Planar Bus Graphs

Till Bruckdorfer[1,a]
bruckdor@informatik.uni-tuebingen.de

Stefan Felsner[2,b]
felsner@math.tu-berlin.de

Michael Kaufmann[3,a]
mk@informatik.uni-tuebingen.de

Abstract

Bus graphs are used for the visualization of hypergraphs, for example in VLSI design. Formally, they are specified by bipartite graphs $G = (B \cup V, E)$. The bus vertices B are realized by horizontal and vertical segments, and the connector vertices V are realized by points and connected orthogonally to the bus segments without any bend; this is called *bus realization.* The decision whether a bipartite graph admits a bus realization, where connections may cross, is NP-complete. In this paper we show that in contrast the question whether a planar bipartite graph admits a planar bus realization can be answered in polynomial time.

First we deal with plane instances, i.e., with the case where a planar embedding is prescribed. We identify three necessary conditions on the partition $B = B_{\mathrm{V}} \cup B_{\mathrm{H}}$ of the bus vertices, here B_{V} denotes the vertical and B_{H} the horizontal buses. We provide a test whether a *good partition*, i.e., a partition obeying these conditions, exists. The test is based on the computation of a maximum matching on some auxiliary graph. Given a good partition we can construct a non-crossing realization of the bus graph on an $O(n) \times O(n)$ grid in linear time. In the second part we use SPQR-trees to solve the problem for general planar bipartite graphs.

1 Introduction

A classical topic in the area of graph visualization is *orthogonal graph drawing*, as such it is covered in all books on graph drawing [6, 21, 25]. In this drawing model each edge consists of a series of subsequent horizontal or vertical line segments. Applications can be found in e.g. VLSI design, cf. [29, 22]. In this application it may also be necessary to model hypergraphs. For example power buses on VLSI chips are often modeled as hyperedges, as well as LANs in computer network visualization. Bus graphs – as being defined later – and their generalizations are a possible approach to represent hyperedges. A bus-style representation might also be used when facing the visualization of highly interconnected parts of a given graph. So, cliques can be represented in a compact and comprehensive way using a bus-style model as an alternative model to edge bundling and confluent drawings [13, 9].

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Tübingen, Germany
[2] Institut für Mathematik, Technische Universität Berlin, D-10623 Berlin, Germany
[3] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Tübingen, Germany

[a] Partially supported by EuroGIGA project GraDR 10-EuroGIGA-OP-003.
[b] Partially supported by DFG grant FE-340/7-2 and ESF EuroGIGA project Compose.

The bus graph approach is very much related to the classical topic of rectilinear Steiner trees, where trees are being used to connect subsets of the vertices [16, 20]. Bus graphs also relate to rectangular drawings, rectangular duals and visibility graphs. The latter graphs are indeed close to bus graphs, because connections in bus graphs enforce visibility constraints in realizations. Corresponding key concepts and surveys can be found in [19, 27, 28, 11] and in chapter 6 of [25]. We will use some of the methods that have been developed in this area.

We are considering the bus graph model introduced by Ada et al. [1]. A *bus graph* is a bipartite graph $G = (B \cup V, E)$, where $E \subseteq V \times B$ and $deg(v) \leq 4$ for all $v \in V$. We call vertices in B *bus vertices* and vertices in V *connector vertices*. A *realization* of a bus graph is a drawing D, where bus vertices are represented as horizontal or vertical line segments (*bus segments*), connector vertices are drawn as points (*connectors*), and the edges are horizontal or vertical segments (*connections*), i.e. connecting a point with a bus segment perpendicularly to that segment. In any realization a bus may not be crossed. To distinguish between bus vertices and edges in a realization, the bus segments are drawn with bold lines, see Figure 1. A *planar realization* is a realization without crossings. A *planar bus graph* is a bus graph that admits a realization without crossings, while a *plane bus graph* is a planar bus graph together with a planar embedding[1]. We always assume to have a connected bus graph, since components can be considered separately.

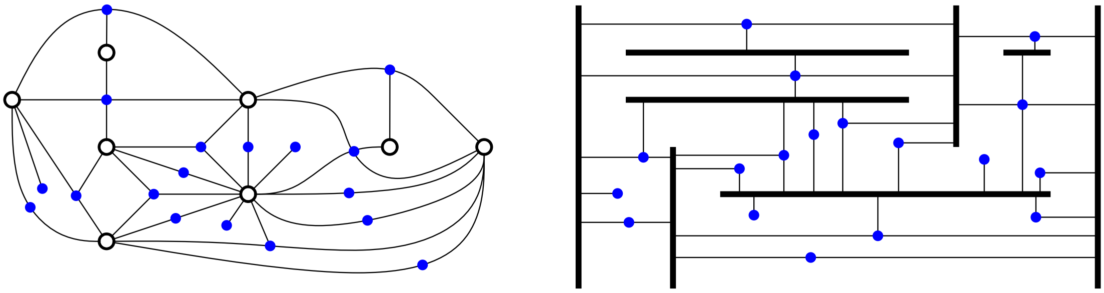

Figure 1: An example of a plane bus graph with a planar realization.

In [1] a relation of bus graphs to hypergraphs is mentioned: the bipartite adjacency matrix of a bipartite graph can be read as incidence matrix of a hypergraph and vice versa. If vertices are contained in at most four hyperedges a bus graph realization may provide a good visualization of the hypergraph. Ada et al. [1] considered the problem to decide if a bus graph has a realization and showed the NP-completeness. In this paper we consider the problem to decide if a planar bus graph has a planar realization. We show that this question in contrast to the previous result can be decided in polynomial time.

The bus segments will be drawn either vertically or horizontally. We assign labels to the bus vertices that determine whether they are to be realized vertically or horizontally and study properties that have to be obeyed by this labeling to actually correspond to a planar realization. In Section 2 we identify necessary conditions for the labeling - if they are satisfied we speak of a *good partition*. In Section 3 we present an algorithm to test whether a maximal plane bus graph admits a good partition. Subsequently in Section 4 we give a linear-time algorithm to produce a planar realization on a grid of size $O(n) \times O(n)$ from a maximal plane bus graph together with a good partition. The approach is based on techniques from [3] and [12]. In the next Section 5 we extend the test for a good partition to non-maximal plane bus graphs. For an extension to (non-embedded) planar bus graphs, we first recall main definitions for SPQR-trees in Section 6 and

[1] A planar embedding is a drawing of the vertices together with a description of the cyclic order of their adjacent vertices. Often a drawing with curves representing the edges is used to illustrate an embedding.

extend the results from Section 5 to biconnected planar bus graphs in an easy form (Section 7) then to biconnected planar graphs without simplifications (Subsection 7.2) and finally to general planar bus graph in Subsection 7.3.

2 Necessary Properties

Clearly non-planar bus graphs cannot be drawn in a planar way. So we focus on finding structures in a planar bus graph. First we assume the bus graph $G = (V \cup B, E)$ to be *plane*, i.e. a planar bus graph together with a fixed planar embedding. Later we concentrate on planar bus graphs without any given embedding. We assume any bus graph to be *simple*, i.e., there are no multiple edges (multiple edges would necessarily be on top of each other).

A planar realization of a plane bus graph $G = (V \cup B, E)$ induces labels H or V for the bus vertices where H indicates that a bus is represented by a horizontal segment in the realization, while V indicates that the bus is represented by a vertical segment. Let $\Pi = (B_{\mathrm{V}}, B_{\mathrm{H}})$ denote the *partition* of B according to the label of the bus vertices. We observe two natural properties of a partition $\Pi = (B_{\mathrm{V}}, B_{\mathrm{H}})$ corresponding to a planar realization:

(P1) Every connector vertex of degree ≥ 3 has neighbors in both classes.

(P2) A connector vertex of degree 4 has cyclically alternating neighbors in both classes.

Properties (P1) and (P2) for a partition of the bus vertices are not sufficient to ensure a planar representation. Also a third property for what we call 'diamonds' must be true. A *diamond* in a plane bus graph is a cycle $z = (b, v, b', v')$ of length four with bus vertices b, b' and connector vertices v, v' such that both v, v' have a third bus neighbour in the bounded region defined by z. Note that the outer cycle of G may be a diamond.

Lemma 1 *Let G be a plane bus graph that has a realization inducing the partition $(B_{\mathrm{V}}, B_{\mathrm{H}})$ of the set of buses B. For any diamond $z = (b, v, b', v')$ the two bus vertices b and b' belong to the same class.*

Proof. Suppose for contradiction that $b \in B_{\mathrm{H}}$ and $b' \in B_{\mathrm{V}}$. The interior of z in the planar bus realization is a polygon with six corners. Four of the corners are at contacts of connector edges and buses and two corners are at the connector vertices. We account for four corners of size $\pi/2$ each, where the edges meet the buses. The other two corners are at v and v'. Since b is horizontal and b' vertical the angles at v and v' have to be either $\pi/2$ or $3\pi/2$. Because v and v' have an additional bus neighbor in the interior the angle at each of v and v' is at least π. Hence, both these angles are of size $3\pi/2$. Therefore, the sum of interior angles is at least $4 \cdot \pi/2 + 2 \cdot 3\pi/2 = 5\pi$. A six-gon, however, has a sum of angles of 4π. The contradiction shows that b and b' belong to the same class of the partition $(B_{\mathrm{V}}, B_{\mathrm{H}})$. □

If a diamond $z = (b, v, b', v')$ of a plane bus graph G with the property that one of the connector vertices has degree four and all its neighbours are in the closed bounded region defined by z, then from Lemma 1 we see that G cannot admit a planar bus representation, since the angles incident to v and v' in the closed bounded region defined by z have to be π. Since our algorithm will identify all diamonds it may reject instances that contain this kind of diamonds right away.

As a consequence of the lemma we state a third property to ensure a planar realization.

(P3) A diamond has both bus vertices in the same class.

Definition 1 *A partition* $\Pi = (B_{\mathrm{V}}, B_{\mathrm{H}})$ *of the buses of a plane bus graph* G *is called a* good partition *if it obeys properties (P1), (P2), and (P3).*

In the next section we consider maximal plane bus graphs and test efficiently whether they admit a good partition. The test is constructive, i.e., if the answer is yes, then a good partition is constructed.

3 Maximal Plane Bus Graphs

A *maximal plane bus graph* G is a plane bipartite bus graph that is a quadrangulation, i.e. all faces have cardinality 4. Let $G = (V \cup B, E)$ be a maximal plane bus graph. We assume w.l.o.g. that G has no bus vertices of degree 1 and no connector vertices of degree 1 or 2, since these connector vertices have no influence on the existence of a good partition. Clearly, if G has a bus vertex or connector vertex of degree 1, then G is not maximal or not simple, and if G has a connector vertex of degree 2, then removing this vertex with its incident edges results in a smaller maximal plane bus graph that has a good partition if and only if G has a good partition. Also recall that diamonds are defined by connector vertices of degree at least 3. Let $\Delta = \Delta(G)$ denote the *degree of a plane bus graph* G which is defined as the maximum degree among the connector vertices of G.

In this section we first assume that G has $\Delta = 3$; we give an algorithm to test if G admits a good partition and if so the algorithm returns a good partition. After that we allow $\Delta = 4$ and reduce this case with a simple modification to the case $\Delta = 3$.

Let $G = (V \cup B, E)$ be a maximal plane bus graph with $\Delta = 3$. The *connector graph* $C_G = (V_C, E_C)$ of G consists of all the connector vertices $V_C = V$ and edges $(v, v') \in E_C$, if v and v' are both incident to the same face of the plane embedding of G. The connector graph is helpful because it allows the translation of the problem of finding a good partition for G to the problem of finding an appropriate perfect matching in C_G, as summarized in Proposition 1 and Proposition 2 and illustrated in Figure 2.

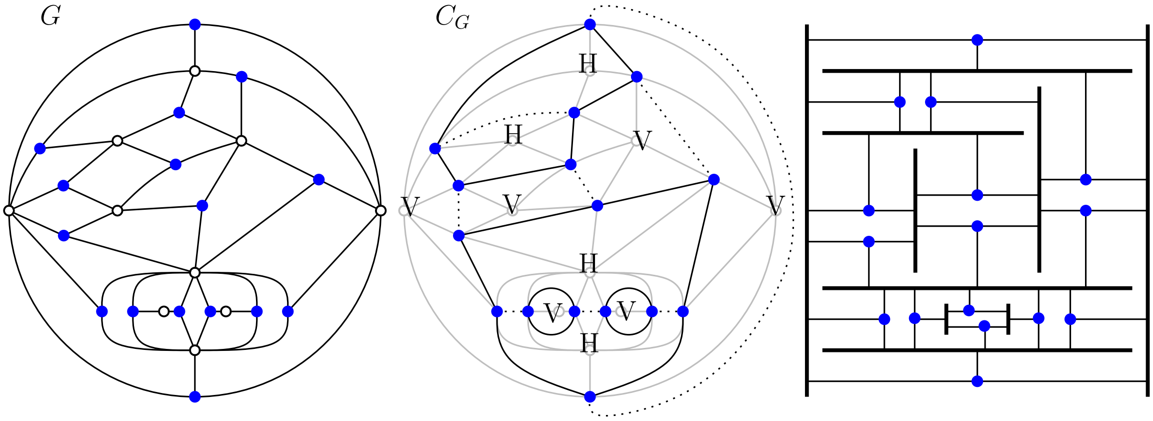

Figure 2: A maximal plane bus graph, the connector graph with a matching (dotted edges) and its good partition, and a corresponding bus representation.

We first show that a good partition Π of G can be used to define a perfect matching of C_G. The first property (P1) of a good partition requires that every connector vertex v has two adjacent bus vertices in one partition class and one in the other. Let b and b' be neighbors of v within the same class. Since every face has cardinality 4 there is a connector vertex v' such that v, b, v', b'

are the vertices of a face of G. The edge (v, v') is called the *reserved* edge of v. Since b and b' are the two adjacent bus vertices of v' of the same class the reserved edge of v' is (v', v), i.e., the same edge. Hence, the collection of reserved edges is a matching covering all the vertices of C_G, i.e., a perfect matching.

Now we show that a perfect matching M of C_G can be used to define a labeling of the bus vertices satisfying (P1). Removing the matching edges from C_G leaves a 2-regular graph, i.e., a disjoint collection of cycles. The regions defined by this collection of cycles can be 2-colored with colors V and H such that each cycle has regions of different colors on its sides. Let B_{V} be the set of bus vertices in faces colored with V, and B_{H} be the set of bus vertices in faces colored with H. This yields a partition satisfying (P1) because every connector vertex is on a cycle and has a bus neighbour in each of the two faces bounded by the cycle. Since $\Delta = 3$, the second property (P2) is void.

Consider a diamond $z = (b, v, b', v')$ in G. Each of v, v', has exactly one edge e, e', in C_G, that corresponds to a face of the outside of z. Let M be a matching defined as described above starting from a good partition. We know from (P3) that b and b' have the equal labels, therefore, e and e' must be the matching edges of v and v' respectively. Conversely, if we aim at defining a good partition as described above starting from a matching M then the two edges e and e' must belong to M, otherwise b and b' would get distinct labels and, hence, violate (P3).

We define the set E_d of *edges forced by diamonds* as the set of edges consisting of the two outside edges e, e' in C_G for each diamond z of G.

The considerations regarding the relation between good partitions and matchings of C_G are summarized in the following proposition.

Proposition 1 *Let G be a maximal plane bus graph with $\Delta = 3$, connector graph C_G, and the set E_d of edges of C_G forced by diamonds. Then G admits a good partition if and only if C_G has a perfect matching M, with $E_d \subseteq M$.*

Now we allow $\Delta = 4$ for a maximal plane bus graph G. To transform G into a plane bus graph G' with $\Delta = 3$, we split every connector vertex v of degree 4 into two connector vertices v', v'', both of degree 3 in the following way: let b_1, b_2, b_3, b_4 be the adjacent bus vertices of v in cyclic order around v. Remove v and its incident edges and introduce new vertices v', v'' with edges (b_1, v'), (b_2, v'), (b_3, v'), (b_3, v''), (b_4, v''), (b_1, v''), as illustrated in Figure 3. The connector graph C_G is obtained from $C_{G'}$ by contracting the edges (v', v'') corresponding to the pairs v', v'' that have been obtained by splitting a vertex of degree 4. Define the set E_s of edges *forced by splits* of $C_{G'}$ as the set of these edges (v', v'').

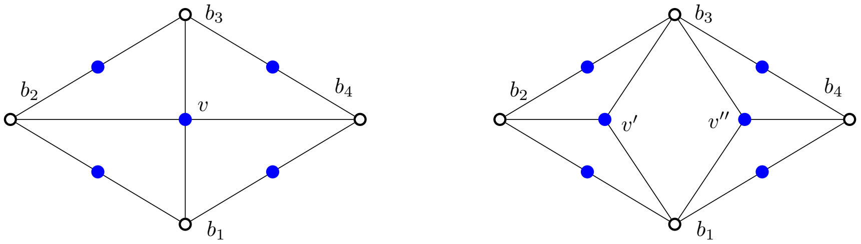

Figure 3: A split of a connector vertex of degree 4 into two connector vertices of degree 3.

If G has a good partition, then by (P2) the labels of the neighbours of v in G alternate. Hence (v', v'') will belong to the matching M of $C_{G'}$ induced by the good partition. Since $(v', v'') \in E_s$ we find that $E_s \subset M$. Conversely if $(v', v'') \in E_s$ belongs to a perfect matching M of $C_{G'}$, then b_1, b_3 have the same label in the partition induced by M. Since v' and v'' both have degree 3 and two of their neighbours have equal label, the third neighbour (for each of them) has a different label, i.e. b_2 and b_4 have both different label compared to the label of b_1, b_3, hence v obeys property (P2).

In summary: a partition Π of G that satisfies (P1) also satisfies property (P2) if and only if all edges of E_s are contained in the perfect matching of $C_{G'}$ corresponding to Π.

For notational simplicity we denote the connector graph $C_{G'}$ of the transformed graph G' by C_G. An example of a maximal plane bus graph with its connector graph showing the edges of $E_d \cup E_s$ is shown in Figure 4.

Proposition 2 *Let G be a maximal plane bus graph with $\Delta = 4$, C_G its connector graph, and E_d, E_s the edges of C_G that are forced by diamonds and splits, respectively. The graph G admits a good partition if and only if C_G has a perfect matching M, with $(E_d \cup E_s) \subseteq M$.*

Proof. The proof almost follows from Proposition 1 and the above considerations. Splitting connector vertices of degree 4, however, may separate diamonds. If $z_1 = (v_l, b_1, v, b_3)$ and $z_2 = (v, b_1, v_r, b_3)$ are diamonds (in this case $z_3 = (v_l, b_1, v_r, b_3)$ is a diamond too), then splitting the connector vertex v of degree 4 is separating them. We claim that separating diamonds does not imply additional restrictions on the matching. Let v be split into v', v'' as above. Any diamond containing v has a cycle z with the two bus vertices b_1 and b_3 belonging to the diamonds. Condition (P3) for this diamond requires that b_1 and b_3 belong to the same class of a good partition. This requirement, however, is already implied by condition (P2) for the original connector vertex v. Hence the claim holds. □

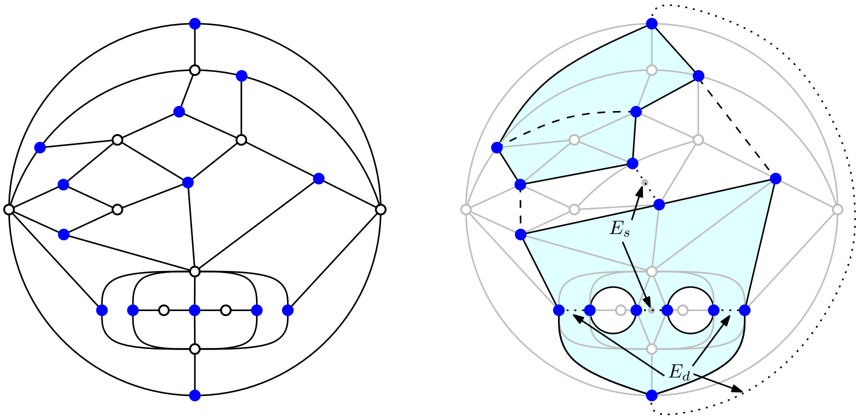

Figure 4: A maximal plane bus graph and its connector graph with the modifications, where the dotted edges are forced and the dashed edges complete the perfect matching.

Theorem 1 *Let G be a maximal plane bus graph. A good partition for G can be computed in $O(n^{3/2})$ time if it exists.*

Proof. By Proposition 2 it suffices to test the connector graph C_G for a perfect matching M that contains $(E_d \cup E_s)$. The extraction of the connector graph C_G from G requires linear time. The set E_s can be computed while constructing the modification G' of the original bus graph G, as well as C_G. To identify diamonds we consider the dual D_G of the connector graph C_G. The vertices of D_G represent the bus vertices of G and edges have corresponding dual edges in C_G. A diamond of G' corresponds to a double edge of D_G; the only exception is the diamond bounding the outer face. Double edges of D_G can be found and grouped so that the set E_d can be constructed in $O(n \text{ polylog}(n))$ time. To force $E_d \cup E_s$ we simply delete all vertices incident to these edges from the graph. If a vertex is incident to two edges from the set, then there is no matching. For constructing a perfect matching of a graph there exist several $O(\sqrt{n}m)$ algorithms, e.g., see [23]. For planar graphs this yields the claimed $O(n^{3/2})$ complexity.[2]

Given the perfect matching the corresponding good partition can again be computed in linear time. □

Notice that a perfect matching in a bridgeless planar 3-regular graph always exists by Petersen's theorem [26] and can be computed in linear time [2]. Unfortunately this efficient algorithm does not support the requirement that edges of $E_s \cup E_d$ have to participate in the matching.

4 Planar Realizations

In this section we show how to construct a planar realization from a maximal plane bus graph G with a good partition. In the proof we use the following theorem about realizations.

Theorem 2 *Planar bipartite graphs admit segment contact representations with interiorly disjoint vertical and horizontal segments.*

This theorem was obtained by Hartman et al. [18] and independently by de Fraysseix et al. [4]. In our application the bipartite graph will be a quadrangulation $Q = (V \cup H, E)$. In the representation vertices of V are represented by vertical line-segments, and vertices of H by horizontal line-segments. The segments of two vertices u and v share a point (i.e., a point of contact where one of the segments ends at an interior point of the other segment) if and only if $(u, v) \in E$. A counting argument shows that all but four of the endpoints of segments are needed to represent all edges of Q. Hence, the segment contact representation of Q can be seen as a dissection of a rectangle into small rectangles. These small rectangles correspond to the faces of Q. A survey on the topic can be found in [11], there and in [5] it is also shown that a segment contact representation of a quadrangulation with n vertices on an $n \times n$ grid can be constructed in linear time.

Now we are ready to show how to construct a planar realization from a maximal plane bus graph G with a good partition.

Theorem 3 *Let G be a maximal plane bus graph admitting a good partition. Then G has a planar realization on a grid of size $O(n) \times O(n)$. The realization can be computed in $O(n)$ time.*

Proof. Let G be a maximal plane bus graph admitting a good partition. We start with some simplifications. First we remove all connector vertices of degree 2 and bus vertices of degree 1. Observe that there are no connector vertices and bus vertices of degree 1, since G is maximal and simple. Second we split all connector vertices of degree 4 into two connector vertices of degree 3. After these two modifications the new graph G' is still a quadrangulation and in particular all connector vertices have degree 3.

The *reduced bus graph* $R' = (B', E_R)$ of G' is the graph on the bus vertices of G' with edges (b, b') if and only if b, b' are incident to a common face and have different labels. Diamonds with different labeled bus vertices are the only substructure that would create double edges in R' but diamonds have identically labeled bus vertices in a good partition. Notice that a double edge in R' infers either a diamond or a degree 2 connector vertex, which does not exist in G'. Hence, there are no double edges in R'. From the three faces incident to a connector vertex exactly two contribute an edge to R', by property (P1). A face that does not contribute an edge to R' is

[2] In [24] a slightly faster randomized algorithm for planar graphs has been proposed.

incident to two connector vertices and each of them contributing two edges to R'. These four edges form a 4-face of R'. Since every face of R' is of this type R' is a quadrangulation. An equivalent view on this is that the edges of the matching M of Proposition 2 are in bijection with the faces of R'.

The construction of a planar realization is as follows. First we identify the two classes B_{V} and B_{H} of the bipartition of the reduced bus graph R' with black and white. Next we construct a segment contact representation of R'. Later the following observation will be important:

($\star$) The rectangles in the segment contact representation correspond bijectively to the faces of R'. Moreover, a vertex b is incident to a face f in R' if and only if the segment S_b contributes a side of the rectangle R_f corresponding to f.

From the segment contact representation of R' we obtain a representation of the bus graph G' in two steps. First clip the endpoints of all segments of the representation so that a disjoint collection of segments remains. These segments serve as the bus segments for the representation of the bus graph G'. It remains to insert the connector vertices and the edges of G' into the picture. To this end recall that each connector vertex belongs to a unique face of R' and each face of R' contains exactly two connector vertices due to the degree 3 of all connector vertices. The two connector vertices contained in a face f can easily be accommodated in the rectangle R_f, because of ($\star$). Figure 5 shows this process.

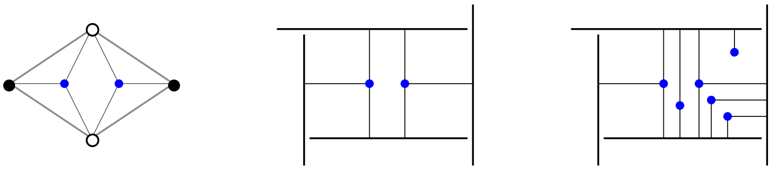

Figure 5: (left) A face f of R' with its two connector vertices. (middle) The placement of the two vertices in R_f. (right) Addition of some connector vertices of degree 2 and 1.

At this point we have a representation of the slightly modified maximal plane bus graph G'. It remains to transform the planar representation of G' into a planar representation of the original input graph G. These are the steps that have to be done:

- Merge pairs of connector vertices that have been created by splitting a connector vertex of degree 4. This is straight-forward, since the two connector vertices are accommodated in the same rectangle of the segment contact representation due to the created edge in E_s forced to be in the matching.

- Insert all bus vertices of degree 1 and all connector vertices of degree 2 that had been deleted. Observe that if a connector vertex v of degree 2 was adjacent to two bus vertices with equal label, then v is placed in the unique rectangle of the segment contact representation given by the bijection in ($\star$) from the face of R', where v was removed. If a connector vertex v of degree 2 was adjacent to two bus vertices with different labels, then v is placed in one of the two rectangles of the segment contact representation given by the bijection in ($\star$) from the two faces f, f' of R', where v was removed. This choice is arbitrary due to routing the edge incident to f and f' arbitrarily.

This yields a representation of the input graph G.

To complete the proof it remains to argue about the complexity. Let $G = (V \cup B, E)$ be a maximal plane bus graph with $n = |V| + |B|$. The bus graph G' is obtained by removing or splitting some vertices, which can be done in linear time. The reduced bus graph R' can be computed from the plane graph G' in $O(n)$ time. Let n' be the number of vertices of R'. A segment contact representation of R' on an $n' \times n'$ grid can be computed in $O(n')$ time [5]. The compression of grid lines that do not contain any connector vertex or endpoint of a bus segment, as well as the reinsertion of connector vertices and bus vertices and the merging of vertices that were created by splitting a connector vertex of degree 4, leads to an $O(n) \times O(n)$ grid. Finally each grid line is occupied by either a connector vertex, or a bus segment along the grid line, or an end point of a bus segment, hence the total number of occupied grid lines is $3|B| + 2|V|$. This is true even if we extend the graph at the beginning to match maximality (see the next section) and remove the inserted vertices at the end. □

Now we finished the construction of a planar realization for a given maximal plane bus graph through the computation of a good partition for the bus vertices. In the next sections we generalize the result from Section 3. In Section 5 we deal with non-maximal plane bus graphs. In Sections 6 and 7 we assume that the graph is biconnected but has no prescribed embedding. Finally, in Subsection 7.3 we discuss the general case, i.e., non-embedded planar connected bus graphs.

In all these cases we aim at arriving at a maximal plane bus graph together with a good partition, if it exists; then Theorem 3 from this section can be applied as a black box.

5 Non-Maximal Plane Bus Graphs

In this section we consider a plane bus graph G with n vertices that is not necessarily maximal. In a first preprocessing step we remove all connector vertices and bus vertices with degree 1, as well as their incident edge. These vertices can easily be integrated in a realization of the remaining graph. Notice that we do not delete any connector vertices of degree 2.

In the following we describe how to augment G to a maximal plane bus graph G^+ containing G as induced subgraph such that G^+ has a good partition if and only if G has a good partition (Lemma 2). The graph G^+ will be called *the quadrangulation* of G.

Let f be a face with cardinality $2k$ in G and let $b_1, \ldots, b_k$ be the bus vertices of f in clockwise order, possibly with $b_i = b_j$ for some $i \neq j$. To *quadrangulate* f we first place a new bus vertex b^*_f in the inside. The bus vertex b^*_f is then connected to the boundary of f by adding a triangular structure for every consecutive pair b_i, b_{i+1} of bus vertices including the pair b_k, b_1. The triangular structure for b_i, b_{i+1} consists of another new bus vertex d_i and three connector vertices v^1_i, v^2_i, v^3_i such that $N(v^1_i) = \{b_i, d_i, b_{i+1}\}$, $N(v^2_i) = \{b_{i+1}, d_i, b^*_f\}$, and $N(v^3_i) = \{b^*_f, d_i, b_i\}$. Figure 6 shows an example.

The graph G^+ is obtained from G by quadrangulating every face f with cardinality greater than 4 including, if necessary, the outer face. The following properties of the quadrangulation G^+ of G are obvious:

- G^+ is planar and has $O(n)$ vertices.
- All diamonds of G^+ are diamonds of G.

Note that the outer face of G^+ has cardinality 4. If the outer face of G has cardinality greater than 4 this is an additional diamond of G^+. We will prove that the condition imposed by this diamond of G^+ does not actually alter the existence of a good partition for G.
In addition we have the following important lemma:

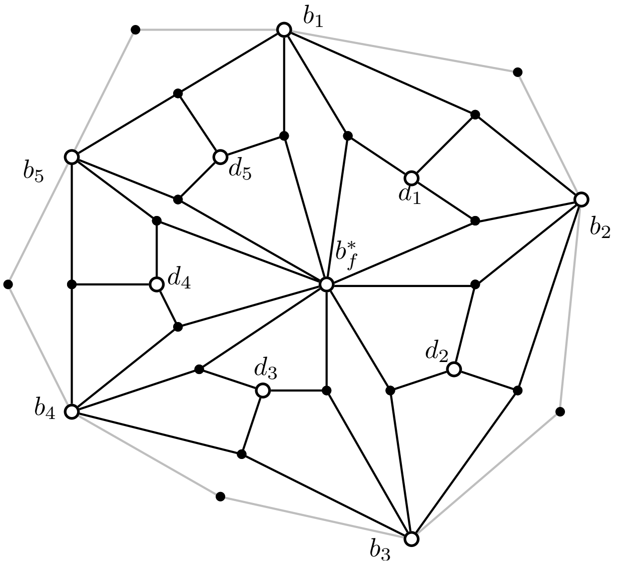

Figure 6: New vertices and edges added to quadrangulate a face f with cardinality 10.

Lemma 2 *Let G be a plane bus graph and G^+ be its quadrangulation. Then G has a good partition if and only if G^+ has a good partition.*

Proof. The three defining properties (P1), (P2), and (P3) are stable under taking induced subgraphs. Hence, a good partition of G^+ immediately yields a good partition of G.

Now assume that G has a good partition. We aim for a partition of the bus vertices of G^+ that extends the given partition of the bus vertices of G. Since all connector vertices of degree 4 and all diamonds of G^+ already belong to G we do not have to care of (P2) and (P3). The following rules define the labels for the new bus vertices:

- Label all central bus vertices b_f^* with V.
- If b_i and b_{i+1} are both labeled H, then the label of d_i is defined to be V. Otherwise the label d_i is H.

It is straightforward to check that (P1) is fulfilled for all new connector vertices, i.e., the construction yields a good partition of G^+.

If vertices have been added to the outer face f^* in the quadrangulation process, then we can choose the outer face of G^+ such that both its incident bus vertices are labeled V and one of them is $b_{f^*}^*$. This change in the outer face does not affect the plane embedding of G. These considerations imply that when looking for a good partition of G^+ we do not fail because of the condition implied by the diamond defined by the outer face of G^+ if this was not already a diamond of G. □

Theorem 4 *Let G be a plane bus graph. A good partition for G can be computed in $O(n^{3/2})$ time if it exists.*

Proof. By Lemma 2 it suffices to test if the quadrangulation G^+ of $G = (V \cup B, E)$ has a good partition. Hence we first compute G^+ in linear time. Simple estimates on the basis of Euler's formula show that in going from G to G^+ at most constant times $|B|$ new vertices have been introduced. Our analysis leads to a constant not larger than 13. Hence, G^+ has $n^+ \in O(n)$ vertices and since G^+ is a maximal plane bus graph we can use the $O(n^{3/2})$ time algorithm from

Theorem 1 to check whether G^+ has a good partition. The algorithm returns a good partition if it exists. Notice that we have to remove all connector vertices of degree 2 from G^+ to match the requirement of Section 3. □

6 Embedding Missing: SPQR-Trees

A planar bus graph can have many planar embeddings. Some of them may allow a planar realization of this bus graph and some may not. From Theorem 3 we know that a plane embedding has a planar realization if and only if the plane embedding admits a good partition. Therefore, we now face the problem of deciding whether a planar bus graph has a plane embedding that admits a good partition.

If the input graph has cut vertices we look at the blocks separately. In Subsection 7.3 we give the details on how to merge solutions for the blocks. The main problem is how to handle each block:

Problem 1 *Given a planar biconnected bus graph find a planar embedding that admits a good partition if such an embedding exists.*

To tackle this problem we use SPQR-trees, a tool developed by Di Battista and Tamassia to describe all combinatorially different planar embeddings of a biconnected graph in one structure. Important references about SPQR-trees are [7, 8, 15]. As sources about SPQR-trees we also used the Wikipedia article and the brief description from [10].

An SPQR-tree for a graph G is a tree $\mathcal{T}$ in which each node $n \in \mathcal{T}$ represents a graph G_n. The vertex set of G_n is a subset of the vertices of G. Some edges of G_n are labeled as *virtual*, while the others are *real*. Removing the virtual edges from G_n yields a subgraph of G and each edge of G appears as a real edge in exactly one of the graphs G_n. If (n, n') is an edge of the SPQR-tree, then G_n and $G_{n'}$ share exactly one edge which is virtual in both; this virtual edge is associated with the tree edge. Each virtual edge of G_n is associated with exactly one tree edge. The given graph G can, hence, be reconstructed by repeatedly taking the union of graphs G_n and $G_{n'}$ belonging to adjacent tree-nodes and deleting the virtual edge associated with the tree-edge (n, n').

The nodes of an SPQR-tree have three types[3]:

[**S**] If n is a S node, then G_n is a cycle of length at least three (the S represents "series").

[**P**] If n is a P node, then G_n is a multigraph with two vertices and three or more edges (the P represents "parallel").

[**R**] If n is a R node, then G_n is a 3-connected graph with more than three vertices (the R represents "rigid").

If we require that no two S nodes and no two P nodes are adjacent in $\mathcal{T}$, then the SPQR-tree of G is unique. Moreover the size of $\mathcal{T}$ is linear in the size of G. Algorithmically the SPQR-tree of a graph can be constructed in linear time.

We continue with some definitions for rooted SPQR-trees. A *rooted SPQR-tree* is an SPQR-tree with a designated root node r, we denote the tree by $\mathcal{T}_r$. Since every node $n \neq r$ has a unique parent n^+ with respect to the root r, there is also a special virtual edge in G_n, which is the virtual edge associated with (n, n^+). In the sequel we will refer to this virtual edge as *the virtual up-edge* of G_n. The two end-vertices of the virtual up-edge are the *poles* of G_n. An example of a rooted SPQR-tree for a planar bus graph is shown in Figure 7.

[3] Some descriptions also use type Q nodes, so that SPQR are the possible types.

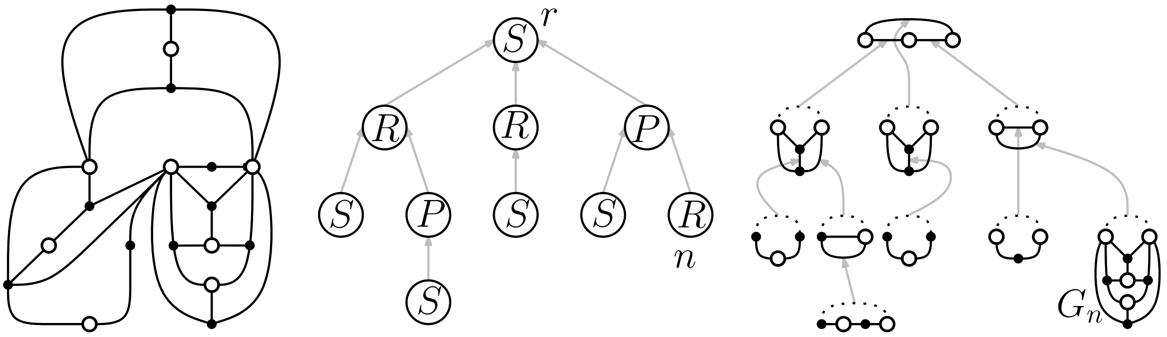

Figure 7: A plane bus graph G with a rooted SPQR-tree $\mathcal{T}$ and the associated graphs for each node. In $\mathcal{T}$ one node r is designated to be the root. The associated graph G_n of each node $n \neq r$ is drawn with one of its feasible embeddings, while the virtual up-edge of n is the dotted edge.

For $n \neq r$ we arbitrarily choose and fix an orientation of the virtual up-edge of G_n and consider the set $G_n^1, \dots, G_n^k$ of all embeddings of G_n that have the outer face to the left of this oriented virtual up-edge. This set will be called the *set of feasible embeddings.* For S nodes we have $k = 1$, for R nodes $k = 2$, and for a P node with $\ell + 1$ parallel edges we have $k = \ell!$.

Lemma 3 *Let G be a biconnected plane graph with SPQR-tree $\mathcal{T}$. Consider the plane graphs G_n associated to the nodes of $\mathcal{T}$, where the plane embedding of G_n is the one induced by the plane embedding of G. All but at most one of these plane graphs G_n have a virtual edge on the outer face.*

Proof. Each of the plane graphs G_n has an outer face. There is at most one node r such that the outer face of G_r contains no vertex from the set $V(G) \setminus V(G_r)$. Now let n be a node with $n \neq r$ and consider a vertex v_r in the outer face. In G there are two disjoint paths from v_r to G_n. These paths can be traced through $\mathcal{T}$ where they both reach n across the same tree edge (n', n). The virtual edge associated with (n', n) is on the outer face of G_n. □

A planar graph G may have exponentially many embeddings. Lemma 3 will be useful to bound the set of embeddings that have to be explored by the algorithm. Indeed, if we correctly guess the root, then we can restrict attention to the set of feasible embeddings of each G_n, since Lemma 3 leads to the implication: if n is not a root of $\mathcal{T}$, then the induced plane embedding of G_n must have the virtual up-edge on the outer face.

In the next section we make some simplifying assumptions and describe the key ideas for the algorithm in the simplified setting.

7 The Algorithm

In the first subsection 7.1 we describe the *basic algorithm* for solving Problem 1 with simplifying assumptions about the input graph G and its SPQR-tree $\mathcal{T}$.

- At every node n of $\mathcal{T}$ all virtual edges of G_n are only incident to bus vertices.
- The input graph G is biconnected.

In Subsection 7.2 we add the details for dealing with virtual edges incident to connector vertices. Finally, in Subsection 7.3 we remove the assumption that the input graph is biconnected.

7.1 The basic algorithm

We are going to work with rooted SPQR-trees but since we do not know which root is a good one we initialize the set R^* of potential roots with the set of all nodes of $\mathcal{T}$. Then we guess a root node $r \in R^*$ and let $\mathcal{T}_r$ be the SPQR-tree with root r.

The tree $\mathcal{T}_r$ will be traversed bottom-up. In the traversal each node has to be processed. However, n can only be processed if all its child nodes have been processed before. To control this condition we use the notion of *eligibility.* At the very beginning all leaf nodes of $\mathcal{T}_r$ are eligible. During the execution of the algorithm a node n of $\mathcal{T}_r$ becomes eligible as soon as all its children have been processed.

When a node n is being processed we want to decide whether the graph G^*_n represented by the subtree $\mathcal{T}_r(n)$ of $\mathcal{T}_r$ rooted at n admits a good partition. (The graph G^*_n is known as the *expansion graph* of n with respect to r, see e.g. [14]). Note that G^*_n still contains the virtual up-edge so it is not even a true bus graph. Moreover G^*_n may have many embeddings. To get to a situation where we can make use of Theorem 3 we substitute all virtual edges including the virtual up-edge with appropriate gadgets.

Actually, we need to know whether there is a good partition with the two poles *in the same class* and whether there is a good partition with the two poles *in different classes.* The information is then passed on from n to the parent node $p(n)$ as a letter σ_n from the set $\{s, d, w\}$ where the meaning is $s =$*same*, $d =$*different*, and $w =$*whatever.*

Moreover, the component may contain a path of length two between the poles where the intermediate connector vertex has degree three or four. Such a 2-path has the potential of being combined with a 2-path from the father component to form a diamond. The potential of forming a diamond on one side or on both sides is passed on from n to n^+ as a number $\tau \in \{0, 1, 2\}$. If replacing the virtual up-edge by a component containing a 2-path may not create an extra diamond, then $\tau = 0$. If such a replacement may generate a diamond but replacing virtual up-edge with the reflected component does not create an extra diamond, then $\tau = 1$. If by adding the component it is unavoidable to create an extra diamond, then $\tau = 2$.

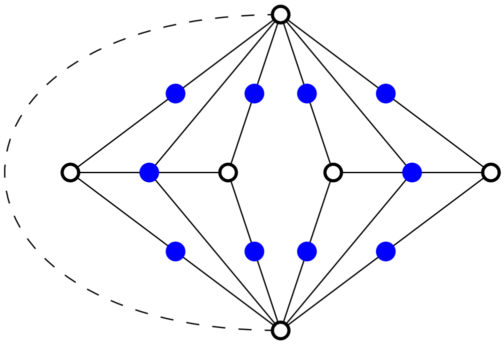

Figure 8: 2 components with $\tau = 2$ imply the creation of a new diamond.

As an example for the use of this data suppose that $(\sigma_n, \tau_n) = (d, 2)$. If $G_{p(n)}$ has a 2-path between the two poles, i.e., between the vertices of the virtual edge of n in $G_{p(n)}$, such that the middle vertex has a neighbor in the 4-cycle defined by the two 2-paths, there is no good partition because the new diamond would force the poles to have the same label but G^*_n forces them to have different labels. If $(\sigma_n, \tau_n) = (d, 1)$ and $G_{p(n)}$ has only one 2-path between the two poles, then the embedding of G_n in $G^*_{p(n)}$ can be reflected to avoid the creation of a diamond.

Depending on the structure of 2-paths in $G_{p(n)}$ the pair (σ_n, τ_n) is condensed into a single letter information $\sigma'_n \in \{s, d, w\}$. This condensation has to be done differently depending on the type of the node n and will be described in detail when we deal with these types.

Figure 9 shows the gadgets that depending on σ'_n are used to replace virtual edges of a *feasible embedding* of G_n.

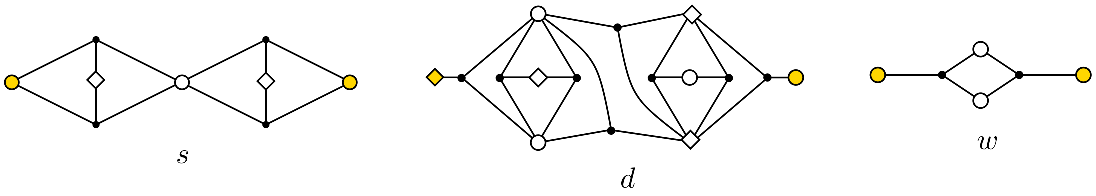

Figure 9: The gadget replacing the virtual edges of G_n. The colored bus vertices are the vertices of the virtual edge. In the s and d gadgets the shapes of the bus vertices represent the two classes of the unique good partition.

(The definition of a feasible embedding was given before Lemma 3). The gadget for the case s is a concatenation of two simple diamonds and forces the two colored bus vertices to belong to the same class (Lemma 1). The gadget for the case d also has two diamonds. The key is that the outer bus vertices of the two diamonds belong to different classes. The gadget for the case w allows any assignment of the colored bus vertices to the classes. Note that all paths between the poles of the gadgets have lengths at least 3, hence they will not be part of potentially new diamonds.

We now describe our algorithm.

Processing an eligible node *n*:

(1) Choose a feasible embedding of G_n.

(2) For each virtual edge consider the set of 2-paths between the endpoints of the virtual edge and compute the condensed letter σ' on the basis of the value of (σ, τ). If it turns out that there is no good partition, then break and call for a new root.

(3) Replace the virtual edges different from the virtual up-edge by the gadgets corresponding to their condensed letter. This results in a graph G_n^+

(4) Run the algorithm to compute a good partition twice: first with the virtual up-edge of G_n^+ replaced by the s gadget and secondly replaced by the d gadget. Define $\sigma_n = s$ if only the first test is positive, and $\sigma_n = d$ if only the latter case is positive, and $\sigma_n = w$ if both tests are positive.

(5) If there was a positive test, pass σ_n to the parent. Otherwise call for a new root.

We now come to a more detailed look at the parts of this algorithm. Depending on the type of the node n in the SPQR-tree the number of feasible embeddings of G_n varies.

***n* is an S node.** This case is easy. There are no 2-paths so that the condensed letter equals the original letter for each virtual edge and $\tau_n = 0$. Moreover, the letter σ_n that has to be passed to the parent of an S node n can be computed directly from the letters obtained from the children. If one of the children submitted a w, then $\sigma_n = w$. Otherwise it depends on the parity of the number of children that submitted a d. If this number is even, then $\sigma_n = s$, otherwise $\sigma_n = d$.

***n* is an R node.** In this case, we have two feasible embeddings G_n^1 and G_n^2 where G_n^2 is obtained from G_n^1 by a reflection of the component minus the virtual edge, i.e., the component stays on the same side of the virtual edge. Before showing that the two embeddings are equivalent with respect to the value of σ_n we have to discuss how to compute the condensed letters for the child components.

Let n' be a child of n. For the condensation we distinguish between *no extra diamond*, in this case $\sigma'_{n'} = \sigma_{n'}$, and *extra diamond*, in this case if $\sigma_{n'} = d$, then there is no good partition, and if $\sigma_{n'} \in \{s, w\}$, then $\sigma'_{n'} = s$. Now we use $\tau_{n'}$ to decide between these cases. If $\tau_{n'} = 0$, then there is no extra diamond. If $\tau_{n'} = 1$, then an extra diamond must be created if there are appropriate 2-paths of G^1_n that connect the poles of n' on both sides of the virtual edge of n'. If $\tau_{n'} = 2$, a single appropriate 2-path of G^1_n forces an extra diamond.

Lemma 4 *$G^{1,+}_n$ and $G^{2,+}_n$ admit the same good partitions. Moreover, the letter σ_n computed on the basis of $G^{1,+}_n$ reflects the behavior of the two poles in good partitions of G^*_n.*

Proof. Since property (P1) is independent from an embedding, $G^{1,+}_n$ and $G^{2,+}_n$ are equivalent with respect to (P1). Properties (P2) and (P3) are invariant under reflections and independent from the virtual edge, hence $G^{1,+}_n$ and $G^{2,+}_n$ are also equivalent with respect to (P2) and (P3).

Now let us look at the behavior of the poles in good partitions of $G^{1,+}_n$ and G^*_n. We assume that for all child nodes n' of n the statement is true, i.e., $\sigma_{n'}$ reflects the behavior of the two poles in good partitions of $G^*_{n'}$. First note that the gadgets have no paths of length 2 between their poles. Therefore, all diamonds of $G^{1,+}_n$ are either diamonds of G^1_n or diamonds inside of gadgets. Diamonds of G^*_n that are formed by a 2-path in G^1_n and a 2-path in the child component are accounted for when computing the condensed letter for the child component. Therefore, replacing the virtual edge of a child by the gadget corresponding to the condensed letter induces label restrictions on the poles of the child in G^1_n which reflect the options of these labels in G^*_n. This shows that the letter σ_n computed on the basis of $G^{1,+}_n$ in step (4) of the procedure reflects the behavior of the two poles in good partitions of G^*_n. □

***n* is an P node.** In this case, we have many feasible embeddings. Let $\Sigma_n = \{\sigma_{n'} : n' \text{ child of } n\}$.

If $\{s, d\} \subseteq \Sigma_n$, then the requirements of two children are contradicting each other so that there is no good partition of G^*_n. If $s \in \Sigma_n \subseteq \{s, w\}$, then there is a child n' requiring that the labels of the two poles are the same and this choice is consistent with all the other children. The choice of s is also compatible with additional diamonds that are formed by 2-paths from different components. Hence, $\sigma_n = s$ in this case.

In the remaining cases d is a valid option for all the children and we have to check whether d is also valid for n. The only obstruction would be a diamond formed by 2-paths from different components. The P node gives us the freedom of rearranging the order of the child components. We order them by increasing value of τ. If two of the components have $\tau = 2$ the creation of an extra diamond is unavoidable and d not a valid option. Otherwise we can use the freedom of reflecting child components with $\tau = 1$ so that each of them would only form an extra diamond with a 2-path further left whose central vertex has a neighbor to the right. This is avoided by our choice of reflections. Based on these considerations we can determine σ_n and of course $\tau_n = \max(\tau_{n'} : n' \text{ child of } n)$.

Note that it follows that for P nodes we can skip the run of the procedure with gadgets replacing the virtual edges and directly compute σ_n and τ_n on the basis of the settings for the child nodes.

The case of no good partition. It remains to discuss what has to be done when the computation shows that there is no good partition for G^*_n with the chosen embedding, i.e., the case where the procedure asks for a new root . If there is an embedding of G that admits a good

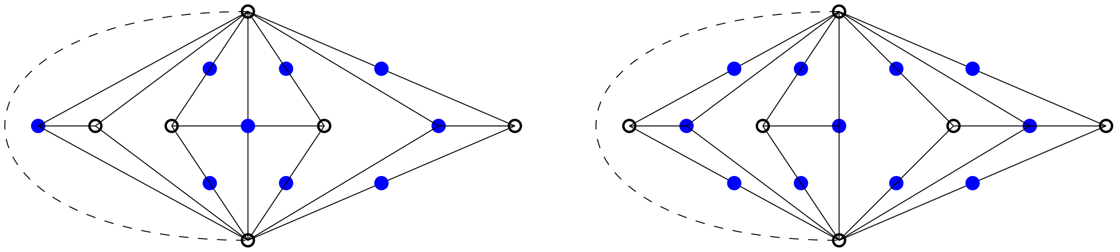

Figure 10: Reordering components with $\tau = 2$ and $\tau = 1$. The embedding at the left had to be changed.

partition, then either n is the root or the virtual up-edge of node n has to be a different one. This implies that the root node for such an embedding belongs to the subtree $\mathcal{T}_r(n)$ of $\mathcal{T}_r$ rooted at n. Therefore, the set R^* of potential roots is redefined to be $R^* \cap V(\mathcal{T}_r(n))$ and a new tentative root r is chosen from the new set R^*.

If R^* becomes empty, then there is no good partition. Otherwise the root r of the tree becomes eligible. Processing the root node is similar to processing any other node. The difference is in the choice of the embedding. In the case of an S node there is only one embedding. In the case of a P node we can argue as before that the embedding does not matter. The interesting case is when the root is an R node. An R node is a 3-connected component and by Whitney's theorem specifying the outer face determines the embedding. Therefore, we can afford to check independently for each embedding.

We have completed the description of the algorithm and conclude with the following lemma.

Lemma 5 *For a given planar biconnected bus graph G with the property that all the poles in the SPQR-Tree are bus vertices, the basic algorithm computes a good partition, if there is any.*

Now we come to the discussion of the running time. Due to the change of root a node n may have to be processed several times. We will see that our algorithm has a drastically reduced running time compared to the straight-forward approach, consisting of testing every node n of $\mathcal{T}$ as root.

Lemma 6 *The number of calls to the procedure that processes the nodes is at most $2|V(\mathcal{T})|$.*

Proof. Consider a tree edge (n_1, n_2). If the root is on the side of n_2 after processing n_1, the information σ_{n_1} is passed to n_2. If the root is on the side of n_1 after processing n_2, the information σ_{n_2} is passed to n_1. This is all the exchange along this edge. Since each processing of a node is followed by passing information over a tree edge, the number of calls to the procedure that processes the nodes is bounded by twice the number of edges of $\mathcal{T}$. □

If the input graph has n vertices, then in linear time we get an SPQR-tree $\mathcal{T}$ whose size is linear in n. Processing a node can be done in $O(n^{3/2})$ time (Theorem 4). The root has at most $2n$ faces, hence processing the root can be done in $O(n^{5/2})$ time. Actually we may have to test more than one root but if $f_1, f_2, \ldots, f_k$ are the numbers of faces of k roots that are checked, then still $\sum f_i \in O(n)$.

Proposition 3 *If a given planar biconnected bus graph G has the property that all the poles in the SPQR-tree are bus vertices, then we can solve Problem 1, i.e., decide whether G admits a good partition in $O(n^{5/2})$ time.*

7.2 Connector vertices as poles

In this subsection we discuss the details for dealing with virtual edges incident to connector vertices, i.e., with situations where at least one of the poles of a tree node n is a connector vertex. The overall idea will be to reduce -if possible- the scenario with connector vertices as poles to the previous scenario with bus vertices as discussed in the previous subsection. This discussion is necessary due to the fact that the order of the neighbors of connector vertices might be important. We will explore two cases when (A) a connector vertex pole has degree 3 and (B) a connector vertex pole has degree 4. Both cases include several subcases depending on the type of node n and on the number of bus vertices adjacent to the connector vertex in pole G_n. The discussion will consider all special cases which do not appear in the simpler case when the poles are just bus vertices. In particular many of the embedding choices are performed when poles are connector vertices. For the case distinction we need the following definition. The neighbors of a connector vertex v in G are called *v-exposed*, note that exposed vertices are bus vertices.

Let (n, n') be an edge in the SPQR-tree $\mathcal{T}$ and let (v, u) be the corresponding virtual edge. We assume that v is a connector vertex in G_n and $G_{n'}$ and if u is also a connector vertex then $\deg(u) \leq \deg(v)$.

We define a so-called *move-operation* as follows: In the case that b is the only v-exposed neighbor in $G_{n'}$, we move edge (v, b) from $G_{n'}$ to G_n such that the new virtual edge corresponding to the tree edge (n, n') is (b, u), cf. Figure 11 for a special case. By this operation, the separating set has one connector pole less than before; we have simplified our scenario.

There are several cases that are treated differently. First we consider connector poles of degree three.

(A) The degree of v is 3. There are two subcases depending whether at least one of n and n' is a P node or none of them is. During the case distinction we disregard whether n' is parent of n or not. For both subcases we discuss the case that n as root at the end.

Neither n nor n' is a P node. The type of one of n and n' has to be R. Assuming that n is of type R we know that G_n contains two v-exposed bus vertices. Let b be the exposed neighbor of v in $G_{n'}$. Since $\{b, u\}$ is a separating set of size two we can conclude that n' is a node of type S and (v, b) is one of the edges of the cycle $G_{n'}$.

We apply the above defined move-operation, cf. Figure 11. Afterwards, we treat n and n' as we treat R and S nodes in the base algorithm, since their new poles with respect to (n, n') are only bus vertices.

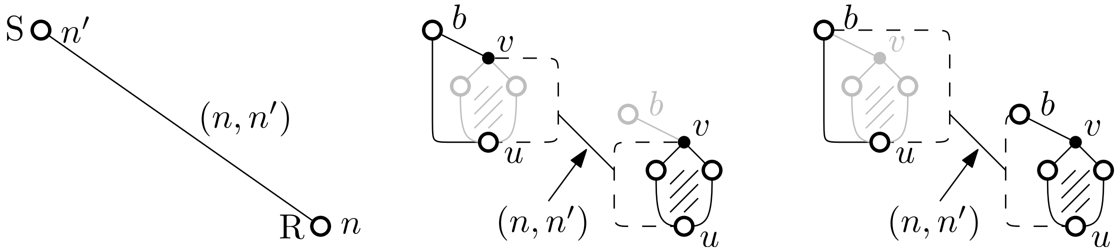

Figure 11: The movement of the virtual edge when n is an R node and n' an S node.

One of n, n' is P node, w.l.o.g. n is P node. In $G_{n'}$ there is only one v-exposed vertex. If there were two, then the third exposed neighbor b and the separator $\{b, u\}$ would reveal that n is of type S. Hence, n has three neighbors n_1, n_2, and n_3 in the SPQR-tree. Let b_i be the neighbor of v in n_i. Since $\{b_i, u\}$ is separating each n_i is of type S.

Let b'_i be the bus vertex closest to u on G_{n_i}, i.e, either $b'_i = u$ or it is the unique u-exposed vertex (recall $\deg(u) \leq \deg(v)$). The relevant information needed from G_{n_i} is the classifying letter σ_{n_i} for the pair b_i, b'_i. This is obtained by running the procedure for eligible nodes.

When n becomes eligible and n is the root, then we can use one of the standard gadgets from Figure 9 for each child together with v and u to check the existence of a good partition (this can even be decided just on the basis of the three letters).

When n becomes eligible and n is not the root, then we skip n, however, when it comes to processing the parent n_1 (a node of type S) we use two of the standard gadgets, one for each of n_2 and n_3 together with v and u. Figure 12 indicates how this is done.

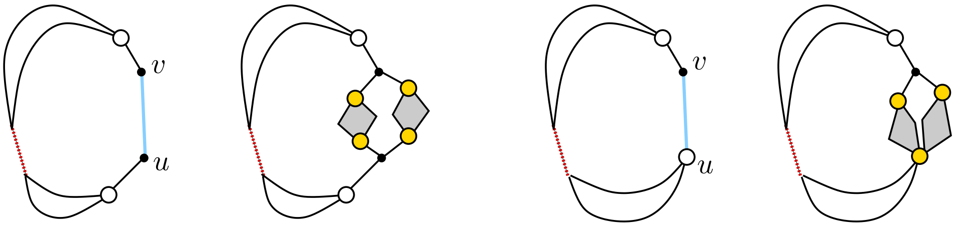

Figure 12: Replacing the virtual edge in G_{n_1} by two gadgets. The left figure covers the case in which v and u both are connector vertices, while at the right, we show the case when v is connector and u is bus vertex. The red dashed edge is the virtual up-edge of n_1.

(B) The degree of v is 4. The critical issue in this case is that we have to care of property (P2) of a good partition, i.e., the cyclic alternation of labels around v.

In the basic case, where poles are bus vertices, see Subsection 7.1 and in the case where poles are connector vertices of degree 3 the existence of a good partition for some embedding of G^*_n could be checked by replacing virtual edges of G_n with some simple gadgets that represent conditions enforced by the descendant nodes and then check for a good partition of the new graph G^+_n.

The present case is more complicated, since now the embedding is more subtle as before. The next example illustrates this fact: Let u be a bus node, let b_1, b_2, b_3 and b_4 be v-exposed, and assume that each $\{b_i, u\}$ is separating. The graph G_{n_i} corresponding to the neighbor n_i of n with separator $\{b_i, u\}$ has two bus vertices as poles, hence, there is a letter $\sigma_i = \sigma_{n_i}$ encoding conditions on appropriate embeddings (for admitting a good partition). Now suppose that $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) = (s, s, d, d)$, then there is a good partition if and only if the G_{n_i} are arranged in the embedding so that the letters alternate between s and d.

We go through several subcases depending on the number of v-exposed vertices in G_n disregarding the type of n (only at the end, we need some extra considerations for the case where n is a P node). Note that if n is not the root v is incident to precisely one virtual edge in G^+_n, since when considering node n all children have been processed before. As in case (A) we disregard whether n' is parent of n or not. For all subcases we discuss the case 'n is root' at the end.

Node n contains 3 v-exposed vertices. Consider the fourth v-exposed vertex b together with u to conclude that n' is a node of type S and (v, b) is one of the edges of the cycle $G_{n'}$. We deal with this case by moving the edge (v, b) from $G_{n'}$ to G_n so that the new virtual edge corresponding to the tree edge (n, n') is (b, u), cf. Figure 11.

Node n contains 1 v-exposed vertex. Let b be this v-exposed vertex together with u to conclude that n is a node of type S and (v, b) is one of the edges of the cycle G_n. We deal with this case by moving the edge (v, b) from G_n to $G_{n'}$ so that the new virtual edge corresponding to the tree edge (n, n') is (b, u), cf. Figure 11 with interchanged roles of n and n'.

Node n contains 2 v-exposed vertices. We consider two further subcases depending on whether u is a bus vertex or not.

Vertex u is a bus vertex. Suppose that G_n contains two v-exposed vertices b_1, b_2. In this case it is necessary that there is a good partition of G_n^* where b_1 and b_2 have different labels. This can be checked when processing node n by inserting a d gadget between b_1 and b_2. If such a test fails, then we have to try with a new root from the subtree rooted at n.

Otherwise G_n^* can be reflected, therefore the alternation condition (P2) for v can be satisfied as soon as the remaining two v-exposed vertices are labeled differently. In the parent node of n it is then enough to replace the virtual edge (v, u) with the gadget shown in the left part of Figure 13. The w gadget encodes the fact that we can use reflections.

Vertex u is a connector vertex. In this case node n contains 2 u-exposed vertices (otherwise we would have applied one of the previously described cases with u playing the role of v). Let b_1, b_2 be the v-exposed vertices and b'_1, b'_2 be the u-exposed vertices of G_n. Insert d gadgets between the pairs b_1, b_2 and b'_1, b'_2. Then check the existence of a good partition twice, first with b_1, b'_1 connected by an s gadget and then by a d gadget. If both fail we have to try with a new root, otherwise let $\sigma_n \in \{s, d, w\}$ be the outcome of the tests.

In the parent node of n it is sufficient to replace the virtual edge (v, u) with the gadget shown in the right part of Figure 13. Since the degree of v and u remains unaffected the alternation condition (P2) is ensured.

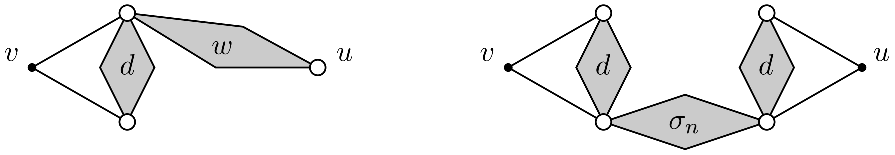

Figure 13: Replacement for a node n with 2 v-exposed vertices and 2 u-exposed vertices.

When n becomes eligible and n is the root, then we just use one of the standard gadgets from Figure 9 for each child together with v and u to check the existence of a good partition.

In the case where n becomes eligible and n is not the root, then there is nothing else to do.

In the above cases for n with 3 v-exposed vertices and with 1 v-exposed vertex, the insertion of gadgets was not necessary, while in the case of 2 v-exposed vertices, it was necessary. Nevertheless in all cases, we have disregarded the type of n. If n is of type S or R, we treat n as in the base algorithm, while if n is of type P, we have to consider property (P2). Therefore we consider this as the final case, since n needs an additional treatment.

Node n is a P node.

Consider the vertex v of degree four. We distinguish two cases: Either v was a pole before we applied the move-operation to the virtual edge, or in the case, that we did not apply the move-operation, v is still a pole and case "‘Node n contains 2 v-exposed vertices"’ applies.

Suppose first that n is the root. Then there are at most four components providing information that can be captured by one of the gadgets from Figure 9 or 13 and thus at most six possible orders are possible to arrange the components cyclically around v in G_n. It is an easy combinatorial sorting problem to find a good partition if it exists.

If n is an inner node then there are at most three children providing information that can be captured by appropriate gadgets. Again the provided information from the children can be captured by one of the gadgets from Figure 9 or 13. We skip considering n and move the gadgets to the parent n' of n, where we have again at most six possible orders of these components cyclically arranged around v. In $G_{n'}$ it is the same combinatorial sorting problem (as if n was root) to find a good partition if it exists.

All in all we have to check a constant number of sortings, i.e. different embeddings, and the result can be encoded by a gadget in the parent node.

Lemma 7 *Let G^*_n be the bus graph represented by $\mathcal{T}_r(n)$ computed according to the above cases. If G admits a plane embedding with a good partition, then G^*_n has a good partition.*

Proof. We consider a connector vertex v and distinguish the cases whether v is a pole or not. In the latter case v is a vertex of G_n for a unique node n in $\mathcal{T}$ with no incident virtual edge. Thus properties (P1), (P2), (P3) are satisfied by Theorem 4 which will be used as black-box. Consider the pole vertex v of degree 3. After the movement of the virtual edge v becomes a vertex in a unique G_n without incident virtual edges and thus again Theorem 4 ensures properties (P1), (P3), while (P2) is irrelevant for v. Consider the pole vertex v of degree 4. We aim at ensuring property (P2), since (P1) is irrelevant and diamonds are handled in the calls of Theorem 3 for the graphs G^+_m associated to tree nodes m, i.e., (P3) is already ensured.

Since v is a pole it belongs to at least two components. Suppose v is in precisely two components of vertices n', n in $\mathcal{T}$. Then n', n are adjacent and of types S,R or R,S or R,R. The first case is covered by “n has 3 v-exposed vertices“ (and n' has 1 v-exposed vertex), the second case is covered by ”n has 1 v-exposed vertex” (and n' has 3 v-exposed vertices), and the last case is covered by “n has 2 v-exposed vertices“ (the 2 v-exposed vertices in n' are fixed due to the embedding and (P2) is ensured due to the black-box (Thm.4) after inserting a gadget for the 2 v-exposed vertices of n).

Suppose v is in more than two components of vertices in $\mathcal{T}$. Then there is exactly one P node n with pole v. The adjacent nodes $n_1, \dots, n_k$ are of type S or R. Note that k may have values $k = 4$ or $k = 3$ ($k \geq 3$ because of definition of P node and $k \leq 4$ because of the degree of v).

If $k = 4$, then $n_1, \dots, n_4$ all have the same type S. This case is covered, when 3 of the nodes were processed and n becomes eligible, i.e. then n contains 3 v-exposed vertices together with the fourth v-exposed vertex due to the movement of the virtual edge. Still n is a P node and (P2) is ensured after finding a feasible sorting of $n_1, \dots, n_4$ in the case ”Node n is a P node”.

If $k = 3$, then let w.l.o.g. n_1 be an R node and let n_2, n_3 be S nodes.

In the case that n has no parent, i.e. n is the root, we first move the virtual edges of n_2, n_3 and then arrange the d gadget from n_1 together with the remaining 2 v-exposed vertices from n_2, n_3. Applying the case “Node n is a P node” we solve the combinatorial problem for the feasible sorting of the v-exposed vertices around v.

Assume n has a parent n_i; we consider the cases $i = 1$ and $i \neq 1$. If $i = 1$, i.e. n_1 is the parent of n, then n_2, n_3 were processed and n becomes eligible with the 2 v-exposed vertices from n_2, n_3 due to the movement of the virtual edge. These 2 v-exposed vertices are linked by a d gadget from Figure 13 which belongs to n_1. In n_1 (P2) is ensured by Theorem 4.

Otherwise if $i \neq 1$, then $i = 2$ or $i = 3$. n_1 and one of n_2, n_3 are processed and n becomes eligible with 3 v-exposed vertices, while two of them (those from n_1) are linked by a d gadget from Figure 13. Since the parent n_i is an S node, its v-exposed vertex is assigned to n due to the movement of the virtual edge. It is again just a combinatorial problem to find a feasible sorting of the v-exposed vertices ensuring (P2), which is covered in “Node n is a P node”.

In the end we considered all possible cases for a connector vertex pole. □

We conclude with

Proposition 4 *Problem 1 can be solved in $O(n^{5/2})$ time, i.e. a good partition for a planar biconnected bus graph can be computed efficiently if it exists.*

7.3 Not biconnected input graphs

Now suppose that the input graph G is not biconnected. We use the *block-cutpoint-tree* of G, i.e., the tree that represents the maximal biconnected components of G, see [17]. We assume that the biconnected components are already processed as described in Subsections 7.1 and 7.2.

Let G_1 and G_2 be two connected bus graphs, possibly with connector vertices of degree one and without fixed embedding. For vertices a_1 of G_1 and a_2 of G_2 that are both bus or both connector vertices, let $G_1\ {}_{a_1}\!\oplus_{a_2} G_2$ denote the graph obtained by gluing G_1 and G_2 together by identifying a_1 and a_2. G is the result of this operation. We summarize the first few cases in the following lemma, and discuss the missing case afterwards:

Lemma 8 *Let G_1 and G_2 be two connected bus graphs, both having a good partition regarding embeddings with G_1 having a_1 on the outer face or G_2 having a_2 on the outer face. If a_1, a_2 are connector vertices with $\deg_{G_1}(a_1) + \deg_{G_2}(a_2) \leq 3$ or a_1, a_2 are bus vertices, then the graph $G_1\ {}_{a_1}\!\oplus_{a_2} G_2$ also has a good partition.*

Proof. Compute good partitions of G_1 and G_2 with corresponding embeddings as required. Either they combine to a good partition of $G_1\ {}_{a_1}\!\oplus_{a_2} G_2$ or exchanging all labels in G_2 yields a labeling that can be combined with the labeling of G_1. □

The remaining case where $G = G_1\ {}_{a_1}\!\oplus_{a_2} G_2$ is a bus graph is when a_1, a_2 are connector vertices with $\deg_{G_1}(a_1) + \deg_{G_2}(a_2) = 4$.

If $\deg_{G_1}(a_1) = 2 = \deg_{G_2}(a_2)$, then a good partition of G exists if and only if each of G_1 and G_2 has a good partition and the neighbors of a_1 in G_1 and the neighbors of a_2 in G_2 are labeled differently. This can be checked by inserting d gadgets before computing good partitions.

If $\deg_{G_1}(a_1) = 3$ and $\deg_{G_2}(a_2) = 1$, then it is again sufficient to have good partitions for G_1 and G_2 separately. Condition (P1) at a_1 requires that both labels appear in the neighborhood of a_1. Embedding G_2 in the face with the two identical labels allows to satisfy (P2) in graph G.

The results are summarized in the following theorem, while a planar embedding will be produced as a byproduct of a good partition if it exists.

Theorem 5 *A good partition for a planar bus graph can be computed in $O(n^{5/2})$ time if it exists.*

8 Conclusion and Future Work

We have considered the class of planar bus graphs that admit a planar realization and have characterized this class by the existence of a good partition of the bus vertices. To test for the existence of a good partition we gave an $O(n^{5/2})$ algorithm based on planar matching and SPQR-trees. Given a good partition the representation can be computed in linear time.

It is still open to characterize the class of bus graphs that admit realizations, where connections are allowed to cross (C_1-realizations) apart from the knowledge that the decision problem is NP-complete.

Another generalization would be to allow connections to cross bus segments (C_2-realizations) or bus segments to cross each other (C_3-realizations). Apart from the question if the decision problem is NP-complete, we could ask here for a given C_j-representation, if there exists a C_i-representation for $i < j$.

Furthermore it is interesting to consider the realizability question with generalized bus graphs, i.e. bipartite graphs where the bus vertices have degree at most six or eight, regarding a realization with vertical and diagonal segments, or with vertical, horizontal and diagonal segments, respectively.

9 Acknowledgments

We thank the anonymous reviewers for their valuable comments and their persistence. With their help the quality of the paper has been improved significantly.

References

[1] A. Ada, M. Coggan, P. D. Marco, A. Doyon, L. Flookes, S. Heilala, E. Kim, J. L. O. Wing, L.-F. Préville-Ratelle, S. Whitesides, and N. Yu, *On bus graph realizability*, in Proc. CCCG, 2007, 229–232. `arXiv:0609127`.

[2] T. C. Biedl, P. Bose, E. D. Demaine, and A. Lubiw, *Efficient algorithms for Petersen's matching theorem*, J. Algorithms, 38 (2001), 110–134.

[3] H. De Fraysseix and P. O. De Mendez, *On topological aspects of orientations*, Discr. Math., 229 (2001), 57–72.

[4] H. de Fraysseix, P. O. de Mendez, and J. Pach, *Representations of planar graphs by segments*, in Intuitive geometry (Szeged, 1991), Colloq. Math. Soc. János Bolyai vol. 63, 1994, 109–117.

[5] H. de Fraysseix, P. Ossona de Mendez, and J. Pach, *A left-first search algorithm for planar graphs*, Discr. and Comput. Geom., 13 (1995), 459–468.

[6] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall, 1999.

[7] G. Di Battista and R. Tamassia, *Incremental planarity testing (extended abstract)*, in Proc. Found. of Comp. Sci., 1989, 436–441.

[8] G. Di Battista and R. Tamassia, *On-line maintenance of triconnected components with SPQR-trees*, Algorithmica, 15 (1996), 302–318.

[9] M. Dickerson, D. Eppstein, M. T. Goodrich, and J. Y. Meng, *Confluent drawings: Visualizing non-planar diagrams in a planar way*, J. Graph Alg. and Appl., 9 (2005), 31–52.

[10] D. Eppstein, *Planar Lombardi drawings for subcubic graphs*, in Proc. Graph Drawing, vol. 7704 of Lect. Notes Comp. Sci., Springer, 2012, 126–137.

[11] S. Felsner, *Rectangle and square representations of planar graphs*, in Thirty Essays in Geometric Graph Theory, J. Pach, ed., vol. 29 of Algorithms and Combinatorics, Springer Verlag, 2012, 213–248.

[12] S. Felsner, C. Huemer, S. Kappes, and D. Orden, *Binary labelings for plane quadrangulations and their relatives*, Discr. Math. and Theor. Comp. Sci., 12 (2010), 115–138.

[13] E. R. Gansner and Y. Koren, *Improved circular layouts*, in Proc. Graph Drawing, vol. 4372 of Lect. Notes Comp. Sci., Springer, 2006, 386–398.

[14] C. Gutwenger, M. Jünger, S. Leipert, P. Mutzel, M. Percan, and R. Weiskircher, *Subgraph induced planar connectivity augmentation*, in Proc. Workshop Graph-Theoret. Concepts Comput. Sci., vol. 2880 of Lect. Notes Comp. Sci., Springer, 2003, 261–272.

[15] C. Gutwenger and P. Mutzel, *A linear time implementation of spqr-trees*, in Proc. Graph Drawing, vol. 1984 of Lect. Notes Comp. Sci., Springer, 2000, 77–90.

[16] M. Hanan, *On Steiner's problem with rectilinear distance*, SIAM J. Appl. Math, (1966), 255–265.

[17] F. Harary and M. Plummer, *On the core of a graph*, Proc. London Math. Soc., 17 (1967), 249–257.

[18] I. B.-H. Hartman, I. Newman, and R. Ziv, *On grid intersection graphs*, Discr. Math., 97 (1991), 41–52.

[19] X. He, *On finding the rectangular duals of planar triangular graphs*, SIAM J. Comput., 22 (1993), 1218–1226.

[20] F. W. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*, vol. 53 of Annals of Discr. Math., North Holland, 1992.

[21] M. Kaufmann and D. Wagner, eds., *Drawing Graphs, Methods and Models*, vol. 2025 of Lect. Notes Comp. Sci., Springer, 2001.

[22] T. Lengauer, *VLSI theory*, in Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A), Elsevier and MIT Press, 1990, 835–868.

[23] S. Micali and V. V. Vazirani, *An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs*, in Proc. Found. of Comp. Sci., 1980, 17–27.

[24] M. Mucha and P. Sankowski, *Maximum matchings in planar graphs via Gaussian elimination*, Algorithmica, 45 (2006), 3–20.

[25] T. Nishizeki and M. S. Rahman, *Planar Graph Drawing*, World Scientific, 2004.

[26] J. P. C. Petersen, *Die Theorie der regulären Graphen*, Acta Mathematica, (1891), 15:193–220.

[27] P. Rosenstiehl and R. E. Tarjan, *Rectilinear planar layouts and bipolar orientations of planar graphs*, Discr. and Comput. Geom., 1 (1986), 343–353.

[28] R. Tamassia and I. G. Tollis, *A unified approach to visibility representation of planar graphs*, Discr. and Comput. Geom., 1 (1986), 321–341.

[29] C. D. Thompson, *A Complexity Theory for VLSI*, PhD thesis, Carnegie-Mellon Univ., 1980.