Mondshein Sequences (a.k.a. (2, 1)-Orders)

Jens M. Schmidt
Institute of Mathematics
TU Ilmenau*

Abstract

Canonical orderings [STOC'88, FOCS'92] have been used as a key tool in graph drawing, graph encoding and visibility representations for the last decades. We study a far-reaching generalization of canonical orderings to non-planar graphs that was published by Lee Mondshein in a PhD-thesis at M.I.T. as early as 1971.

Mondshein proposed to order the vertices of a graph in a sequence such that, for any i, the vertices from 1 to i induce essentially a 2-connected graph while the remaining vertices from $i+1$ to n induce a connected graph. Mondshein's sequence generalizes canonical orderings and became later and independently known under the name *non-separating ear decomposition*. Surprisingly, this fundamental link between canonical orderings and non-separating ear decomposition has not been established before. Currently, the fastest known algorithm for computing a Mondshein sequence achieves a running time of $O(nm)$; the main open problem in Mondshein's and follow-up work is to improve this running time to subquadratic time.

After putting Mondshein's work into context, we present an algorithm that computes a Mondshein sequence in optimal time and space $O(m)$. This improves the previous best running time by a factor of n. We illustrate the impact of this result by deducing linear-time algorithms for five other problems, for four out of which the previous best running times have been quadratic. In particular, we show how to

- compute three independent spanning trees in a 3-connected graph in time $O(m)$, improving a result of Cheriyan and Maheshwari [J. Algorithms 9(4)],
- improve the preprocessing time from $O(n^2)$ to $O(m)$ for the output-sensitive data structure by Di Battista, Tamassia and Vismara [Algorithmica 23(4)] that reports three internally disjoint paths between any given vertex pair,
- derive a very simple $O(n)$-time planarity test once a Mondshein sequence has been computed,
- compute a nested family of contractible subgraphs of 3-connected graphs in time $O(m)$,
- compute a 3-partition in time $O(m)$, while the previous best running time is $O(n^2)$ due to Suzuki et al. [IPSJ 31(5)].

1 Introduction

Canonical orderings are a fundamental tool used in graph drawing, graph encoding and visibility representations; we refer to [2] for a wealth of applications. For maximal planar graphs, canonical orderings were introduced by de Fraysseix, Pach and Pollack [9, 10] in 1988. Kant then generalized canonical orderings to 3-connected planar graphs [23, 24]. In polyhedral combinatorics, canonical

*This research was partly done at Max Planck Institute for Informatics, Saarbrücken. An extended abstract of this paper has been published at ICALP'14.

orders are in addition related to shellings of (dual) convex 3-dimensional polytopes [42]; however, such shellings are often, as in the Bruggesser-Mani theorem, dependent on the geometry of the polytope. A combinatorial generalization to arbitrary planar graphs was given by Chiang, Lin and Lu [7].

Surprisingly, the concept of canonical orderings can be traced back much further, namely to a long-forgotten PhD-thesis at M.I.T. by Lee F. Mondshein [29] in 1971. In fact, Mondshein proposed a sequence that generalizes canonical orderings to non-planar graphs, hence making them applicable to arbitrary 3-connected graphs. *Mondshein's sequence* was, independently and in a different notation, found later by Cheriyan and Maheshwari [6] under the name *non-separating ear decompositions* and is sometimes also called *(2,1)-order* (e.g., see [5]). In addition, Mondshein sequences provide a generalization of Schnyder's famous woods to non-planar 3-connected graphs. One key contribution of this paper is to establish the above fundamental link between canonical orderings and non-separating ear decompositions in detail.

Computationally, it is an intriguing question how fast a Mondshein sequence can be computed. Mondshein himself gave an involved algorithm with running time $O(m^2)$. Cheriyan showed that it is possible to achieve a running time of $O(nm)$ by using a theorem of Tutte that proves the existence of non-separating cycles in 3-connected graphs [36]. Both works state as main open problem, whether it is possible to compute a Mondshein sequence in subquadratic time (see [29, p. 1.2] and [6, p. 532]).

We present the first algorithm that computes a Mondshein sequence in optimal time and space $O(m)$, hence solving the above 45-year-old problem. The interest in such a computational result stems from the fact that 3-connected graphs play a crucial role in algorithmic graph theory. We illustrate this in five applications by giving linear-time algorithms. For four of them, the previous best running times have been quadratic.

We start by giving an overview of Mondshein's work and its connection to canonical orderings and non-separating ear decompositions in Section 3. Section 4 explains the linear-time algorithm and proves its main technical lemma, the Path Replacement Lemma. Section 5 covers five applications of our linear-time algorithm.

2 Preliminaries

We use standard graph-theoretic terminology and assume that all graphs are simple.

Definition 1 ([26, 40])**.** An *ear decomposition* of a graph $G = (V, E)$ is a sequence $(P_0, P_1, \ldots, P_k)$ of subgraphs of G that partition E such that P_0 is a cycle and every P_i, $1 \leq i \leq k$, is a path that intersects $P_0 \cup \cdots \cup P_{i-1}$ in exactly its endpoints. Each P_i is called an *ear*. An ear is *short* if it is an edge and *long* otherwise.

According to Whitney [40], every ear decomposition has exactly $m-n+1$ ears and G has an ear decomposition if and only if G is 2-connected. For any i, let $G_i := P_0 \cup \cdots \cup P_i$ and $\overline{V_i} := V - V(G_i)$. We write $\overline{G_i}$ to denote the graph induced by $\overline{V_i}$. Note that $\overline{G_i}$ does not necessarily contain all edges in $E - E(G_i)$; in particular, there may be short ears in $E - E(G_i)$ that have both endpoints in G_i.

For a path P and two vertices x and y in P, let $P[x, y]$ be the subpath in P from x to y. A path with endpoints v and w is called a *vw-path*. A vertex x in a vw-path P is an *inner vertex* of P if $x \notin \{v, w\}$. For convenience, every vertex in a cycle is called an *inner vertex* of that cycle.

For an ear P, let $inner(P)$ be the set of its inner vertices. The inner vertex sets of the ears in an ear decomposition of G play a special role, as they partition V. Every vertex of G is contained in exactly one long ear as inner vertex. This readily gives the following characterization of $\overline{V_i}$.

Observation 2. *For every i, $\overline{V_i}$ is the union of the inner vertices of all long ears P_j with $j > i$.*

We will compare vertices and edges of G by their first occurrence in a fixed ear decomposition.

Definition 3. Let $D = (P_0, P_1, \ldots, P_{m-n})$ be an ear decomposition of G. For an edge $e \in G$, let $birth_D(e)$ be the index i such that P_i contains e. For a vertex $v \in G$, let $birth_D(v)$ be the minimal i such that P_i contains v (thus, $P_{birth_D(v)}$ is the ear containing v as an inner vertex). Whenever D is clear from the context, we will omit D.

Clearly, for every vertex v, the ear $P_{birth(v)}$ is long, as it contains v as an inner vertex.

3 Generalizing Canonical Orderings

Although canonical orderings of (maximal or 3-connected) planar graphs are traditionally defined as vertex partitions, we will define them as special ear decompositions. This will allow for an easy comparison of canonical orderings to the more general Mondshein sequences, which extend them to non-planar graphs. We assume that the input graphs are 3-connected and, when talking about canonical orderings, planar. It is well-known that maximal planar graphs (which were considered in [9] in this setting) form a subclass of 3-connected graphs, apart from the triangle-graph.

Definition 4. An ear decomposition is *non-separating* if, for every long ear P_i except the last one, every inner vertex of P_i has a neighbor in $\overline{G_i}$.

The name *non-separating* refers to the following helpful property.

Lemma 5. *In a non-separating ear decomposition D, $\overline{G_i}$ is connected for every i.*

Proof. For all i satisfying $\overline{G_i} = \emptyset$ the claim is true, in particular if i is at least the index of the last long ear. Otherwise, i is such that the inner vertex set A of the last long ear in D is contained in $\overline{G_i}$. Consider any vertex x in $\overline{G_i}$. In order to show connectedness, we exhibit a path from x to A in $\overline{G_i}$. If $x \in A$, we just take the path of length zero. Otherwise, the vertex x has a neighbor in $\overline{G_{birth(x)}}$, since D is non-separating. According to Observation 2, this neighbor is an inner vertex of some ear P_j with $j > birth(x)$. Applying induction on j gives the desired path to A. □

A *plane graph* is a graph that is embedded into the plane. In particular, a plane graph has a fixed outer face. We define canonical orderings as follows.

Definition 6 (canonical ordering)**.** Let G be a 3-connected plane graph and let rt and ru be edges of its outer face. A *canonical ordering* through rt and avoiding u is an ear decomposition D of G such that

1. $rt \in P_0$,
2. $P_{birth(u)}$ is the last long ear, contains u as its only inner vertex and does not contain ru, and
3. D is non-separating.

The fact that D is non-separating plays a key role for both canonical orderings and their generalization to non-planar graphs. E.g., Lemma 5 implies that the plane graph G can be constructed from P_0 by successively inserting the ears of D into only one dedicated face of the current embedding, a routine that is heavily applied in graph drawing and embedding problems. Put simply, the second condition forces u to be "added last" in D. Further motivations are given by 3-connectivity: If we would not restrict u to be the only vertex in $P_{birth(u)}$, other vertices in the same ear could have degree two, as the non-separateness does not imply any later neighbors for the last ear.

The condition $ru \notin P_{birth(u)}$ ensures that u has degree at least three in G (which is necessary for 3-connectivity) and will also lead to the existence of a third independent spanning tree (see Application 1 in Section 5).

We note that forcing one edge rt in P_0 is optimal in the sense that two edges rz and rt cannot be forced: Let W be a sufficiently large wheel graph with center vertex r and rim vertices t and z such that t and z are not adjacent. Then a canonical ordering with $rt, rz \in P_0$ and avoiding u does not exist, as any inner vertex on the rim-path from t to z not containing u has no larger neighbor with respect to *birth*, and thus violates the non-separateness.

The original definition of canonical orderings by Kant [24] states the following additional properties.

Lemma 7 (further properties)**.** *For every $0 \leq i \leq m-n$ in a canonical ordering,*

4. the outer face C_i of the plane subgraph $G_i \subseteq G$ is a (simple) cycle that contains rt,

5. G_i is 2-connected and every separation pair of G_i has both its vertices in C_i, and

6. for $i > 0$, the neighbors of $inner(P_i)$ in G_{i-1} are contained consecutively in C_{i-1}.

Further, the canonical ordering implies the existence of one satisfying the following property:

7. if $|inner(P_i)| \geq 2$, each inner vertex of P_i has degree two in $G - \overline{V_i}$

Properties 4–6 can be easily deduced from Definition 6 as follows: Every G_i is a 2-connected plane subgraph of G, as G_i has an ear decomposition. According to [34, Corollary 1.3], all faces of a 2-connected plane graph form cycles. Thus, every C_i is a cycle and Property 4 follows directly from the fact that rt is assumed to be in the fixed outer face of G. Property 5 is implied by the 3-connectivity of G and Property 4. Property 6 follows from Property 4, the fact that every inner vertex of P_i must be outside C_{i-1} (in G) and the Jordan Curve Theorem.

For the sake of completeness, we show how Property 7 is derived. Although it is not directly implied by Definition 6 (in that sense our definition is more general), the following lemma shows that we can always find a canonical ordering satisfying it.

Lemma 8. *Every canonical ordering can be transformed to a canonical ordering satisfying Property 7.7 in linear time.*

Proof. First, consider any ear $P_i \neq P_0$ with $|inner(P_i)| \geq 2$ such that an inner vertex x of P_i has a neighbor y in $G - \overline{V_i}$ that is different from its predecessor and successor in P_i. Then $P_{birth(xy)} = xy$ and $birth(xy) > i$. If y is in P_i, let Z be the path obtained from P_i by replacing $P_i[x,y] \subseteq P_i$ with xy; we call this latter operation *short-cutting*. We replace P_i with the two ears Z and $P_i[x,y]$ in that order and delete $P_{birth(xy)} = xy$. This preserves Properties 1–3 (note that $u \notin P_i$, as $|inner(P_i)| \geq 2$) and therefore the canonical ordering. If y is not in P_i, let Z_1 be a shortest path in P_i from an endpoint of P_i to x and let Z_2 be the path in P_i from x to the remaining endpoint. Replace P_i with the two ears $Z_1 \cup xy$ and Z_2 in that order and delete $P_{birth(xy)}$. This preserves Properties 1–3.

Now, consider a vertex $x \in P_0$ not having degree 2 in $G - \overline{V_0}$, i.e. x has a non-consecutive neighbor y in P_0 in the graph that is vertex-induced by $V(P_0)$. If $x \in \{r,t\}$, we replace P_0 with the shortest cycle C in $P_0 \cup xy$ that contains r, t and y, delete $P_{birth(xy)} = xy$ and add the remaining path from x to y in $P_0 - E(C)$ as new ear directly after C. This clearly preserves Properties 1–3. If $x \notin \{r,t\}$, we can shortcut P_0 in a similar way. The above operations can be computed in linear total time. ☐

Our definition of canonical orderings uses planarity only in one place: $tr \cup ru$ is assumed to be part of the outer face of G. Note that the essential part of this assumption is that $tr \cup ru$ is part of *some* face of G, as we can always choose an embedding for G having this face as outer face. Hence, there is a natural generalization of canonical orderings to non-planar graphs G: We merely require rt and ru to be edges of G! The following ear-based definition is similar to the one given in [6] but does not need additional degree-constraints.

Definition 9 ([29, 6])**.** Let G be a graph with edges rt and ru. A *Mondshein sequence through* rt and *avoiding* u (see Figure 1) is an ear decomposition D of G such that

1. $rt \in P_0$,
2. $P_{birth(u)}$ is the last long ear, contains u as its only inner vertex and does not contain ru, and
3. D is non-separating.

This definition is in fact equivalent to the one Mondshein used 1971 to define a *(2,1)-sequence* [29, Def. 2.2.1], but which he gave in the notation of a special vertex ordering. This vertex ordering actually refines the partial order $inner(P_0), \dots, inner(P_{m-n})$ by enforcing an order on the inner vertices of each path according to their occurrence on that path (in any direction). The statement that canonical orderings can be extended to non-planar graphs can also be found in [14, p.113], however, no further explanation is given.

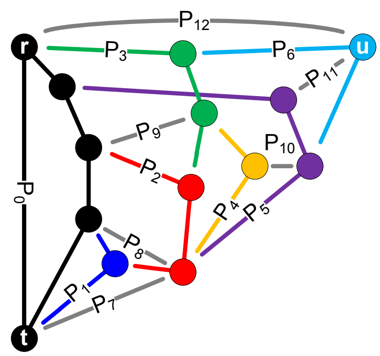


Figure 1: A Mondshein sequence of a non-planar 3-connected graph.

Note that Definition 9 implies $u \notin P_0$, as $P_0 \neq P_{birth(u)}$, since $P_{birth(u)}$ contains only one inner vertex. As a direct consequence of this and the fact that D is non-separating, G must have minimum degree at least 3 in order to have a Mondshein sequence. Mondshein proved that every 3-connected graph has a Mondshein sequence. In fact, also the converse is true.

Theorem 10 (compare also [6, 41])**.** *Let rt and ru be edges of G. Then G is 3-connected if and only if G has three internally vertex-disjoint paths between t and u and a Mondshein sequence through rt and avoiding u.*

We state two additional facts about Mondshein sequences. For the first, let G be planar. Clearly, every canonical ordering of an embedding of G is also a Mondshein sequence. Conversely, let D be a Mondshein sequence of G through rt and avoiding u. Then Theorem 10 implies that G is 3-connected. If G has an embedding in which $tr \cup ru$ is contained in a face, we can choose this face as outer face and get an embedding of G for which D is a canonical ordering. This embedding must be unique, as Whitney proved that any 3-connected planar graph has a unique embedding (up to flipping) [39]. Otherwise, there is no embedding of G such that $tr \cup ru$ is contained in some face. Since the faces of a 3-connected planar graph are precisely its non-separating cycles [36], we conclude the following observation.

Observation 11. *For a planar graph G and edges tr and ru, the following statements are equivalent:*

- *There is a planar embedding of G whose outer face contains $tr \cup ru$, and D is a canonical ordering of this (unique) embedding through rt and avoiding u.*
- *D is a Mondshein sequence through rt and avoiding u, and $tr \cup ru$ is contained in a non-separating cycle of G.*

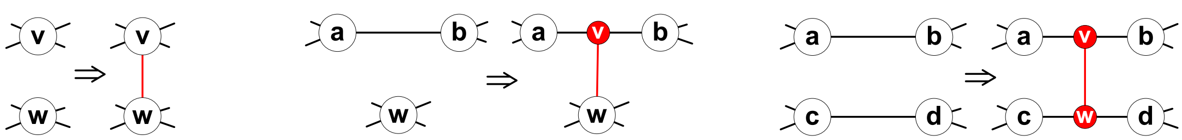
For the second fact, let a *chord* of an ear P_i be an edge in G that joins two non-adjacent vertices of P_i. Note that the definition of a Mondshein sequence allows chords for every P_i. Once having a Mondshein sequence, one can aim for a slightly stronger structure. Let a Mondshein sequence be *induced* if P_0 is induced in G and every ear $P_i \neq P_0$ has no chord, except possibly the one joining the endpoints of P_i. It has been shown [6] that every Mondshein sequence can be made induced. The following lemma shows the somewhat stronger statement that we can always expect Mondshein sequences to satisfy Property 7.7. In fact, its proof is precisely the same as the one for Lemma 8, since none of its arguments uses planarity.

Lemma 12. *Every Mondshein sequence can be transformed into a Mondshein sequence D satisfying Property 7.7 in linear time. In particular, D is induced.*

4 Computing a Mondshein Sequence

Mondshein gave an involved algorithm [29] that computes his sequence in time $O(m^2)$. Independently, Cheriyan and Maheshwari gave an algorithm that runs in time $O(nm)$ and which is based on a theorem of Tutte. At the heart of our linear-time algorithm is the following classical construction sequence for 3-connected graphs due to Barnette and Grünbaum [3] and Tutte [37, Thms. 12.64 and 12.65].

Definition 13. The following operations on simple graphs are *BG-operations* (see Figure 2).
(a) *vertex-vertex-addition*: Add an edge between two distinct non-adjacent vertices
(b) *edge-vertex-addition*: Subdivide an edge ab, $a \neq b$, with a vertex v and add the edge vw for a vertex $w \notin \{a, b\}$
(c) *edge-edge-addition*: Subdivide two distinct edges (the edges may intersect in one vertex) with vertices v and w, respectively, and add the edge vw



(a) vertex-vertex-addition (b) edge-vertex-addition (c) edge-edge-addition

Figure 2: BG-operations

Theorem 14 ([3, 37])**.** *A graph is 3-connected if and only if it can be constructed from K_4 using BG-operations.*

Hence, applying a BG-operation on a 3-connected graph preserves it to be simple and 3-connected. Let a *BG-sequence* of a 3-connected graph G be a sequence of BG-operations that constructs G from K_4. It has been shown that such a BG-sequence can be computed efficiently.

Theorem 15 ([31, Thms. 6.(2) and 52])**.** *A BG-sequence of a 3-connected graph can be computed in time $O(m)$.*

The outline of our algorithm is as follows. Assume we want a Mondshein sequence of G through $r\overline{t}$ and avoiding $\overline{u}$. We will first compute a suitable BG-sequence of G using Theorem 15 and start with a Mondshein sequence of its first graph, the K_4. The crucial part is then a careful analysis that a Mondshein sequence of a 3-connected graph can be modified to one of G', where G' is obtained from the former by applying a BG-operation.

In more detail, we need a special BG-sequence to harness the dynamics of the vertices r, $\overline{t}$ and $\overline{u}$ throughout the BG-sequence. A BG-sequence is determined by an (arbitrary) DFS-tree and two fixed incident edges of its root. We choose a DFS-tree with root r and fix the edges $r\overline{t}$ and $r\overline{u}$. This way the initial K_4 will contain the vertex r and r will never be relabeled [30, Section 5].

However, $\overline{t}$ and $\overline{u}$ are not necessarily vertices of the K_4. This is a problem, as we have to specify an edge rt and vertex u of K_4 which the Mondshein sequence of K_4 goes through and avoids, respectively, for induction purposes. Fortunately, the relation between the graphs in a BG-sequence and subdivisions of these graphs in G [30, Section 4] gives us such replacement vertices for $\overline{t}$ and $\overline{u}$ efficiently: We find vertices t and u of the initial K_4 such that the following labeling process ends with the input graph G in which $t = \overline{t}$ and $u = \overline{u}$: For every BG-operation of the BG-sequence from K_4 to G that subdivides the edge rt or ru, we label the subdividing vertex with t or u, respectively (the old vertex t or u is then given a different label). As desired, the final t and u upon completion of the BG-sequence will be $\overline{t}$ and $\overline{u}$. We refer to [30, Section 4] for details on how to efficiently compute such a labeling scheme.

For the K_4, it is easy to compute a Mondshein sequence through rt and avoiding u efficiently. We iteratively proceed to a Mondshein sequence of the next graph in the sequence. The following modifications and their computational analysis are the main technical contribution of this paper and depend on the various positions in the sequence in which the vertices and edges that are involved in the BG-operation can occur.

Note that any short ear xy in a Mondshein sequence can be moved to an arbitrary position of the sequence without destroying the Mondshein property, as long as both x and y are created at an earlier position. Thus, the essential information of a Mondshein sequence is its order on long ears. We will prove that there is always a modification that is local in the sense that the only long ears that are modified are the ones containing a vertex that is involved in the BG-operation.

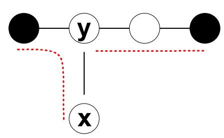
Lemma 16 (Path Replacement Lemma)**.** *Let G be a 3-connected graph with edges rt and ru and let $D = (P_0, P_1, \ldots, P_{m-n})$ be a Mondshein sequence of G through rt and avoiding u. Let G' be obtained from G by applying a BG-operation Γ and let rt' and ru' be the edges of G' that correspond to rt and ru in G. Then a Mondshein sequence D' of G' through rt' and avoiding u' can be computed from D using only constantly many (amortized) constant-time modifications.*

We split the proof into three parts. First, we state two preprocessing routines $leg()$ and $belly()$ on D that will reduce the number of subsequent cases considerably. Second, we show how to modify D to D' using these routines and, third, we discuss computational issues.

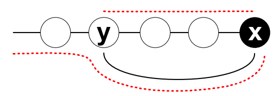
From now on, let vw be the edge that was added by Γ such that v subdivides $ab \in E(G)$ and w subdivides $cd \in E(G)$ (if applicable). Thus, the vertex u' in G' is either u, v or w, and likewise t' in G' is either t, v or w. By symmetry, we assume w.l.o.g. that $birth(a) \leq birth(b)$, $birth(c) \leq birth(d)$ and $birth(d) \leq birth(b)$. Recall that $\{a, b\}$ may intersect $\{c, d\}$ in at most one vertex. If not stated otherwise, the $birth$-operator refers always to D in this section.

We need some notation for describing the modifications. Suppose P_i is an ear containing an inner vertex z. If an orientation of P_i is given, let $P_i[, z]$ be the prefix of P_i ending at z in this orientation and let $P_i[z,]$ be the suffix of P_i starting at z. Occasionally, the orientation does not matter; if none is given, an arbitrary orientation can be taken. For paths A and B that end and start at a unique common vertex, let $A + B$ be the *concatenation* of A and B. Similarly, for disjoint paths A and B such that exactly one endpoint x of A is a neighbor of exactly one endpoint y of B, let $A + B$ be the path $A \cup xy \cup B$.

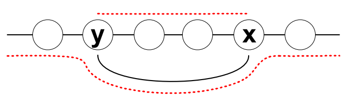
Of legs and bellies: We describe two preprocessing routines. These will be used on D in the next section to ensure that $ab \in P_{birth(b)}$ and $cd \in P_{birth(d)}$ (up to some special cases). Let an edge $xy \notin P_{birth(y)}$ be a *leg* of $P_{birth(y)}$ if $xy \neq ru$ and $birth(x) < birth(y)$. For each such leg, $P_{birth(y)}$ is a long ear, xy is a short ear, and x is either not contained in $P_{birth(y)}$ or an endpoint of $P_{birth(y)}$ (see Figures 3a and 3b). In the first case, if y is not the only inner vertex of $P_{birth(y)}$, orient $P_{birth(y)}$ such that the successor of y is also an inner vertex of $P_{birth(y)}$; this will preserve the non-separateness at y for some later cases. In the latter case, orient $P_{birth(y)}$ toward x.



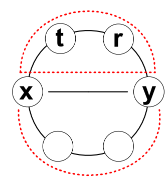
(a) A leg xy with $x \notin P_{birth(y)}$ and the result of Operation $leg(x, y)$ (dashed lines).



(b) A leg xy with $x \in P_{birth(y)}$ and the result of Operation $leg(x, y)$.



(c) A belly xy with $birth(y) > 0$ and the result of Operation $belly(x, y)$.



(d) A belly xy with $birth(y) = 0$ and the result of Operation $belly(x, y)$.

Figure 3

A leg xy of $P_{birth(y)}$ has the feature that it may be incorporated into $P_{birth(y)}$ such that the resulting sequence is still a Mondshein sequence: Let $leg(x, y)$ be the operation that deletes the short ear xy in the sequence D and replaces the long ear $P_{birth(y)}$ by the two ears $P_{birth(y)}[, y] + x$ and $P_{birth(y)}[y,]$ in that order. We prove that the resulting sequence $\overline{D}$ is a Mondshein sequence. Clearly, $\overline{D}$ is an ear decomposition. In addition, we still have $rt \in P_0$, as P_0 did not change due to $birth(y) > birth(x) \geq 0$. Since every inner vertex of the two new ears is also an inner vertex of $P_{birth(y)}$, it has a neighbor in some larger ear (with respect to $birth$) in $\overline{D}$; thus $\overline{D}$ is non-separating by Definition 4. Since $xy \neq ru$, the last long ear in $\overline{D}$ does not contain ru. The last long ear in $\overline{D}$ may be different from the one in D if $y = u$, but since the replacement does not introduce any new inner vertex, it will still contain the same vertex u as only inner vertex. Hence, $\overline{D}$ is a Mondshein sequence through rt and avoiding u by Definition 9.

Let an edge xy of G be a *belly* of $P_{birth(y)}$ if $birth(x) = birth(y) \neq birth(xy)$. Then $P_{birth(y)}$ contains both x and y as inner vertices, but does not contain xy; hence xy is a short ear (see Figures 3c and 3d).

For a belly xy, we can again find a Mondshein sequence that ensures $xy \in P_{birth(y)}$. First, consider the case $birth(y) > 0$, in which we orient $P_{birth(y)}$ from y to x. For this case, let $belly(x, y)$ be the operation that deletes the short ear xy in the sequence D and replaces the long ear $P_{birth(y)}$ by the two long ears $P_{birth(y)}[, y] + P_{birth(y)}[x,]$ and $P_{birth(y)}[y, x]$ in that order (see Figure 3c). For the same reasons as before, the resulting sequence $\overline{D}$ is an ear decomposition and non-separating.

Since $P_{birth(y)}$ contains two inner vertices, we have $birth(y) \neq birth(u)$, and it follows that the last long ear in $\overline{D}$ is exactly the last long ear of D. In addition, $rt \in P_0$, as P_0 did not change due to $birth(y) > birth(x) \geq 0$. Hence, $\overline{D}$ is a Mondshein sequence through rt and avoiding u.

Now consider the case $birth(y) = 0$. The vertices x and y cut P_0 into two distinct paths A and B having endpoints x and y; let A be the one containing rt. Let $belly(x, y)$ be the operation that deletes the short ear xy in D and replaces P_0 by the two long ears $A \cup xy$ and B in that order (see Figure 3d). This preserves P_0 to be a cycle that contains rt and, thus, gives also a Mondshein sequence through rt and avoiding u. Note that both operations $leg()$ and $belly()$ leave the vertices u, r and t unchanged.

Modifying D to D': We use the operations $leg()$ and $belly()$ for a preprocessing on the subdivided edges ab and cd (if applicable) by Γ. Suppose first that $ru \notin \{ab, cd\}$; we will solve the remaining case $ru \in \{ab, cd\}$ later. Assume $birth(ab) \neq birth(b)$ and recall that $birth(a) \leq birth(b)$. If $birth(a) < birth(b)$, ab is a leg of $P_{birth(b)}$ and we apply the operation $leg(a, b)$. Otherwise, $birth(a) = birth(b)$ and we apply the operation $belly(a, b)$. In both cases, this leaves a Mondshein sequence in which $birth(ab) = birth(b)$, i.e. ab is contained in the long chain $P_{birth(b)}$.

Similarly, if $birth(cd) \neq birth(d)$, we want to apply either $leg(c, d)$ or $belly(c, d)$ to obtain $birth(cd) = birth(d)$. However, doing this without any restrictions may result in loosing $birth(ab) = birth(b)$, e.g. when cd is a belly of $P_{birth(b)}$. Thus, we apply $leg(c, d)$ or $belly(c, d)$ only if $birth(d) < birth(b)$, as then d is no inner vertex of $P_{birth(b)}$. Since $birth(d) \leq birth(b)$, we have therefore $birth(d) \in \{birth(b), birth(cd)\}$. Subdivide the edge ab in G and $P_{birth(ab)}$ with v and likewise subdivide cd with w if applicable for Γ. Call the resulting sequence D; D satisfies $birth(v) = birth(b)$ and $birth(d) \in \{birth(b), birth(w)\}$. We obtain the desired Mondshein sequence D' through rt' and avoiding u from D by distinguishing the following cases (see Figure 4).

(1) Γ *is a vertex-vertex-addition*
Obtain D' from D by adding the new short ear vw to the end of D. This way v and w exist when vw is born.

(2) Γ *is an edge-vertex-addition* $\triangleright$ $birth(v) = birth(b)$

(a) $birth(w) > birth(b)$ $\triangleright$ $w \notin G_{birth(b)}$
Obtain D' from D by adding the new ear vw to the end of D. Since $birth(w) > birth(b)$, v has a larger neighbor with respect to $birth$.

(b) $birth(w) < birth(b)$
Then $wv \neq ru'$, as otherwise we would have $w = r$ and $v = u'$ and thus $ab = ru$, which contradicts our assumption. Hence, wv is a leg of $P_{birth(v)}$. We apply $leg(w, v)$. By the orientation assigned to $P_{birth(v)}$, this ensures that v has a larger neighbor with respect to $birth$ (e.g., b).

(c) $birth(w) = birth(b)$
Then $wv \notin P_{birth(v)}$, since v is adjacent to only a and b in $P_{birth(v)}$ and $w \notin \{a, b\}$ for edge-vertex-additions. Thus, $birth(w) = birth(v) \neq birth(wv)$ and hence wv is a belly of $P_{birth(v)}$. We apply $belly(w, v)$. By the orientation assigned to $P_{birth(v)}$, this ensures that v has a larger neighbor.

(3) Γ *is an edge-edge-addition* $\triangleright$ $birth(v) = birth(b)$ and $birth(d) \in \{birth(b), birth(w)\}$

(a) $birth(d) < birth(b)$ $\triangleright$ $d \in G_{birth(b)-1}$ and $birth(b) > 0$
Then $birth(c) \leq birth(d) = birth(w) < birth(b) = birth(v)$. We further have $vw \neq ru'$, as otherwise we would have $w = r$ and $v = u'$ and thus $r \in \{a, b\}$ which contradicts $r = w$. Hence, wv is a leg of $P_{birth(b)}$. Obtain D' from D by applying $leg(w, v)$.

(b) $birth(d) = birth(b) = birth(w)$ $\triangleright\ d, w \in inner(P_{birth(b)})$
Then vw is a belly of $P_{birth(b)}$. Obtain D' from D by applying $belly(v, w)$.

(c) $birth(d) = birth(b) \neq birth(w)$ and $birth(c) = birth(b)$ $\triangleright\ c, d \in inner(P_{birth(b)}) \not\ni w$
Then $birth(w) > birth(b)$ and thus $P_{birth(w)} = cw \cup wd$. Let Z be a shortest path in $P_{birth(b)}$ that contains c, d and v, but not the edge rt' (the latter is only relevant for $birth(b) = 0$). Let z be the inner vertex of Z that is contained in $\{c, d, v\}$. At least one of the two paths $Z[;z]$ and $Z[z;]$, say $Z[z;]$, contains an inner vertex, as otherwise Γ would not be a BG-operation. Obtain D' from D by deleting $P_{birth(w)}$, replacing the path Z in $P_{birth(b)}$ with the two edges connecting w to the endpoints of Z, and adding the two new ears $Z[;z] + w$ and $Z[z;]$ directly afterward in that order. Clearly, $rt' \in P_0$ in D'.

(d) $birth(d) = birth(b) \neq birth(w)$ and $birth(c) \neq birth(b)$ $\triangleright\ d \in inner(P_{birth(b)}) \not\ni c, w$
Then $birth(c) < birth(d) < birth(w)$ and hence $birth(b) > 0$ and $P_{birth(w)} = cw \cup wd$. One of the paths $P_{birth(b)}[;v]$ and $P_{birth(b)}[v;]$, say $P_{birth(b)}[v;]$, contains d as an inner vertex. Obtain D' from D by replacing $P_{birth(b)}$ with the two ears $P_{birth(b)}[;v]+w+c$ and $P_{birth(b)}[v;]$ in that order and replacing $P_{birth(w)}$ with the short ear wd. If $birth(b) \neq birth(u)$, it follows directly that $u' = u$ and thus that D' avoids $u' = u$. Otherwise $birth(b) = birth(u)$, which implies $u = b = d$ and $c \neq r$, since we assumed $cd \neq ru$. Thus, in this case D' avoids $u' = u = b$ as well.

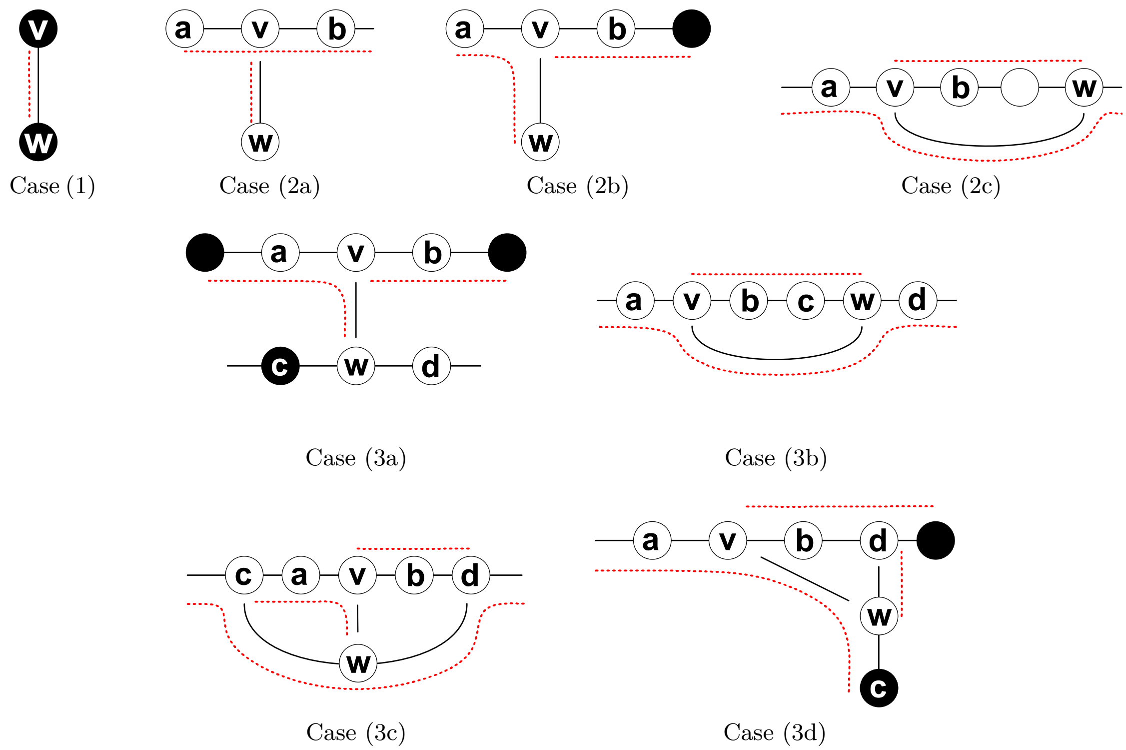


Figure 4: Cases when modifying D to D'. Black vertices are endpoints of ears that are contained in $G_{birth(b)}$. The dashed paths depict (parts of) the ears in D'.

In all these cases, we obtain a Mondshein sequence D' through rt' and avoiding u' as desired. Now consider the remaining case $ru \in \{ab, cd\}$. If $birth(d) = birth(b)$ (for an edge-edge-addition), we have $b = d = u$ and can w.l.o.g. assume $ru = ab$. Otherwise, $birth(d) < birth(b)$ and it follows directly that we have in all cases, even for edge-vertex-additions, $r = a$ and $u = b$. If cd is a short ear, we move cd to the position in D directly after $P_{birth(d)}$; this preserves a Mondshein sequence. As before, subdivide ab and cd with v and w.

Let Γ be an edge-vertex-addition. Then $u' = v$ and hence $birth(w) < birth(u) < birth(v)$. Obtain D' from D by replacing $P_{birth(v)}$ with the long ear $uv \cup vw$ and adding the short ear $av = ru'$ directly afterward. Then D' avoids u'.

Let Γ be an edge-edge-addition and suppose first that $birth(w) \neq birth(u)$. Then $u' = v$ and $birth(w) < birth(v) > birth(u)$. Obtain D' from D by replacing $P_{birth(v)}$ with the long ear $uv \cup vw$ and adding the short ear $av = ru'$ directly afterward. Then D' avoids u'. Now suppose that $birth(w) = birth(u)$. Then $b = d = u$, $u' = v$ and $birth(u) = birth(w) < birth(v)$. Obtain D' from D by replacing $P_{birth(v)}$ with the long ear $uv \cup vw$ and adding the short ear $av = ru'$ directly afterward. Hence, in all cases, we obtain a Mondshein sequence D' through rt' and avoiding u'.

Computational Complexity: For proving the Path Replacement Lemma 16, it remains to show that each modification can be computed in amortized constant time. Note that ears may become arbitrarily long in the path replacement process and therefore may contain up to $\Theta(n)$ vertices. Moreover, we have to maintain the birth-values of all vertices that are involved in future BG-operations in order to compute which of the subcases in Case (1)–(3) applies. Thus, we cannot use the standard approach of storing the ears of D explicitly by using doubly-linked lists, as then the birth-values of linearly many vertices may change for every modification.

Instead, we will represent the ears as the sets of a data structure for *set splitting*, which maintains disjoint sets online under an intermixed sequence of find and split operations. Gabow and Tarjan [15] discovered the first data structure for set splitting with linear space and constant amortized time per operation. Their and our model of computation is the standard unit-cost word-RAM. Imai and Asano [20] enhanced this data structure to an *incremental variant*, which additionally supports adding single elements to certain sets in constant amortized time. In both results, all sets are restricted to be intervals of some total order. To represent the Mondshein sequence D in the path replacement process, we will use the following more general data structure due to Djidjev [12, Section 3.2], which does not have that restriction but still supports the add-operation.

The data structure maintains a collection P of edge-disjoint paths under the following operations:

`new_path(x,y)`: Creates a new path that consists of the edge xy. The edge xy must not be in any other path of P.

`find(e)`: Returns the integer-label of the path containing the edge e.

`split(xy)`: Splits the path containing the edge xy into the two subpaths from x to one endpoint and from x to the other endpoint of that path.

`sub(x,e)`: Modifies the path containing e by subdividing e with the vertex x.

`replace(x,y,e)`: Neither x nor y may be an endpoint of the path Z containing e. This operation cuts Z into the subpath from x to y and the path that consists of the two remaining subpaths of Z joined by the new edge xy.

`add(x,yz)`: The vertex y must be an endpoint of the path Z containing the edge yz and x is either a new vertex or not in Z. Add the new edge xy to Z.

Note that all ears are not only edge-disjoint but also internally disjoint. Djidjev proved that each of the above operations can be computed in amortized constant time [12, Theorem 1]. We will only represent long ears in this data structure; the remaining short ears do not contain any essential birth-value information and can therefore be maintained simply as edges. As the data structure can only store paths, we need to clarify how the unique cycle P_0 in D can be maintained: We store P_0 as paths, namely as the two paths in P_0 with endpoints r and t. For every ear different from P_0, we store its two endpoints at its `find()`-label. These endpoints can therefore be accessed and updated in constant time.

Now we initialize the data structure with the Mondshein sequence of K_4 in constant time using the above operations. Every modification of the Cases (1)–(3) and $ru \in \{ab, cd\}$ can then be realized with a constant number of operations of the data structure, and hence in amortized constant time.

Additionally, we need to maintain the order of ears in D. The *incremental list order-maintenance problem* is to maintain a total order subject to the operations of (i) *inserting* an element after a given element and (ii) *comparing* two distinct given elements by returning the one that is smaller in the order. Tsakalidis [35] and Bender et al. [4] showed a simple solution with amortized constant time per operation (the latter holds even if, additionally, *deletions* of elements are supported); we will call this the *order data structure*. It is easy to see that the Path Replacement Lemma inserts in every step at most two new ears directly after $P_{birth(b)}$ and at most one new short ear at the end of D. Hence, we can maintain the order of ears in D by applying the order data structure to the `find()`-labels of ears; this costs amortized constant time per step.

For deciding which of the subcases in (1)–(3) and $ru \in \{ab, cd\}$ applies, we additionally need to maintain the birth-values of the vertices and edges in D. In fact, it suffices to support the queries "$birth(x) < birth(y)$" and "$birth(x) = birth(y)$", where x and y may be arbitrary edges or vertices in D. If x and y are edges, both queries can be computed in constant amortized time by comparing the labels `find(x)` and `find(y)` in the order data structure. In order to allow birth-queries on vertices, we will store pointers at every vertex x to the two edges e_1 and e_2 that are incident to x in $P_{birth(x)}$. The desired query involving $birth(x)$ can then be computed by comparing `find(e`$_1$`)` in the order data structure.

For any new vertex x that is added to D, we can find e_1 and e_2 in constant time, as these are in $\{av, vb, cw, wd, vw\}$. Since $P_{birth(x)}$ may change over time, we have to update e_1 and e_2 after each step. The only situation in which $P_{birth(x)}$ may loose e_1 or e_2 (but not both) is a `split` or `replace` operation on $P_{birth(x)}$ at x (the split operation must be followed by an add operation on x, as x is always inner vertex of some ear). This cuts $P_{birth(x)}$ into two paths, each of which contains exactly one edge in $\{e_1, e_2\}$. Checking `find(e`$_1$`)=find(e`$_2$`)` recognizes this case efficiently. Dependent on the particular case, we compute a new consistent pair $\{e'_1, e'_2\}$ that differs from $\{e_1, e_2\}$ in exactly one edge. This allows to check the desired comparisons in amortized constant time.

We conclude that D' can be computed from D in amortized constant time; this proves the Path Replacement Lemma. Thus, we deduce the following theorem.

Theorem 17. *Given edges rt and ru of a 3-connected graph G, a Mondshein sequence D of G through rt and avoiding u can be computed in time $O(m)$.*

The above algorithm is *certifying* in the sense of [27]: First, check in linear time that D is an ear decomposition of G. Second, check the side constraints on the first and last ear. Third, check in linear time that D is non-separating by testing that every ear satisfies Definition 4.

5 Applications

Application 1: Independent Spanning Trees
Let k spanning trees of a graph be *independent* if they all have the same root vertex r and, for every vertex $x \neq r$, the paths from x to r in the k spanning trees are *internally disjoint* (i.e., vertex-disjoint except for their endpoints; see Figure 5). The following conjecture from 1988 due to Itai and Rodeh [21] has received considerable attention in graph theory throughout the past decades.

Conjecture (Independent Spanning Tree Conjecture [21]). Every k-connected graph contains k independent spanning trees.

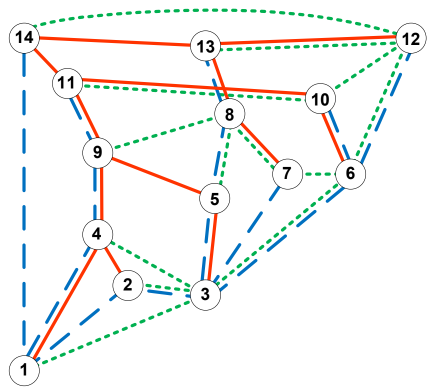


Figure 5: Three independent spanning trees in the graph of Figure 1, which were computed from its Mondshein sequence (vertex numbers depict a consistent tr-numbering).

The conjecture has been proven for $k \leq 2$ [21], $k = 3$ [6, 41] and $k = 4$ [8], with running times $O(m)$, $O(n^2)$ and $O(n^3)$, respectively, for computing the corresponding independent spanning trees. For every $k \geq 5$, the conjecture is open. For planar graphs, the conjecture has been proven by Huck [19].

We show how to compute three independent spanning trees in linear time, using an idea of [6]. This improves the previous best quadratic running time. It may seem tempting to compute the spanning trees directly and without using a Mondshein sequence, e.g. by local replacements in an induction over BG-operations or inverse contractions. However, without additional restrictions this is bound to fail, as shown in Figure 6.

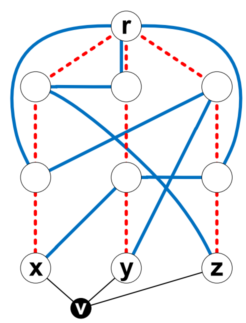


Figure 6: A 3-connected graph G (some edges are not drawn). G is obtained from the 3-connected graph $G' := (G - v) \cup xy$ by performing a BG-operation (or inverse contraction) that adds the vertex v (with added edge vy). Two of the three independent spanning trees of G' are given, rooted at r (thick edges). However, not both of them can be extended to v.

Compute a Mondshein sequence through rt and avoiding u, as described in Theorem 17. Choose r as the common root vertex of the three spanning trees and let $x \neq r$ be an arbitrary vertex.

First, we show how to obtain two internally disjoint paths from x to r that are both contained in the subgraph $G_{birth(x)}$. A *tr-numbering* $<$ is a total order $v_1 < \cdots < v_n$ of the vertices of a graph such that $t = v_1$, $r = v_n$, and every other vertex has both a higher-numbered and a lower-numbered neighbor. Let a tr-numbering $<$ be *consistent* [6] to a Mondshein sequence if $<$ is a tr-numbering for every graph G_i, $0 \le i \le m-n$. We can compute a consistent tr-numbering $<$ in linear time as follows: Let $<_0$ be the total order on $V(P_0)$ from t to r; then $<_0$ is a consistent tr-numbering of G_0. We maintain $<_{i-1}$ in the *order data structure* of [4] (see the computational complexity paragraph). Now we add iteratively the next ear P_i and obtain $<_i$ from $<_{i-1}$ by ordering the new inner vertices of P_i from the lower to the larger endpoint of P_i in $<_{i-1}$ (such that $inner(P_i)$ is between these endpoints in $<_i$). This takes amortized time proportional to the length of P_i and, hence, gives a total linear running time.

According to $<$, every vertex $x \neq r$ has a higher-numbered neighbor in $G_{birth(x)}$ and every vertex $x \notin \{r, t\}$ a lower-numbered neighbor in $G_{birth(x)}$. Fixing arbitrary such neighbors, the first two spanning trees T_1 and T_2 then consist of the incident edges to higher neighbors and of the edge tr and the incident edges to lower neighbors, respectively. Clearly, T_1 and T_2 are independent due to the numbering used.

We construct the third independent spanning tree T_3. As a Mondshein sequence is non-separating, every vertex $x \neq \{r, u\}$ has an incident edge with an endpoint in $\overline{G_{birth(x)}}$ (as seen before, iterating this argument gives a path to u in $\overline{G_{birth(x)}}$). Let T_3 consist of arbitrary such incident edges and of the edge ru. Since $G_{birth(x)}$ and $\overline{G_{birth(x)}}$ are vertex-disjoint, T_3 is independent from T_1 and T_2.

Remark. We remark that the three independent spanning trees constructed this way satisfy the following additional condition: Due to the fact that T_2 and T_3 are extended to r by one single edge, all incident edges of r are contained in at most one of T_1, T_2, T_3. In particular, no edge of G is contained in all three independent trees, which is a fact that cannot be derived from the definition of independent spanning trees (an edge that is incident to r may be contained in all three trees).

Application 2: Output-Sensitive Reporting of Disjoint Paths

Given two vertices x and y of an arbitrary graph, a *k-path query* reports k internally disjoint paths between x and y or outputs that these do not exist. Di Battista, Tamassia and Vismara [11] give data structures that answer k-path queries for $k \le 3$. A key feature of these data structures is that every k-path query has an *output-sensitive* running time, i.e., a running time of $O(\ell)$ if the total length of the reported paths is ℓ (and running time $O(1)$ if the paths do not exist). The preprocessing time of these data structures is $O(m)$ for $k \le 2$, but $O(n^2)$ for $k = 3$.

For $k = 3$, Di Battista et al. show how the input graph can be restricted to be 3-connected using a standard decomposition. For every 3-connected graph we can compute a Mondshein sequence, which allows us to compute three independent spanning trees T_1–T_3 in a linear preprocessing time, as shown in Application 1. If x or y is the root r of T_1–T_3, this gives a straight-forward output-sensitive data structure that answers 3-path queries: we just store T_1–T_3 and extract one path from each tree per query.

In order to extend these queries to k-path queries between arbitrary vertices x and y, [11] gives a case distinction that shows that the desired paths can be found efficiently in the union of the six paths in T_1–T_3 that join either x with r or y with r. This case distinction can be used for the desired output-sensitive reporting in time $O(\ell)$ without changing the preprocessing. We conclude that the preprocessing time of $O(n^2)$ for allowing k-path queries with $k \le 3$ in arbitrary graphs can be improved to $O(n + m)$.

Application 3: Planarity Testing

We give a conceptually very simple planarity test based on Mondshein's sequence for any 3-connected graph G in time $O(n)$. The 3-connectivity requirement is not crucial, as the planarity of G can be reduced to the planarity of all 3-connected components of G, which in turn are computed as a side-product from the computation of the BG-sequence [28, Appendix 2]. Alternatively, one could also use standard algorithms [18, 16] for reducing G to be 3-connected.

If $m > 3n - 6$, G is not planar due to Euler's formula and we reject the instance, so let $m \leq 3n - 6$. Let rt be an edge of G. We will find an embedding whose outer face is left of rt, unless G is non-planar. Due to Whitney [39], this embedding is unique. In light of Observation 11, we need to pick an edge $ru \neq rt$ such that $tr \cup ru$ is in a non-separating cycle. We can easily find such an edge by computing a Mondshein sequence through rt and avoiding some vertex $u' \notin \{r, t\}$, and then taking the edge that is incident to r in $P_0 - rt$ (alternatively, any linear-time algorithm that computes a non-separating cycle containing rt like the one in [6] can be used).

Now we compute a Mondshein sequence D through rt and avoiding u that satisfies Property 7.7 in time $O(n)$. If G is planar, Observation 11 ensures that D is a canonical ordering of our fixed embedding; in particular, the last vertex u and the edge rt will be embedded in the outer face. Due to Property 7.7, P_0 has no chords and every short ear xy satisfies $birth(x) \neq birth(y)$. For the embedding process, we rearrange the order of short ears in D such that all short ears xy with $birth(x) < birth(y)$ are direct successors of the long ear $P_{birth(y)}$ (this can be done in linear time using bucket sort).

We start with a planar embedding M_0 of P_0. Step by step, we attempt to augment M_i with the next long ear P_j in D as well as all short ears directly succeeding P_j in order to construct a planar embedding M_j of G_j.

Once the current embedding M_i contains u, we have added all edges of G and are done. Otherwise, u is contained in $\overline{G_i}$, according to Definition 6.2. Then $\overline{G_i}$ contains a path from each inner vertex of P_j to u, according to Lemma 5. Since u is contained in the outer face of the unique embedding of G, adding the long ear P_j to M_i can preserve planarity only when it is embedded into the outer face f of M_i. Thus, we only have to check that both endpoints of P_j are contained in f (this is easy to test by maintaining the vertices of the outer face). For the same reason, the short ears directly succeeding P_j can preserve planarity only if the set S of their endpoints in G_i is contained in f. Note that, if there is at least one such short ear, P_j has precisely one inner vertex v due to Property 7.7 and all short ears directly succeeding P_j have v as endpoint.

Thus, if the endpoints of P_j and S are contained in f, we embed P_j and the short ears into f in the only possible way, i.e. as a path or as one new vertex v with the short ears and the two edges of P_j as incident edges. Otherwise, we output "not planar". If desired, a Kuratowski-subdivision can then be easily extracted in time $O(n)$, as shown in [32, Lemma 5] (the extraction is even simpler, as we do not make use of adding "claws").

Application 4: Contractible Subgraphs in 3-Connected Graphs
A connected subgraph H of a 3-connected graph G is called *contractible* if contracting H to a single vertex generates a 3-connected graph. It is easy to show that a connected subgraph H is contractible if and only if $G - V(H)$ is 2-connected. While many structural results about contractible subgraphs are known in graph theory, we are not aware of any non-trivial result that computes them.

Using a Mondshein sequence, we can identify a nested family of $m - n$ contractible induced subgraphs in linear time, namely the subgraphs $\overline{G_i}$ for every $0 \leq i < m - n$. Clearly, these subgraphs are contractible, as $G - \overline{G_i}$ is 2-connected due to Lemma 7.5. Moreover, for each $i > 0$, $\overline{G_i}$ is an induced subgraph of the induced subgraph $\overline{G_{i-1}}$. In particular, every $\overline{G_i}$ contains u, since $\overline{V_{m-n-1}} = \{u\}$ due to Definition 9.2.

Application 5: The *k-Partitioning* Problem
Given vertices $a_1, \ldots, a_k$ of a graph G and natural numbers $n_1, \ldots, n_k$ with $n_1 + \cdots + n_k = n$, we

want to find a partition of V into sets $A_1, \dots, A_k$ with $a_i \in A_i$ and $|A_i| = n_i$ for every i such that every set A_i induces a connected graph in G. We call this a *k-partition*.

If the conditions $a_i \in A_i$ are ignored, the problem becomes NP-hard even for $k = 2$ and bipartite input graph G [13]; although often stated otherwise, this does not seem to imply an NP-hardness proof for the k-partitioning problem directly. If the input graph is k-connected, however, Györi [17] and Lovász [25] proved that there is always a k-partition. Thus, let G be k-connected. If $k = 2$, the k-partitioning problem is easy to solve: If G does not contain the edge a_1a_2, add this edge to G. Compute an a_1a_2-numbering $a_1 = v_1, v_2, \dots, v_n = a_2$ and observe that, for any vertex v_i (in particular for v_{n_1}), the graphs induced by $\{v_1, \dots, v_i\}$ and by $\{v_{i+1}, \dots, v_n\}$ are connected. For every $k \geq 4$, the k-partitioning problem on a k-connected input graph is not even known to be in P (although its decision variant is), so we will focus on the 3-partitioning problem of a 3-connected input graph.

This problem can be solved in quadratic time [33] and, if the graph is additionally *planar*, even in linear time [22]. As suggested in [38, 1], the problem (as well as a related extension) can be solved with the aid of a non-separating ear decomposition. For planar graphs, it thus suffices with Observation 11 to compute just a canonical ordering, which simplifies previous algorithms considerably.

More generally, we get the first $O(m)$ time algorithm for arbitrary 3-connected graphs as follows. Consider a Mondshein sequence through a_1a_2 and avoiding a_3 (if the edges a_1a_2 and a_1a_3 do not exist in G, we add them in advance). If G_i contains exactly $n_1 + n_2$ vertices for some i, we set $A_3 := \overline{G_i}$ and compute A_1 and A_2 by solving the 2-partitioning problem on G_i in linear time using an a_1a_2-numbering, as described above. Otherwise, let P_i be the first ear such that $|V(G_i)| > n_1 + n_2$.

We partition $inner(P_i)$ into the vertex sets B_1, B_3 and B_2 (designated to be part of A_1, A_3 and A_2, respectively) of three consecutive paths in $P_i - a_1a_2$ such that $|B_3| = n_3 - |\overline{V_i}|$. In particular, $0 < |B_3| < |inner(P_i)|$. Let $l := |B_1| + |B_2|$; then there are $l + 1$ choices for B_3. For any such choice, setting $A_3 := B_3 \cup \overline{V_i}$ satisfies the claim for A_3, as A_3 contains a_3, has cardinality n_3 and is connected, as a Mondshein sequence is non-separating.

We specify how to compute B_1; this determines the sets B_3 and B_2. If $i = 0$, choose B_1 as the path in $P_0 - a_1a_2$ that starts at a_1 and consists of n_1 vertices. The desired 2-partition of $G - A_3$ is then given by $A_1 := B_1$ and $A_2 := B_2$. If $i > 0$, we aim for a coloring of G_{i-1} into *blue* and *red* vertices such that A_1 consists of B_1 and the blue vertices, and A_2 consists of B_2 and the red vertices. In order to make A_1 connected, we have to prevent that both endpoints of G_{i-1} are colored red as long as $|B_1| > 0$. Clearly, $|B_1| < n_1$, as a_1 has to be in A_1; similarly, $|B_2| < n_2$, which implies $|B_1| > l - n_2$. Hence, the valid choices for $|B_1|$ are between $max\{0, l - n_2 + 1\}$ and $min\{l, n_1 - 1\}$.

For every $max\{0, l - n_2 + 1\} \leq |B_1| \leq min\{l, n_1 - 1\}$, we compute a 2-partition of G_{i-1} into $n_1 - |B_1|$ blue and $n_2 - |B_2|$ red vertices. The first 2-partition for $|B_1| = max\{0, l - n_2 + 1\}$ can be computed in linear time using an a_1a_2-numbering as described above. For each increase of $|B_1|$ by one, we can construct the new 2-partition in constant time from the old one, as exactly one blue vertex is recolored red. If the coloring of one of these choices for $|B_1|$ colors the endpoints x and y of P_i differently, we choose B_1 as the path in P_i next to the blue endpoint that consists of $|B_1|$ vertices. Then A_1 and A_2 as stated above give a 3-partition.

Otherwise, x and y have always the same color. Moreover, this color is identical, say red by symmetry, for every computed choice of $|B_1|$, since only one vertex is recolored per increase of $|B_1|$. Consider the smallest choice $|B_1| := max\{0, l - n_2 + 1\}$. As x and y are red, $n_2 - |B_2| \geq 2$, which implies $|B_1| > l - n_2 + 1$. Hence, $|B_1| = 0$ and we choose $B_1 := \emptyset$. Then A_1 and A_2 as stated above give the desired 3-partition.

Acknowledgments. I wish to thank Joseph Cheriyan for valuable hints, the anonymous referee who drew my attention to Lee F. Mondshein's work, David R. Wood for suggesting the graph in Figure 6, and the anonymous reviewers that gave me very valuable feedback, which led to a decrease of the number of cases in the Path Replacement Lemma.

References

[1] T. Awal and M. S. Rahman. A linear algorithm for resource tripartitioning triconnected planar graphs. *INFOCOMP Journal of Computer Science*, 9(2):39–48, 2010.

[2] M. Badent, U. Brandes, and S. Cornelsen. More canonical ordering. *Journal of Graph Algorithms and Applications*, 15(1):97–126, 2011.

[3] D. W. Barnette and B. Grünbaum. On Steinitz's theorem concerning convex 3-polytopes and on some properties of planar graphs. In *The Many Facets of Graph Theory*, pages 27–40, 1969.

[4] M. A. Bender, R. Cole, E. D. Demaine, M. Farach-Colton, and J. Zito. Two simplified algorithms for maintaining order in a list. In *Proceedings of the 10th European Symposium on Algorithms (ESA'02)*, pages 152–164, 2002.

[5] T. Biedl and M. Derka. The (3,1)-ordering for 4-connected planar triangulations. *Journal of Graph Algorithms and Applications*, 20(2):347–362, 2016.

[6] J. Cheriyan and S. N. Maheshwari. Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs. *Journal of Algorithms*, 9(4):507–537, 1988.

[7] Y.-T. Chiang, C.-C. Lin, and H.-I. Lu. Orderly spanning trees with applications. *SIAM Journal on Computing*, 34(4):924–945, 2005.

[8] S. Curran, O. Lee, and X. Yu. Finding four independent trees. *SIAM Journal on Computing*, 35(5):1023–1058, 2006.

[9] H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting Fáry embeddings of planar graphs. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 426–433, 1988.

[10] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[11] G. Di Battista, R. Tamassia, and L. Vismara. Output-sensitive reporting of disjoint paths. *Algorithmica*, 23(4):302–340, 1999.

[12] H. N. Djidjev. A linear-time algorithm for finding a maximal planar subgraph. *SIAM J. Discrete Math.*, 20(2):444–462, 2006.

[13] M. E. Dyer and A. M. Frieze. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10:139–153, 1985.

[14] H. d. Fraysseix and P. O. de Mendez. Regular orientations, arboricity, and augmentation. In *Proceedings of the DIMACS International Workshop on Graph Drawing 1994*, volume LNCS 894, pages 111–118, 1995.

[15] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences*, 30(2):209–221, 1985.

[16] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In *Proceedings of the 8th International Symposium on Graph Drawing (GD'00)*, pages 77–90, 2001.

[17] E. Győri. Partition conditions and vertex-connectivity of graphs. *Combinatorica*, 1(3):263–273, 1981.

[18] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973.

[19] A. Huck. Independent trees in planar graphs. *Graphs and Combinatorics*, 15(1):29–77, 1999.

[20] H. Imai and T. Asano. Dynamic orthogonal segment intersection search. *Journal of Algorithms*, 8(1):1–18, 1987.

[21] A. Itai and M. Rodeh. The multi-tree approach to reliability in distributed networks. *Information and Computation*, 79:43–59, 1988.

[22] L. Jou, H. Suzuki, and T. Nishizeki. A linear algorithm for finding a non-separating ear decomposition of triconnected planar graphs. Technical Report AL40-3 (in Japanese), Information Processing Society of Japan, Tokyo, 1994.

[23] G. Kant. Drawing planar graphs using the lmc-ordering. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS'92)*, pages 101–110, 1992.

[24] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996.

[25] L. Lovász. A homology theory for spanning trees of a graph. *Acta Mathematica Hungarica*, 30(3-4):241–251, 1977.

[26] L. Lovász. Computing ears and branchings in parallel. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS'85)*, pages 464–467, 1985.

[27] R. M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011.

[28] K. Mehlhorn, A. Neumann, and J. M. Schmidt. Certifying 3-edge-connectivity. *Algorithmica*, 77(2):309–335, 2017.

[29] L. F. Mondshein. *Combinatorial Ordering and the Geometric Embedding of Graphs*. PhD thesis, M.I.T. Lincoln Laboratory / Harvard University, 1971. Technical Report available at www.dtic.mil/cgi-bin/GetTRDoc?AD=AD0732882.

[30] J. M. Schmidt. Construction sequences and certifying 3-connectedness. In *Proceedings of the 27th Symposium on Theoretical Aspects of Computer Science (STACS'10)*, pages 633–644, 2010.

[31] J. M. Schmidt. Contractions, removals and certifying 3-connectivity in linear time. *SIAM Journal on Computing*, 42(2):494–535, 2013.

[32] J. M. Schmidt. A planarity test via construction sequences. In *38th International Symposium on Mathematical Foundations of Computer Science (MFCS'13)*, pages 765–776, 2013.

[33] H. Suzuki, N. Takahashi, T. Nishizek, H. Miyano, and S. Ueno. An algorithm for tripartitioning 3-connected graphs. *Information Processing Society of Japan (IPSJ)*, 31(5):584–592, 1990. (In Japanese).

[34] C. Thomassen. Kuratowski's theorem. *Journal of Graph Theory*, 5(3):225–241, 1981.

[35] A. K. Tsakalidis. Maintaining order in a generalized linked list. *Acta Informatica*, 21(1):101–112, 1984.

[36] W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–767, 1963.

[37] W. T. Tutte. Connectivity in graphs. In *Mathematical Expositions*, volume 15. University of Toronto Press, 1966.

[38] K. Wada and K. Kawaguchi. Efficient algorithms for tripartitioning triconnected graphs and 3-edge-connected graphs. In *19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'93)*, pages 132–143, 1993.

[39] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932.

[40] H. Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34(1):339–362, 1932.

[41] A. Zehavi and A. Itai. Three tree-paths. *Journal of Graph Theory*, 13(2):175–188, 1989.

[42] G. M. Ziegler. *Lectures on Polytopes.* Springer, corrected second printing, 1998.