Simultaneous PQ-Ordering with Applications to Constrained Embedding Problems

THOMAS BLÄSIUS and IGNAZ RUTTER, Karlsruhe Institute of Technology (KIT)

16

In this article, we define and study the new problem of SIMULTANEOUS PQ-ORDERING. Its input consists of a set of PQ-trees, which represent sets of circular orders of their leaves, together with a set of child-parent relations between these PQ-trees, such that the leaves of the child form a subset of the leaves of the parent. SIMULTANEOUS PQ-ORDERING asks whether orders of the leaves of each of the trees can be chosen *simultaneously*; that is, for every child-parent relation, the order chosen for the parent is an extension of the order chosen for the child. We show that SIMULTANEOUS PQ-ORDERING is $\mathcal{NP}$-complete in general, and we identify a family of instances that can be solved efficiently, the *2-fixed instances*. We show that this result serves as a framework for several other problems that can be formulated as instances of SIMULTANEOUS PQ-ORDERING. In particular, we give linear-time algorithms for recognizing simultaneous interval graphs and extending partial interval representations. Moreover, we obtain a linear-time algorithm for PARTIALLY PQ-CONSTRAINED PLANARITY for biconnected graphs, which asks for a planar embedding in the presence of PQ-trees that restrict the possible orderings of edges around vertices, and a quadratic-time algorithm for SIMULTANEOUS EMBEDDING WITH FIXED EDGES for biconnected graphs with a connected intersection. Both results can be extended to the case where the input graphs are not necessarily biconnected but have the property that each cutvertex is contained in at most two nontrivial blocks. This includes, for example, the case where both graphs have a maximum degree of 5.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.1 [**Discrete Mathematics**]: Combinatorics; G.2.2 [**Discrete Mathematics**]: Graph Theory

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Ordering problem, PQ-trees, planar embeddings, simultaneous embedding, interval graphs

ACM Reference Format:
Thomas Bläsius and Ignaz Rutter. 2015. Simultaneous PQ-ordering with applications to constrained embedding problems. ACM Trans. Algorithms 12, 2, Article 16 (December 2015), 46 pages.
DOI: http://dx.doi.org/10.1145/2738054

1. INTRODUCTION

Ordering objects in a specific way is a fundamental concept behind many applications. Probably the most basic ordering problem is sorting a totally ordered set. However, there may be less restrictive requirements on an order of elements in a set than a total order. Examples for such requirements are partially ordered sets or the requirement

Part of this work was done within GRADR – EUROGIGA project no. 10-EuroGIGA-OP-003.
A preliminary version of this article appeared as T. Bläsius, I. Rutter. Simultaneous PQ-Ordering with Applications to Constrained Embedding Problems. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13),* pages 1030–1043, 2013.
Authors' addresses: T. Bläsius, Prof.-Dr.-Helmert-Straße 2-3, Room A 1.4, 14482 Potsdam; email: thomas.blaesius@hpi.de; I. Rutter, Am Fasanengarten 5, Room 319, 76131 Karlsruhe; email: rutter@kit.edu.

DOI: http://dx.doi.org/10.1145/2738054

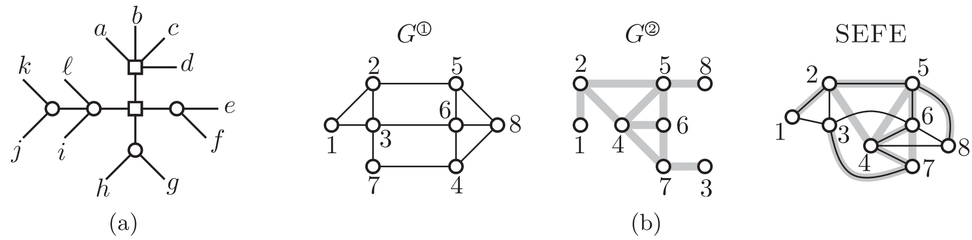


Fig. 1. (a) A PQ-tree with leaves $\{a, \ldots, \ell\}$ where P- and Q-nodes are depicted as circles and boxes, respectively. For example, the degree-5 Q-node at the top enforces the leaves a, b, c, h to occur in this or its reversed order. Furthermore, the two P-nodes on the left enforce the leaves 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.

that subsets of elements have to appear consecutively. Such requirements yield sets of possible (circular or linear) orders, and, in the two mentioned examples, these sets admit compact representations (i.e., polynomial in the number of elements although the set of orderings may be exponentially large). More precisely, the possible orders for a partially ordered set may be represented by a Directed Acyclic Graph (DAG), and all orders in which some specific subsets of elements appear consecutively can be represented by a PQ-tree [Booth and Lueker 1976]. A PQ-tree represents orders of its leaves by allowing edges around inner nodes to be either ordered arbitrarily (P-nodes) or by fixing this order up to reversal (Q-nodes); see Figure 1(a). Similarly, a matching on a set of vertices describes a set of possible orders; namely, all orders where no pair of matched vertices alternates. In this work, we do not consider the case where the order of elements of a single set is restricted in a specific way, but we introduce the concept of simultaneous orders for a family of sets. Namely, given sets of orders $\mathcal{L}_1, \ldots, \mathcal{L}_n$ on element sets $L_1, \ldots, L_n$, we seek orderings $O_i \in \mathcal{L}_i$ such that the common elements are ordered consistently. Note that this is generally $\mathcal{NP}$-hard if the sets of orders $\mathcal{L}_i$ are given as compact representations because it contains the $\mathcal{NP}$-hard problem Cyclic Ordering [Galil and Megiddo 1977].

Nevertheless, many special cases with interesting applications admit polynomial-time algorithms. For example, Klavík et al. [2011] essentially find a simultaneous ordering of a partially ordered set and a superset constrained by a PQ-tree to extend partial interval representations of graphs (an *interval representation* assigns an interval to each vertex such that intervals intersect if and only if the corresponding vertices are adjacent). Haeupler et al. [2010] solve a special case of the simultaneous embedding problem Simultaneous Embedding with Fixed Edges (SEFE), which asks for planar drawings of two graphs $G^{①}$ and $G^{②}$ such that their intersection G is drawn the same in both drawings (see Figure 1(b)) by repeatedly finding simultaneous orders for two PQ-trees. Angelini et al. [2012] show that a more general case of SEFE is equivalent to finding simultaneous orderings for a PQ-tree and two matchings. To find simultaneous interval representations of two graphs (where common vertices are represented by the same intervals), Jampani and Lubiw [2010] seek compatible clique orderings represented by a pair of PQ-trees. Angelini et al. [2010] and Gutwenger et al. [2008] find planar embeddings subject to constraints on orderings of edges around vertices. The problem Partially PQ-Constrained Planarity combines these problems by restricting the orders of subsets of edges using PQ-trees. Such constrained embedding problems also fall into the domain of simultaneous ordering problems for the following reason. Planar embeddings of graphs are determined by circular orderings of edges around vertices; thus, for each vertex, there is a set of possible orders. However, to obtain a planar embedding by choosing an ordering for each vertex, extensive compatibility conditions need to be satisfied, yielding a simultaneous ordering problem.

In this article, we make a first step to unify simultaneous ordering problems within a common framework. We consider the case where all orders are represented by PQ-trees leading to the problem SIMULTANEOUS PQ-ORDERING that is defined as follows. Let $D = (N, A)$ be a DAG with nodes $N = \{T_1, \dots, T_k\}$, where T_i is an unrooted PQ-tree representing a set $\mathcal{L}_i$ of circular orderings of its leaves L_i. Each arc $a \in A$ consists 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 $O_1, \dots, O_k$ 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. Note that this strictly generalizes the just described simultaneous ordering problem for PQ-trees; consistent orderings of common elements can be enforced by introducing common children for all pairs of trees sharing elements, using $\varphi = \mathrm{id}$. On the other hand, the injective maps can express more general relations between elements.

1.1. Related Work

Since this work touches on several different topics, we consider related work on constrained embedding problems, simultaneous embedding problems, PQ-trees and interval graphs separately.

PQ-Trees. PQ-Trees were originally introduced by Booth and Lueker [1976]. They were designed to decide whether a set L has the CONSECUTIVE ONES property with respect to a family $\mathcal{S} = \{S_1, \dots, S_k\}$ 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. In addition to testing planarity in linear time, they also showed how to decide in linear time whether a given graph is an interval graph. In the original approach by Booth and Lueker, the PQ-trees were rooted, representing linear orders of their leaves. However, Tucker [1971, Theorem 1] showed that CONSECUTIVE ONES orderings can also be used to represent circular orderings with a circular CONSECUTIVE ONES property. By considering PQ-trees to be unrooted, Haeupler and Tarjan [2008] give a corresponding interpretation of PQ-trees as a representation of circular orders. Unrooted PQ-trees are sometimes also called PC-trees [Hsu 2001; Hsu and McConnell 2001, 2003]. In most cases, we will use unrooted PQ-trees representing circular orders. However, the same results can be achieved for rooted PQ-trees representing linear orders by simply adding a single leaf to each tree (see Section 2.3 for further details).

Interval Graphs. Fulkerson and Gross [1965] gave a characterization of interval graphs in terms of the CONSECUTIVE ONES property, enabling Booth and Lueker [1976] to recognize them in linear time using PQ-trees. More recently, Klavík et al. [2011] give an $\mathcal{O}(n^2)$ time algorithm testing whether a given interval representation of a subgraph can be extended to an interval representation of the whole graph. Note that interval graphs are a special kind of intersection graphs in which vertices are represented by geometric objects such that vertices are adjacent if and only if their corresponding objects intersect. Jampani and Lubiw [2009] introduce a notion of simultaneous representations for any class of intersection graphs; namely, intersection representations where common vertices are represented by the same objects. They study the recognition problem for several graph classes. In a companion paper [Jampani and Lubiw 2010], they give an $\mathcal{O}(n^2 \log n)$-time algorithm for recognizing simultaneous interval graphs.

Constrained Embedding. Constrained embedding problems in general ask for a given planar graph and whether it can be drawn without crossings in the plane satisfying some additional constraints. Pach and Wenger [1998] show that every planar graph can be drawn crossing-free even if the vertex positions are prespecified by the application. Unfortunately, such a drawing may require linearly many bends per edge. Kaufmann and Wiese [2002] prove 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. Another constrained embedding problem is PARTIALLY EMBEDDED PLANARITY, which asks whether a planar drawing of a subgraph can be extended to a planar drawing of the whole graph. Angelini et al. [2010] give a linear-time algorithm for testing PARTIALLY EMBEDDED PLANARITY, and Jelínek et al. [2011] give a characterization by forbidden substructures similar to Kuratowski's theorem. 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. [2008] show that PQ-CONSTRAINED PLANARITY can be solved in linear time by simply replacing every vertex by a gadget and testing the planarity of the resulting graph (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 orientations of some Q-nodes are fixed.

Simultaneous Embedding. In addition to SEFE, there are other simultaneous embedding problems, such as SIMULTANEOUS EMBEDDING, only requiring the common vertices to be drawn at the same position, and SIMULTANEOUS GEOMETRIC EMBEDDING, requiring the edges to be straight-line segments [Erten and Kobourov 2005]. For an extensive survey on simultaneous embeddings of planar graphs see Bläsius et al. [2013]. We only consider SEFE, and we focus on the computational complexity of this problem. Gassner et al. [2006] show that it is $\mathcal{NP}$-complete to decide whether three or more graphs have an SEFE. Fowler et al. [2009] show how to solve SEFE efficiently if $G^{①}$ and G have at most two and one cycles, respectively. Haeupler et al. [2010] solve SEFE in linear time for the case that the common graph is biconnected. Angelini et al. [2012] obtain the same result with a completely different approach. They additionally solve the case where the common graph is a star.

All these approaches (and also the approach we present in this article) only consider orderings of edges around vertices. However, if the common graph G is disconnected, one additionally has to ensure that the relative positions of connected components to each other are the same with respect to the embeddings of $G^{①}$ and $G^{②}$. Bläsius and Rutter [2015] assume the other extreme, ignoring edge orderings and only considering relative positions. They show that SEFE can be solved in $\mathcal{O}(n)$ time if G is a set of disjoint cycles and in $\mathcal{O}(n^2)$ time if the embedding of each connected component of G is fixed. In a recent paper [Bläsius et al. 2014], this result is combined with the techniques by Angelini et al. [2012] and with the techniques presented in this article to solve cases of SEFE where one has to deal with relative positions and edge orderings at the same time. Schaefer [2013] introduces a completely different algebraic approach to SEFE based on the independent odd crossing number. This approach also leads to a polynomial time algorithm for different cases where the graph G has several non-trivial connected components.

1.2. Contribution and Outline

We first define basic notation and present known results, which we use throughout this article, in Section 2. In Section 3, we first give a precise problem definition for

Simultaneous PQ-Ordering and show that it is $\mathcal{NP}$-complete in general; see Section 3.1. In the remainder of that section, which forms the main part of this article, we characterize a subset of "simple" instances, the so-called *2-fixed instances*, for which a solution can be computed efficiently; namely, in quadratic time. We present several applications in Section 4, where we show how to formulate various problems as 2-fixed instances within the framework of Simultaneous PQ-Ordering, thus yielding efficient algorithms to solve them. The algorithms obtained in this way either solve problems that were not known to be efficiently solvable or significantly improve over the previously best running times.

In particular, we show that Partially PQ-Constrained Planarity can be solved in linear time for biconnected graphs; see Section 4.2. Note that this problem can be seen as a common generalization of the constrained embedding problems Partially Embedded Planarity [Angelini et al. 2010; Jelínek et al. 2011] and PQ-Constrained Planarity [Gutwenger et al. 2008]. The former completely fixes the order of some edges around a vertex; the latter partially fixes the order of all edges around a vertex. Partially PQ-Constrained Planarity partially fixes the order of some edges. Similar to the work of Gutwenger et al., we can also handle the case where some Q-nodes have a fixed orientation. In addition to that, SEFE can be formulated as a 2-fixed instance of Simultaneous PQ-Ordering if both graphs are biconnected and the common graph is connected, thus yielding a quadratic-time algorithm for this case; see Section 4.3. This strictly extends the results requiring that the common graph is biconnected [Haeupler et al. 2010; Angelini et al. 2012] for the following reason. If the intersection G of two graphs $G^{①}$ and $G^{②}$ is biconnected, it is completely contained in a single maximal biconnected component of $G^{①}$ and $G^{②}$, respectively. Thus, testing SEFE for $G^{①}$ and $G^{②}$ is equivalent to testing it for these two biconnected components since all remaining biconnected components can be attached if and only if they are planar. Moreover, we improve the previously best algorithms for recognizing simultaneous interval graphs [Jampani and Lubiw 2010] from $O(n^2 \log n)$ to linear (Section 4.4) and for extending partial interval representations [Klavík et al. 2011] from $O(n^2)$ to $O(n + m)$ (Section 4.5). We show that the results for Partially PQ-constrained Planarity and SEFE still hold if the input graphs have the property that each cutvertex is contained in at most two nontrivial blocks in Section 4.6. We conclude with some prospects for future work and some open question in Section 5.

We emphasize that all applications follow easily from the main results in Section 3. The formulations as instances of Simultaneous PQ-Ordering we use are straightforward and can easily be verified to be 2-fixed, at which point the machinery developed in the main part of this article takes over.

2. PRELIMINARIES

In this section, 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 are 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 biconnected planar graphs. In Section 2.5, we describe the relation between PQ-trees and SPQR-trees.

2.1. Graphs, Planar Graphs, DAGs and Trees

A graph $G = (V, E)$ 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*, respectively. 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 (with respect to inclusion) of G are called *connected components,* and the maximal biconnected subgraphs are called *blocks*. A complete subgraph of G is called a *clique*. A clique is *maximal* if it is not contained in a larger 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 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 directed cycle in G (with the face to its right), 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. A combinatorial embedding together with the choice of an outer face is a *planar embedding*. In most cases, we do not care about which face is the outer face, thus we mean a combinatorial embedding by simply saying embedding.

In a directed graph, we call the edges *arcs* and an arc from the *source* u to the *target* v is denoted by (u, v). A directed graph G without 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 1: $\text{depth}(u) < \text{depth}(v)$; $\text{level}(u) \geq \text{level}(v) - 1$. By the level and the depth of the DAG G itself, we mean the largest level and the largest depth of any vertex in G, respectively.

An *(unrooted) tree* T is a connected graph without cycles. The degree-1 vertices of T are called *leaves,* and the others 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 all edges towards 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). We use the following lemma, which for trees that do not contain degree-2 vertices bounds the tree size in terms of the number of leaves.

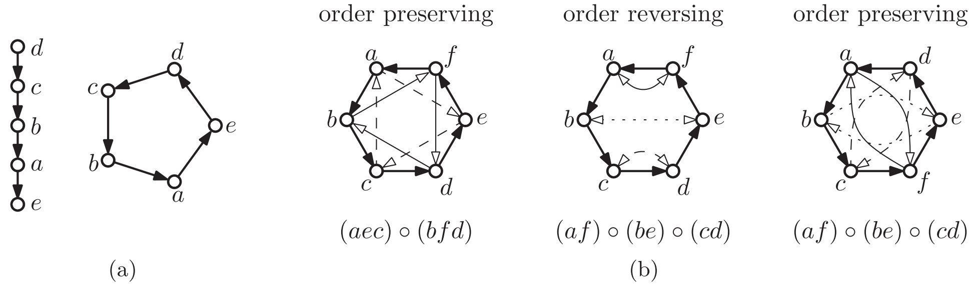


Fig. 2. (a) The interpretation of the linear and circular order *dcbae* as simple path and simple cycle, respectively. (b) The permutation $\varphi = (aec) \circ (bfd)$ on the left can be seen as as a clockwise rotation by 2 of the circular order *abcdef*, and thus is order preserving, whereas the permutation $\varphi = (af) \circ (be) \circ (cd)$ 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 distinguished by solid, dashed, and dotted lines.

Lemma 2.1. *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 maximum number of edges possible. Then every inner vertex in T has degree 3, because a vertex with four incident edges $e_1, \dots, e_4$ 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. Clearly, T has also the maximum number of inner vertices for the fixed number of leaves n_1. Let now n and m denote the total number of vertices and edges of T, respectively, and let n_3 denote the number of vertices of degree 3 in T. Since every vertex of T has either degree 3 or is a leaf, we have $n = n_1 + n_3$. Since T is a tree, we have $m = n - 1$, and, by counting the edge incidences, we get $2m = n_1 + 3n_3$. Together, these three equations imply $n_3 = n_1 - 2$, 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. Note that this is an equivalence relation that is independent from the chosen element ℓ. The equivalence classes are called *circular orders*. For example, for $L = \{a, \dots, e\}$ the orders $O_1 = baedc$ and $O_2 = dcbae$ are circularly 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(a) 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 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* $\varphi = (\ell_1\varphi(\ell_1)\dots\varphi^{k_1}(\ell_1)) \circ \cdots \circ (\ell_r\varphi(\ell_r)\dots\varphi^{k_r}(\ell_r))$. 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 described earlier (i.e., 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(b) 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. Additionally, we want to find an order that is preserved or reversed, respectively. Note that not fixing the order has the effect, for example, 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 2.2. *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.

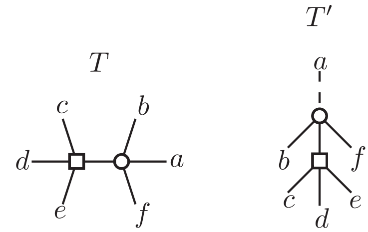
$$\ell_1\dots\ell_r \quad \varphi(\ell_1)\dots\varphi(\ell_r) \quad \dots \quad \varphi^k(\ell_1)\dots\varphi^k(\ell_k).$$

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 2.3. *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 $\varphi = (\ell_1\ell_1') \circ \cdots \circ (\ell_r\ell_r')$, $\varphi = (\ell) \circ (\ell_1\ell_1') \circ \cdots \circ (\ell_r\ell_r')$ or $\varphi = (\ell) \circ (\ell') \circ (\ell_1\ell_1') \circ \cdots \circ (\ell_r\ell_r')$. 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. □



(a) (b)

Fig. 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 $L' = \{b, \dots, f\}$. By choosing an arbitrary order for $b, \square, f$ where $\square$ represents either 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.

It is clear that the characterizations given in Lemma 2.2 and Lemma 2.3 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 (i.e., having a fixed combinatorial embedding), the circular order of the leaves (as they occur along the outer face of the embedding) 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 3(a). Note that not every set of orders is PQ-representable; for example, every PQ-representable set of orderings must be closed under 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. Haeupler and Tarjan [2008] show that there is an equivalence between unrooted and rooted PQ-trees; for completeness, we repeat their construction. 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 a 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 can work with its rooted equivalent instead by choosing one leaf to be the special leaf; see Figure 3 for an example. Conversely, rooted PQ-trees can be represented by unrooted ones by inserting a single leaf adjacent to the root. In most cases, we will work with unrooted PQ-trees representing sets of circular orders. Unless stated otherwise, we thus refer to *circular orders* and *unrooted PQ-trees* if we write orders and PQ-trees, respectively.

PQ-trees were introduced by Booth and Lueker [1976] 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 $\mathcal{L}$ is the empty set if in no order all subsets S_i appear consecutively. In this case, $\mathcal{L}$ 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 earlier, 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. 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 those orders in which all elements in S appear consecutively. The reduction can be seen as the operation for which PQ-trees were designed by Booth and Lueker [1976]. 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. Since the reductions with L and $S \setminus L$ are equivalent, we may assume without loss of generality that $\ell \notin S$, and we obtain the reduction of T by reducing T' with S, reinserting ℓ, and unrooting T' again. 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 [1975] showed how they can be applied consuming time linear in the size of T_1 and T_2. 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 straightforward. 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 whole 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 subscript 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* 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 order 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 notion 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 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 $\{s, t\}$ such that $G - s - t$ consists of two connected components. Let H_1 and H_2 be the two connected 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

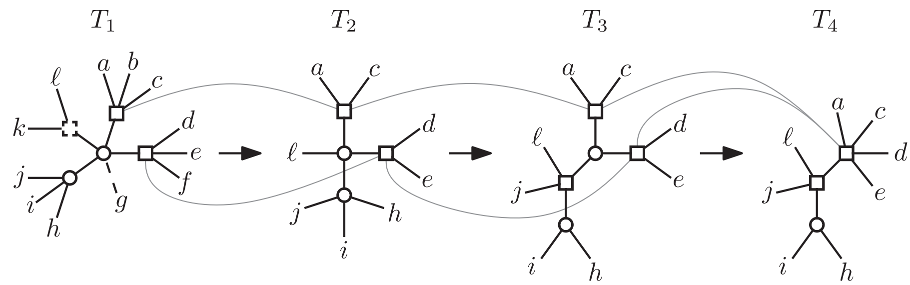


Fig. 4. We start with the PQ-tree T_1 on the left and project it to $L \setminus \{b, f, g, k\}$ yielding T_2. There is one Q-node and one edge incident to a P-node, both drawn dashed, that do not appear in T_2 and hence are free. The trees T_3 and T_4 are obtained by applying reductions with $\{\ell, j\}$ and $\{c, d\}$ to T_2. Note that the arrows (and even their transitive closure) can be interpreted as child-parent relations between the PQ-trees. Every fixed Q-node has a representative depicted by gray lines, whereas it is not possible to find something similar for the P-nodes.

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 skel(μ_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 skel(μ_1) and skel(μ_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 kind of replacement can, of course, be applied iteratively to the skeletons yielding a tree with more nodes but smaller skeletons associated with the nodes. Applying such a decomposition in a systematic way yields the SPQR-tree as introduced by Di Battista and Tamassia [1996a, 1996b]. The SPQR-tree $\mathcal{T}$ of a biconnected planar graph G contains four types of nodes. First, the P-nodes having a bundle of at least three parallel edges as skeleton and a combinatorial embedding 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. In addition to 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 [2001] showed how to compute it in linear time. Figure 5(a) 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; the order of the sets can be chosen arbitrarily. 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 in this case. Hence, there is a bijection between the P- and R-nodes of the SPQR-tree containing v and the P- and Q-nodes

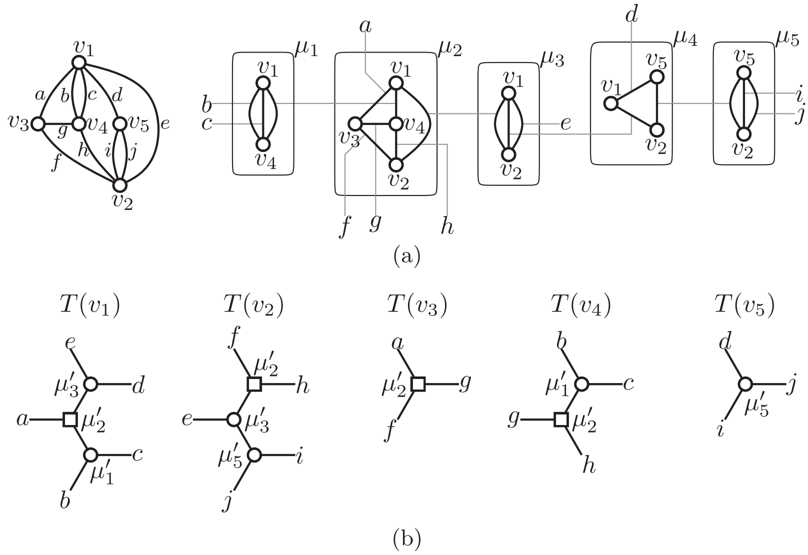


Fig. 5. (a) A biconnected planar graph on the left and its SPQR-tree on the right. The Q-nodes are depicted as single letters, whereas μ_1, μ_3, and μ_5 are P-nodes; μ_2 is an R-node, and μ_4 is an S-node. The embeddings chosen for the skeletons yield the embedding shown for the graph on the left. (b) The embedding trees of the five vertices, where the inner nodes are named according to the nodes in the SPQR-tree that they stem from.

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 5 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 $T(v)$. Conversely, for every order represented by $T(v)$, there exists 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 5(b) 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 any planar 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, thus 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 we have seen, all planar embeddings of a biconnected planar graph G can be expressed in terms of PQ-trees $T(v_1), \ldots, T(v_n)$, called the "embedding trees" and describing the orders of incident edges around every vertex, if we respect some additional constraints between the nodes of the embedding trees stemming from the same node of 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 $a \in A$ consists 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. If this is the case, we say that the orders O_i and O_j satisfy the arc $(T_i, T_j; \varphi)$. 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 lose 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 is possible 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 2.1. 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 a common 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 stemming from the same P-node of the SPQR-tree are ordered the same, but what we really want is that the two P-nodes are ordered oppositely. Therefore, we also need *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 the possible 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 that we can solve 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

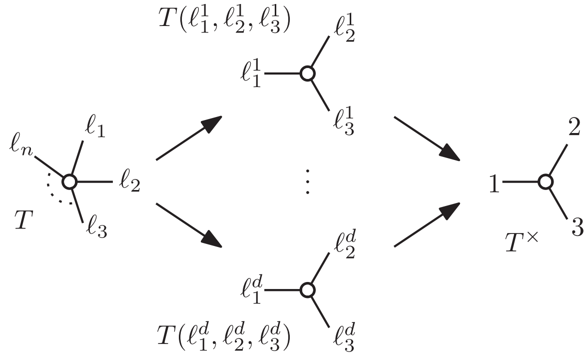


Fig. 6. The instance $D(L, \Delta)$ of SIMULTANEOUS PQ-ORDERING corresponding to the instance (L, Δ) of CYCLIC ORDERING with $L = \{\ell_1, \dots, \ell_n\}$ and $\Delta = \{(\ell_1^1, \ell_2^1, \ell_3^1), \dots, (\ell_1^d, \ell_2^d, \ell_3^d)\}$.

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 analysis of the running time, 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 (i.e., 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 [1977] proved that CYCLIC ORDERING is $\mathcal{NP}$-complete.

THEOREM 3.1. 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 whether 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 $\{\ell_1^i, \ell_2^i, \ell_3^i\}$ and 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 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 6 illustrates this construction. It is clear that the size of $D(L, \Delta)$ is linear in the size of (L, Δ) and that it can be computed in linear time. 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 that 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 in general, we give in this section a strategy for how to solve it for special instances. Later, in Section 3.3, we show that this strategy indeed leads to a polynomial-time algorithm for a certain 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. However, T can have several children $T_1, \dots, T_\ell$, and orders $O_i \in \mathcal{L}_i$ represented by T_i for $i = 1, \dots, \ell$ 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 $x_\mu \neq x_{\text{rep}(\mu)}$. Haeupler et al. [2010] use a similar technique to enforce consistent orientations of Q-nodes over several PQ-trees. 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 [Krom 1967] and even linear [Even et al. 1976; Aspvall et al. 1979] time. It is furthermore clear that we can now easily fix the orientation of some Q-nodes by adding corresponding clauses. 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 order the fixed edges incident to μ as given by O_i and add the free edges arbitrarily. 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 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

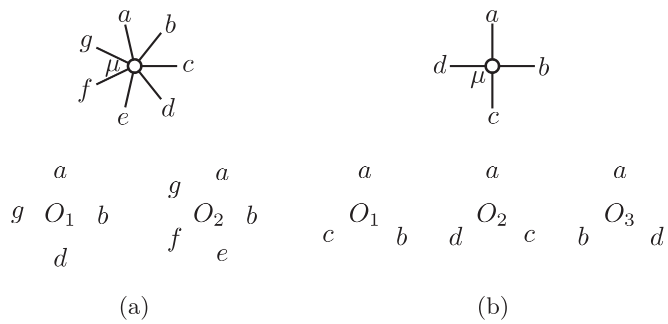


Fig. 7. (a) We can find an order for the P-node μ extending the orders O_1 and O_2 if and only if the commonly fixed edges a, b, and g are ordered the same. (b) Although for every pair $\{O_i, O_j\}$ of orders out of the three orders O_1, O_2, and O_3 the commonly fixed edges are ordered the same, we cannot extend all three orders simultaneously.

in at most one critical triple; that is, it is fixed with respect to at most two children of T. See Figure 7 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 3.2. *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 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 remaining 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 $T(\mu, T_1, T_2)$

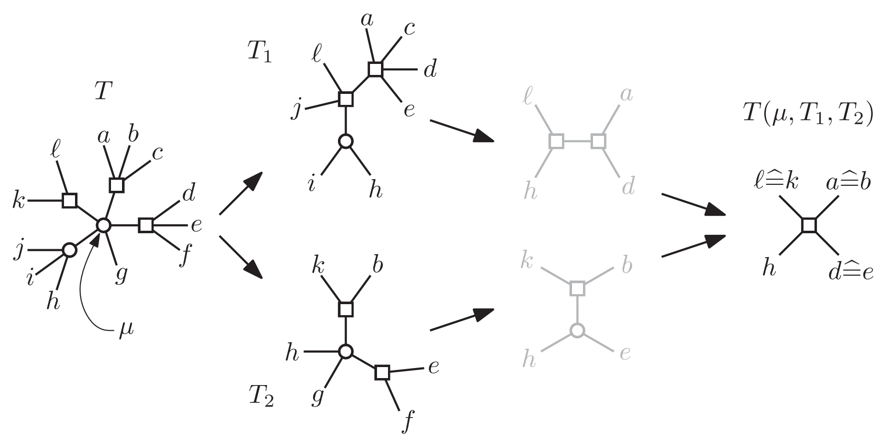


Fig. 8. The P-node μ in the PQ-tree T is fixed with respect to the children T_1 and T_2. We first project T_1 and T_2 to representatives of the common fixed edges incident to μ and intersect the result to obtain $T(\mu, T_1, T_2)$. Note that the gray shaded projections only illustrate an intermediate step; they are not inserted.

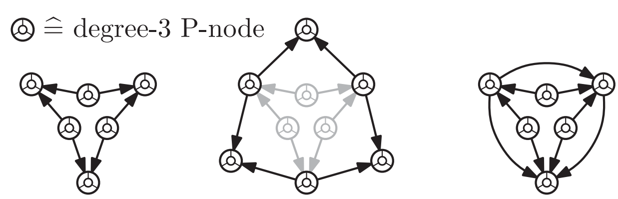


Fig. 9. Consider the instance of Simultaneous PQ-Ordering on the left, where every PQ-tree consists of a single P-node with degree 3. The DAG in the middle shows the result after expanding three times. The so far processed part is shaded gray, and, for the remaining part, we are in the same situation 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.

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 8 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. In other words, we ensure that O_1 and O_2 satisfy the critical triple (μ, T_1, T_2). By Lemma 3.2, we know that satisfying the critical triple is necessary; thus, we do not lose 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 then find simultaneous orders bottom-up. Unfortunately, it can happen that the expansion does not terminate and thus yields an infinite graph; see Figure 9 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, 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 from 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 earlier. If the conditions for a finalizing step are given (i.e., 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. In this case, we omit the loop and mark $(T, T'; \varphi_1)$ and $(T, T'; \varphi_2)$ as a *critical double arc* with the associated permutation φ. When choosing orders bottom-up in the DAG, we have to explicitly ensure that all critical triples stemming from critical double arcs are satisfied. To simplify this, we ensure that all targets of critical double arcs are sinks in the expansion graph. This follows from the construction, except for the case when the critical double arc is already contained in the input instance. In this case, we apply one additional expansion step, which essentially clones the double arc. We thus 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. Since we want to apply Lemma 3.2 by choosing orders bottom-up, it is a problem that the critical triples belonging to critical double arcs are not satisfied automatically. However, if every P-node is contained in at most one critical triple, our construction ensures 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' (which, by construction, consists of a single P-node), and we will use Lemma 2.2 (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 3.5 that D and D_{exp} are equivalent, we need to show that D_{exp} is well-defined (i.e., it is unique and finite). Lemma 3.3 essentially states that the P-nodes become smaller at least every second expansion step. We will use this result in Lemma 3.4 to show finiteness.

Lemma 3.3. *Let D be an instance of* Simultaneous PQ-Ordering *and let D_{exp} be its expansion graph. Furthermore, let 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 both consist 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 3.4. *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 only depends 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 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 3.3, the degree of μ_i is strictly 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 earlier: namely, that D and D_{exp} are equivalent.

Lemma 3.5. *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 lose solutions by applying expansion or finalizing steps. 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. Afterward, 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 $T(\mu, T_1, T_2)$ be 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. $\square$

For now, we know that we can consider the expansion graph instead of the original instance to solve SIMULTANEOUS PQ-ORDERING. Lemma 3.2 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. Moreover, the expansion graph may become exponentially large in general. In the following section, we 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 and 2-Fixed 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 3.2 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 3.6. *Let D be a 1-critical instance of* SIMULTANEOUS PQ-ORDERING *with expansion graph D_{exp}. Furthermore, let $(T, T'; \varphi_1)$ and $(T, T'; \varphi_2)$ be a critical double arc. Then T' is a sink in D_{exp}.*

PROOF. Recall that, as a target of a critical double arc, T' consists of a single P-node. Hence, 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 that is 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. $\square$

LEMMA 3.7. *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 3.5, 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 3.2 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, thus satisfying the Q-constraints by solving a linear size instance of 2-SAT [Even et al. 1976; Aspvall et al. 1979]. 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; 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 2.2. 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}.

For each sink T in D_{exp}, we choose a circular ordering as follows. 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. Hence, we can use Lemma 2.2 to choose an order that is preserved by φ. Otherwise, orient every Q-node μ in T as determined by the variable x_μ representing it. Additionally, choose an arbitrary order for every P-node in T. Afterward, mark T as processed. We consider the remaining PQ-trees in a bottom-up order. Let T be a PQ-tree in D_{exp} for which all of its children $T_1, \dots, T_\ell$ are already processed. Since $T_1, \dots, T_\ell$ are processed, orders $O_1, \dots, O_\ell$ 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 3.6 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 3.2 to extend the orders $O_1, \dots, O_\ell$ simultaneously to an order O represented by T. This extension can clearly be computed in polynomial time; 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 earlier, 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 (i.e., 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 3.8. *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. Furthermore, let n and m denote the number of vertices and edges in T, respectively. We obtain the following equation by double counting.

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

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 (1):

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

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 (2) yields the claim. □

LEMMA 3.9. *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 $\mathcal{O}(|D|^2)$. Afterward, 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 2.1. 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 this more precise, 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 3.3. Otherwise, T contains k P-nodes $\mu_1, \dots, \mu_k$ 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 for all the PQ-trees for which $\mu_1, \dots, \mu_k$ are responsible, 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 3.8 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 μ 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 3.7 and Lemma 3.9 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 3.10. SIMULTANEOUS PQ-ORDERING *can be solved in polynomial time for 1-critical instances.*

Actually, Theorem 3.10 tells us how to solve 1-critical instances, which was the main goal of this section. However, the characterization of 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 we 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 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. Furthermore, let 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' that 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 define the *fixedness* $\text{fixed}(\mu)$ of μ to be the number of children fixing it.

Now let μ be a P-node of some internal PQ-tree T of D with parents $T_1, \dots, T_\ell$. Each of the trees T_i contains exactly one P-node μ_i that is fixed by μ. Additionally, let k' be the number of children fixing μ. We set $\text{fixed}(\mu) = k' + \sum(\text{fixed}(\mu_i) - 1)$. We say that a P-node μ is *k-fixed*, if $\text{fixed}(\mu) \leq k$ and an instance D is *k-fixed* for some integer k if all its P-nodes are k-fixed. 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 3.11. *Every 2-fixed instance of* SIMULTANEOUS PQ-ORDERING *is 1-critical.*

PROOF. Let D be a 2-fixed instance of SIMULTANEOUS PQ-ORDERING and let D_{exp} be its expansion graph. 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 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 $T_1, \dots, T_\ell$ be the parents of T and let $\mu_1, \dots, \mu_\ell$ be the corresponding P-nodes that μ stems from. Furthermore, letμ 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 $i = 1, \dots, \ell$ 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 earlier, 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 3.10 and Theorem 3.11 together provide a framework for solving problems that can be formulated as instances of SIMULTANEOUS PQ-ORDERING. We can use Theorem 3.11 to prove that the instances our application produces are 1-critical, whereas Theorem 3.10 tells us that we can solve these instances in polynomial time. Note that, since the Q-constraints are expressed as a 2-SAT formula, the algorithm still works if the orientation of some Q-nodes is given with the input.

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 3.9, 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 as 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 straightforward. Afterward, 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 remainder of this section, let $D = (N, A)$ be a 1-critical instance of SIMULTANEOUS PQ-ORDERING with the expansion graph $D_{\text{exp}} = (N_{\text{exp}}, A_{\text{exp}})$. Furthermore, let $|D|$, $|N|$, $|A|$, $|D_{\text{exp}}|$, $|N_{\text{exp}}|$, 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 earlier, we want to compute and store some additional information in addition to 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 that 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 that 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 3.12. *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 3.9, we already showed that the expansion graph of a 1-critical instance has quadratic size. However, this can be done more precisely.

LEMMA 3.13. *Let D be a 1-critical instance of* SIMULTANEOUS PQ-ORDERING *with the expansion graph $D_{\exp}$. Then $|D_{\exp}| \in \mathcal{O}(p_{\max} \cdot |N| + |A|)$, where $p_{\max}$ is the degree of the largest P-node in D.*

PROOF. The proof of Lemma 3.9 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 $\mu_1, \ldots, \mu_\ell$ 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 3.9, the size of all newly created arcs in $D_{\exp}$ is linear in the size of all nodes in $D_{\exp}$. Thus, we obtain $|D_{\exp}| \in \mathcal{O}(p_{\max} \cdot |N| + |A|)$ 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 whether it is contained in both projections. This consumes $\mathcal{O}(\deg(\mu))$ 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 is stored. However, this projection was already computed and can simply 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 3.14. *The expansion graph $D_{\exp}$ of a 1-critical instance $D = (N, A)$ of* SIMULTANEOUS PQ-ORDERING *can be computed in $\mathcal{O}(|N|^2)$ time.*

PROOF. As mentioned earlier, 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

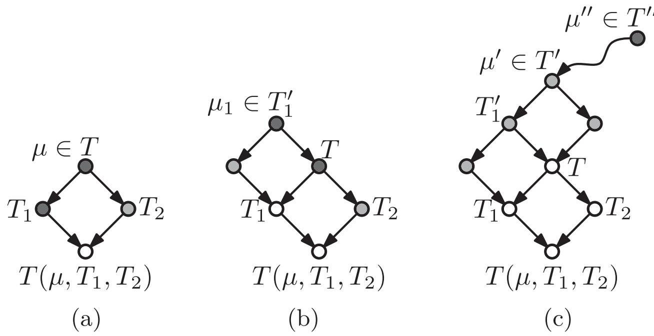


Fig. 10. Nodes in the original graph are shaded dark gray and expansion trees white. Light gray is used where it does not matter. (a) The case where T_1 is contained in the original graph. (b) The case where T_1 is an expansion graph but T containing μ is not. (c) The case where neither T_1 nor T are expansion graphs.

different parts of D by 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 cost $\mathcal{O}(|N|)$ 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 $T_1 \in N$ (i.e., 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_{exp}, we obtain that the total cost is in $\mathcal{O}(|N|^2)$. Figure 10(a) illustrates this case.

If $T_1 \notin N$ but $\mu \in T \in N$ (i.e., 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 10(b). 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 $\mu \in T \notin N$ (i.e., 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 10(c). 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 must 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 in $\deg(\mu'')$ to μ''. As shown in the proof of Lemma 3.9, 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 3.7, 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. This is shown in the following lemma.

LEMMA 3.15. *Let D be a 1-critical instance of* SIMULTANEOUS PQ-ORDERING *with expansion graph $D_{\exp}$. In $\mathcal{O}(|D_{\exp}|)$ 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 3.7. 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 that 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 that 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 that 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 every P-node μ in T for up to two subsets of edges incident to μ, orders are stored. If we have two orders, these orders need to be merged, which can clearly be done in linear time. Thus, we can assume to have only one order on a subset of edges. All remaining free edges can be simply added arbitrarily. This takes overall $\mathcal{O}(\deg(\mu))$ time. Hence, we need, for each node in T, linear time in its degree and hence $\mathcal{O}(|T|)$ time for the whole tree. All together, 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 $D_{\exp} = (N_{\exp}, A_{\exp})$ takes $\mathcal{O}(|N_{\exp}| + |A_{\exp}|) = \mathcal{O}(|D_{\exp}|)$ time. □

Overall Running Time. For applications producing instances of SIMULTANEOUS PQ-ORDERING it may be possible that reconsidering the runtime analysis containing normalization, size, 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 Lemmas 3.12, 3.13, 3.14, and 3.15 together. Note that the running time is dominated by the computation of the expansion graph.

THEOREM 3.16. 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 whether 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 $N = \{T_1, \dots, T_k\}$, 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 $O_1, \dots, O_k$ 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 reverse 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. We first give a rough overview. The definitions of the Q-constraints and the critical triples can be modified in a straightforward manner such that Lemma 3.2, 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, still holds. 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 3.3, 3.4, and 3.5 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 3.6, stating that critical double arcs have a sink as target, works as before. In Lemma 3.7, 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 be done in the same way as before. 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 2.3 instead of Lemma 2.2. Afterward, it is again ensured that every critical triple is satisfied; hence, we can choose orders bottom-up as before. Lemma 3.9 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 3.7 and Lemma 3.9 together yielding that SIMULTANEOUS PQ-ORDERING WITH REVERSING ARCS can be solved in polynomial time for 1-critical instances as stated before in Theorem 3.10 for SIMULTANEOUS PQ-ORDERING. Theorem 3.11 provides an easy criterion that an instance that is 1-critical works exactly the same as before.

In the following, we highlight some important details. Let us start with the Q-constraints. Let μ be a Q-node in T that is fixed with respect to the child T' of T and let $\operatorname{rep}(\mu)$ be its representative in T'. To ensure that μ is ordered as determined by $\operatorname{rep}(\mu)$, we introduced either the constraint $x_\mu = x_{\operatorname{rep}(\mu)}$ or $x_\mu \neq x_{\operatorname{rep}(\mu)}$. Now, if the arc (T, T') is reversing, we simply negate this constraint, thus ensuring that μ is orientated oppositely to the orientation determined by $\operatorname{rep}(\mu)$. 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 symbolize 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 from O_2 for the commonly fixed edges incident to μ. With these straightforwardly adapted definitions, it is clear that the proof of Lemma 3.2 works exactly as before. To improve readability, we cite this lemma here.

LEMMA 3.17. *Let T be a PQ-tree with children $T_1, \dots, T_\ell$, such that every P-node in T is contained in at most one critical triple, and let $O_1, \dots, O_\ell$ be orders represented by $T_1, \dots, T_\ell$. An order O that is represented by T and simultaneously extends the orders $O_1, \dots, O_\ell$ 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 triples are satisfied. This 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, thus 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, thus 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, all structural results on the expansion graph of an instance of Simultaneous PQ-Ordering still hold if we allow reversing arcs. Particularly, we have that the expansion graph is well-defined (Lemmas 3.3 and 3.4), that the target of every unsatisfied double arc is a sink if D is 1-critical (Lemma 3.6), that $|D_{\text{exp}}|$ is polynomial in $|D|$ if D is 1-critical (Lemma 3.9), and that D is 1-critical if it is at most 2-fixed (Theorem 3.11). 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 3.5) 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 3.7). In the proof of Lemma 3.5, 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 3.7 can be adapted straightforwardly, since Lemma 3.2 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 2.2. 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 2.3 instead of Lemma 2.2. Finally, Lemma 3.7 also works if we allow reversing arcs and hence we obtain the following theorem analogously to Theorem 3.16.

Theorem 3.18. Simultaneous PQ-Ordering with Reversing Arcs *can be solved in* $\mathcal{O}(|N|^2)$ *time for a 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 no longer distinguish between these two problems. Thus, if we create 1-critical instances of Simultaneous PQ-Ordering in our applications, we implicitly 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 SEFE for biconnected graphs with a connected intersection (Section 4.3). Furthermore, we show in Sections 4.4 and 4.5 how Simultaneous PQ-Ordering can be used to recognize simultaneous interval graphs and extend partial interval representations in linear time. Finally, in Section 4.6, we reconsider the embedding representation and relax the biconnectivity requirement of the input graph. This results in efficient algorithms for Partially PQ-Constrained Planarity and SEFE for a larger class of graphs, including, for example, maxdeg-5 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 call them *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. 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 $(T(v_2), -P(\mu))$.

When we solve the PQ-embedding representation $D(G)$ as an instance of Simultaneous PQ-Ordering, we choose orders bottom-up. Thus, we 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 11 depicts the

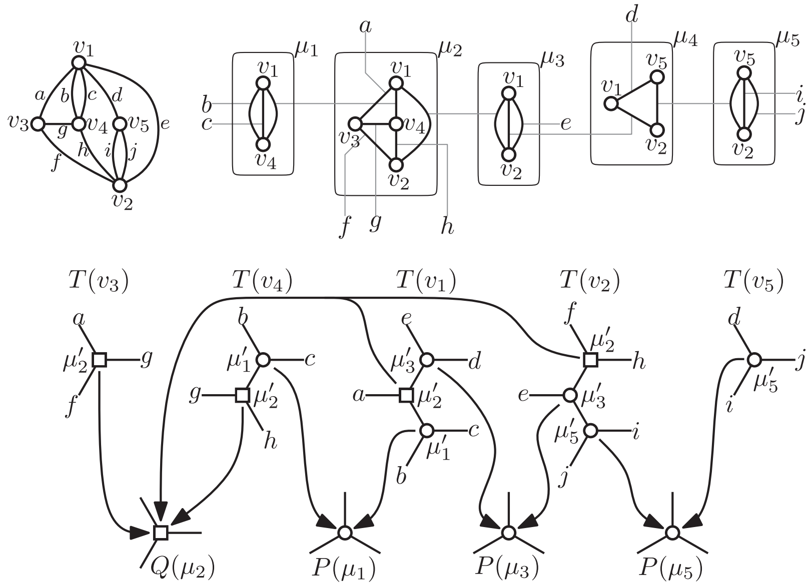


Fig. 11. A biconnected planar graph and its SPQR-tree on the top and the corresponding PQ-embedding representation on the bottom. The injective maps on the edges are not explicitly depicted, but the starting points of the arcs suggest which maps are suitable.

PQ-embedding representation for the example we had before in Figure 5(b). 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 with G biconnected, it is straightforward to formulate it as an instance of Simultaneous PQ-Ordering. 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 12 depicts an example instance of Partially PQ-Constrained Planarity formulated as an 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

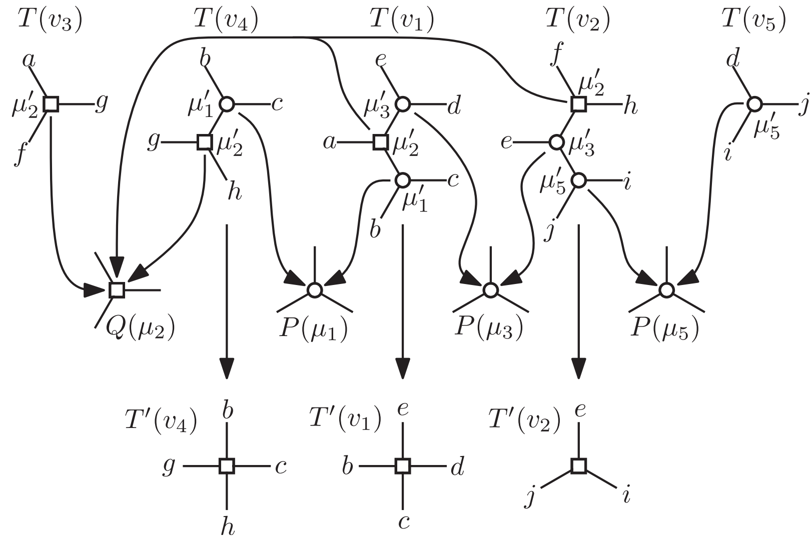


Fig. 12. The PQ-embedding representation from Figure 11 together with the constraint trees provided by an instance of Partially PQ-Constrained Planarity.

equivalent, which is quite obvious, and that $D(G, C)$ is an at most 2-fixed instance of Simultaneous PQ-Ordering.

Theorem 4.1. 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. Furthermore, let $D(G, C)$ 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 3.16 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 3.11, 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 (i.e., $\text{fixed}(\mu) \leq 2$). 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 $\text{fixed}(\mu') = \text{fixed}(\mu) - 1$. Since μ is a P-node in an embedding tree, we obtain $\text{fixed}(\mu') \leq 1$. 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 from 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 $\text{fixed}(\mu) = (\text{fixed}(\mu_1) - 1) + (\text{fixed}(\mu_2) - 1)$. Since μ_1 and μ_2 are P-nodes in embedding trees, this yields $\text{fixed}(\mu) \leq 2$. 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 3.11 yielding that $D(G, C)$ is 1-critical and thus can be solved quadratic time due to Theorem 3.16. □

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

Theorem 4.2. Partially PQ-Constrained Planarity *can be solved in linear time for biconnected graphs.*

Proof. As described 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 3.12), the expansion graph has quadratic size in worst case (Lemma 3.13), its computation consumes quadratic time (Lemma 3.14), and, finally, choosing borders bottom-up needs linear time in the size of the expansion graph (Lemma 3.15).

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 by 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 4.1, 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 $E = E^{①} \cap E^{②}$. SEFE 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 [2009, Theorem 4] 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.

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 straightforward 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 4.3. SEFE *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. Furthermore, let $D(G^{①}, G^{②})$ be the corresponding instance of Simultaneous PQ-Ordering as defined earlier. 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 4.1. 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 $T(v)$ that 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. At this point, we can apply Theorems 3.11 and 3.16 to obtain a solution in quadratic time. □

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 13(a) 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

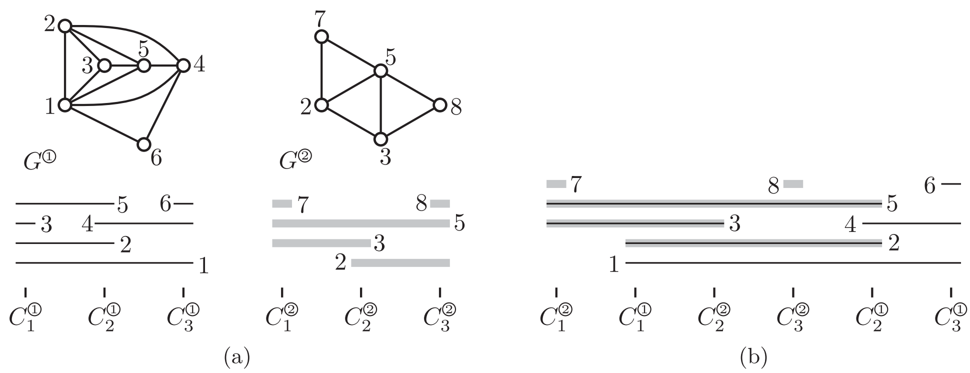


Fig. 13. (a) Two interval graphs $G^{①}$ and $G^{②}$ with interval representations. The maximal cliques are $C_1^{①}, C_2^{①}, C_3^{①}$ and $C_1^{②}, C_2^{②}, C_3^{②}$, respectively. (b) Interval representations of $G^{①}$ and $G^{②}$ such that common vertices are represented by the same interval in both representations; in other words, a simultaneous interval representation of $G^{①}$ and $G^{②}$.

common vertices are represented by the same intervals in both representations; see Figure 13(b) for an example. The problem to decide whether a pair $(G^{①}, G^{②})$ of graphs are simultaneous interval graphs is called Simultaneous Interval Representation.

The first algorithm recognizing interval graphs in linear time was given by Booth and Lueker [1976] and was based on a characterization by Fulkerson and Gross [1965]. 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. The problem Simultaneous Interval Representation was first considered by Jampani and Lubiw [2010], who show how to solve it in $\mathcal{O}(n^2 \log n)$ time.

In Theorem 4.4, 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 4.5. 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 $C_1, \dots, C_\ell$ 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 4.4 ([Fulkerson and Gross 1965]). *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 $C = \{v_1, \dots, v_k\}$ be a maximal clique in G. It is clear that there must be a position x such that x is contained in the intervals $I(v_1), \dots, I(v_k)$. Additionally, x is not contained in any interval represented by another vertex since the clique C is maximal. By fixing such positions $x_1, \dots, x_\ell$ for each of the maximal cliques $C_1, \dots, C_\ell$ 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 $I(v)$ necessarily contains x_i and x_k, and, hence, also x_j. This 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 $O = C_1 \dots C_\ell$ 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 maximal cliques $C_1, \dots, C_\ell$ at the natural numbers $1, \dots, \ell$, since, for each clique $C_i = \{v_1, \dots, v_k\}$, the position i is contained in all the intervals $I(v_1), \dots, I(v_k)$. 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 $1, \dots, \ell$ all edges in G are represented and no edges not in G. Furthermore, all intervals $I(v)$ containing a noninteger position $1 < x < \ell$ contain also $\lceil x \rceil$ and $\lfloor x \rfloor$, yielding that no edge is defined due to position x that 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 4.5. *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, $I(v)$ be the interval representing v. Assume $\mathcal{C}^{①} = \{C^{①}_1, \dots, C^{①}_k\}$ and $\mathcal{C}^{②} = \{C^{②}_1, \dots, C^{②}_\ell\}$ are the maximal cliques in $G^{①}$ and $G^{②}$, respectively. When considering $G^{①}$ for itself, we again obtain for every maximal clique $C^{①} = \{v_1, \dots, v_r\}$ 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 4.4, 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. This 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 4.4.

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 $C_1 \dots C_{k+\ell}$ 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 noninteger position x is only contained in intervals also containing $\lceil x \rceil$ and $\lfloor x \rfloor$. Thus, we only need to consider the positions $1, \dots, k+\ell$. Let i be such an integral position and assume, without loss of generality, that $C_i = \{v_1, \dots, v_r\}$ is a clique of $G^{①}$. Then i is contained in all the intervals $I(v_1), \dots, I(v_r)$ 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 in the graph 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. This can be achieved as mentioned in the preliminaries about PQ-trees (Section 2.3). Consider an instance of SIMULTANEOUS PQ-ORDERING having rooted PQ-trees as nodes. By introducing for every PQ-tree a new leaf ℓ, the special leaf, on top of the root, unrooting the PQ-tree and setting $\varphi(\ell) = \ell$ for every arc $(T, T'; \varphi)$, we obtain an equivalent instance of SIMULTANEOUS PQ-ORDERING having unrooted PQ-trees as nodes. Thus, solving SIMULTANEOUS PQ-ORDERING for the case that the PQ-trees are rooted reduces to the case where the PQ-trees are unrooted. Note that the other direction does not work as simply since we cannot necessarily find a single leaf ℓ contained in every PQ-tree. The PQ-trees mentioned in the remaining part of this section are assumed to be rooted, representing linear orders.

THEOREM 4.6. SIMULTANEOUS INTERVAL REPRESENTATION *can be solved 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 4.5, 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 3.16 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. □

4.5. Extending Partial Interval Representations

Let G be a graph, let $H = (V, E)$ be a subgraph of G, and let I be an interval representation of H. The problem PARTIAL INTERVAL GRAPH EXTENSION asks whether there exists an interval graph representation I' of G such that, for all $v \in V$, we have that $I'(v) = I(v)$. We call an instance (G, H, I) of PARTIAL INTERVAL GRAPH EXTENSION a *partial interval graph*.

Klavík et al. [2011] show that PARTIAL INTERVAL GRAPH EXTENSION can be solved in time $O(n^2)$, where $n = |V(G)|$. We show that PARTIAL INTERVAL GRAPH EXTENSION can be reduced in $O(n + m)$ time to an instance of SIMULTANEOUS INTERVAL REPRESENTATION. It then follows from Theorem 4.6 that the partial interval graph extension problem can be solved in $O(n + m)$ time, where $m = |E(G)|$.

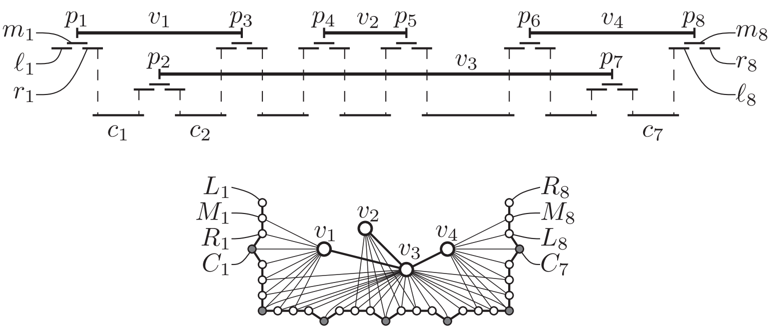


Fig. 14. An example graph H containing the vertices $v_1, \ldots, v_4$ with prescribed interval representation I together with the markers ℓ_i, m_i, r_i and the connectors c_i on the top. The resulting graph G' with the new vertices L_i, M_i, R_i, and C_i on the bottom.

Without loss of generality, we assume that the endpoints of all intervals $I(v), v \in V(H)$ are distinct. For $v \in V(H)$, let $\ell(v)$ and $r(v)$ denote the left and right endpoints of $I(v)$, respectively. Further, let $S(I)$ denote the sequence of these endpoints in increasing order of coordinate. We call this order the *signature* of I. We say that two interval representations I and I' of the same graph H are *equivalent* if they have the same signature. Klavík et al. [2011] show that Partial Interval Graph Extension for a partial interval graph (G, H, I) is equivalent to deciding whether there exists an interval representation I' of G whose restriction to H is equivalent to I. In the following, we construct an interval graph G' containing H as an induced subgraph such that every interval representation of G' induces an interval representation of H that is equivalent to I.

Let $p_1, \ldots, p_{2n}$ denote the interval endpoints of I in increasing order. We now add several intervals to the representation. Namely, for each point p_i, we put three intervals of length ε. The interval ℓ_i is to the left of p_i, interval r_i is to the right of p_i, and m_i contains p_i and intersects both ℓ_i and r_i. We choose ε small enough so that no two intervals of distinct points p_i and p_j intersect. We call these intervals *markers*. Finally, we add $2n-1$ *connectors*, where the connector c_i for $i = 1, \ldots, 2n-1$ lies strictly between p_i and p_{i+1} and intersects r_i and ℓ_{i+1}; see Figure 14 for an example. Now consider the graph G' given by this interval representation. We denote the vertices corresponding to the intervals ℓ_i, m_i, r_i, and c_i by L_i, M_i, R_i, and C_i, respectively. Note that removing all these vertices from G' yields the graph H, which is hence an induced subgraph of G'. Now the pair (G, G') defines an instance of Simultaneous Interval Representation corresponding to the instance (G, H, I) of Partial Interval Graph Extension and we obtain the following theorem by showing their equivalence.

Theorem 4.7. *The problem* Partial Interval Graph Extension *can be solved in linear time.*

Proof. Let (G, H, I) be an instance of Partial Interval Graph Extension and let (G, G') be the corresponding instance of Simultaneous Interval Representation as defined earlier. We show that these two instances are equivalent and that (G, G') has size linear in the size of (G, H, I).

Obviously G' contains H as an induced subgraph. We claim that, in any interval representation I' of G', the subrepresentation $I'|_H$ is equivalent to I. First, note that the sequence $L_1, M_1, R_1, C_1, \ldots, C_{2n-1}, L_{2n}, M_{2n}, R_{2n}$ is an induced path in G'. Hence, in every representation of G', the starting points (and also the endpoints) of their intervals occur either in this or in the reverse order. In particular, the marker intervals $I'(M_i)$

are pairwise disjoint and sorted. Let v_i denote the vertex whose interval has p_i as an endpoint. Since M_i is adjacent to L_i and R_i, exactly one of which is adjacent v_i, it follows that $I'(M_i)$ contains an endpoint of $I'(v_i)$. Since this holds for each marker M_i, the claim follows.

With this result, the equivalence of the instance (G, H, I) and (G, G') is easy to see. If (G, H, I) admits an interval representation of G, then the preceding construction shows how to construct a corresponding simultaneous representation of (G, G'). On the other hand, if G and G' admit a simultaneous interval representation, then the endpoints of the intervals corresponding to vertices of H must occur in the same order as in I, and hence the interval representation of G extends I.

It remains to show that G' has size linear in the size of H. To this end, we revisit the construction of G' from H. Let H' be the subgraph of G' obtained by removing the vertices corresponding to connectors. We first show that the size of H' is linear in the size of H.

Clearly, H' contains exactly six additional vertices for each vertex of H (three for each endpoint of an interval representing a vertex of H), and thus $|V(H')| = 7n$. Now consider the edges of H'. We denote by $I(p)$ the set of vertices whose intervals contain p in the *interior*. Let again $p_1, \dots, p_{2n}$ denote the endpoints of the intervals in the interval representation I of H. Recall that, for each such endpoint, we add three vertices, which are represented by the intervals ℓ_i, m_i, and r_i, respectively. Note that the endpoints p_{i-1} and p_{i+1} (if they exist) lie to the left of ℓ_i and to the right of r_i, respectively, and hence do not intersect with these intervals. The neighbors of L_i, M_i, and R_i belonging to H are contained in $I(p_i) \cup \{v_i\}$. This implies that the degree of L_i, M_i, and R_i is linear in the degree of v_i in H, and hence the total number of edges in H' is linear in $|E(H)|$.

For the step from H' to G', we add the connectors. Consider the ith connector C_i, which is adjacent to R_i and L_{i+1}. Since no other intervals start or end in-between, the vertex corresponding to the connector C_i is adjacent to the same vertices as R_i and L_{i+1}. Thus, the size of G' is linear in the size of H', and the claim follows. Moreover, it is clear that, assuming the intervals of I are given in sorted order, then G' can be constructed from G in $O(n + m)$ time. □

4.6. Generalization to Non-Biconnected Graphs

In this section, we return to the embedding representation derived in Section 4.1 and relax the condition that the graph must be biconnected to have such a representation. The reason that our solutions for Partially PQ-Constrained Planarity and SEFE are restricted to the case where the graphs are biconnected is that the set of possible orders of edges around a cutvertex may not be PQ-representable. However, this is not really necessary. Assume we have a representation of all embeddings of a planar graph as an instance of Simultaneous PQ-Ordering with the following two properties. First, this instance contains a PQ-tree $T(v)$ for every vertex v having the edges incident to v as leaves. And, second, this instance remains 1-critical even if we introduce an additional child to $T(v)$. If this is the case, Partially PQ-Constrained Planarity can be solved by introducing the constraint tree $T'(v)$ as child of $T(v)$. Similarly, in the setting of SEFE, common edges around a vertex v can be enforced to be ordered the same by introducing a common embedding tree $T(v)$ having the common edges incident to v as leaves as child of the trees $T(v^{①})$ and $T(v^{②})$, where $T(v^{①})$ and $T(v^{②})$ have the edges incident to v in $G^{①}$ and $G^{②}$ as leaves, respectively. We show that all embeddings can be represented by such an instance for the special case that every cutvertex is contained in only two blocks. Furthermore, this extends to the case where each block containing the cutvertex v consists of a single edge except for up to two blocks.

Consider a cutvertex v in a planar graph G that is contained in two blocks $B_1(v)$ and $B_2(v)$ and let $E_1(v)$ and $E_2(v)$ be the edges incident to v contained in $B_1(v)$ and

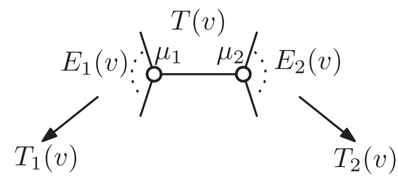


Fig. 15. Representation of the possible orders of edges around a cutvertex v for the special case that v is contained in two blocks in terms of an instance of Simultaneous PQ-Ordering.

$B_2(v)$, respectively. As before, the orders of $E_1(v)$ around v that can occur in a planar drawing can be represented by a PQ-tree $T_1(v)$ with $E_1(v)$ as leaves; call $T_1(v)$ the *block embedding tree* with respect to B_1. Let $T_2(v)$ be the block embedding tree of v with respect to the second block B_2. It is clear that, in a planar drawing of the whole graph, the edges $E_1(v)$ (and with it also $E_2(v)$) appear consecutively around v. This condition can be formulated independently from the PQ-trees $T_1(v)$ and $T_2(v)$ by an other PQ-tree $T(v)$ consisting of two P-nodes μ_1 and μ_2 with the edge $\{\mu_1, \mu_2\}$ and leaves $E_1(v)$ and $E_2(v)$ attached to μ_1 and μ_2, respectively. It is clear that the instance of Simultaneous PQ-Ordering consisting of the PQ-tree $T(v)$ with $T_1(v)$ and $T_2(v)$ as children represents all possible circular orders of edges around v in the sense that, in every planar embedding, the order of edges around v induces a solution of this instance and vice versa; Figure 15 depicts this instance of Simultaneous PQ-Ordering. We call the PQ-tree $T(v)$ the *combined embedding tree* of v. Of course, the order of edges around each vertex cannot be chosen independently, but, since the block embedding trees are embedding trees of biconnected components, the P- and Q-nodes stem from P- and R-nodes in the SPQR-tree. We can thus again ensure consistency by introducing the consistency trees for each block. This yields an extension of the PQ-embedding representation to the case that G may contain cutvertices that are contained in two blocks. Note that the embedding tree of a vertex that is not a cutvertex can be seen as combined and block embedding tree at the same time.

It is easy to see that this representation satisfies the conditions mentioned earlier. First, the combined embedding tree $T(v)$ has the edges incident to v as leaves. Second, if an additional child is introduced to every combined embedding tree, the instance remains 2-fixed, which can be seen as follows. The combined embedding tree has three children, the two block embedding trees and the additional child. However, each P-node in $T(v)$ is fixed with respect to only one of the block embedding trees, thus it is 2-fixed. Every P-node in the block embedding trees is fixed with respect to one child, the corresponding consistency tree, thus it is 2-fixed since it has the combined embedding tree as parent. The P-nodes in consistency trees are also 2-fixed since they have two 2-fixed parents. Hence, we obtain a 2-fixed instance if we use this extended PQ-embedding representation to formulate Partially PQ-Constrained Planarity or SEFE as an instance of Simultaneous PQ-Ordering. Furthermore, the runtime analysis yielding linear time for Partially PQ-Constrained Planarity in Theorem 4.2 works analogously.

Assume now that all blocks containing v consist of a single edge except for up to two blocks B_1 and B_2. A block consisting of a single edge is identified with this edge and called a *bridge*. It is clear that each bridge can be attached arbitrarily to an embedding of $B_1 + B_2$. Hence, we can modify the extension of the PQ-embedding representation defined earlier by introducing a single P-node containing all edges incident to v as parent of the combined embedding tree. This tree then represents exactly the possible orderings of edges around v in any planar embedding. The analysis presented previously works analogously, yielding the following two theorems.

THEOREM 4.8. PARTIALLY PQ-CONSTRAINED PLANARITY *can be solved in linear time if each vertex is contained in up to two blocks not consisting of a single edge.*

THEOREM 4.9. SEFE *can be solved in quadratic time if, in both graphs, every vertex is contained in at most two blocks not consisting of a single edge, and the common graph is connected.*

Note that this special case always applies if the cutvertices have degree at most 5. In particular, for SEFE, we obtain the following corollary.

COROLLARY 4.10. SEFE *can be solved in quadratic time for maxdeg-5 graphs whose intersection is connected.*

5. CONCLUSION

In this work, we introduced a new problem called SIMULTANEOUS PQ-ORDERING. Its input consists of 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 of representing 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 showed that certain "simple" instances (the 1-critical instances) can be solved in polynomial time. To achieve this result, we showed that satisfying the Q-constraints and the critical triples is sufficient to extend orders of several children simultaneously to a parent if each P-node is contained in at most one critical triple. Inserting additional PQ-trees—the expansion trees—enforces that the critical triples are satisfied when choosing orders bottom-up. 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. This allows us to solve a 1-critical instance of SIMULTANEOUS PQ-ORDERING in polynomial time, essentially by choosing orders bottom-up in the expansion graph. We showed how this framework can be applied to solve PARTIALLY PQ-CONSTRAINED PLANARITY for biconnected graphs and SEFE for biconnected graphs with a connected intersection in polynomial time (linear and quadratic, respectively), which were both not known to be efficiently solvable earlier. Furthermore, we showed how to solves SIMULTANEOUS INTERVAL REPRESENTATION and PARTIAL INTERVAL GRAPH EXTENSION in linear time, which improves over the best known algorithms with running times $\mathcal{O}(n^2 \log n)$ and $\mathcal{O}(n^2)$ algorithm, respectively. We stress that all these results can be obtained in a straightforward way from the main result of this work, the algorithm for SIMULTANEOUS PQ-ORDERING for 2-fixed instances.

Open problems. Several questions remain open for the applications as well as for problems related to SIMULTANEOUS PQ-ORDERING. Since the set of possible orders of edges around a cutvertex in a planar drawing is not necessarily PQ-representable, our solutions for PARTIALLY PQ-CONSTRAINED PLANARITY and SEFE cannot handle graphs containing cutvertices, except for the special cases discussed in Section 4.6. We believe that understanding edge orderings around cutvertices can lead to substantial progress for the SEFE problem. Thus, solving the simpler problem PARTIALLY PQ-CONSTRAINED PLANARITY for graphs containing cutvertices seems to be worthwhile.

An obvious open problem concerning SIMULTANEOUS PQ-ORDERING is whether our result can be extended to instances that are not 1-critical or generalize it in the sense that structures different from PQ-trees are used as nodes in the DAG. Questions forming the basis of such an approach could be of the following kind. Given three PQ-trees having some leaves in common, can we find an order for each of the trees such that the three resulting orders can be extended to a common order? Note that testing this for three fixed orders can be done efficiently. Does it make the problem easier if we consider rooted PQ-trees representing linear orders? Can we find structures similar to PQ-trees that represent orders of edges around cutvertices?

REFERENCES

Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. 2010. Testing planarity of partially embedded graphs. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*. Society for Industrial and Applied Mathematics, 202–221.

Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. 2012. Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph. *Journal of Discrete Algorithms* 14 (2012), 150–172.

Bengt Aspvall, Michael F. Plass, and Robert E. Tarjan. 1979. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters* 8, 3 (1979), 121–123.

Thomas Bläsius, Annette Karrer, and Ignaz Rutter. 2014. Simultaneous embedding: Edge orderings, relative positions, cutvertices. In *Proceedings of the 21st International Symposium on Graph Drawing (GD'13) (Lecture Notes in Computer Science)*, Vol. 8242. Springer, Berlin, 220–231.

Thomas Bläsius, Stephen G. Kobourov, and Ignaz Rutter. 2013. *Handbook of Graph Drawing and Visualization*. Chapman and Hall/CRC, Chapter Simultaneous Embedding of Planar Graphs, 349–381.

Thomas Bläsius and Ignaz Rutter. 2015. Disconnectivity and relative positions in simultaneous embeddings. *Computational Geometry: Theory and Applications* 48, 6 (2015), 459–478.

Kellogg S. Booth. 1975. *PQ-Tree Algorithms*. Ph.D. Dissertation. University of California, Berkeley.

Kellogg S. Booth and George S. Lueker. 1976. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-Tree algorithms. *Journal of Computer System Science* 13 (1976), 335–379. Issue 3.

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

Giuseppe Di Battista and Roberto Tamassia. 1996b. On-line planarity testing. *SIAM Journal on Computing* 25, 5 (1996), 956–997.

Cesim Erten and Stephen G. Kobourov. 2005. Simultaneous embedding of planar graphs with few bends. In *Proceedings of the 12th International Symposium on Graph Drawing (GD'04) (Lecture Notes in Computer Science)*, Vol. 3383. Springer, Berlin, 195–205.

Shimon Even, Alon Itai, and Adi Shamir. 1976. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing* 5, 4 (1976), 691–703.

J. Joseph Fowler, Carsten Gutwenger, Michael Jünger, Petra Mutzel, and Michael Schulz. 2009. An SPQR-tree approach to decide special cases of simultaneous embedding with fixed edges. In *Proceedings of the 16th International Symposium on Graph Drawing (GD'08) (Lecture Notes in Computer Science)*, Vol. 5417. Springer, Berlin, 157–168.

Delbert R. Fulkerson and Oliver A. Gross. 1965. Incidence matrices and interval graphs. *Pacific Journal of Mathematics* 15, 3 (1965), 835–855.

Zvi Galil and Nimrod Megiddo. 1977. Cyclic ordering is NP-complete. *Theoretical Computer Science* 5, 2 (1977), 179–182.

Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz. 2006. Simultaneous graph embeddings with fixed edges. In *Proceedings of the 32nd Workshop on Graph-Theoretic Concepts in Computer Science (WG'06) (Lecture Notes in Computer Science)*, Vol. 4271. Springer, Berlin, 325–335.

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

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

Bernhard Haeupler, Krishnam Raju Jampani, and Anna Lubiw. 2010. Testing simultaneous planarity when the common graph is 2-connected. In *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC'10) (Lecture Notes in Computer Science)*, Vol. 6507. Springer, Berlin, 410–421.

Bernhard Haeupler and Robert E. Tarjan. 2008. Planarity algorithms via PQ-Trees (Extended abstract). *Electronic Notes in Discrete Mathematics* 31 (2008), 143–149.

Wen-Lian Hsu. 2001. PC-trees vs. PQ-trees. In *Proceedings of the 7th Annual International Conference on Computing and Combinatorics (COCOON'01) (Lecture Notes in Computer Science)*. Springer, Berlin, 207–217.

Wen-Lian Hsu and Ross M. McConnell. 2001. PQ-Trees, PC-Trees, and Planar Graphs. Retrieved from www.cs.colostate.edu/ rmm/pc2.pdf.

Wen-Lian Hsu and Ross M. McConnell. 2003. PC-trees and circular-ones arrangements. *Theoretical Computer Science* 296, 1, 99–116.

Krishnam Raju Jampani and Anna Lubiw. 2009. The simultaneous representation problem for chordal, comparability and permutation graphs. In *Proceedings of the 11th International Symposium on Algorithms and Data Structures (WADS'09) (Lecture Notes in Computer Science)*, Vol. 5664. Springer, Berlin, 387–398.

Krishnam Raju Jampani and Anna Lubiw. 2010. Simultaneous interval graphs. In *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC'10) (Lecture Notes in Computer Science)*, Vol. 6506. Springer, Berlin, 206–217.

Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. 2011. A Kuratowski-type theorem for planarity of partially embedded graphs. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG'11)*. ACM, 107–116.

Michael Jünger and Michael Schulz. 2009. Intersection graphs in simultaneous embedding with fixed edges. *Journal of Graph Algorithms and Applications* 13, 2 (2009), 205–218.

Michael Kaufmann and Roland Wiese. 2002. Embedding vertices at points: Few bends suffice for planar graphs. *Journal of Graph Algorithms and Applications* 6, 1 (2002), 115–129.

Pavel Klavík, Jan Kratochvíl, and Tomáš Vyskočil. 2011. Extending partial representations of interval graphs. In *Proceedings of the 8th Annual Conference on Theory and Applications of Models of Computation (TAMC'11) (Lecture Notes in Computer Science)*, Vol. 6648. Springer, Berlin, 276–285.

M. R. Krom. 1967. 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, 1–2 (1967), 15–20.

János Pach and Rephael Wenger. 1998. Embedding planar graphs at fixed vertex locations. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98) (Lecture Notes in Computer Science)*, Vol. 1547. Springer, Berlin, 263–274.

Marcus Schaefer. 2013. Toward a theory of planarity: Hanani-Tutte and planarity variants. In *Proceedings of the 20th International Symposium on Graph Drawing (GD'12) (Lecture Notes in Computer Science)*, Vol. 7704. Springer, Berlin, 162–173.

Alan Tucker. 1971. Matrix characterizations of circular-arc graphs. *Pacific Journal of Mathematics* 39, 2 (1971), 535–545.

Received August 2013; accepted February 2015