preserved or reversed if the given permutation is order preserving or order reversing, respectively.

2.3 PQ-Trees

Given an unrooted tree T with leaves L having a fixed circular order of edges around every vertex, that is, having a fixed combinatorial embedding, then the circular order of the leaves is also fixed. In an *unrooted PQ-tree* for some inner nodes, the *Q-nodes*, the circular order of incident edges is fixed up to reversal, for the other nodes, the *P-nodes*, this order can be chosen arbitrarily. Hence, an unrooted PQ-tree represents a set of circular orders of its leaves. Given a set L, a set of circular orders $\mathcal{L}$ of L is called *PQ-representable*, if there is an unrooted PQ-tree with leaves L representing it. Formally, the empty set, saying that no order is possible, is represented by the *null tree*, whereas the *empty tree* has the empty set as leaves and represents the set containing only the empty order. A simple example for an unrooted PQ-tree is shown in Figure 2.3a. Note that not every set of orders is PQ-representable. For example a PQ-tree representing a particular order also represents its reversal.

In the same way, we can define a *rooted PQ-tree* representing sets of linear orders by replacing *circular* by *linear* and additionally choosing an inner node of the PQ-tree as root. There is an equivalence between unrooted and rooted PQ-trees in the following sense. Let T be an unrooted PQ-tree with leaves L, representing the set of circular orders $\mathcal{L}$. If we choose one leaf $\ell \in L$ to be the *special leaf*, every circular order in $\mathcal{L}$ can be seen as linear order of $L' := L - \ell$ by breaking the cycle at ℓ. Since every circular order in $\mathcal{L}$ yields a different linear order, we obtain a bijection to a set of linear orders $\mathcal{L}'$. We can construct a rooted PQ-tree T' with the leaves L' representing $\mathcal{L}'$ as follows. First, we choose the special leaf ℓ to be the root of T. Then, for every Q-node we obtain a linear order from the given circular order by breaking the cycle at the (unique) parent. Finally, we remove ℓ and choose its (unique) child as the new root. Hence, given an unrooted PQ-tree, we

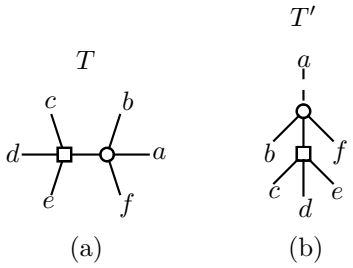

Figure 2.3: (a) An unrooted PQ-tree T with leaves $L = \{a, \dots, f\}$, where P- and Q-nodes are drawn as circles and boxes, respectively. By choosing an order for a, b, f and concatenating it with cde or edc, we obtain all circular orders in $\mathcal{L}$. (b) Choosing a as the special leaf yields the rooted PQ-tree T' with leaves $L' = \{b, \cdots, f\}$. By choosing an arbitrary order for $b, \square, f$ where $\square$ stands for cde or edc, we obtain all orders in $\mathcal{L}'$. Note that this simply means to break the cyclic orders in $\mathcal{L}$ at the special leaf a.

can work with its rooted equivalent instead, by choosing one leaf to be the special leaf; see Figure 2.3 for an example. Conversely, rooted PQ-trees can be represented by unrooted ones by inserting a single leaf. In most cases we can work with the unrooted version of a PQ-tree representing circular sets of orders. Unless stated otherwise, we refer to circular orders and unrooted PQ-trees if we write orders and PQ-trees, respectively.

PQ-trees were introduced by Booth and Lueker [BL76] in the rooted version. Let L be a finite set and let $\mathcal{S} = \{S_1, \dots, S_k\}$ be a family of subsets $S_i \subseteq L$. Booth and Lueker showed that the set $\mathcal{L}$ containing all linear orders in which the elements in each set S_i appear consecutively is PQ-representable. Note that $\mathcal{L}$ could be the empty set, since in no order all subsets S_i appear consecutively, then $\mathcal{L}$ is represented by the null tree. This result can be easily extended to unrooted PQ-trees and circular orders in which the subsets $\mathcal{S}$ appear consecutively, which will become clearer in a moment.

As mentioned above, not every set of orders $\mathcal{L}$ is PQ-representable, but we will see three operations on sets of orders that preserve the property of being PQ-representable. Given a subset $S \subseteq L$, the *projection* of $\mathcal{L}$ to S is the set of orders of S achieved by restricting every order in $\mathcal{L}$ to S. The *reduction* with S is the subset of $\mathcal{L}$ containing the orders where the elements of S appear consecutively. Given two sets of orders $\mathcal{L}_1$ and $\mathcal{L}_2$ on the same set L, their *intersection* is simply $\mathcal{L}_1 \cap \mathcal{L}_2$. That projection, reduction and intersection preserve the property of being PQ-representable can be shown constructively. But first we introduce the following notation, making our life a bit easier. Let T be a PQ-tree with leaf set L, representing $\mathcal{L}$, and let μ be an inner node with incident edges $\varepsilon_1, \dots, \varepsilon_k$. Removing ε_i splits T into two components. We say that the leaves contained in the component not containing μ *belong* to ε_i with respect to μ, and we denote the set of these leaves by $L_{\varepsilon_i,\mu}$. In most cases it is clear which node μ we refer to, so we simply write L_{ε_i}. Note that the sets L_{ε_i} form a partition of L.

Projection Let T be a PQ-Tree with leaves L, representing the set of orders $\mathcal{L}$. The projection to $S \subseteq L$ is represented by the PQ-tree T' that is obtained form T by removing all leaves not contained in S and simplifying the result, where simplifying means, that former inner nodes now having degree 1 are removed iteratively and that degree-2 nodes together with both incident edges are iteratively replaced by single edges. We denote the tree resulting from the projection of T to S by $T|_S$ and we often call $T|_S$ itself the projection of T to S.

Reduction Recall that the reduction with a set S reduces a set of orders to these orders in which all elements in S appear consecutively. The reduction can be seen as the

operation PQ-trees were designed for by Booth and Lueker [BL76]. They showed for a rooted PQ-tree T representing the linear orders $\mathcal{L}$ that the reduction to S is again PQ-representable and the PQ-tree representing it can be computed in $\mathcal{O}(|L|)$ time. For an unrooted PQ-tree T we can consider the rooted PQ-tree T' instead by choosing $\ell \in L$ as special leaf. Note that the reduction with S is equivalent to the reduction with $L \setminus S$ in the unrooted case. There are two possibilities, either ℓ is contained in S or it is not. If it is not, the reduction of T' with S yields the reduction of T with S by simply reinserting ℓ and unrooting T'. If $\ell \in S$, we reduce T' with $L \setminus S$ instead. This shows for a family of subsets $\mathcal{S} = \{S_1, \dots, S_k\}$ that the set containing all circular orders in which each subset $S_i \subseteq L$ appears consecutively can be represented by an unrooted PQ-tree T. Thus, applying a reduction with S to a given PQ-tree T can be seen as adding the subset S to $\mathcal{S}$. Therefore, we denote the result of the reduction of T with S by $T + S$ and we often call $T + S$ itself the reduction of T with S.

Intersection For an inner node μ, all leaves L_ε belonging to an incident edge ε appear consecutively in every order contained in $\mathcal{L}$. Furthermore, if μ is a Q-node with two consecutive incident edges ε and ε', all leaves in $L_\varepsilon \cup L_{\varepsilon'}$ need to appear consecutively. On the other hand, if we have an order of L satisfying these conditions for every inner node, it is contained in $\mathcal{L}$. Hence, T can be seen as a sequence of reductions applied to the set of all orders, which is represented by the star with a P-node as center. Now, given two unrooted PQ-trees T_1 and T_2 with the same leaves, we obtain their intersection by applying the sequence of reductions given by T_1 to T_2. Note that the size of all these reductions can be quadratic in the size of T_1. However Booth showed how they can be applied consuming time linear in the size of T_1 and T_2 [Boo75]. We denote the intersection of T_1 and T_2 by $T_1 \cap T_2$.

Let $T|_S$ be the projection of T to $S \subseteq L$. The extension of an order of S represented by $T|_S$ to an order of L represented by T is straight forward. An inner node in T is either contained in $T|_S$ or it was removed in the simplification step. If a Q-node in T is also contained in $T|_S$, its orientation is determined by the orientation chosen in $T|_S$ and we call it *fixed*, otherwise its orientation can be chosen arbitrarily and we call it *free*. For a P-node not contained in $T|_S$ the order of incident edges can be chosen arbitrarily. If a P-node is contained in $T|_S$, every incident edge is either also contained, was removed or replaced (and the replacement was not removed). The order of the contained and replaced edges is fixed, and the removed edges can be inserted arbitrarily. We call the removed edges (and the edges incident to removed P-nodes) *free* and all other edges *fixed*.

Let $T + S$ be the reduction of a PQ-tree T with leaves L with the subset $S \subseteq L$. Choosing an order in the reduction $T + S$ of course determines an order of the complete leaf set L. Hence, it determines the order of incident edges for every inner node in T. For every Q-node μ in T there exists exactly one Q-node in $T + S$ determining its orientation, we call it the *representative* of μ with respect to the reduction with S and denote it by $\text{rep}_S(\mu)$, where the index is omitted, if it is clear from the context. Note that one Q-node in $T + S$ can be the representative of several Q-nodes in T. For a P-node μ we cannot find such a representative in $T + S$ since it may depend on several nodes in $T + S$. However, if we consider a P-node μ' in $T + S$ there is exactly one P-node μ in T that depends on μ'. We say that μ' *stems* from this P-node μ.

The considerations concerning a PQ-tree T with leaves L together with another PQ-tree T' with leaves $L' \subseteq L$ that is a projection or a reduction of T can of course be extended to the case where T' is obtained from T by a projection followed by a sequence of reductions. This can be further generalized to the case where T and T' are arbitrary PQ-trees with leaves L and L' with an injective map $\varphi : L' \to L$. Note that the injective map ensures that L' can be treated as a subset of L. In this case, we call T' a *child* of T and T a *parent*

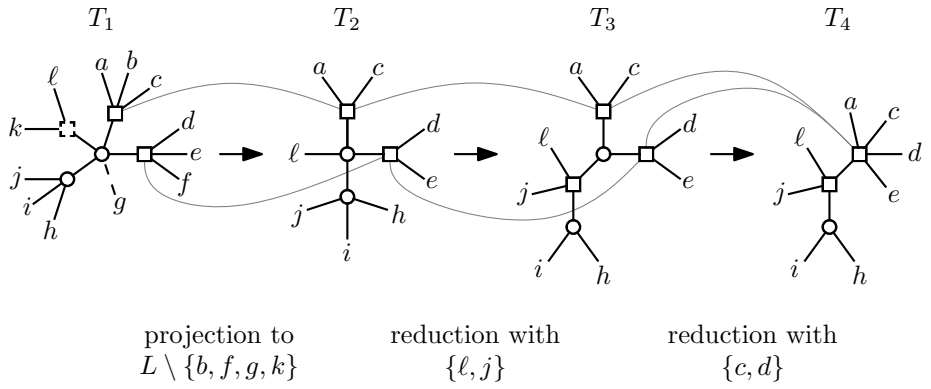

Figure 2.4: We start with the PQ-tree T_1 on the left and project it to $L \setminus \{b, f, g, k\}$ yielding T_2. There is one Q-node and one edge incident to a P-node, both drawn dashed, that do not appear in T_2 and hence are free. The trees T_3 and T_4 are obtained by applying reductions with $\{\ell, j\}$ and $\{c, d\}$ to T_2. Note that the arrows (and even their transitive closure) can be interpreted as child-parent relation between the PQ-trees. Every fixed Q-node has a representative depicted by gray lines, whereas it is not so easy to find something similar for the P-nodes.

of T'. Choosing an order for the leaves L of T induces an order for the leaves L' of T', whereas an order of L' only partially determines an order of L. Now we are interested in all the orders of the leaves L that are represented by T and additionally induce an order for the leaves L' that is represented by T'. Informally spoken, we want to find orders represented by T' and T simultaneously, fitting to one another. It is clear that T' can be replaced by $T' \cap T|_{L'}$ without changing the possible orders, since each possible orders of the leaves L' is of course represented by the projection $T|_{L'}$ of T to L'. Hence this general case reduces to the case where T' is obtained from T by applying a projection and a sequence of reductions. We can extend the notation of free and fixed nodes to this situation as follows. An edge incident to a P-node in the parent T is free with respect to the child T' if and only if it is free with respect to the projection $T|_{L'}$. If all edges are free, the whole P-node is called free. Similarly, a Q-node is free with respect to T' if and only if it is free with respect to $T|_{L'}$. Again, every fixed Q-node μ has a representative $\text{rep}(\mu)$ in T' (which is also a Q-node). Figure 2.4 shows an example PQ-tree together with a projection and a sequence of reductions applied to it.

2.4 SPQR-Trees

Consider a biconnected planar graph G and a split pair $\{s, t\}$, that is, $G - s - t$ is disconnected. Let H_1 and H_2 be two subgraphs of G such that $H_1 \cup H_2 = G$ and $H_1 \cap H_2 = \{s, t\}$. Consider the following tree containing the two nodes μ_1 and μ_2 associated with the graphs $H_1 + \{s, t\}$ and $H_2 + \{s, t\}$, respectively. These graphs are called *skeletons* of the nodes μ_i, denoted by $\text{skel}(\mu_i)$ and the special edge $\{s, t\}$ is said to be a *virtual edge*. The two nodes μ_1 and μ_2 are connected by an edge, or more precisely, the occurrence of the virtual edges $\{s, t\}$ in both skeletons are linked by this edge. Now a combinatorial embedding of G uniquely induces a combinatorial embedding of $\text{skel}(\mu_1)$ and $\text{skel}(\mu_2)$. Furthermore, arbitrary and independently chosen embeddings for the two skeletons determine an embedding of G, thus the resulting tree can be used to represent all embeddings of G by the combination of all embeddings of two smaller planar graphs. This replacement can of course be applied iteratively to the skeletons yielding a tree with more nodes but smaller skeletons associated with the nodes. Applying this kind of decomposition in a systematic way yields the SPQR-tree as introduced by Di Battista and Tamassia [DT96a, DT96b]. The SPQR-tree $\mathcal{T}$ of a biconnected planar graph G contains four types of nodes. First,

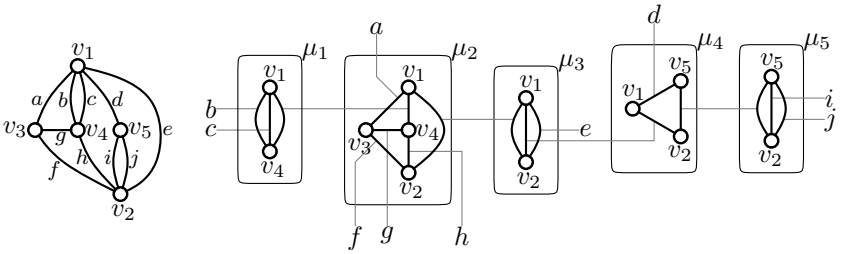

Figure 2.5: A biconnected planar graph on the left and its SPQR-tree on the right. The Q-nodes are depicted as single letters, whereas μ_1, μ_3 and μ_5 are P-nodes, μ_2 is an R-node and μ_4 is an S-node. The embeddings chosen for the skeletons yield the embedding shown for the graph on the left.

the P-nodes having a bundle of at lest three parallel edges as skeleton and a combinatorial embedding is given by any order of these edges. Second, the skeleton of an R-node is triconnected having exactly two embeddings, and third, S-nodes have a simple cycle as skeleton without any choice for the embedding. Finally, every edge in a skeleton representing only a single edge in the original graph G is formally also considered to be a virtual edge linked to a Q-node in $\mathcal{T}$ representing this single edge. Note that all leaves of the SPQR-tree $\mathcal{T}$ are Q-nodes. Besides from being a nice way to represent all embeddings of a biconnected planar graph, the SPQR-tree has only linear size and Gutwenger and Mutzel showed how to compute it in linear time [GM01]. Figure 2.5 shows a biconnected planar graph together with its SPQR-tree.

2.5 Relation between PQ- and SPQR-Trees

Given the SPQR-tree of a biconnected graph, it is easy to see that the set of all possible orders of edges around a vertex is PQ-representable. For a vertex v and a P-node in the SPQR-tree containing v in its skeleton, every virtual edge represents a set of edges incident to v that need to appear consecutively around v. For an R-node in the SPQR-tree containing v, again every virtual edge represents a set of edges that needs to appear consecutively, additionally the order of the virtual edges is fixed up to reversal. Hence, there is a bijection between the P- and R-nodes of the SPQR-tree containing v and the P- and Q-nodes of the PQ-tree representing the possible orders of edges around v, respectively. Note that the occurrence of v in the skeleton of an S-node enforces the edges belonging to one of the two virtual edges incident to v to appear consecutively around v. But since this would introduce a degree-2 node yielding no new constraints, we can ignore the S-nodes. We call the resulting PQ-tree representing the possible circular orders of edges around a vertex v the *embedding tree* of v and denote it by $T(v)$. Figure 2.6 depicts a planar graph together with its SPQR-tree and the resulting embedding trees.

For every planar embedding of G, the circular order of edges around every vertex v is represented by the embedding tree $T(v)$, and for every order represented by $T(v)$ we can find a planar embedding realizing this order. However, we cannot choose orders for the embedding trees independently. Consider for example the case that the order of edges around v_1 in Figure 2.6 is already chosen. Since the embedding tree $T(v_1)$ contains nodes stemming from the P-nodes μ_1 and μ_3 and the Q-node μ_2 in the SPQR-tree, the embedding of the skeletons in these nodes is already fixed. Since every other embedding tree except for $T(v_5)$ contains nodes stemming from one of these three nodes the order of the incident edges around v_2, v_3 and v_4 is at least partially determined. In general, every P-node μ contains two vertices v_1 and v_2 in its skeleton, thus there are two embedding trees $T(v_1)$ and $T(v_2)$ containing the P-nodes μ_1 and μ_2 stemming from μ. The order of virtual edges in skel(μ) around v_1 is the opposite of the order of virtual edges around v_2 for a fixed

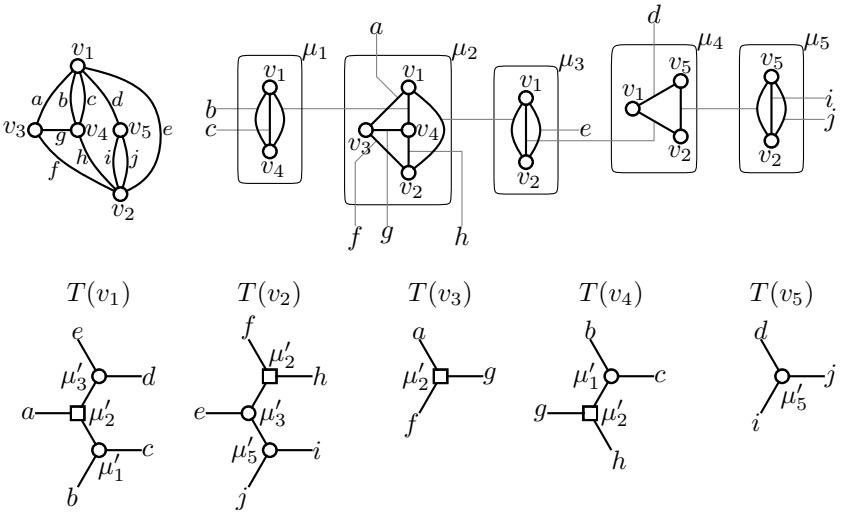

Figure 2.6: The top row shows an example graph G together with its (unrooted) SPQR-tree. The bottom row shows the embedding trees of the five vertices, where the inner nodes are named according to the nodes in the SPQR-tree they stem from.

embedding of skel(μ). Hence, in every planar embedding of G the edges around μ_1 in $T(v_1)$ are ordered oppositely to the order of edges around μ_2 in $T(v_2)$. Similarly, all Q-nodes in the embedding trees stemming from the same R-node in the SPQR-tree need to be oriented the same, if we choose the orders induced by one of the two embeddings of the skeleton as reference orders of the Q-nodes. On the other hand, if every two P-nodes stemming from the same P-node are ordered oppositely and all Q-nodes stemming from the same R-node are oriented the same, we can simply use these orders and orientations to obtain embeddings for the skeleton of every node in the SPQR-tree, yielding a planar embedding of G. Hence, all planar embeddings of G can be expressed in terms of the PQ-trees $T(v_1), \ldots, T(v_n)$, if we respect the additional constraints between nodes stemming from the same node in the SPQR-tree.

3. Simultaneous PQ-Ordering

As described in Section 2.5, we can express all planar embeddings of a biconnected planar graph G in terms of PQ-trees $T(v_1), \ldots, T(v_n)$, called the embedding trees, describing the orders of incident edges around every vertex, if we respect some additional constraints between the nodes in the embedding trees stemming from the same node in the SPQR-tree. In this section, we show how to get completely rid of the SPQR-tree by providing a way to express these additional constraints also in terms of PQ-trees.

The problem Simultaneous PQ-Ordering is defined as follows. Let $D = (N, A)$ be a DAG with nodes $N = \{T_1, \ldots, T_k\}$, where T_i is a PQ-tree representing the set of orders $\mathcal{L}_i$ on its leaves L_i. Every arc $a \in A$ consist of a source T_i, a target T_j and an injective map $\varphi : L_j \to L_i$, and it is denoted by $(T_i, T_j; \varphi)$. Simultaneous PQ-Ordering asks whether there are orders $O_1, \ldots, O_k$ with $O_i \in \mathcal{L}_i$ such that an arc $(T_i, T_j; \varphi) \in A$ implies that $\varphi(O_j)$ is a suborder of O_i. Normally, we want every arc to represent a projection followed by a sequence of reductions, which is not ensured by this definition. Hence, we say that an instance $D = (N, A)$ of Simultaneous PQ-Ordering is *normalized*, if an arc $(T_i, T_j; \varphi) \in A$ implies that $\mathcal{L}_i$ contains an order O_i extending $\varphi(O_j)$ for every order $O_j \in \mathcal{L}_j$. It is easy to see that every instance of Simultaneous PQ-Ordering can be normalized. If there is an order $O_j \in \mathcal{L}_j$ such that $\mathcal{L}_i$ does not contain an extension of $\varphi(O_j)$, then O_j cannot be contained in any solution. Hence, we do not loose solutions by applying the reductions, given by T_i, to T_j. Applying these reductions for every arc in A top down yields an equivalent normalized instance. From now on, all instances of Simultaneous PQ-Ordering we consider are assumed to be normalized. In most cases it is not important to consider the map φ explicitly, hence we often simply write (T_i, T_j) instead of $(T_i, T_j; \varphi)$ and say that O_i is an extension of O_j instead of $\varphi(O_j)$.

Note that we cannot measure the size of an instance D of Simultaneous PQ-Ordering by the number of vertices plus the number of arcs, as it is usual for simple graphs, since the nodes and arcs in D are not of constant size in our setting. The size of every node in D consisting of a PQ-tree T is linear in the number of nodes in T or even linear in the number of leaves by Lemma 1. For every arc $(T_i, T_j; \varphi) \in A$ we need to store the injective map φ from the leaves of T_j to the leaves of T_i. Thus, the size of this arc is linear in the number of leaves in T_j. Finally, the size of D, denoted by $|D|$, can be measured by the size of all nodes plus the sizes of all arcs.

To come back to the embedding trees introduced in Section 2.5, we can now create a PQ-tree consisting of a single Q-node as child of all embedding trees containing a Q-node

stemming from the same R-node in the SPQR-tree. With the right injective maps this additional PQ-tree ensures, that all these Q-nodes are oriented the same. Similarly, we can ensure that two P-nodes are ordered the same, but what we really want is that two P-nodes are ordered oppositely. Therefor we would need something like *reversing arcs* not ensuring that an order is enforced to be the extension of the order provided by the child, but requiring that it is an extension of the reversal of this order. To improve readability we do not consider reversing arcs for now. We will come back to this in Section 3.5 showing what changes if we allow reversing arcs.

Since Simultaneous PQ-Ordering is $\mathcal{NP}$-hard, which will be shown in Section 3.1, we will not solve it in general, but we will give a class of instances for that we can solve it efficiently. In Section 3.2 we figure out the main problems in general instances and provide an approach to solve Simultaneous PQ-Ordering for "simple" instances. In Section 3.3 we make precise which instances we can solve and show how to solve them. In Section 3.4 we give a detailed runtime analysis and in Section 3.5 we show that the results on Simultaneous PQ-Ordering can be extended to the case where we allow "reversing arcs", that is, arcs ensuring that the order of the source is an extension of the reversed order of the target.

3.1 $\mathcal{NP}$-Completeness of Simultaneous PQ-Ordering

Let $L = \{\ell_1, \dots, \ell_n\}$ be a set of elements and let $\Delta = \{(\ell_1^1, \ell_2^1, \ell_3^1), \dots, (\ell_1^d, \ell_2^d, \ell_3^d)\}$ be a set of triples such that each triple $(\ell_1^i, \ell_2^i, \ell_3^i)$ specifies a circular order for these three elements. The problem Cyclic Ordering is to decide whether there is a circular order of all elements in L respecting the circular order specified for every triple in Δ. Galil and Megiddo proved that Cyclic Ordering is $\mathcal{NP}$-complete [GM77].

Theorem 1. Simultaneous PQ-Ordering *is $\mathcal{NP}$-complete.*

Proof. It is clear that Simultaneous PQ-Ordering is in $\mathcal{NP}$ since it can be tested in polynomial time, if the conditions provided by the arcs are satisfied by given circular orders. We show $\mathcal{NP}$-hardness by reducing Cyclic Ordering to Simultaneous PQ-Ordering. Let (L, Δ) be an instance of Cyclic Ordering. We define the corresponding instance $D(L, \Delta)$ of Simultaneous PQ-Ordering as follows. We create one PQ-tree T consisting of a single P-node with leaves L. For every triple $(\ell_1^i, \ell_2^i, \ell_3^i)$ we create a PQ-tree $T(\ell_1^i, \ell_2^i, \ell_3^i)$ consisting of a single node (it does not matter if P- or Q-node) with leaves $\{\ell_1^i, \ell_2^i, \ell_3^i\}$ with an incoming arc $(T, T(\ell_1^i, \ell_2^i, \ell_3^i); \mathrm{id})$, where id is the identity map. With this construction it is still possible to choose an arbitrary order for each of the triples. To

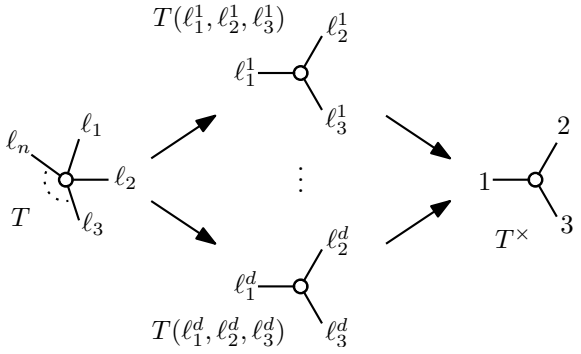

Figure 3.1: The instance $D(L, \Delta)$ of Simultaneous PQ-Ordering corresponding to the instance (L, Δ) of Cyclic Ordering with $L = \{\ell_1, \dots, \ell_n\}$ and $\Delta = \{(\ell_1^1, \ell_2^1, \ell_3^1), \dots, (\ell_1^d, \ell_2^d, \ell_3^d)\}$.

ensure that they are all ordered the same, we introduce an additional PQ-tree $T^\times$ consisting of a single node with three leaves 1, 2 and 3 and an incoming arc $(T(\ell_1^i, \ell_2^i, \ell_3^i), T^\times; \varphi)$ with $\varphi(j) = \ell_j^i$ for every triple $(\ell_1^i, \ell_2^i, \ell_3^i)$. Figure 3.1 illustrates this construction. It is clear that the size of $D(L, \Delta)$ is linear in the size of (L, Δ). It remains to show that the instance (L, Δ) of Cyclic Ordering and the instance $D(L, \Delta)$ of Simultaneous PQ-Ordering are equivalent.

Assume we have a solution of (L, Δ), that is, we have a circular order O of L such that every triple $(\ell_1^i, \ell_2^i, \ell_3^i) \in \Delta$ has the circular order $\ell_1^i \ell_2^i \ell_3^i$. The PQ-tree T in $D(L, \Delta)$ has the leaves L, thus we can choose O as the order of the leaves of T. For every triple $(\ell_1^i, \ell_2^i, \ell_3^i)$ there is an incoming arc from T to $T(\ell_1^i, \ell_2^i, \ell_3^i)$ inducing the circular order $\ell_1^i \ell_2^i \ell_3^i$ on its leaves. Furthermore, there is an outgoing arc to $T^\times$ inducing the order 123. Since all of these arcs having $T^\times$ as target induce the same circular order 123, these orders are a solution of the instance $D(L, \Delta)$ of Simultaneous PQ-Ordering.

Conversely, assume we have a solution for $D(L, \Delta)$. If the order of leaves in $T^\times$ is 132, we obtain another solution by reversing all orders. Thus, we can assume without loss of generality that the leaves of $T^\times$ have the order 123. Hence, the leaves of the tree $T(\ell_1^i, \ell_2^i, \ell_3^i)$ are ordered $\ell_1^i \ell_2^i \ell_3^i$ for every triple $(\ell_1^i, \ell_2^i, \ell_3^i)$ implying that the order on the leaves L of T, which is an extension of all these orders, is a solution of the instance (L, Δ) of Cyclic Ordering. □

3.2 Critical Triples and the Expansion Graph

Although Simultaneous PQ-Ordering is $\mathcal{NP}$-complete we give in this section a strategy how to solve it for special instances. Afterwards, in Section 3.3, we show that this strategy really leads to a polynomial time algorithm for a class of instances. Let $D = (N, A)$ be an instance of Simultaneous PQ-Ordering and let $(T, T_1) \in A$ be an arc. By choosing an order $O_1 \in \mathcal{L}_1$ and extending O_1 to an order $O \in \mathcal{L}$, we ensure that the constraint given by the arc (T, T_1) is satisfied. Hence, our strategy will be to choose orders bottom up, which can always be done for a single arc since our instances are normalized. Unfortunately, T can have several children $T_1, \ldots, T_\ell$, and orders $O_i \in \mathcal{L}_i$ represented by T_i for $i = 1, \ldots, \ell$ cannot always be simultaneously extended to an order $O \in \mathcal{L}$ represented by T. We derive necessary and sufficient conditions for the orders O_i to be simultaneously extendable to an order $O \in \mathcal{L}$ under the additional assumption that every P-node in T is fixed with respect to at most two children. We consider the Q- and P-nodes in T separately.

Let μ be a Q-node in T. If μ is fixed with respect to T_i, there is a unique Q-node $\text{rep}(\mu)$ in T_i determining its orientation. By introducing a boolean variable x_η for every Q-node η, which is TRUE if η is oriented the same as a fixed reference orientation and FALSE otherwise, we can express the condition that μ is oriented as determined by its representative by $x_\mu = x_{\text{rep}(\mu)}$ or $x_\mu \neq x_{\text{rep}(\mu)}$. For every Q-node in T that is fixed with respect to a child T_i we obtain such an (in)equality and we call the resulting set of (in)equalities the *Q-constraints*. It is obvious that the Q-constraints are necessary. On the other hand, if the Q-constraints are satisfied, all children of T fixing the orientation of μ fix it in the same way. Note that the Q-constraints form an instance of 2-Sat that has linear size in the number of Q-nodes, which can be solved in polynomial [Kro67] and even linear [EIS76, APT79] time. Hence, we only need to deal with the P-nodes, which is not as simple.

Let μ be a P-node in T. If μ is fixed with respect to only one child T_i, we can simply choose the order given by O_i. If μ is additionally fixed with respect to T_j, it is of course necessary that the orders O_i and O_j induce the same order for the edges incident to μ that are fixed with respect to both, T_i and T_j. We call such a triple (μ, T_i, T_j), where μ is a P-node in T

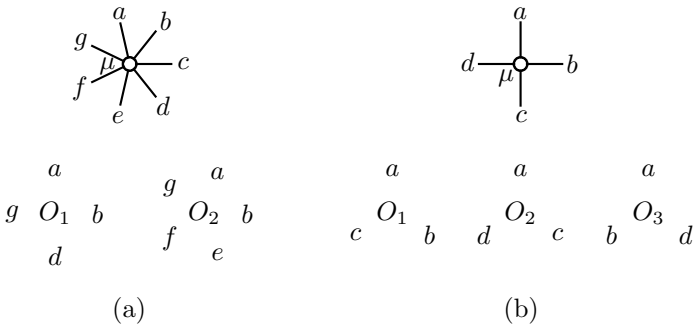

Figure 3.2: (a) We can find an order for the P-node μ extending the orders O_1 and O_2 if and only if the commonly fixed edges a, b and g are ordered the same.
(b) Although for every pair $\{O_i, O_j\}$ of orders out of the three orders O_1, O_2 and O_3 the commonly fixed edges are ordered the same, we cannot extend all three orders simultaneously.

fixed with respect to the children T_i and T_j a *critical triple*. We say that the critical triple (μ, T_i, T_j) is *satisfied* if the orders O_i and O_j induce the same order for the edges incident to μ commonly fixed with respect to T_i and T_j. If we allow multiple arcs, we can also have a critical triple (μ, T', T') for two parallel arcs $(T, T'; \varphi_1)$ and $(T, T'; \varphi_2)$. Clearly, all critical triples need to be satisfied by the orders chosen for the children to be able to extend them simultaneously. Note that this condition is not sufficient, if μ is contained in more than one critical triple, which is one of the main difficulties of Simultaneous PQ-Ordering for general instances. However, the following lemma shows that satisfying all critical triples is not only necessary but also sufficient, if every P-node is contained in at most one critical triple, that is, it is fixed with respect to at most two children of T. See Figure 3.2 for two simple examples, illustrating that satisfying critical triples is sufficient if every P-node is contained in at most one critical triple, whereas the general case is not as simple.

Lemma 4. *Let T be a PQ-tree with children $T_1, \ldots, T_\ell$, such that every P-node in T is contained in at most one critical triple, and let $O_1, \ldots, O_\ell$ be orders represented by $T_1, \ldots, T_\ell$. An order O that is represented by T and simultaneously extends the orders $O_1, \ldots, O_\ell$ exists if and only if the Q-constraints and all critical triples are satisfied.*

Proof. The only if part is clear, since an order O represented by T extending the orders $O_1, \ldots, O_\ell$ yields an assignment of TRUE and FALSE to the variables x_η satisfying the Q-constraints. Additionally, for every critical triple (μ, T_i, T_j) the common fixed edges are ordered the same in O as in O_i and in O_j and hence (μ, T_i, T_j) is satisfied.

Now, assume that we have orders $O_1, \ldots, O_\ell$ satisfying the Q-constraints and every critical triple. We show how to construct an order O represented by T, extending all orders $O_1, \ldots, O_\ell$ simultaneously. The variable assignments for the variables stemming from Q-nodes in each of the children $T_1, \ldots, T_\ell$ imply an assignment of every variable stemming from a fixed Q-node in T, and hence an orientation of this Q-node. Since the Q-constraints are satisfied, all children fixing a Q-node in T imply the same orientation. The orientation of free Q-nodes can be chosen arbitrarily. For a P-node μ in T that is fixed with respect to at most one child of T, we can simply choose the order of fixed edges incident to μ as determined by the child and add the free edges arbitrarily. Otherwise, μ is contained in exactly one critical triple (μ, T_i, T_j). We first choose the order of edges incident to μ that are fixed with respect to T_i as determined by O_i. From the point of view of T_j, some of the fixed edges incident to μ are already ordered, but this order is consistent with the order induced by O_j, since (μ, T_i, T_j) is satisfied. Additionally, some edges that are free

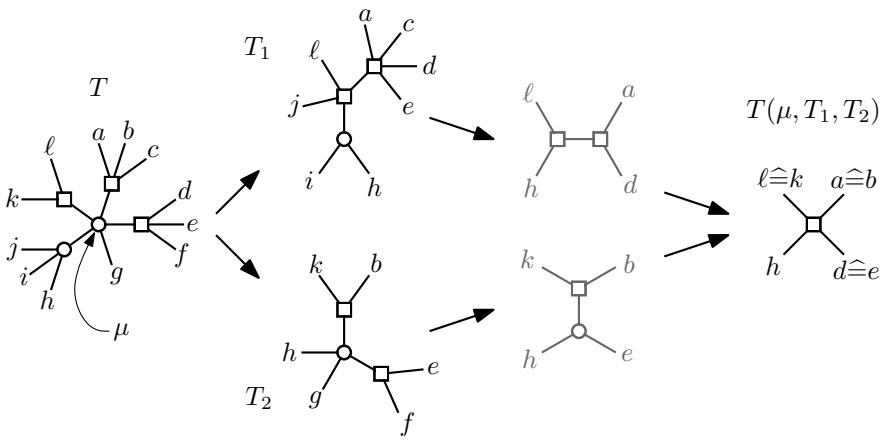

Figure 3.3: The P-node μ in the PQ-tree T is fixed with respect to the children T_1 and T_2. We first project T_1 and T_2 to representatives of the common fixed edges incident to μ and intersect the result to obtain $T(\mu, T_1, T_2)$. Note that the gray shaded projections only illustrate an intermediate step, they are not inserted.

with respect to T_j are already ordered. Of course, the remaining edges incident to μ that are fixed with respect to T_j can be added as determined by O_j, and the free edges can be added arbitrarily. □

Since testing whether the Q-constraints are satisfiable is easy, we concentrate on satisfying the critical triples. Let μ be a P-node in a PQ-tree T such that μ is fixed with respect to two children T_1 and T_2, that is, (μ, T_1, T_2) is a critical triple. By projecting T_1 and T_2 to representatives of the common fixed edges incident to μ and intersecting the result, we obtain a new PQ-tree $T(\mu, T_1, T_2)$. There are natural injective maps from the leaves of $T(\mu, T_1, T_2)$ to the leaves of T_1 and T_2, hence we can add $T(\mu, T_1, T_2)$ together with incoming arcs from T_1 and T_2 to our instance D of Simultaneous PQ-Ordering. This procedure of creating $T(\mu, T_1, T_2)$ is called *expansion step* with respect to the critical triple (μ, T_1, T_2), and the resulting new PQ-tree $T(\mu, T_1, T_2)$ is called the *expansion tree* with respect to that triple; see Figure 3.3 for an example of the expansion step. We say that the P-node μ in T is *responsible* for the expansion tree $T(\mu, T_1, T_2)$. Note that every expansion tree has two incoming and no outgoing arcs at the time it is created.

We introduce the expansion tree for the following reason. If we find orders O_1 and O_2 represented by T_1 and T_2 that both extend the same order represented by the expansion tree $T(\mu, T_1, T_2)$, we ensure that the edges incident to μ fixed with respect to both, T_1 and T_2, are ordered the same in O_1 and O_2, or in other words, we ensure that O_1 and O_2 satisfy the critical triple (μ, T_1, T_2). By Lemma 4, we know that satisfying the critical triple is necessary, thus we do not loose solutions by adding expansion trees to an instance of Simultaneous PQ-Ordering. Furthermore, it is also sufficient, if every P-node is contained in at most one critical triple (if we forget about the Q-nodes for a moment). Hence, given an instance D of Simultaneous PQ-Ordering, we would like to expand D iteratively until no unprocessed critical triples are left and find simultaneous orders bottom up. Unfortunately, it can happen that the expansion does not terminate and thus yields an infinite graph; see Figure 3.4 for an example. Thus, we need to define a special case where we do not expand further. Let μ be a P-node of T with outgoing arcs $(T, T_1; \varphi_1)$ and $(T, T_2; \varphi_2)$ such that (μ, T_1, T_2) is a critical triple. Denote the leaves of T_1 and T_2 by L_1 and L_2, respectively. If T_i (for $i = 1, 2$) consists only of a single P-node, the image of φ_i is a set of representatives of the edges incident to μ that are fixed with respect to T_i. Hence φ_i is a bijection between L_i and the fixed edges incident to μ. If

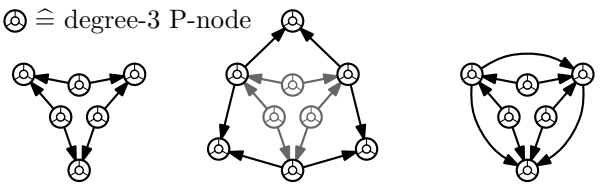

Figure 3.4: Consider the instance of Simultaneous PQ-Ordering on the left, where every PQ-tree consists of a single P-node with degree 3. The DAG in the center shows the result after expanding three times. The so far processed part is shaded gray and for the remaining part we are in the same situation as before, hence iterated expansion would yield an infinite DAG. To prevent infinite expansion we apply finalizing steps resulting in the DAG on the right.

additionally the fixed edges with respect to both, T_1 and T_2, are the same, we obtain a bijection $\varphi : L_1 \to L_2$. Assume without loss of generality that there is no directed path from T_2 to T_1 in the current DAG. If there is neither a directed path from T_1 to T_2 nor form T_2 to T_1, we achieve uniqueness by assuming that T_1 comes before T_2 with respect to some fixed order of the nodes in D. Instead of an expansion step we apply a *finalizing step* by simply creating the arc $(T_1, T_2; \varphi)$. This new arc ensures that the critical triple (μ, T_1, T_2) is satisfied if we have orders for the leaves L_1 and L_2 respecting $(T_1, T_2; \varphi)$. Since no new node is inserted, we do not run into the situation where we create the same PQ-tree over and over again.

For the case that (μ, T', T') is a critical triple resulting from two parallel arcs $(T, T'; \varphi_1)$ and $(T, T'; \varphi_2)$, we can apply the expansion step as described above. If the conditions for a finalizing step are given, that is T' consists of a single P-node and both maps φ_1 and φ_2 fix the same edges incident to μ, a finalizing step would introduce a self loop with the permutation φ associated with it. We distinguish between the two cases that T' is an expansion tree and that it was already contained in D. If it is an expansion tree, we do nothing and mark the critical triple as processed. Otherwise, we apply an expansion step having the effect that the resulting expansion tree again satisfies the conditions to apply a finalizing step and additionally is an expansion tree. Note that doing nothing instead of the finalizing step has the effect that the critical triple (μ, T', T') is not automatically satisfied by choosing orders bottom up. We say that the two arcs $(T, T'; \varphi_1)$ and $(T, T'; \varphi_2)$ form a *critical double arc* with the permutation φ belonging to it. Since we want to apply Lemma 4 by choosing orders bottom up, it is a problem that the critical triple belonging to a critical double arc is in general not satisfied. Fortunately, we can show for the instances we want to solve that the target T' of a critical double arc is a sink and no further expansion or finalizing steps can change that. Hence, we are free to choose any order for the leaves of T' and we will use Lemma 2 (about order preserving permutations) to choose it in a way satisfying the critical triple or decide that this is impossible.

To sum up, we start with an instance D of Simultaneous PQ-Ordering. As long as D contains unprocessed critical triples (μ, T_1, T_2) we apply expansion steps (or finalizing steps if T_1 and T_2 are essentially the same) and mark (μ, T_1, T_2) as processed. The resulting graph is called the *expansion graph* of D and is denoted by D_{exp}. Note that D_{exp} is also an instance of Simultaneous PQ-Ordering. Before showing in Lemma 7 that D and D_{exp} are equivalent, we need to show that D_{exp} is well defined, that is, it is unique and finite. Lemma 5 essentially states that the P-nodes become smaller at least every second expansion step. We will use this result in Lemma 6 to show finiteness.

Lemma 5. *Let D be an instance of* Simultaneous PQ-Ordering *and let D_{exp} be its expansion graph. Let further T be a PQ-tree in D_{exp} containing a P-node μ. If μ is responsible for an expansion tree T' containing a P-node μ' with $\deg(\mu') = \deg(\mu)$, then μ'*

itself is not responsible for an expansion tree T'' containing a P-node μ'' with $\deg(\mu'') = \deg(\mu') = \deg(\mu)$.

Proof. Since T' is created by first projecting a child of T to representatives of edges incident to μ, it can contain at most $\deg(\mu)$ leaves. Thus, if T' contains a P-node μ' with $\deg(\mu') = \deg(\mu)$, it contains no other inner node. Now assume that μ' is responsible for another expansion tree T'' containing a P-node μ'' with $\deg(\mu'') = \deg(\mu') = \deg(\mu)$ and let (μ', T_1, T_2) be the corresponding critical triple. Again T'' consists only of the single P-node μ''. Since T_1 and T_2 lie on a directed path from T' to T'' they also need to consist of single P-nodes with $\deg(\mu')$ incident edges. Thus, T_1 and T_2 consist both of a single P-node having the same degree and they fix the same, namely all, edges incident to μ'. Hence we would have applied a finalizing step instead of creating the expansion tree T''; a contradiction. □

Lemma 6. *The expansion graph D_{exp} of an instance $D = (N, A)$ of* Simultaneous PQ-Ordering *is unique and finite.*

Proof. If we apply an expansion or a finalizing step due to a critical triple (μ, T_1, T_2), where μ is a P-node of the PQ-tree T, the result does only depend on the trees T, T_1 and T_2 and the arcs (T, T_1) and (T, T_2). By applying other expansion or finalizing steps, we of course do not change these trees or arcs, thus it does not matter in which order we expand and finalize a given DAG D. Hence, D_{exp} is unique and we can talk about *the* expansion graph D_{exp} of an instance D of Simultaneous PQ-Ordering.

To prove that D_{exp} is finite, we show that $\text{level}(D_{\text{exp}}) \leq \text{level}(D) + 4 \cdot (p_{\max} + 1)$, where $p_{\max}$ is the degree of the largest P-node in D. To simplify the notation denote $p_{\max} + 1$ by $p^+_{\max}$. Recall that the level of a node in D was defined as the the shortest directed path from a sink to this node and $\text{level}(D)$ is the largest level occurring in D. Note that all sources in D_{exp} are already contained in D, since every expansion tree has two incoming arcs. Showing that the level of D_{exp} is finite is sufficient since there are only finitely many sources in D_{exp} and no node has infinite degree. Assume we have a PQ-tree T_1 in D_{exp} with $\text{level}(T_1) > \text{level}(D) + 4 \cdot p^+_{\max}$. Then T_1 is of course an expansion tree and there is a unique P-node μ_2 that is responsible for T_1. Denote the PQ-tree containing μ_2 by T_2. Since there is a directed path of length 2 from T_2 to T_1, we have $\text{level}(T_2) \geq \text{level}(T_1) - 2 > \text{level}(D) + 4 \cdot p^+_{\max} - 2$. Due to its level, T_2 itself needs to be an expansion tree and we can continue, obtaining a sequence $T_1, \ldots, T_{2 \cdot p^+_{\max}}$ of expansion trees containing P-nodes μ_i, such that μ_i is responsible for T_{i-1}. Due to Lemma 5 the degree of μ_i is larger than the degree of μ_{i-2}, hence $\deg(\mu_{2 \cdot p^+_{\max}}) \geq p^+_{\max} > p_{\max}$, which is a contradiction to the assumption that the largest P-node in D has degree $p_{\max}$. □

Now that we know that the expansion graph D_{exp} of a given instance D of Simultaneous PQ-Ordering is well defined, we can show what we already mentioned above, namely that D and D_{exp} are equivalent.

Lemma 7. *An instance D of* Simultaneous PQ-Ordering *admits simultaneous PQ-orders if and only if its expansion graph D_{exp} does.*

Proof. It is clear that D is a subgraph of D_{exp}. Hence, if we have simultaneous orders for the expansion graph D_{exp}, we of course also have simultaneous orders for the original instance D.

It remains to show that we do not loose solutions by applying expansion or finalizing steps. Therefore, assume we have simultaneous orders for the original instance D. Since

every expansion tree is a descendant of a PQ-tree in D, for which the order is already fixed, there is no choice left for the expansion trees. Thus, we only need to show that for every expansion tree all parents induce the same order on its leaves and that this order is represented by the expansion tree. We first show this for the expansion graph without the arcs inserted due to finalizing steps. Afterwards, we show that adding these arcs preserves valid solutions.

Consider an expansion tree $T(\mu, T_1, T_2)$ introduced due to the critical triple (μ, T_1, T_2) such that T_1, T_2 and the tree T containing μ are not expansion trees. By construction $T(\mu, T_1, T_2)$ represents the edges incident to μ fixed with respect to T_1 and T_2. Since the orders chosen for T_1, T_2 and T are valid simultaneous orders, T_1 and T_2 induce the same order for the leaves of $T(\mu, T_1, T_2)$. Since $T(\mu, T_1, T_2)$ has no other incoming arcs, we do not need to consider other parents. The induced order is of course represented by the projection of T_1 and T_2 to the commonly fixed edges incident to μ, and hence it is of course also represented by their intersection $T(\mu, T_1, T_2)$. For the case that T, T_1 or T_2 are expansion trees, we can assume by induction that the orders chosen for T, T_1 and T_2 are valid simultaneous orders, yielding the same result that T_1 and T_2 induce the same order represented by $T(\mu, T_1, T_2)$. It remains to show, that the arcs introduced by a finalizing step respect the chosen orders. Let $T(\mu, T_1, T_2)$ a critical triple such that T_1 and T_2 consist of single P-nodes both fixing the same edges in μ. It is clear that the order chosen for μ induces the same order for T_1 and T_2 with respect to the canonical bijection φ between the leaves of T_1 and T_2. Hence, adding an arc $(T_1, T_2; \varphi)$ preserves simultaneous PQ-orders. □

For now, we know that we can consider the expansion graph instead of the original instance to solve Simultaneous PQ-Ordering. Lemma 4 motivates that we can solve the instance given by the expansion graph by simply choosing orders bottom up, if additionally the Q-constraints are satisfiable. However, this only works for "simple" instances since satisfying critical triples is no longer sufficient for a P-node that is fixed with respect to more than two children. And there is another problem, namely that the expansion graph can become exponentially large. In the following section we will define precisely what "simple" means and additionally address the second problem by showing that the expansion graph has polynomial size for these instances.

3.3 1-Critical Instances

The expansion graph was introduced to satisfy the critical triples simply by choosing orders bottom up, which can then be used to apply Lemma 4, if the additional condition that every P-node is contained in at most one critical triple is satisfied. Let D be an instance of Simultaneous PQ-Ordering and let D_{exp} be its expansion graph. We say that D is a *1-critical instance*, if in its expansion graph D_{exp} every P-node is contained in at most one critical triple. We will first prove a lemma helping us, to deal with critical double arcs. Afterwards, we show how to solve 1-critical instances efficiently.

Lemma 8. *Let D be a 1-critical instance of* Simultaneous PQ-Ordering *with expansion graph D_{exp}. Let further $(T, T'; \varphi_1)$ and $(T, T'; \varphi_2)$ be a critical double arc. Then T' is a sink in D_{exp}.*

Proof. Since T' consists only of a single P-node, there is exactly one P-node μ in T that is fixed with respect to T'. Due to the double arc, μ is contained in the critical triple (μ, T', T'). The tree T' is an expansion tree by construction, hence at the time T' is created it has only the two incoming arcs $(T, T'; \varphi_1)$ and $(T, T'; \varphi_2)$ and no outgoing arc. Assume that we can introduce an outgoing arc to T' by applying an expansion or finalizing

step. Then T' needs to be contained in another critical triple than (μ, T', T') and since T is its only parent and μ is the only P-node in T fixed with respect to T', this critical triple must also contain μ. But then μ is contained in more than one critical triple, which is a contradiction to the assumption that D is 1-critical. □

Lemma 9. *Let D be a 1-critical instance of* Simultaneous PQ-Ordering *with expansion graph D_{exp}. In time polynomial in $|D_{\text{exp}}|$ we can compute simultaneous PQ-orders or decide that no such orders exist.*

Proof. Due to Lemma 7, we can solve the instance D_{exp} of Simultaneous PQ-ordering instead of D itself. Of course we cannot find simultaneous PQ-orders for the PQ-trees in D_{exp} if any of these PQ-trees is the null tree. Additionally, Lemma 4 states that the Q-constraints are necessary. We can check in linear time whether there exists an assignment of TRUE and FALSE to the variables x_μ, where μ is a Q-node, satisfying the Q-constraints by solving a linear size instance of 2-Sat [EIS76, APT79]. Hence, if D_{exp} contains the null tree or the Q-constraints are not satisfiable, we know that there are no simultaneous PQ-orders. Additionally, we need to deal with the critical double arcs. Let $(T, T'; \varphi_1)$ together with $(T, T'; \varphi_2)$ be a critical double arc. By construction, the target T' consists of a single P-node fixing the same edges incident to a single P-node μ in T with respect to both edges. Thus, φ_1 and φ_2 can be seen as bijections between the leaves L' of T' and the fixed edges incident to μ and hence they define a permutation φ on L' with $\varphi = \varphi_2^{-1} \circ \varphi_1$. To satisfy the critical triple (μ, T', T'), we need to find an order O' of L' such that $\varphi_1(O') = \varphi_2(O')$. This equation is equivalent to $\varphi_1 \circ \varphi(O') = \varphi_2 \circ \varphi(O')$, and hence also to $\varphi(O') = O'$. Thus, the critical triple (μ, T', T') is satisfied if and only if φ is order preserving with respect to O'. Whether φ is order preserving with respect to any order can be tested in $\mathcal{O}(|L'|)$ time by applying Lemma 2. Now assume we have a variable assignment satisfying the Q-constraints, no PQ-tree is the null tree and every permutation φ corresponding to a critical double arc is order preserving. We show how to find simultaneous PQ-orders for all PQ-trees in D_{exp}.

Start with a sink T in D_{exp}. If T is the target of a critical double arc, it is a single P-node and its corresponding permutation φ is order preserving by assumption and hence we can use Lemma 2 to choose an order that is preserved by φ. Otherwise, orientate every Q-node μ in T as determined by the variable x_μ stemming from it. Additionally, choose an arbitrary order for every P-node in T. Afterwards mark T as processed. We continue with a PQ-tree T in D_{exp} for which all of its children $T_1, \ldots, T_\ell$ are already processed, that is, we traverse D_{exp} bottom up. Since $T_1, \ldots, T_\ell$ are processed, orders $O_1, \ldots, O_\ell$ for their leaves were already chosen. Consider a P-node μ in T contained in a critical triple (μ, T_i, T_j). If there is the expansion tree $T(\mu, T_i, T_j)$, it guarantees that the edges incident to μ fixed with respect to T_i and T_j are ordered the same in O_i and O_j and hence the critical triple is satisfied. If we had to apply a finalizing step due to the critical triple (μ, T_i, T_j), we have an arc from T_i to T_j (or in the other direction), again ensuring that O_i and O_j induce the same order on the fixed edges incident to μ. In the special case that (μ, T_i, T_j) corresponds to a critical double arc, we know due to Lemma 8 that $T_i = T_j$ is a sink. Then the critical triple is also satisfied, since we chose an order that is preserved by the permutation φ corresponding to the critical double arc. Thus, all critical triples containing P-nodes in T are satisfied. Additionally, the Q-constraints are satisfied and since D is 1-critical every P-node μ in T is contained in at most one critical triple. Hence, we can apply Lemma 4 to extend the orders $O_1, \ldots, O_\ell$ simultaneously to an order O represented by T. This extension can clearly be computed in polynomial time and hence D_{exp} can be traversed bottom up choosing an order for every PQ-tree in polynomial time in the size of D_{exp}. □

As mentioned above, the expansion graph can be exponentially large for instances that are not 1-critical, which can be seen as follows. Assume a P-node μ in the PQ-tree T is fixed with respect to three children T_1, T_2 and T_3. Then this P-node is responsible for the three expansion trees $T(\mu, T_1, T_2)$, $T(\mu, T_1, T_3)$ and $T(\mu, T_2, T_3)$. So every layer can be three times larger than the layer above, hence the expansion graph may be exponentially large even if there are only linearly many layers. But if we can ensure that μ is fixed with respect to at most two children of T, that is, it is contained in at most one critical triple, it is responsible for only one expansion tree. Of course, the resulting expansion tree can itself contain several P-nodes that can again be responsible for new expansion trees. We first prove a technical lemma followed by a lemma stating that the size of the expansion graph remains quadratic in the size of D for 1-critical instances.

Lemma 10. *If μ is a P-node responsible for an expansion tree T containing the P-nodes $\mu_1, \dots, \mu_k$, the following inequality holds.*

$$\sum_{i=1}^{k} \deg(\mu_i) \leq \deg(\mu) + 2k - 2$$

Proof. Let $\eta_1, \dots, \eta_\ell$ be the Q-nodes contained in T and let n_1 be the number of leaves in T. Let further n and m denote the number of vertices and edges in T, respectively. We obtain the following equation by double counting.

$$n_1 + \sum_{i=1}^{k} \deg(\mu_i) + \sum_{i=1}^{\ell} \deg(\eta_i) = 2m \tag{3.1}$$

Since T is a tree, we can replace m by $n-1$ and due to the fact that every node in T is either a leaf, a P-node or a Q-node, we can replace n further by $n_1 + k + \ell$. With some additional rearrangement we obtain the following from Equation (3.1).

$$\sum_{i=1}^{k} \deg(\mu_i) = n_1 + 2k - 2 + 2\ell - \sum_{i=1}^{\ell} \deg(\eta_i) \tag{3.2}$$

The tree T has at most $\deg(\mu)$ leaves since it is obtained by projecting some PQ-tree to representatives of the edges incident to μ, yielding the inequality $n_1 \leq \deg(\mu)$. Additionally, we have the inequality $2\ell - \sum \deg(\eta_i) \leq 0$ since $\deg(\eta_i) \geq 3$. Plugging these two inequalities into Equation (3.2) yields the claim. □

Lemma 11. *Let D be a 1-critical instance of* Simultaneous PQ-Ordering. *The size of its expansion graph D_{exp} is quadratic in $|D|$.*

Proof. We first show that the total size of all expansion trees is in $\mathcal{O}(|D|^2)$. Afterwards, we show that the size of all arcs that are contained in D_{exp} but not in D is linear in the total size of all expansion trees in D_{exp}.

Every expansion tree T in D_{exp} has a P-node that is responsible for it. If this P-node is itself contained in an expansion tree, we can again find another responsible P-node some layers above. Thus, we finally find a P-node μ that was already contained in D, which is *transitively responsible* for the expansion tree T. Every PQ-tree for which μ is transitively responsible can have at most $\deg(\mu)$ leaves, thus its size is linear in $\deg(\mu)$ due to Lemma 1. Furthermore, we show that μ can only be transitively responsible for $\mathcal{O}(\deg(\mu))$ expansion trees, and thus for expansion trees of total size $\mathcal{O}(\deg(\mu)^2)$. With this estimation it is clear that the size of all expansion trees is quadratic in the size of D. To make it more precisely, denote the number of PQ-trees μ is transitively responsible for by $\text{resp}(\mu)$. We show by induction over $\deg(\mu)$ that $\text{resp}(\mu) \leq 3\deg(\mu) - 8$.

A P-node μ with $\deg(\mu) = 3$ can be responsible for at most one PQ-tree, thus $\text{resp}(\mu) \leq 3\deg(\mu) - 8$ is satisfied. If μ has $\deg(\mu) > 3$ incident edges, it is directly responsible for at most one expansion tree T, since our instance is 1-critical. In the special case that T consists of a single P-node μ' with $\deg(\mu') = \deg(\mu)$, the PQ-tree for which μ' is responsible cannot again contain a P-node of degree $\deg(\mu)$ due to Lemma 5. Otherwise, T contains k P-nodes $\mu_1, \ldots, \mu_k$ with $\deg(\mu_i) < \deg(\mu)$. In the special case, $\text{resp}(\mu) = \text{resp}(\mu') + 1$ holds and we show the inequality $\text{resp}(\mu) \leq 3\deg(\mu) - 8$ for both cases by showing $\text{resp}(\mu) \leq 3\deg(\mu) - 9$ for the second case. In the second case, μ is transitively responsible for T and all the PQ-trees $\mu_1, \ldots, \mu_k$ are responsible for, yielding the following equation.

$$\text{resp}(\mu) = 1 + \sum_{i=1}^{k} \text{resp}(\mu_i)$$

Plugging in the induction hypothesis $\text{resp}(\mu_i) \leq 3\deg(\mu_i) - 8$ yields the following inequality.

$$\text{resp}(\mu) \leq 1 + 3\sum_{i=1}^{k} \deg(\mu_i) - 8k$$

If $k = 1$, this inequality directly yields the claim $\text{resp}(\mu) \leq 3\deg(\mu) - 9$ since $\deg(\mu_1) \leq \deg(\mu) - 1$. Otherwise, we can use Lemma 10 to obtain $\text{resp}(\mu) \leq 3\deg(\mu) - 5 - 2k$. This again yields the claim $\text{resp}(\mu) \leq 3\deg(\mu) - 9$ since $k > 1$. Finally, we have that the induction hypothesis holds for μ, and hence every P-node is transitively responsible for $\mathcal{O}(\deg(\mu))$ expansion trees of size $\mathcal{O}(\deg(\mu))$.

For an arc that is contained in D_{exp} but not in D consider the critical triple (μ, T_1, T_2) that is responsible for it. Since μ is not contained in another critical triple, it is only responsible for the arcs $(T_1, T(\mu, T_1, T_2))$ and $(T_2, T(\mu, T_1, T_2))$ or (T_1, T_2) in the case of a finalizing step. The size of these arcs is in $\mathcal{O}(\deg(\mu))$ since the expansion tree contains at most $\deg(\mu)$ leaves and, if the finalizing step is applied, T_1 and T_2 are single P-nodes of degree at most $\deg(\mu)$. Hence, the size of newly created arcs in D_{exp} is linear in the size of all PQ-trees in D_{exp}, which concludes the proof. □

Putting Lemma 9 and Lemma 11 together directly yields the following theorem. For a detailed runtime analysis see Section 3.4, showing that quadratic time is sufficient, which is not as obvious as it seems to be.

Theorem 2. Simultaneous PQ-Ordering *can be solved in polynomial time for 1-critical instances.*

Actually, Theorem 2 tells us how to solve 1-critical instances, which was the main goal of this section. However, the characterization of the 1-critical instances is not really satisfying, since we need to know the expansion graph, which may be exponentially large, to check whether an instance is 1-critical or not. For our applications we can ensure that all instances are 1-critical and hence do not need to test it algorithmically. But to prove for an application that all instances are 1-critical, it would be much nicer to have conditions for 1-criticality of an instance that are defined for the instance itself and not for some other structure derived from it. In the remaining part of this section we will provide sufficient conditions for an instance to be 1-critical that do not rely on the expansion graph.

Let $D = (N, A)$ be an instance of Simultaneous PQ-Ordering. Let further T be a PQ-tree with a parent T' and let μ be a P-node in T. Recall that there is exactly one P-node μ' in T' it stems from, that is, μ' is fixed with respect to μ and no other P-node in T' is fixed with respect to μ. Note that there may be several P-nodes in T stemming

from μ'. Consider a P-node μ in the PQ-tree $T \in N$ such that T is a source in D. We say that the P-node μ is *k-fixed* if it is fixed with respect to k children of T. Alternatively, we say that the *fixedness* of μ is k and denote it by $\text{fixed}(\mu) = k$. We recursively extend this definition to the case where T has parents $T_1, \ldots, T_\ell$ as follows. The P-node μ stems from exactly one P-node μ_i in T_i, for every $i = 1, \ldots, \ell$. Assume μ itself is fixed with respect to k' children of T. Then μ is defined to be k-fixed for $k = k' + \sum(\text{fixed}(\mu_i) - 1)$. The motivation for this definition is that a P-node with fixedness k in D is fixed with respect to at most k children in the expansion graph D_{exp}. We obtain the following theorem providing sufficient conditions for D to be a 1-critical instance.

Theorem 3. *Let D be an instance of* Simultaneous PQ-Ordering*, such that every P-node in every PQ-tree in D is at most 2-fixed. Then D is 1-critical.*

Proof. Let D_{exp} be the expansion graph of D. We need to show for every P-node μ in D_{exp} that it is contained in at most one critical triple, that is, it is fixed with respect to at most two children. We will show that separately for the cases where the tree T containing μ is already contained in D and where T is an expansion tree.

Assume that T is already contained in D. It is clear that μ is fixed with respect to at most two children in D, since it is at most 2-fixed, but it may happen that T has additional children in D_{exp}. We will show by induction over the depth of the node T in D_{exp} that μ has at most $\text{fixed}(\mu)$ children fixing it in D_{exp}. Recall that the depth of a node in a DAG is defined as the length of the longest directed path from a source to this node. For sources in D it is clear that the number of children fixing a P-node does not increase by expanding D, which shows the base case. For the general case let $T_1, \ldots, T_\ell$ be the parents of T and let $\mu_1, \ldots, \mu_\ell$ be the corresponding P-nodes μ stems from. Let further μ be fixed with respect to k' children of T in D. By the definition of fixedness we have $\text{fixed}(\mu) = k' + \sum(\text{fixed}(\mu_i) - 1)$. Note that $\text{fixed}(\mu_i) \geq 1$ for every $i = 1, \ldots, \ell$ since μ_i is at least fixed with respect to T and note further, that T_i has by induction at most $\text{fixed}(\mu_i)$ children fixing μ_i. Thus, μ_i can be contained in at most $\text{fixed}(\mu_i) - 1$ critical triples also containing T, which means, that μ_i can be responsible for at most $\text{fixed}(\mu_i) - 1$ children of T in D_{exp}. Hence, T can have in D_{exp} at most $k' + \sum(\text{fixed}(\mu_i) - 1) = \text{fixed}(\mu)$ children fixing μ. By the assumption that $\text{fixed}(\mu) \leq 2$ we obtain that μ is contained in at most one critical triple in D_{exp}.

Now consider the case where T is an expansion tree with P-node μ. At the time T is created, it has two incoming and no outgoing arcs, denote the parents by T_1 and T_2, and the P-nodes μ stems from by μ_1 and μ_2, respectively. Again we show by induction over the depth of T in D_{exp} that T has at most two children fixing μ. In the base case, T_1 and T_2 are both already contained in D. As shown above, μ_1 and μ_2 can each be contained in at most one critical triple, hence expansion can introduce at most two children fixing μ. In the general case, a parent T_i for $i = 1, 2$ is either contained in D or an expansion graph. In the first case it again can introduce at most one child fixing μ, in the second case we can apply the induction hypothesis with the same result. Note that in a finalizing step for one of the trees a new incoming arc is created instead of an outgoing arc. But this incoming arc can itself of course be responsible for at most one outgoing arc, hence the number of children fixing a P-node cannot become larger than two. Finally, we have that every P-node in every PQ-tree in D_{exp} is fixed with respect to at most two children, hence D is 1-critical. □

Theorem 3 and Theorem 2 together provide a framework to solve problems that can be formulated as instances of Simultaneous PQ-Ordering. We can use Theorem 3 to prove that the instances our application produces are 1-critical, whereas Theorem 2 tells us that we can solve these instances in polynomial time.

3.4 Implementation Details

To solve an instance of Simultaneous PQ-Ordering, we first normalize the instance, then compute the expansion graph and finally choose orders bottom up. As shown in Lemma 11 the size of the expansion graph is quadratic in the size of D. All other steps that need to be applied are simple, such as projection, intersection or the extension of an order. All these steps run in linear time, but unfortunately linear in the size of the parent. For example, in the normalization step the projection of a tree T to the leaves of its child T' must be computed, consuming linear time in $|T|$. Since T can be a large PQ-tree with many small children we need quadratic time. A similar problem arises when computing an expansion tree due to a critical triple (μ, T_1, T_2). To compute $T(\mu, T_1, T_2)$ the trees T_1 and T_2 need to be projected to representatives of the commonly fixed edges incident to μ, consuming $\mathcal{O}(|T_1| + |T_2|)$ time. Since the resulting expansion tree $T(\mu, T_1, T_2)$ can be arbitrarily small, these costs cannot be expressed in terms of $|T(\mu, T_1, T_2)|$. But since T_1 and T_2 can have linearly many expansion trees as children we potentially need quadratic time for each PQ-tree in D_{exp} to compute the expansion graph, yielding an $\mathcal{O}(|D|^4)$ time algorithm. Another problem is the extension of orders bottom up. If a PQ-tree T has one child T' with chosen order, it is easy to extend this order to T in $|T|$ time. However, T can have linearly many children, yielding an algorithm consuming quadratic time per PQ-tree and thus overall again $\mathcal{O}(|D|^4)$ time. However, if additionally the projection $T|_{L'}$ of T to the leaves L' of T' is known, the order chosen for T' can be extended in $\mathcal{O}(|T'|)$ time to $T|_{L'}$. Furthermore, the extension of orders from several projections of T to T can be done in time linear in the size of all projections, if some additional projection information are stored. In this section we show how to compute the normalization in quadratic time, which is straight forward. Afterwards, we give a more detailed estimation for the size of the expansion graph of 1-critical instances. Then, we show that computing the expansion graph for 1-critical instances actually runs in quadratic time. Furthermore, we show for the normalization and the expansion that for every arc the projection of the parent to the leaves of the child together with additional projection information can be computed and stored without consuming additional time. This information can then be used to choose orders bottom up in linear time in the size of the expansion graph. Altogether, this yields a quadratic time algorithm to solve 1-critical instances of Simultaneous PQ-Ordering.

In the remaining part of this section let $D = (N, A)$ be a 1-critical instance of Simultaneous PQ-Ordering with the expansion graph $D_{\text{exp}} = (N_{\text{exp}}, A_{\text{exp}})$. Let further $|D|$, $|N|$, $|A|$, $|D_{\text{exp}}|$, $|N_{\text{exp}}|$ and $|A_{\text{exp}}|$ denote the size of D, N, A, D_{exp}, N_{exp} and A_{exp}, respectively. Recall that the size of a node is linear in the size of the contained PQ-tree and the size of an arc is linear in the size of its target, which is due to the injective map that needs to be stored for every arc. Furthermore, let $p_{\max}$ be the degree of the largest P-node in D and let $\#N$ denote the number of nodes in D.

Normalization

As mentioned above, we want to compute and store some additional information besides computing the normalization. In detail, let (T, T') be an arc and let L' be the leaves of T'. For every node in the projection $T|_{L'}$ of T to the leaves of T' there is a node in T it stems from and for every edge incident to a P-node in the projection there is an edge incident to the corresponding P-node in T it stems from. We say that the arc (T, T') has *additional projection information*, if $T|_{L'}$ with a pointer from every node and edge to the node and edge in T it stems from is known. Note that the arc (T, T') does not become asymptotically larger due to additional projection information. In the following, being a normalized instance of Simultaneous PQ-Ordering includes that every arc has additional projection information. The following lemma is not really surprising.

Lemma 12. *An instance $D = (N, A)$ of* Simultaneous PQ-Ordering *can be normalized in $\mathcal{O}(\#N \cdot |N|)$ time.*

Proof. To normalize an instance D of Simultaneous PQ-Ordering we need to project T to the leaves of T' and intersect the result with T' for every arc (T, T') in D. The projection can be done in $\mathcal{O}(|T|)$ time, while the intersection consumes $\mathcal{O}(|T'|)$ time. Note that the additional projection information can be simply stored directly after computing the projection. Since T may have $\#N$ children all these projections consume $\mathcal{O}(\#N \cdot |T|)$ time. Summing over all PQ-trees yields $\mathcal{O}(\#N \cdot |N|)$ for the normalization of D. □

Size of the Expansion Graph

In Lemma 11 we already showed that the expansion graph of a 1-critical instance has quadratic size. However, this can be done more precisely.

Lemma 13. *Let D be a 1-critical instance of* Simultaneous PQ-Ordering *with the expansion graph D_{exp}. It holds $|D_{\text{exp}}| \in \mathcal{O}(p_{\max} \cdot |N| + |A|)$, where $p_{\max}$ is the degree of the largest P-node in D.*

Proof. The proof of Lemma 11 shows that every P-node μ can be transitively responsible for at most $3\deg(\mu) - 8$ expansion trees where each of these expansion trees has size $\mathcal{O}(\deg(\mu))$. Thus, μ is responsible for expansion trees of total size $\mathcal{O}(\deg(\mu)^2)$. To compute the total size of all expansion trees we need to sum over all P-nodes $\mu_1, \dots, \mu_\ell$ that are already contained in D. The following estimations show the claimed size of $\mathcal{O}(p_{\max} \cdot |N|)$.

$$\sum_{i=1}^{\ell} \deg(\mu_i)^2 \leq p_{\max} \cdot \sum_{i=1}^{\ell} \deg(\mu_i) \leq p_{\max} \cdot |N|$$

As mentioned in the proof of Lemma 11 the size of all newly created arcs in D_{exp} is linear in the size of all nodes in D_{exp}. Thus we obtain $|D_{\text{exp}}| \in \mathcal{O}(p_{\max} \cdot |N| + |A|)$ for the whole expansion graph. □

Computing the Expansion Graph

When computing the expansion tree $T(\mu, T_1, T_2)$ due to the critical triple (μ, T_1, T_2) we need to project T_1 and T_2 to the representatives of the commonly fixed edges incident to μ. Let T denote the tree containing μ and let L_1 and L_2 be the leaves of T_1 and T_2, respectively. First, we need to find the commonly fixed edges and a representative for each. Assume that the projections $T|_{L_1}$ and $T|_{L_2}$ are stored as ensured by the normalization. Then for every edge incident to μ it can be easily tested in constant time, if it is contained in both projections, consuming $\mathcal{O}(\deg(\mu))$ time overall. With a simple traversal of $T|_{L_i}$ (for $i = 1, 2$) representatives of these commonly fixed edges can be found in $\mathcal{O}(|T_i|)$ time and the projection of T_i to these representatives can also be done in $\mathcal{O}(|T_i|)$ time. The intersection of the two projections yields $T(\mu, T_1, T_2)$ in $\mathcal{O}(|T(\mu, T_1, T_2)|)$ time, which can be neglected. For the two newly created arcs $(T_1, T(\mu, T_1, T_2))$ and $(T_2, T(\mu, T_1, T_2))$ we again need to ensure that the additional projection information are stored. However, this projection was already computed and can simple be stored without additional running time. Hence the total running time for computing the expansion tree $T(\mu, T_1, T_2)$ is in $\mathcal{O}(\deg(\mu) + |T_1| + |T_2|)$. Thus, a superficial analysis yields quadratic running time in the size of the expansion graph. However, we can do better, as shown in the following lemma.

Lemma 14. *The expansion graph D_{exp} of an 1-critical instance $D = (N, A)$ of* Simultaneous PQ-Ordering *can be computed in $\mathcal{O}(|N|^2)$ time.*

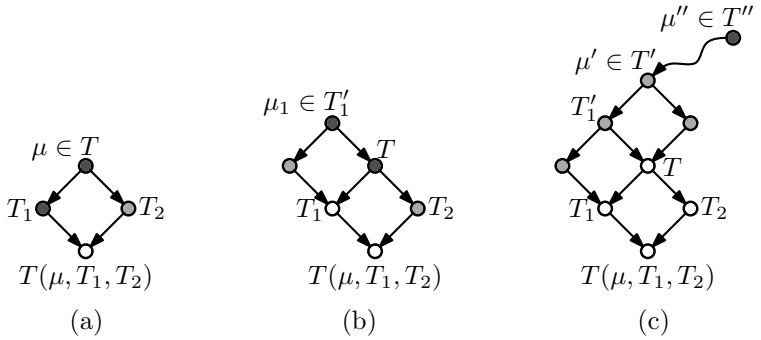

Figure 3.5: Nodes in the original graph are shaded dark gray and expansion trees white. Light gray is used where it does not matter. (a) The case where T_1 is contained in the original graph. (b) The case where T_1 is an expansion graph but T containing μ is not. (c) The case where neither T_1 nor T are expansion graphs.

Proof. As mentioned above, computing the expansion tree $T(\mu, T_1, T_2)$ consumes $\mathcal{O}(\deg(\mu) + |T_1| + |T_2|)$ time. We consider this time as cost and show how to assign it to different parts of D defining them to be responsible for this cost. The cost $\mathcal{O}(\deg(\mu))$ can be simply assigned to μ. Since every P-node μ is contained in at most one critical triple this can happen at most once, yielding linear cost in total. Assume without loss of generality that $|T_1| \geq |T_2|$. In this case we only need to assign the cost $\mathcal{O}(|T_1|)$. To do that, we consider three cases.

If $T_1 \in N$, that is, T_1 is not an expansion tree, then we assign the cost $\mathcal{O}(|T_1|)$ to T_1. This can happen at most as many times as T_1 occurs in a critical triple. In each of these critical triples there necessarily is a P-node that is contained in a PQ-tree in a parent of T_1. There can be $\mathcal{O}(|N|)$ of these P-nodes and since every P-node is contained in at most one critical triple the total cost assigned to T_1 is in $\mathcal{O}(|N| \cdot |T_1|)$. Note that no expansion tree is responsible for any cost, thus by summing over all PQ-trees in D we obtain that the total cost is in $\mathcal{O}(|N|^2)$. Figure 3.5a illustrates this case.

If $T_1 \notin N$ but $\mu \in T \in N$, that is, T_1 is an expansion tree, but the P-node μ is contained in the original graph D. Then T_1 has exactly two parents, like every other expansion tree, and of course one of them is the tree T containing the P-node μ. Furthermore, there is a P-node μ_1 responsible for T_1; let T'_1 be the PQ-tree containing μ_1. Thus T_1 was created due to a critical triple containing μ_1 and T, and T'_1 containing μ_1 needs to be a parent of T as depicted in Figure 3.5b. In this case we assign the cost $\mathcal{O}(|T_1|)$ to T'_1 or more precisely to μ_1. Since T was already contained in the original graph, we also have $T'_1 \in N$, thus again, only PQ-trees from the original graphs are responsible for any costs. Since T_1 is obtained by projecting T and its other parent to representatives of edges incident to μ_1 we have that $|T_1| \in \mathcal{O}(\deg(\mu_1))$. Due to the fact that μ_1 is contained in at most one critical triple it is overall responsible for $\mathcal{O}(\deg(\mu_1))$ cost and hence we obtain only linear cost by summing over all P-nodes in all PQ-trees in D.

If $T_1 \notin N$ and $\mu \in T \notin N$, that is, T_1 is an expansion tree and μ is contained in an expansion tree. In other words, we are somehow "far away" from the original graph. With the same argument as before, we can find a P-node μ' in a PQ-tree T' that is responsible for the PQ-tree T containing μ and this PQ-tree needs to be a parent of the PQ-tree T'_1; see Figure 3.5c. If T' again is an expansion tree, we can find a P-node responsible for it and so on, until we reach a P-node μ'' in the PQ-tree T'' that is transitively responsible for T and T', such that T'' is already contained in the graph D. Then we assign the cost $\mathcal{O}(|T_1|)$ to T'' or more precisely to μ''. Since T_1 is a child of T its size needs to be linear in $|T|$. Furthermore, since μ'' is transitively responsible for T, we have $|T| \in \mathcal{O}(\deg(\mu''))$. Thus

we assign cost linear to $\deg(\mu'')$ to μ''. As shown for Lemma 11 μ'' can be transitively responsible for at most $3\deg(\mu'') - 8$ expansion trees, thus it is overall responsible for $\mathcal{O}(\deg(\mu'')^2)$ cost. Note that again only PQ-trees in D are responsible for any costs. Thus by summing over all P-nodes in all PQ-trees we obtain $\mathcal{O}(p_{\max} \cdot |N|)$.

To sum up, the costs from the first case are dominating, hence we obtain a running time of $\mathcal{O}(|N|^2)$ for computing the expansion graph D_{exp} of a 1-critical instance $D = (N, A)$ of Simultaneous PQ-Ordering. □

Extending Orders

As shown in Lemma 9 Simultaneous PQ-Ordering can be solved for 1-critical instances in time polynomial in the size of the expansion graph. There are three things to do, first the Q-constraints need to be satisfied, which can be checked in linear time, second the critical double arcs need to be satisfied, which again can be done in linear time if possible, and finally orders for the edges around P-nodes need to be chosen bottom up. This is not obviously possible in linear time. However, the additional projection information that is stored for every arc makes it possible, which is shown in the following lemma.

Lemma 15. *Let D be a 1-critical instance of* Simultaneous PQ-Ordering *with expansion graph D_{exp}. In $\mathcal{O}(|D_{\text{exp}}|)$ time we can compute simultaneous PQ-orders or decide that no such orders exist.*

Proof. The major work for this lemma was already done in the proof of Lemma 9. It remains to show how orders for the P-nodes can be chosen bottom up in the expansion graph in linear time.

Consider a PQ-tree T in the expansion graph D_{exp} having the PQ-trees $T_1, \dots, T_\ell$ as children. Assume further that orders $O_1, \dots, O_\ell$ are already chosen for the children. The obvious approach to extend these orders simultaneously to an order represented by T would take $\mathcal{O}(\ell \cdot |T|)$ time, yielding a worst case quadratic running time per PQ-tree in the expansion tree. However, it can also be done in $\mathcal{O}(|T| + |T_1| + \cdots + |T_\ell|)$ time, which can be seen as follows. Let T_i be one of the children of T and let T_i' be the projection of T to the leaves of T_i, which was stored for the arc (T, T_i) while normalizing and expanding. Since T_i' has as many leaves as T_i, we can apply the order O_i to T_i' in $\mathcal{O}(|T_i|)$ time, inducing an order of incident edges around every P-node of T_i'. Now let μ_i be a P-node of T_i' and let μ be the P-node in T it stems from. Recall that we can find μ in constant time and furthermore for an edge incident to μ_i we can find the edge incident to μ it stems from in constant time. Thus, we can simply take the order of incident edges around μ_i and replace each edge by the edge incident to μ it stems from. This order is then stored for μ. Note that μ may store up to two orders in this way since it is fixed with respect to at most two children. It is clear that this can be done in $\mathcal{O}(\deg(\mu_i))$ time, thus processing all nodes in T_i takes $\mathcal{O}(|T_i|)$ time. Now assume we have processed all children of T. Then for the free P-nodes in T there is nothing stored, for a P-node μ fixed with respect to one child there is one order given for a subset of edges incident to μ and for the P-nodes fixed with respect to two children there are two such orders. In the first case, we can simply choose an arbitrary order for the edges incident to μ, taking $\mathcal{O}(\deg(\mu))$ time. In the second case, the free edges are added in an arbitrary way to the already ordered edges, which can again be done in $\mathcal{O}(\deg(\mu))$ time. If we have two orders, these orders need to be merged, which can clearly be done in linear time. Afterwards the free edges can be added in an arbitrary way. This again consumes $\mathcal{O}(\deg(\mu))$ time. Hence, we need for each node in T linear time in its degree and hence $\mathcal{O}(|T|)$ for the whole tree. Altogether we obtain the claimed $\mathcal{O}(|T| + |T_1| + \cdots + |T_\ell|)$ running time for extending the orders $O_1, \dots, O_\ell$ to an order O represented by T. Recall, that $|T_i|$ is linear in the size of the arc

(T, T_i). Thus, extending orders bottom up in the expansion graph $D_{\text{exp}} = (N_{\text{exp}}, A_{\text{exp}})$ takes $\mathcal{O}(|N_{\text{exp}}| + |A_{\text{exp}}|) = \mathcal{O}(|D_{\text{exp}}|)$ time. □

Overall Running Time

For applications producing instances of Simultaneous PQ-Ordering it may be possible that reconsidering the runtime analysis containing normalization, size and computation time of the expansion graph and order extension yields a better running time then $\mathcal{O}(|N|^2)$. However, for the general case we obtain the following theorem by putting Lemma 12, Lemma 13, Lemma 14 and Lemma 15 together. Note that the running time is dominated by the computation of the expansion graph.

Theorem 4. Simultaneous PQ-Ordering *can be solved in* $\mathcal{O}(|N|^2)$ *time for a 1-critical instance* $D = (N, A)$.

3.5 Simultaneous PQ-Ordering with Reversing Arcs

As mentioned in Section 2.5 we can express all embeddings of a biconnected planar graph in terms of PQ-trees by considering the embedding tree $T(v)$ describing all possible orders of incident edges around v, if we additionally ensure that Q-nodes stemming from the same R-node in the SPQR-tree $\mathcal{T}$ are oriented the same and pairs of P-nodes stemming from the same P-node in $\mathcal{T}$ are ordered oppositely. Forcing edges to be ordered the same can be easily achieved with an instance of Simultaneous PQ-Ordering by inserting a common child. However, we want to enforce edges around P-nodes to be ordered oppositely and not the same. Note that this cannot be achieved by simply choosing an appropriate injective mapping from the leaves of the child to the leaves of the parent, since it depends on the order if such a map reverses it.

To solve this problem we introduce Simultaneous PQ-Ordering with Reversing Arcs, which is an extension of the problem Simultaneous PQ-Ordering. Again, we have a DAG $D = (N, A)$ with nodes $N = \{T_1, \ldots, T_k\}$, such that every node T_i is a PQ-tree and every arc consists of a source T_i, a target T_j and an injective map $\varphi : L_j \to L_i$, where L_i and L_j are the leaves of T_i and T_j, respectively. In addition to that, every arc can be a *reversing arc*. Reversing arcs are denoted by $(T_i, -T_j; \varphi)$, whereas normal arcs are denoted by $(T_i, T_j; \varphi)$ as before. Simultaneous PQ-Ordering with Reversing Arcs asks whether there exist orders $O_1, \ldots, O_k$ such that every normal arc $(T_i, T_j; \varphi) \in A$ implies that $\varphi(O_j)$ is a suborder of O_i, whereas every reversing arc $(T_i, -T_j; \varphi) \in A$ implies that the reversal of $\varphi(O_j)$ is a suborder of O_i. As for Simultaneous PQ-Ordering, we define an instance of Simultaneous PQ-Ordering with Reversing Arcs to be *normalized*, if a normal arc $(T_i, T_j; \varphi)$ implies that $\mathcal{L}_i$ contains an order O_i extending $\varphi(O_j)$ for every order $O_j \in \mathcal{L}_i$ and a reversing arc $(T_i, -T_j; \varphi)$ implies that $\mathcal{L}_i$ contains an order O_i extending the reversal of $\varphi(O_j)$ for every order $O_j \in \mathcal{L}_j$, where $\mathcal{L}_i$ and $\mathcal{L}_j$ are the sets of orders represented by T_i and T_j, respectively. Since $\mathcal{L}_i$ is represented by a PQ-tree, it is closed with respect to reversing orders. Thus, if $\mathcal{L}_i$ contains an order extending $\varphi(O_j)$, it also contains an order extending the reversal order of $\varphi(O_j)$. Hence, we can normalize an instance of Simultaneous PQ-Ordering with Reversing Arcs in the same way we normalize an instance of Simultaneous PQ-Ordering by ignoring that some of the arcs are reversing.

In the following we show how to adapt the solution for Simultaneous PQ-Ordering presented in the previous sections to solve Simultaneous PQ-Ordering with Reversing Arcs. To give a rough overview, the definitions of the Q-constraints and the critical triples can be modified in a straight-forward manner, such that Lemma 4, stating that satisfying the Q-constraints and the critical triples is necessary and sufficient to be able

to extend orders chosen for several PQ-trees to an order of a common parent, is still true. By declaring some of the created arcs to be reversing, the definitions of expansion and finalizing step can be easily adapted such that the resulting expansion trees and the newly created arcs ensure that the responsible critical triples are satisfied. Thus, again the only critical triples that are not automatically satisfied by choosing orders bottom up correspond to critical double arcs. Lemmas 5, 6 and 7 showing that the expansion graph is well defined and equivalent to the original instance work in exactly the same way. For the definition of 1-critical instances there is no need to change anything. Lemma 8 stating that critical double arcs have a sink as target works as before. In Lemma 9 we showed how to solve 1-critical instances by testing whether the Q-constraints are satisfiable and whether we can choose orders for the critical double arcs satisfying the corresponding critical triple. If this was the case, we simply chose orders bottom up. Testing the Q-constraints can now be done in the same way. For the critical double arcs we can do the same as before if both arcs are normal or both are reversing. If one of them is normal and the other is reversing, we need to check if the corresponding permutation is order reversing instead of order preserving, hence we use Lemma 3 instead of Lemma 2. Afterwards, it is again ensured that every critical triple is satisfied, hence we can choose orders bottom up as before. Lemma 11 stating that the expansion graph has quadratic size for 1-critical instances works as before, since the only change in the definition of the expansion graph is that some arcs are reversing arcs instead of normal arcs, which of course does not change the size of the graph. Finally, we can put Lemma 9 and Lemma 11 together yielding that Simultaneous PQ-Ordering with Reversing Arcs can be solved in polynomial time for 1-critical instances as stated before in Theorem 2 for Simultaneous PQ-Ordering. Theorem 3 providing an easy criterion that an instance is 1-critical works exactly the same as before.

Let us start with the Q-constraints in more detail. Let μ be a Q-node in T that is fixed with respect to the child T' of T and let $\text{rep}(\mu)$ be its representative in T'. To ensure that μ is ordered as determined by $\text{rep}(\mu)$, we introduced either the constraint $x_\mu = x_{\text{rep}(\mu)}$ or $x_\mu \neq x_{\text{rep}(\mu)}$. Now if the arc (T, T') is reversing, we simply negate this constraint, ensuring that μ is orientated oppositely to the orientation determined by $\text{rep}(\mu)$. Let μ be a P-node in the PQ-tree T that is fixed with respect to two children T_1 and T_2 of T. Then μ, T_1 and T_2 together form again a critical triple. If both arcs (T, T_1) and (T, T_2) are normal arcs, we denote this critical triple by (μ, T_1, T_2) as before. If $(T, -T_i)$ is a reversing arc, we symbolise that by a minus sign in the critical triple, for example if we have the arcs (T, T_1) and $(T, -T_2)$, we denote the critical triple by $(\mu, T_1, -T_2)$. Assume we have orders O_1 and O_2 represented by T_1 and T_2, respectively. In the case that both arcs are normal or both are reversing, we say that the critical triple is satisfied, if the edges incident to μ fixed with respect to T_1 and T_2 are ordered the same in both orders O_1 and O_2, which is the same definition as before. In the case that one of the arcs is normal and the other is reversing, we define a critical triple to be satisfied if the order O_1 induces the opposite order than O_2 for the commonly fixed edges incident to μ. With these straight-forwardly adapted definitions it is clear that the proof of Lemma 4 works exactly as before. To improve readability we cite this lemma here.

Lemma 4. *Let T be a PQ-tree with children $T_1, \ldots, T_\ell$, such that every P-node in T is contained in at most one critical triple, and let $O_1, \ldots, O_\ell$ be orders represented by $T_1, \ldots, T_\ell$. An order O that is represented by T and simultaneously extends the orders $O_1, \ldots, O_\ell$ exists if and only if the Q-constraints and all critical triples are satisfied.*

This lemma implies that we can choose orders bottom up, if we ensure that the Q-constraints and the critical triple are satisfied, which leads us to the definition of the expansion graph. If we have a critical triple $(\mu, (-)T_1, (-)T_2)$, in general we apply an expansion

step as before, that is, we project T_1 and T_2 to representatives of the commonly fixed edges incident to μ and intersect the result to obtain the expansion tree $T(\mu, (-)T_1, (-)T_2)$. Additionally, we add arcs from T_1 and T_2 to the expansion tree. The only thing we need to change is that the arc from T_i (for $i = 1, 2$) to $T(\mu, (-)T_1, (-)T_2)$ is reversing if the arc $(T, -T_i)$ is reversing. Consider for example the critical triple $(\mu, -T_1, T_2)$. Then we have the reversing arcs $(T, -T_1)$ and $(T_1, -T(\mu, -T_1, T_2))$ and the normal arcs (T, T_2) and $(T_2, T(\mu, -T_1, T_2))$. If we choose an order for the leaves of $T(\mu, -T_1, T_2)$ representing the common fixed edges incident to μ, this order is reversed when it is extended to an order O_1 represented by T_1 and it remains the same by extension to an order O_2 represented by T_2. Hence, the edges incident to μ fixed with respect to T_1 and T_2 are ordered oppositely in O_1 and O_2 implying that the critical triple $(\mu, -T_1, T_2)$ is satisfied. In other words by extending an order represented by $T(\mu, -T_1, T_2)$ to an order of T containing μ it is reversed twice over the path containing T_1 yielding the same order as an extension over the path containing T_2 not reversing it at all. The other three configurations work analogously. The finalizing step can be handled similarly. If for a critical triple $(\mu, (-)T_1, (-)T_2)$ both PQ-trees T_1 and T_2 consist of a single P-node fixing the same edges incident to μ, we obtain a bijection φ between the leaves of T_1 and the leaves of T_2. As before, we create an arc from T_2 to T_1 with the map φ. This new arc is a normal arc if both arcs $(T, (-)T_1)$ and $(T, (-)T_2)$ are normal or if both are reversing. If one is reversing and one is normal, the new arc $(T_1, -T_2; \varphi)$ is reversing. Again this new arc ensures that the critical triple $(\mu, (-)T_1, (-)T_2)$ is satisfied if we choose orders bottom up. Note that we need to consider the special case where we have a critical triple $(\mu, (-)T', (-)T')$ due to a double arc. As before we apply expansion steps as if the children were different, ensuring that the critical triple is satisfied. Again a finalizing step would introduce a self loop, thus we simply prune expansion here (if T' is an expansion tree, otherwise we apply one more expansion step), introducing an unsatisfied double arc. The only difference to the unsatisfied double arcs we had before is that the arcs may be reversing.

For an instance D of Simultaneous PQ-Ordering with Reversing Arcs, we obtain the expansion graph D_{exp} by iteratively applying expansion and finalizing steps. Denote the expansion graph that we would obtain from D if we assume that all arcs are normal by D'_{exp}. It is clear that the only difference between D_{exp} and D'_{exp} is that some arcs in D_{exp} are reversing arcs. Hence, everything we proved for the structure of the expansion graph of an instance of Simultaneous PQ-Ordering still holds if we allow reversing arcs. Particularly, we have that the expansion graph is well defined (Lemma 5 and Lemma 6), that the target of every unsatisfied double arc is a sink if D is 1-critical (Lemma 8), that $|D_{\text{exp}}|$ is polynomial in $|D|$ if D is 1-critical (Lemma 11) and that D is 1-critical if it is at most 2-fixed (Theorem 3). Furthermore, all the implementation details provided in Section 3.4 still work. Note that we say that an instance D is 1-critical if every P-node in every PQ-tree in D_{exp} is contained in at most one critical triple, which is exactly the same definition as before.

It remains to show, that the instances D and D_{exp} are still equivalent (Lemma 7) and that we can solve D_{exp} by checking the Q-constraints, dealing with the unsatisfied double arcs and finally choosing orders bottom up, if D is 1-critical (Lemma 9). In the proof of Lemma 7 we had to show that simultaneous PQ-orders for all PQ-trees in D induce simultaneous PQ-orders for D_{exp}. That can be done analogously for the case where we allow reversing arcs. Most parts of the proof for Lemma 9 can be adapted straight forwardly since Lemma 4 still holds if we allow reversing arcs. The only difference is that the arcs in an unsatisfied double arc can be reversing. Consider an unsatisfied double arc $(T, (-)T'; \varphi_1)$ and $(T, (-)T'; \varphi_2)$ together with the corresponding permutation φ on the leaves of T'. If both arcs are normal or both are reversing, we need to check if φ is order preserving and choose an order that is preserved by φ, which can be done due to Lemma 2. If, however,

one of the arcs is normal and the other is reversing, we need to check if φ is order reversing and then choose an order that is reversed. This is something we have not done before, but it can be easily done by applying Lemma 3 instead of Lemma 2. Finally, Lemma 9 also works if we allow reversing arcs and hence we obtain the following theorem analogously to Theorem 4

Theorem 5. Simultaneous PQ-Ordering with Reversing Arcs *can be solved in* $\mathcal{O}(|N|^2)$ *time for an 1-critical instances* $D = (N, A)$.

Now that we know that 1-critical instances of Simultaneous PQ-Ordering with Reversing Arcs can be solved essentially in the same way as 1-critical instances of Simultaneous PQ-Ordering we do not longer distinguish between these two problems. Thus, if we create 1-critical instances of Simultaneous PQ-Ordering in our applications, we allow them to contain reversing arcs.

4. Applications

As mentioned in Section 2.5 and again in Section 3.5 to motivate why reversing arcs are necessary, we want to express all combinatorial embeddings of a biconnected planar graph in terms of PQ-trees or more precisely in terms of an instance of Simultaneous PQ-Ordering. A detailed description of this instance is given in Section 4.1. This representation is then used to solve Partially PQ-Constrained Planarity for biconnected graphs (Section 4.2) and Simultaneous Embedding with Fixed Edges for biconnected graphs with a connected intersection (Section 4.3). Furthermore, we show in Section 4.4 how Simultaneous PQ-Ordering can be used to recognize simultaneous Interval graphs.

4.1 PQ-Embedding Representation

Let $G = (V, E)$ be a planar biconnected graph and let $\mathcal{T}$ be its SPQR-tree. We want to define an instance $D(G) = (N, A)$ of Simultaneous PQ-Ordering called the *PQ-embedding representation* containing the embedding trees representing the circular order of edges around every vertex as defined in Section 2.5, such that it is ensured that every set of simultaneous PQ-orders corresponds to an embedding of G and vice versa. For every R-node η in $\mathcal{T}$, we define the PQ-tree $Q(\eta)$ consisting of a single Q-node with three edges and for every P-node μ in $\mathcal{T}$ with k virtual edges in skel(μ) we define the PQ-tree $P(\mu)$ consisting of a single P-node of degree k. The trees $Q(\eta)$ and $P(\mu)$ will ensure that embedding trees of different vertices sharing R- or P-nodes in the SPQR-tree are ordered consistently, thus we will call them the *consistency trees*. The node set N of the PQ-embedding representation contains the consistency trees $Q(\eta)$ and $P(\mu)$ and the embedding trees $T(v)$ for $v \in V$. If we consider an R-node η in the SPQR-tree $\mathcal{T}$, then there are several Q-nodes in different embedding trees stemming from it and we need to ensure that all these Q-nodes are oriented the same or in other words we need to ensure that they are all oriented the same as $Q(\eta)$, which can be done by simply adding arcs from the embedding trees to $Q(\eta)$ with suitable injective maps. Similarly, the skeleton of every P-node μ in $\mathcal{T}$ contains two vertices v_1 and v_2. Thus, the embedding trees $T(v_1)$ and $T(v_2)$ contain P-nodes μ_1 and μ_2 stemming from μ and every incident edge corresponds to a virtual edge in skel(μ). We need to ensure that the order of incident edges around μ_1 is the reversal of the order of edges around μ_2, or in other words, we need to ensure that the order for μ_1 is the same and the order for μ_2 is the opposite to any order chosen for $P(\mu)$, which can be ensured by a normal arc $(T(v_1), P(\mu))$ and a reversing arc $(T(v_2), -P(\mu))$.

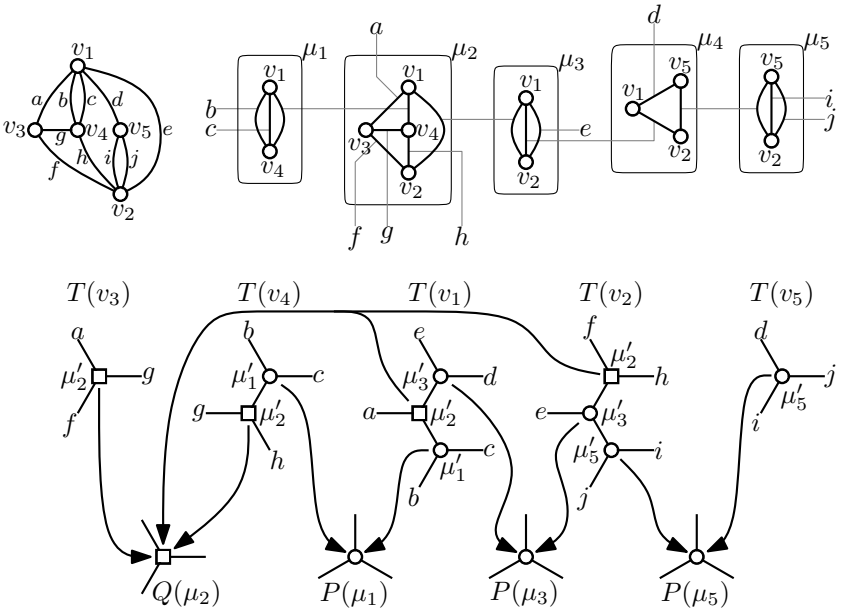

Figure 4.1: A biconnected planar graph and its SPQR-tree on the top and the corresponding PQ-embedding representation on the bottom. The injective maps on the edges are not explicitly depicted, but the starting points of the arcs suggests which maps are suitable.

If we solve the PQ-embedding representation $D(G)$ as instance of Simultaneous PQ-Ordering we would choose orders bottom up. Thus, we would first choose orders for the trees $P(\mu)$ and $Q(\mu)$, which corresponds to choosing orders for the P-nodes and orientations for the R-nodes in the SPQR-tree. For the embedding trees there is no choice left, since all nodes are fixed by some children, which is not surprising since the planar embedding is already chosen. Hence, extending the chosen orders to orders of the embedding trees can be seen as computing the circular orders of edges around every vertex for given embeddings of the skeletons of every node in $\mathcal{T}$. Figure 4.1 depicts the PQ-embedding representation for the example we had before in Figure 2.6. Note that the size of the PQ-embedding representation $D(G)$ is obviously linear in the size of the SPQR-tree $\mathcal{T}$ of G, and thus linear in the size of the planar graph G itself.

The PQ-embedding representation is obviously less elegant than the SPQR-tree, also representing all embeddings of a biconnected planar graph. At least for a human, the planar embeddings of a graph are easy to understand by looking at the SPQR-tree, whereas the PQ-embedding representation does not really help. However, with the PQ-embedding representation it is easier to formulate constraints concerning the order of incident edges around a vertex, since these orders are explicitly expressed by the embedding trees.

4.2 Partially PQ-Constrained Planarity

Let $G = (V, E)$ be a planar graph and let $C = \{T'(v_1), \ldots, T'(v_n)\}$ be a set of PQ-trees, such that for every vertex $v_i \in V$ the leaves of $T(v_i)$ are a subset $E'(v_i) \subseteq E(v_i)$ of edges incident to v_i. We call $T'(v_i)$ the *constraint tree* of the vertex v_i. The problem Partially PQ-Constrained Planarity asks whether a planar embedding of G exists, such that the order of incident edges $E(v_i)$ around every vertex v_i induces an order on $E'(v_i)$ that is represented by the constraint tree $T'(v_i)$.

Given an instance (G, C) of Partially PQ-Constrained Planarity, it is straightforward to formulate it as an instance of Simultaneous PQ-Ordering if G is biconnected.

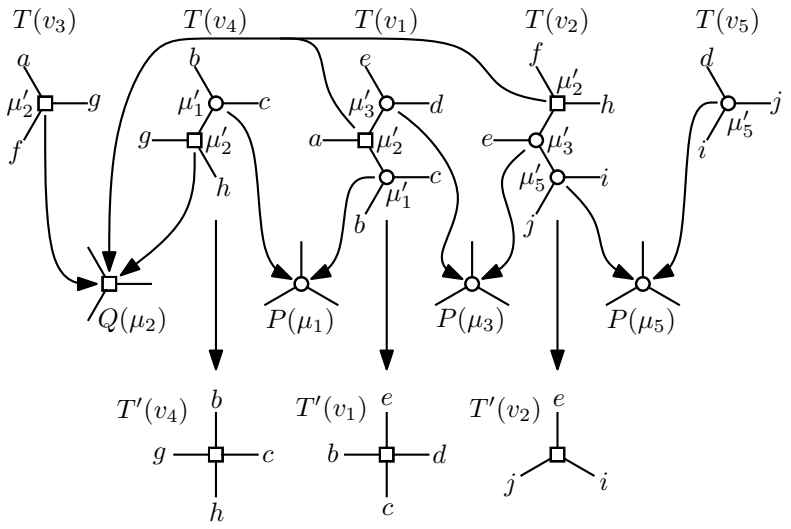

Figure 4.2: The PQ-embedding representation from Figure 4.1 together with the constraint trees provided by an instance of Partially PQ-Constrained Planarity.

Simply take the PQ-embedding representation $D(G)$ of G and add the constraint trees together with an arc $(T(v), T'(v); \mathrm{id})$ from the embedding tree to the corresponding constraint tree. Denote the resulting instance of Simultaneous PQ-Ordering by $D(G, C)$. Figure 4.2 depicts an example instance of Partially PQ-Constrained Planarity formulated as instance of Simultaneous PQ-Ordering. Note that we can leave the orders of edges around a vertex unconstrained by choosing the empty PQ-tree as its constraint tree. To obtain the following theorem, we need to show that (G, C) and $D(G, C)$ are equivalent, which is quite obvious, and that $D(G, C)$ is an at most 2-fixed instance of Simultaneous PQ-Ordering.

Theorem 6. Partially PQ-Constrained Planarity *can be solved in quadratic time for biconnected graphs.*

Proof. Consider (G, C) to be an instance of Partially PQ-Constrained Planarity where G is a biconnected planar graph and C the set of constraint trees. Let further $D(G, C)$ be the corresponding instance of Simultaneous PQ-Ordering. Since $D(G, C)$ contains the PQ-embedding representation $D(G)$, every solution of $D(G, C)$ yields a planar embedding of G. Additionally, this planar embedding respects the constraint trees since the order of edges around every vertex is an extension of an order of the leaves in the corresponding constraint tree. On the other hand, it is clear that a planar embedding of G respecting the constraint trees yields simultaneous orders for all trees in $D(G, C)$. Since the size of $D(G, C)$ is linear in the size of (G, C), we can solve (G, C) in quadratic time using Theorem 4, if $D(G, C)$ is 1-critical. We will show that the instance $D(G, C)$ is at most 2-fixed, and hence, due to Theorem 3 also 1-critical.

To compute the fixedness of every P-node in every PQ-tree in $D(G, C)$, we distinguish between three kinds of trees, the embedding trees, the consistency trees and the constraint trees. If we consider a P-node μ in an embedding tree $T(v)$, this P-node is fixed with respect to exactly one consistency tree, namely the tree that represents the P-node in the SPQR-tree μ stems from. In addition to the consistency trees, $T(v)$ has the constraint tree $T'(v)$ as child, thus μ can be fixed with respect to $T'(v)$. Since $T(v)$ has no parents and no other children, μ is at most 2-fixed, that is $\mathrm{fixed}(\mu) \leq 2$. Consider a P-node μ' in a constraint tree $T'(v)$. Since $T'(v)$ has no children and its only parent is $T(v)$ containing the P-node μ that is fixed by μ', we have by the definition of fixedness that

fixed(μ') = fixed(μ) − 1. Since μ is a P-node in an embedding tree we obtain fixed(μ') ≤ 1. We have two kinds of consistency trees, some stem from P- and some from R-nodes in the SPQR-tree. We need to consider only trees $P(\mu)$ stemming form P-nodes since the consistency trees stemming from R-nodes only contain a single Q-node. Denote the single P-node in $P(\mu)$ also by μ and let μ_1 and μ_2 be the two P-nodes in the embedding trees $T(v_1)$ and $T(v_2)$ that are fixed with respect to $P(\mu)$. Since $P(\mu)$ has no child and only these two parents, we obtain fixed(μ) = (fixed(μ_1) − 1) + (fixed(μ_2) − 1). Since μ_1 and μ_2 are P-nodes in embedding trees this yields fixed(μ) ≤ 2. Hence, all P-nodes in all PQ-trees in $D(G, C)$ are at most 2-fixed, thus $D(G, C)$ itself is 2-fixed. Finally, we can apply Theorem 3 yielding that $D(G, C)$ is 1-critical and thus can be solved quadratic time, due to Theorem 4. □

Since $D(G, C)$ is a special instance of Simultaneous PQ-Ordering, which seems to be quite simple, it is worth to make a more detailed runtime analysis, yielding the following theorem.

Theorem 7. Partially PQ-Constrained Planarity *can be solved in linear time for biconnected graphs.*

Proof. As figured out in Section 3.4 about the implementation details, there are four major parts influencing the running time. First, a given instance needs to be normalized consuming quadratic time (Lemma 12), the expansion graph has quadratic size in worst case (Lemma 13) and its computation consumes quadratic time (Lemma 14) and finally choosing borders bottom up needs linear time in the size of the expansion graph (Lemma 15).

In an instance $D(G, C)$ of Simultaneous PQ-Ordering stemming from an instance (G, C) of Partially PQ-Constrained Planarity there are two kinds of arcs. First, arcs from embedding trees to consistency trees, and second, arcs from embedding trees to constraint trees. When normalizing an arc from an embedding tree to a consistency tree there is nothing to do, since there is a bijection between the consistency tree and an inner node of the embedding tree. The arcs from embedding trees to constraint trees can be normalized as usual consuming only linear time, since each embedding tree has only one consistency tree as child. Hence, normalization can be done in linear time. When computing the expansion graph, the fixedness of the nodes is important. As seen in the proof of Theorem 6, the P-nodes in embedding and consistency trees are at most 2-fixed, whereas the P-nodes in constraint trees are at most 1-fixed. Note that every critical triple (μ, T_1, T_2) in $D(G, C)$ is of the kind that μ is contained in an embedding tree, T_1 is a constraint tree and T_2 is a consistency tree. Thus, the expansion tree $T(\mu, T_1, T_2)$ created due to such a triple has two parents where one of them is at most 1-fixed and the other at most 2-fixed. Hence, by the definition of fixedness, $T(\mu, T_1, T_2)$ itself is at most 1-fixed. After creating these expansion trees, all newly created critical triple must contain a P-node μ in a consistency tree and two expansion trees. By creating expansion trees for these critical triples no new critical triple are created and hence the expansion stops. It is clear that the resulting expansion graph has only linear size and can be computed in linear time. Choosing orders bottom up takes linear time in the size of the expansion graph, as before. Hence we obtain the claimed linear running time. □

4.3 Simultaneous Embedding with Fixed Edges

Let $G^① = (V^①, E^①)$ and $G^② = (V^②, E^②)$ be two planar graphs sharing a common subgraph $G = (V, E)$ with $V = V^① \cap V^②$ and $E = E^① \cap E^②$. Simultaneous Embedding with Fixed Edges asks, whether there exist planar drawings of $G^①$ and $G^②$ such that their

intersection G is drawn the same in both. Jünger and Schulz show that this is equivalent to the question whether combinatorial embeddings of $G^{①}$ and $G^{②}$ inducing the same combinatorial embedding for their intersection G exist [JS09, Theorem 4].

Assume that $G^{①}$ and $G^{②}$ are biconnected and G is connected. Then the order of incident edges around every vertex determines the combinatorial embedding, which is not the case for disconnected graphs. Thus, we can reformulate the problem as follows. Can we find planar embeddings of $G^{①}$ and $G^{②}$ inducing for every common vertex $v \in V$ the same order of common incident edges $E(v)$ around v? Since both graphs are biconnected, they both have a PQ-embedding representation and it is straight forward to formulate an instance $(G^{①}, G^{②})$ of SEFE as an instance $D(G^{①}, G^{②})$ of Simultaneous PQ-Ordering. The instance $D(G^{①}, G^{②})$ contains the PQ-embedding representations $D(G^{①})$ and $D(G^{②})$ of $G^{①}$ and $G^{②}$, respectively. Every common vertex $v \in V$ occurs as $v^{①}$ in $V^{①}$ and as $v^{②}$ in $V^{②}$, thus we have the two embedding trees $T(v^{①})$ and $T(v^{②})$. By projecting these two embedding trees to the common edges incident to v and intersecting the result, we obtain a new tree $T(v)$ called the *common embedding tree* of v. If we add the arcs $(T(v^{①}), T(v))$ and $(T(v^{②}), T(v))$ to the instance $D(G^{①}, G^{②})$ of Simultaneous PQ-Ordering, we ensure that the common edges incident to v are ordered the same in both graphs. Note that this representation is quite similar to the representation of an instance of Partially PQ-Constrained Planarity. Every common embedding tree can be seen as a constraint tree for both graphs simultaneously. To obtain the following theorem, we need to show that the instances $(G^{①}, G^{②})$ of SEFE and the instance $D(G^{①}, G^{②})$ of Simultaneous PQ-Ordering are equivalent and that $D(G^{①}, G^{②})$ is at most 2-fixed.

Theorem 8. Simultaneous Embedding with Fixed Edges *can be solved in quadratic time, if both graphs are biconnected and the common graph is connected.*

Proof. Let $(G^{①}, G^{②})$ be an instance of SEFE with the common graph G such that $G^{①}$ and $G^{②}$ are biconnected and G is connected. Let further $D(G^{①}, G^{②})$ be the corresponding instance of Simultaneous PQ-Ordering as defined above. Since $D(G^{①}, G^{②})$ contains the PQ-embedding representations $D(G^{①})$ and $D(G^{②})$, every solution of $D(G^{①}, G^{②})$ yields planar embeddings of $G^{①}$ and $G^{②}$. Furthermore, the common edges incident to a common vertex $v \in V$ are ordered the same in the two embedding trees $T(v^{①})$ and $T(v^{②})$ since both orders extend the same order of common edges represented by the common embedding tree $T(v)$. Thus, the embeddings for $G^{①}$ and $G^{②}$ induced by a solution of $D(G^{①}, G^{②})$ induce the same embedding on the common graph and hence are a solution of $(G^{①}, G^{②})$. On the other hand, if we have a SEFE of $G^{①}$ and $G^{②}$, these embeddings induce orders for the leaves of all PQ-trees in $D(G^{①}, G^{②})$ and since the common edges around every common vertex are ordered the same in both embeddings, all constraints given by arcs in $D(G^{①}, G^{②})$ are satisfied.

To compute the fixedness of every P-node in every PQ-tree in $D(G^{①}, G^{②})$ we distinguish between three kinds of trees, the embedding trees, the consistency trees and the common embedding trees. The proof that $\text{fixed}(\mu) \leq 2$ for every P-node μ in every embedding and consistency tree works as in the proof of Theorem 6. For a P-node μ in a common embedding tree $T(v)$ we have two P-nodes $\mu^{①}$ and $\mu^{②}$ in the parents $T(v^{①})$ and $T(v^{②})$ of $T(v)$ it stems from. Since $T(v)$ has no other parents and no children, we obtain $\text{fixed}(\mu) = (\text{fixed}(\mu^{①}) - 1) + (\text{fixed}(\mu^{②}) - 1)$ by the definition of fixedness. Since $\mu^{①}$ and $\mu^{②}$ are P-nodes in embedding trees, we know that their fixedness is at most 2. Thus, we have $\text{fixed}(\mu) \leq 2$. Hence, all P-nodes in all PQ-trees in $D(G^{①}, G^{②})$ are at most 2-fixed, thus $D(G^{①}, G^{②})$ itself is 2-fixed. □

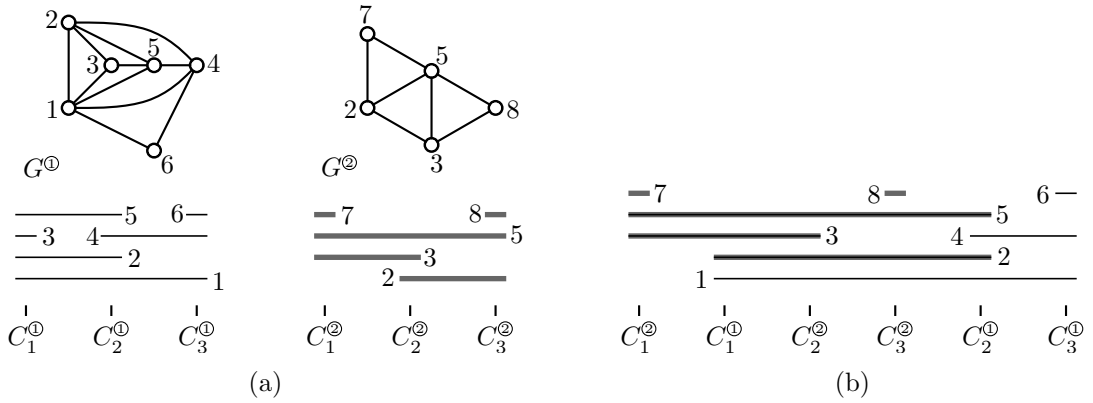

Figure 4.3: (a) Two interval graphs $G^{①}$ and $G^{②}$ with interval representations. The maximal cliques are $C_1^{①}, C_2^{①}, C_3^{①}$ and $C_1^{②}, C_2^{②}, C_3^{②}$, respectively.
(b) Interval representations of $G^{①}$ and $G^{②}$ such that common vertices are represented by the same interval in both representations, in other words, a simultaneous interval representation of $G^{①}$ and $G^{②}$.

4.4 Simultaneous Interval Graphs

A graph G is an *interval graph* if each vertex v can be represented as an interval $I(v) \subset \mathbb{R}$ such that two vertices u and v are adjacent if and only if their intervals intersect, that is, $I(u) \cap I(v) \neq \emptyset$. Such a representation is called *interval representation* of G; see Figure 4.3a for two examples. Two graphs $G^{①}$ and $G^{②}$ sharing a common subgraph are *simultaneous interval graphs* if $G^{①}$ and $G^{②}$ have interval representations such that the common vertices are represented by the same intervals in both representations; see Figure 4.3b for an example.

The first algorithm recognizing interval graphs in linear time was given by Booth and Lueker [BL76] and was based on a characterization by Fulkerson and Gross [FG65]. This characterization says that G is an interval graph if and only if there is a linear order of all its maximal cliques such that for each vertex v all cliques containing v appear consecutively. It is easy to see that an interval graph can have only linearly many maximal cliques thus it is clear how to recognize interval graphs in linear time by using PQ-trees. Simultaneous interval graphs were first considered by Jampani and Lubiw [JL10] who show how to recognize them in $\mathcal{O}(n^2 \log n)$ time.

In Theorem 9 we give a proof of the characterization by Fulkerson and Gross that can then be extended to a characterization of simultaneous interval graphs in Theorem 10. With this characterization it is straightforward to formulate an instance of Simultaneous PQ-Ordering that can be used to test whether a pair of graphs are simultaneous interval graphs in linear time, improving the so far known result. The following definition simplifies the notation. Let $C_1, \cdots, C_\ell$ be sets (for example maximal cliques) and let v be an element contained in some of these sets. We say that a linear order of these sets is *v-consecutive* if the sets containing v appear consecutively.

Theorem 9 (Fulkerson and Gross [FG65])**.** *A graph G is an interval graph if and only if there is a linear order of all maximal cliques of G that is v-consecutive with respect to every vertex v.*

Proof. Assume G is an interval graph with a fixed interval representation. Let $C = \{v_1, \ldots, v_k\}$ be a maximal clique in G. It is clear that there must be a position x such that x is contained in the intervals $I(v_1), \ldots, I(v_k)$. Additionally x is not contained in any interval represented by another vertex since the clique C is maximal. By fixing such

positions $x_1, \ldots, x_\ell$ for each of the maximal cliques $C_1, \ldots, C_\ell$ in G, we define a linear order on all maximal cliques. Assume this order is not v-consecutive for some vertex v. Then there are cliques C_i, C_j, C_k with $x_i < x_j < x_k$ such that $v \in C_i, C_k$ but $v \notin C_j$. However, since v is in C_i and C_k its interval $I(v)$ needs to contain x_i and x_k, and hence also x_j, which is a contradiction to the construction of the position x_j. Hence the defined linear order of all maximal cliques is v-consecutive with respect to every vertex v.

Now assume $O = C_1 \ldots C_\ell$ is a linear order of all maximal cliques of G that is v-consecutive for every vertex v. Let v be a vertex and let C_i and C_j be the leftmost and rightmost cliques containing v, respectively. Then define $I(v) = [i, j]$ to be the interval representing v. With this representation, we obtain all edges contained in the the maximal cliques $C_1, \ldots, C_\ell$ at the natural numbers $1, \ldots, \ell$, since for each clique $C_i = \{v_1, \ldots, v_k\}$ the position i is contained in all the intervals $I(v_1), \ldots, I(v_k)$. Furthermore, there is no vertex $u \notin C_i$ such that $I(u)$ also contains i, because such a vertex would need to be contained in a clique on the left and in a clique on the right to C_i, which is a contradiction since the order O is u-consecutive. Thus, at the integer positions $1, \ldots, \ell$ all edges in G are represented and no edges not in G. Furthermore, all intervals $I(v)$ containing a non integer position $1 < x < \ell$ contain also $\lceil x \rceil$ and $\lfloor x \rfloor$, yielding that no edge is defined due the position x which is not already defined due to an integer position. Hence, this definition of intervals is an interval representation of G showing that G is an interval graph. □

We can extend this characterization of interval graphs to a characterization of simultaneous interval graphs by using the same arguments as follows.

Theorem 10. *Two graphs $G^①$ and $G^②$ are simultaneous interval graphs if and only if there are linear orders of the maximal cliques of $G^①$ and $G^②$ that are v-consecutive with respect to every vertex v in $G^①$ and $G^②$, respectively, such that they can be extended to an order of the union of maximal cliques that is v-consecutive with respect to every common vertex v.*

Proof. Assume $G^①$ and $G^②$ are simultaneous interval graphs and let for every vertex v be $I(v)$ the interval representing v. Assume $\mathcal{C}^① = \{C_1^①, \ldots, C_k^①\}$ and $\mathcal{C}^② = \{C_1^②, \ldots, C_\ell^②\}$ are the maximal cliques in $G^①$ and $G^②$ respectively. When considering $G^①$ for itself, we again obtain for every maximal clique $C^① = \{v_1, \ldots, v_r\}$ a position x such that x is contained in $I(v_i)$ for every $v_i \in C^①$ but in no other interval representing a vertex in $G^①$. The same can be done for the maximal cliques of $G^②$, yielding a linear order O of all maximal cliques $\mathcal{C} = \mathcal{C}^① \cup \mathcal{C}^②$. It is clear that the projection of this order to the cliques in $G^①$ is v-consecutive for every vertex v in $G^①$ due to Theorem 9 and the same holds for $G^②$. It remains to show that O is v-consecutive for each common vertex v. Assume O is not v-consecutive for some common vertex v. Then there need to be three cliques C_i, C_j and C_k, no matter if they are maximal cliques in $G^①$ or in $G^②$, with positions x_i, x_j and x_k such that $x_i < x_j < x_k$ and $v \in C_i, C_k$ but $v \notin C_j$. However, since the interval $I(v)$ contains x_i and x_k it also contains x_j, which is a contradiction to the construction of the position x_j for the clique C_j since v is a common vertex. Note that this is the same argument as used in the proof of Theorem 9.

Conversely, we need to show how to construct an interval representation from a given linear order of all maximal cliques. Assume we have a linear order O of all maximal cliques satisfying the conditions of the theorem. Rename the cliques such that $C_1 \ldots C_{k+\ell}$ is this order, neglecting for a moment from which graph the cliques stem. Let v be a vertex in $G^①$ or $G^②$ and let C_i and C_j be the leftmost and rightmost clique in O containing v. Then we define the interval $I(v)$ to be $[i, j]$, as in the case of a single graph. Our claim is that this yields a simultaneous interval representation of $G^①$ and $G^②$. Again, it is easy

to see that a non integer position x is only contained in intervals also containing $\lceil x \rceil$ and $\lfloor x \rfloor$. Thus we only need to consider the positions $1, \ldots, k + \ell$, let i be such an integral position. Assume without loss of generality that $C_i = \{v_1, \ldots, v_r\}$ is a clique of $G^①$. Then i is contained in all the intervals $I(v_1), \ldots, I(v_r)$ by definition. The position i may be additionally contained in the interval $I(u)$ for a vertex that is exclusively contained in $G^②$ but this does not create an edge between vertices in $G^①$. However, there is no vertex $u \notin C_i$ contained in $G^①$ such that i is contained in $I(u)$ since this would violate the u-consecutiveness either of the whole order or of the projection to the cliques in $G^①$. Since the same argument works for cliques in $G^②$, all edges in maximal cliques of $G^①$ and $G^②$ are represented by the defined interval representation and at the integer positions no edges not contained are represented. Hence, this definition of intervals is a simultaneous interval representation of $G^①$ and $G^②$. □

With this characterization it is straightforward to formulate the problem of recognizing simultaneous interval graphs as an instance of Simultaneous PQ-Ordering. Furthermore, the resulting instance is so simple that it can be solved in linear time. Since we want to represent linear orders instead of circular orders we need to use rooted PQ-trees instead of unrooted ones. Note that this can be easily done by introducing an additional leaf, the special leaf, as described in the preliminaries about PQ-trees (Section 2.3). The PQ-trees mentioned in the remaining part of this section are assumed to be rooted, representing linear orders.

Theorem 11. *Whether two graphs $G^①$ and $G^②$ with a common subgraph are simultaneous interval graphs can be tested in linear time.*

Proof. Let $\mathcal{C}^① = \{C_1^①, \ldots, C_k^①\}$ and $\mathcal{C}^② = \{C_1^②, \ldots, C_\ell^②\}$ be the maximal cliques of $G^①$ and $G^②$ respectively and let $\mathcal{C} = \mathcal{C}^① \cup \mathcal{C}^②$ be the set of all maximal cliques. We define three PQ-trees T, $T^①$ and $T^②$ having $\mathcal{C}$, $\mathcal{C}^①$ and $\mathcal{C}^②$ as leaves, respectively. The tree T is defined such that it represents all linear orders of $\mathcal{C}$ that are v-consecutive with respect to all common vertices v. The trees $T^①$ and $T^②$ are defined to represent all linear orders of $\mathcal{C}^①$ and $\mathcal{C}^②$ that are v-consecutive with respect to all vertices v in $G^①$ and $G^②$, respectively. Note that $T^①$ and $T^②$ are the PQ-trees that would be used to test whether $G^①$ and $G^②$ themselves are interval graphs. By the characterization in Theorem 10 it is clear that $G^①$ and $G^②$ are simultaneous interval graphs if and only if we can find an order represented by T extending orders represented by $T^①$ and $T^②$. Hence $G^①$ and $G^②$ are simultaneous interval graphs if and only if the instance D of Simultaneous PQ-Ordering consisting of the nodes T, $T^①$ and $T^②$ and the arcs $(T, T^①)$ and $(T, T^②)$ has a solution. This can be checked in quadratic time using Theorem 4 since D is obviously 1-critical. Furthermore, normalization can of course be done in linear time and the expansion tree of linear size can be computed in linear time since expansion stops after a single expansion step. Hence the instance D of Simultaneous PQ-Ordering can be solved in linear time, which concludes the proof. □

5. Conclusion

In this work we introduced a new problem called Simultaneous PQ-Ordering. It has as input a set of PQ-trees with a child-parent relation (a DAG with PQ-trees as nodes) and asks, whether for every PQ-tree a circular order can be chosen such that it is an extension of the orders of all its children. This was motivated by the possibility to represent the possible circular orders of edges around every vertex of a biconnected planar graph by a PQ-tree. Unfortunately, Simultaneous PQ-Ordering turned out to be $\mathcal{NP}$-complete in general. However, we were able to find an algorithm solving Simultaneous PQ-Ordering in polynomial time for "simple" instances, the 1-critical instances. To achieve this result we showed that satisfying the Q-constraints and the critical triples is sufficient to be able to extend orders of several children simultaneously to a parent, if each P-node is contained in at most one critical triple. We were able to ensure that a critical triple is automatically satisfied when choosing orders bottom up by inserting a new PQ-tree, the expansion tree. Creating the expansion trees iteratively for every critical triple led to the expansion graph that turned out to have polynomial size for 1-critical instances. Hence, we are able to solve a 1-critical instance of Simultaneous PQ-Ordering in polynomial time essentially by choosing orders bottom up in the expansion graph. With this framework we were able to solve Partially PQ-Constrained Planarity for biconnected graphs and Simultaneous Embedding with Fixed Edges for biconnected graphs with a connected intersection in polynomial time (linear and quadratic, respectively), which were both not known to be efficiently solvable. Furthermore, we are able to recognize simultaneous interval graphs in linear time, which is an improvement to the so far known $\mathcal{O}(n^2 \log n)$ algorithm. Note that all these results are really simple, once we have the algorithm for Simultaneous PQ-Ordering.

However, several questions remain open. Since the PQ-embedding representation can only represent the embeddings of biconnected planar graphs, our solutions for Simultaneous PQ-Ordering and SEFE cannot handle graphs containing cutvertices. This restriction is due to the fact that the set of possible orders of edges around cutvertices is not PQ-representable. Thus, a question naturally raising from our results is whether such sets of orders can be represented by a data structure similar to PQ-trees. For example the possible orders of edges around a cutvertex v could be represented by a set of PQ-trees, one PQ-tree for each block (maximal biconnected component) incident to v. Then orders for all PQ-trees can be chosen arbitrarily and independently, while merging these orders to one order is restricted as follows. Let a_1 and a_2 be leaves of a PQ-tree and let b_1 and b_2 be two leaves of another PQ-tree. Then the suborder $a_1 b_1 a_2 b_2$ must not appear. The

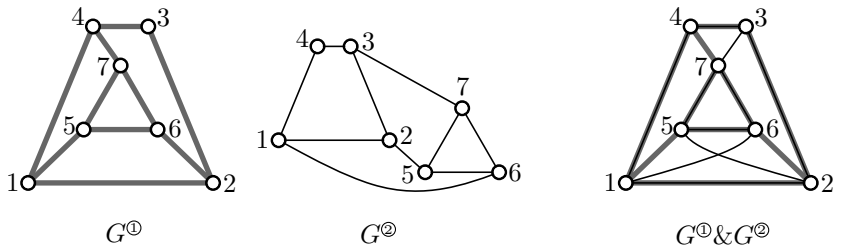

Figure 5.1: An instance of SEFE with graphs $G^{①}$ and $G^{②}$ on the left. Although the embeddings of $G^{①}$ and $G^{②}$ on the left induce for the common graph G the same circular orders of edges around every vertex, they induce different combinatorial embeddings and thus cannot be drawn simultaneously, as one can see on the right.

main question for such a representation would be, whether something similar to Lemma 4 can be shown, that is, under which conditions chosen orders of children can be extended simultaneously to an order represented by a parent.

If we keep the restriction that the graphs need to be biconnected, we could try to solve SEFE for the case that the common graph does not need to be connected. Unfortunately, the formulation of an instance $(G^{①}, G^{②})$ of SEFE is not as straight forward if we allow the common graph to be disconnected, although $G^{①}$ and $G^{②}$ are still biconnected. The difference is that we need to ensure that the circular orders of edges around every vertex chosen for $G^{①}$ and $G^{②}$ really induce the same embedding on the common subgraph. Figure 5.1 shows two graphs that do not have a SEFE, although there are embeddings inducing the same circular order of common edges around every vertex. That the two depicted graphs do not have a SEFE is easy to see, since both graphs are triconnected (except for a subdivision vertex) and hence have only two embeddings. An approach to solve this problem could be of the following kind. Assume there are two edges $e^{①}$ and $e^{②}$ that belong exclusively to $G^{①}$ and $G^{②}$ incident to the common vertex v. Assume further that there are simple paths in $G^{①}$ and $G^{②}$, not using any common edges, from v to a connected component in G not containing v and starting with the edges $e^{①}$ and $e^{②}$, respectively. Then $e^{①}$ and $e^{②}$ need to be embedded into the same face of the common graph G. In terms of the circular order of edges around v this means that $e^{①}$ and $e^{②}$ need to be located between the same pair of common edges in the circular order of common edges. Such a constraint can be added to an instance of Simultaneous PQ-Ordering by introducing an additional PQ-tree having the common edges incident to v as leaves plus one additional leaf. Then a double arc from the embedding tree $T(v)$ to this new PQ-tree associating the additional leaf with $e^{①}$ for the first and with $e^{②}$ for the second arc ensures that $e^{①}$ and $e^{②}$ are embedded into the same face. One can easily see that additional constraints of this kind are not only necessary but also sufficient. However, the resulting instance of Simultaneous PQ-Ordering is no longer 1-critical. There are two obvious approaches to address this problem. First, one could try to find a different construction of SEFE as instance of Simultaneous PQ-Ordering that is 1-critical. The second approach would be to find algorithms solving Simultaneous PQ-Ordering for instances that are not 1-critical. There are two major problems that need to be addressed, if a P-node μ is contained in more than one critical triple. First, the expansion graph may become exponentially large and second, satisfying critical triple is not sufficient to be able to extend orders of children simultaneously, as depicted in Figure 3.2b. However, the second problem does not arise, if all children fixing the P-node μ fix the same edges incident to μ up to a single edge that is unique for each child, which is the case for the above mentioned construction. Hence, there is the hope that Simultaneous PQ-Ordering can help to solve SEFE also for the case where the intersection is disconnected.

Bibliography

[ADF+10] Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter, *Testing planarity of partially embedded graphs*, Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), SODA '10, Society for Industrial and Applied Mathematics, 2010, pp. 202–221.

[ADF11a] Patrizio Angelini, Giuseppe Di Battista, and Fabrizio Frati, *Simultaneous embedding of embedded planar graphs*, Algorithms and Computation - 22nd International Symposium ISAAC 2011, 2011.

[ADF+11b] Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter, *Testing the simultaneous embeddability of two graphs whose intersection is a biconnected graph or a tree*, Combinatorial Algorithms (Costas Iliopoulos and William Smyth, eds.), Lecture Notes in Computer Science, vol. 6460, Springer Berlin / Heidelberg, 2011, pp. 212–225.

[ADF+11c] ———, *Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph*, To appear, 2011.

[AGKN11] Patrizio Angelini, Markus Geyer, Michael Kaufmann, and Daniel Neuwirth, *On a tree and a path with no geometric simultaneous embedding*, Graph Drawing (Ulrik Brandes and Sabine Cornelsen, eds.), Lecture Notes in Computer Science, vol. 6502, Springer Berlin / Heidelberg, 2011, pp. 38–49.

[APT79] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan, *A linear-time algorithm for testing the truth of certain quantified boolean formulas*, Information Processing Letters **8** (1979), no. 3, 121–123.

[BCD+07] Peter Brass, Eowyn Cenek, Cristian A. Duncan, Alon Efrat, Cesim Erten, Dan P. Ismailescu, Stephen G. Kobourov, Anna Lubiw, and Joseph S. B. Mitchell, *On simultaneous planar graph embeddings*, Computational Geometry: Theory and Applications **36** (2007), 117–130.

[Bie91] Daniel Bienstock, *Some provably hard crossing number problems*, Discrete & Computational Geometry **6** (1991), 443–459.

[BK06] Glencora Borradaile and Philip Klein, *An O(n log n) algorithm for maximum st-flow in a directed planar graph*, Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm (New York, NY, USA), SODA '06, ACM, 2006, pp. 524–533.

[BL76] Kellogg S. Booth and George S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, Journal of Computer and System Sciences **13** (1976), 335–379.

[BM99] John Boyer and Wendy Myrvold, *Stop minding your P's and Q's: A simplified O(n) planar embedding algorithm*, Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms (Philadelphia, PA, USA), SODA '99, Society for Industrial and Applied Mathematics, 1999, pp. 140–146.

[Boo75] K.S. Booth, *PQ-tree algorithms*, Ph.D. thesis, University of California, Berkeley, 1975.

[Bra09] Ulrik Brandes, *The left-right planarity test*, Unpublished but available at http://www.inf.uni-konstanz.de/algo/publications/b-lrpt-sub.pdf, 2009.

[CON85] Norishige Chiba, Kazunori Onoguchi, and Takao Nishizeki, *Drawing plane graphs nicely*, Acta Informatica **22** (1985), 187–201.

[CP95] M. Chrobak and T. H. Payne, *A linear-time algorithm for drawing a planar graph on a grid*, Information Processing Letters **54** (1995), no. 4, 241–246.

[CYN84] Norishige Chiba, Tadashi Yamanouchi, and Takao Nishizeki, *Linear algorithms for convex drawings of planar graphs*, Progress in Graph Theory, 1984, pp. 153–173.

[ddR06] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl, *Depth-first search and planarity*, International Journal of Foundations of Computer Science **17** (2006), 1017–1029.

[DL07] Emilio Di Giacomo and Giuseppe Liotta, *Simultaneous embedding of outerplanar graphs, paths, and cycles*, International Journal of Computational Geometry and Applications **17** (2007), no. 2, 139–160.

[dR82] Hubert de Fraysseix and Pierre Rosenstiehl, *A depth-first-search characterization of planarity*, Graph Theory, Proceedings of the Conference on Graph Theory (Béla Bollobás, ed.), North-Holland Mathematics Studies, vol. 62, North-Holland, 1982, pp. 75–80.

[dR85] Hubert de Fraysseix and Pierre Rosenstiehl, *A characterization of planar graphs by Trémaux orders*, Combinatorica **5** (1985), 127–135.

[DT96a] Giuseppe Di Battista and Roberto Tamassia, *On-line maintenance of triconnected components with SPQR-trees*, Algorithmica **15** (1996), no. 4, 302–318.

[DT96b] ———, *On-line planarity testing*, SIAM Journal on Computing **25** (1996), no. 5, 956–997.

[EBFK09] Alejandro Estrella-Balderrama, J. Fowler, and Stephen Kobourov, *Graph simultaneous embedding tool, GraphSET*, Graph Drawing (Ioannis Tollis and Maurizio Patrignani, eds.), Lecture Notes in Computer Science, vol. 5417, Springer Berlin / Heidelberg, 2009, pp. 169–180.

[EBGJ+08] Alejandro Estrella-Balderrama, Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz, *Simultaneous geometric graph embeddings*, Graph Drawing (Seok-Hee Hong, Takao Nishizeki, and Wu Quan, eds.), Lecture Notes in Computer Science, vol. 4875, Springer Berlin / Heidelberg, 2008, pp. 280–290.

[EIS76] S. Even, A. Itai, and A. Shamir, *On the complexity of timetable and multicommodity flow problems*, SIAM Journal on Computing **5** (1976), no. 4, 691–703.

[EK05] Cesim Erten and Stephen Kobourov, *Simultaneous embedding of planar graphs with few bends*, Graph Drawing (János Pach, ed.), Lecture Notes in Computer Science, vol. 3383, Springer Berlin / Heidelberg, 2005, pp. 195–205.

[EKLN04] Cesim Erten, Stephen Kobourov, Vu Le, and Armand Navabi, *Simultaneous graph drawing: Layout algorithms and visualization schemes*, Graph Drawing (Giuseppe Liotta, ed.), Lecture Notes in Computer Science, vol. 2912, Springer Berlin / Heidelberg, 2004, pp. 437–449.

[FG65] D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific Journal of Mathematics **15** (1965), no. 3, 835–855.

[FGJ+09] J. Joseph Fowler, Carsten Gutwenger, Michael Jünger, Petra Mutzel, and Michael Schulz, *An SPQR-tree approach to decide special cases of simultaneous embedding with fixed edges*, Graph Drawing (Ioannis Tollis and Maurizio Patrignani, eds.), Lecture Notes in Computer Science, vol. 5417, Springer Berlin / Heidelberg, 2009, pp. 157–168.

[FJKS11] J. Joseph Fowler, Michael Jünger, Stephen G. Kobourov, and Michael Schulz, *Characterizations of restricted pairs of planar graphs allowing simultaneous embedding with fixed edges*, Computational Geometry **44** (2011), no. 8, 385–398.

[Fár48] István Fáry, *On straight-line representation of planar graphs*, Acta Scientiarum Mathematicarum **11** (1948), 229–233.

[Fra07] Fabrizio Frati, *Embedding graphs simultaneously with fixed edges*, Graph Drawing (Michael Kaufmann and Dorothea Wagner, eds.), Lecture Notes in Computer Science, vol. 4372, Springer Berlin / Heidelberg, 2007, pp. 108–113.

[GJ83] M. R. Garey and D. S. Johnson, *Crossing number is NP-complete*, SIAM Journal on Algebraic and Discrete Methods **4** (1983), no. 3, 312–316.

[GJP+06] Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz, *Simultaneous graph embeddings with fixed edges*, Graph-Theoretic Concepts in Computer Science (Fedor Fomin, ed.), Lecture Notes in Computer Science, vol. 4271, Springer Berlin / Heidelberg, 2006, pp. 325–335.

[GKM07] Carsten Gutwenger, Karsten Klein, and Petra Mutzel, *Planarity testing and optimal edge insertion with embedding constraints*, Proceedings of the 14th international conference on Graph drawing (Berlin, Heidelberg), GD'06, Springer-Verlag, 2007, pp. 126–137.

[GKV09] Markus Geyer, Michael Kaufmann, and Imrich Vrt'o, *Two trees which are self-intersecting when drawn simultaneously*, Discrete Mathematics **309** (2009), no. 7, 1909–1916, 13th International Symposium on Graph Drawing, 2005, 13th International Symposium on Graph Drawing, 2005.

[GM77] Zvi Galil and Nimrod Megiddo, *Cyclic ordering is NP-complete*, Theoretical Computer Science **5** (1977), no. 2, 179–182.

[GM01] Carsten Gutwenger and Petra Mutzel, *A linear time implementation of SPQR-trees*, Proceedings of the 8th International Symposium on Graph Drawing (GD '00), Lecture Notes in Computer Science, vol. 1984, Springer, 2001, pp. 77–90.

[GM04] Carsten Gutwenger and Petra Mutzel, *An experimental study of crossing minimization heuristics*, Graph Drawing (Giuseppe Liotta, ed.), Lecture Notes in Computer Science, vol. 2912, Springer Berlin / Heidelberg, 2004, pp. 13–24.

[GMW01] Carsten Gutwenger, Petra Mutzel, and René Weiskircher, *Inserting an edge into a planar graph*, Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, SODA '01, Society for Industrial and Applied Mathematics, 2001, pp. 246–255.

[Guy72] Richard Guy, *Crossing numbers of graphs*, Graph Theory and Applications (Y. Alavi, D. Lick, and A. White, eds.), Lecture Notes in Mathematics, vol. 303, Springer Berlin / Heidelberg, 1972, pp. 111–124.

[HJL10] Bernhard Haeupler, Krishnam Jampani, and Anna Lubiw, *Testing simultaneous planarity when the common graph is 2-connected*, Algorithms and Computation (Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, eds.), Lecture Notes in Computer Science, vol. 6507, Springer Berlin / Heidelberg, 2010, pp. 410–421.

[HM01] Wen-Lian Hsu and Ross M. McConnell, *PQ trees, PC trees, and planar graphs*, 2001.

[HM03] _____, *PC trees and circular-ones arrangements*, Theoretical Computer Science **296** (2003), 99–116.

[HN09] Seok-Hee Hong and Hiroshi Nagamochi, *Two-page book embedding and clustered graph planarity*, Tech. Report 2009-004, Department of Applied Mathematics & Physics, Kyoto University, 2009.

[Hsu01] Wen-Lian Hsu, *PC-trees vs. PQ-trees*, Proceedings of the 7th Annual International Conference on Computing and Combinatorics, COCOON '01, 2001, pp. 207–217.

[HT74] John Hopcroft and Robert Tarjan, *Efficient planarity testing*, Journal of the ACM **21** (1974), 549–568.

[HT08] Bernhard Haeupler and Robert E. Tarjan, *Planarity algorithms via PQ-trees (extended abstract)*, Electronic Notes in Discrete Mathematics **31** (2008), 143–149, The International Conference on Topological and Geometric Graph Theory.

[JKR11] Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter, *A Kuratowski-type theorem for planarity of partially embedded graphs*, Proceedings of the 27th annual ACM symposium on Computational geometry (New York, NY, USA), SoCG '11, ACM, 2011, pp. 107–116.

[JL10] Krishnam Raju Jampani and Anna Lubiw, *Simultaneous interval graphs*, Computing Research Repository **abs/1009.3502** (2010), 1–25.

[JS09] Michael Jünger and Michael Schulz, *Intersection graphs in simultaneous embedding with fixed edges*, Journal of Graph Algorithms and Applications **13** (2009), no. 2, 205–218.

[KP05] Stephen Kobourov and Chandan Pitta, *An interactive multi-user system for simultaneous graph drawing*, Graph Drawing (János Pach, ed.), Lecture Notes in Computer Science, vol. 3383, Springer Berlin / Heidelberg, 2005, pp. 492–501.

[Kro67] M. R. Krom, *The decision problem for a class of first-order formulas in which all disjunctions are binary*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **13** (1967), no. 1-2, 15–20.

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

[KW02] Michael Kaufmann and Roland Wiese, *Embedding vertices at points: Few bends suffice for planar graphs*, Journal of Graph Algorithms and Applications **6** (2002), no. 1, 115–129.

[LEC67] A. Lempel, S. Even, and I. Cederbaum, *An algorithm for planarity testing of graphs*, Theory of Graphs (P. Rosenstiehl, ed.), 1967, pp. 215–232.

[PW98] János Pach and Rephael Wenger, *Embedding planar graphs at fixed vertex locations*, Graph Drawing (Sue Whitesides, ed.), Lecture Notes in Computer Science, vol. 1547, Springer Berlin / Heidelberg, 1998, pp. 263–274.

[Sch89] Walter Schnyder, *Planar graphs and poset dimension*, Order **5** (1989), 323–343.

[SH93] Wei-Kuan Shih and Wen-Lian Hsu, *A simple test for planar graphs*, Proceedings of the International Workshop on Discrete Mathematics and Algorithms, 1993, pp. 110–122.

[SWK90] Wei-Kuan Shih, Sun Wu, and Y.S. Kuo, *Unifying maximum cut and minimum cut of a planar graph*, IEEE Transactions on Computers **39** (1990), no. 5, 694–697.

[Tut63] W. T. Tutte, *How to draw a graph*, Proceedings of the London Mathematical Society, vol. 13, 1963, pp. 743–768.