

## Planarity Testing and Optimal Edge Insertion with Embedding Constraints

*Carsten Gutwenger   Karsten Klein   Petra Mutzel*

University of Dortmund  
Chair of Algorithm Engineering  
Otto-Hahn-Str. 14, 44227 Dortmund, Germany

<http://ls11-www.cs.uni-dortmund.de/>

[carsten.gutwenger@cs.uni-dortmund.de](mailto:carsten.gutwenger@cs.uni-dortmund.de)   [karsten.klein@cs.uni-dortmund.de](mailto:karsten.klein@cs.uni-dortmund.de)  
[petra.mutzel@cs.uni-dortmund.de](mailto:petra.mutzel@cs.uni-dortmund.de)

### Abstract

The planarization method has proven to be successful in graph drawing. The output, a combinatorial planar embedding of the so-called planarized graph, can be combined with state-of-the-art planar drawing algorithms. However, many practical applications have additional constraints on the drawings that result in restrictions on the set of admissible planar embeddings. In this paper, we consider embedding constraints that restrict the admissible order of incident edges around a vertex. Such constraints occur in applications, e.g., from side or port constraints. We introduce a set of *hierarchical embedding constraints* that include *grouping*, *oriented*, and *mirror* constraints, and show how these constraints can be integrated into the planarization method. For this, we first present a linear time algorithm for testing if a given graph  $G$  is *ec-planar*, i.e., admits a planar embedding satisfying the given embedding constraints. In the case that  $G$  is ec-planar, we provide a linear time algorithm for computing the corresponding *ec-embedding*. Otherwise, an ec-planar subgraph is computed. The critical part is to re-insert the deleted edges subject to the embedding constraints so that the number of crossings is kept small. For this, we present a linear time algorithm which is able to insert an edge into an ec-planar graph  $H$  so that the insertion is crossing minimal among all ec-planar embeddings of  $H$ . As a side result, we characterize the set of all possible ec-planar embeddings using BC- and SPQR-trees.

Article Type	Communicated by	Submitted	Revised
Regular paper	M. Kaufmann and D. Wagner	December 2006	July 2007

## 1 Introduction

In many application domains information visualization is based on graph representations. Examples include software engineering, database modeling, business process modeling, VLSI-design, and bioinformatics. The computation of concise graph layouts by automatic layout systems facilitates the readability and immediate understanding of the displayed information. These layout systems need to take into account application specific as well as user-defined layout rules in addition to the aesthetic criteria they try to optimize. In database diagrams, for example, links between attributes should enter the tables only at the left or right side of the corresponding attributes, the placement of reactants in chemical reactions or biological pathways should reflect their role within the displayed reactions, and in UML class diagrams, generalization edges should leave a class object at the top and enter a base class object at the bottom. Many of these layout rules impose restrictions on the admissible embeddings for a drawing. Even more important is the possibility to use drawing restrictions in order to express the user's preferences and to guide the layout phase. A general survey of constraints in graph drawing algorithms is given in [22].

In this paper, we consider restrictions on the allowed order of incident edges around a vertex, e.g., to specify groups of edges that have to appear consecutively around the vertex or that have a fixed clockwise order in any admissible embedding. Such constraints occur, e.g., in the form of *side constraints*, where incident edges are assigned to the four sides of a rectangular vertex, or *port constraints* where edges have prescribed attachment points at a vertex. In particular, we introduce three types of constraints which may be arbitrarily nested: *grouping*, *oriented* (prescribed clockwise order), and *mirror* constraints (prescribed reversible order). We call a planar embedding that fulfills the given set of constraints an *ec-planar* embedding.

An important optimization goal for the computation of graph layouts is the minimization of crossings. The problem of minimizing the number of crossings in a drawing is NP-hard [14] and no practically efficient method exists so far. In practice, the problem is attacked via the *planarization* approach [1] which first deletes a number of edges until the remaining graph is planar and then carefully reinserts them (iteratively) so that the number of crossings is minimized, see for example [16]. This paper deals with the integration of the embedding constraint concept into the planarization approach. The first step can be solved by successive ec-planarity testing. Our first contribution therefore is a linear time algorithm for testing if a graph with a set of embedding constraints is ec-planar. The main challenge is to incorporate oriented constraints, where a given clockwise order of (groups of) incident edges needs to be satisfied. Furthermore, we characterize all possible ec-planar embeddings using BC- and SPQR-trees, which also yields a linear time algorithm for computing an ec-planar embedding.

The second step of the planarization approach can be tackled by repeatedly solving the *optimal edge insertion problem*: This problem asks for inserting an edge  $e = (v, w)$  into a planar graph so that all crossings involve  $e$  and their number is minimized. Alternatively, the problem can be stated as finding an

embedding of a planar graph  $G$  where the given edge can be inserted with the minimum number of crossings. Recently, the problem has been solved in linear time using the SPQR-tree data structure [17]. The algorithm essentially computes a shortest path  $\Psi$  between those nodes in the SPQR-tree  $\mathcal{T}$  of  $G$  whose skeletons contain  $v$  and  $w$ , respectively. The optimal insertion path is then constructed by simply concatenating locally optimal insertion paths of the tree nodes on  $\Psi$ .

However, if embedding constraints have to be observed, i.e., restrictions on the order of the edges around the vertices of  $G$  are given, locally optimal solutions need not lead to globally optimal solutions and the greedy approach cannot be applied anymore. The best local decision now depends on the decisions for other parts of the edge insertion path. Our second contribution thus is a new linear time algorithm to solve the *optimal ec-edge insertion problem*: Given an ec-planar graph  $G$  with an additional edge  $e$  and a set of embedding constraints  $C$  for the graph  $G + e$ , our algorithm computes an *ec-planar* embedding of  $G$  together with a crossing minimal edge insertion path for  $e$  that observes  $C$ .

Even though constraint handling is an important issue because of its relevance in practical applications, e.g., in interactive graph drawing (see, e.g., [2, 21, 6, 5]), there is only few previous work concerning constraints on the admissible embeddings of a graph. Di Battista et al. [9] consider embedding constraints that appear in database schemas, where table attributes are arranged from top to bottom within a rectangular vertex representing a table, and links connecting attributes may attach at the left or right hand side of these attributes. The integer linear programming approach in [13] considers side constraints in the shape computation phase of orthogonal graph drawing. Dornheim [12] studies the problem of computing embeddings satisfying topological constraints that consist of a cycle together with two sets of edges that have to be embedded inside or outside the cycle, respectively. Buchheim et al. [7] describe how to adapt the planarization approach for directed graphs when incoming and outgoing edges have to appear consecutively around each vertex. On the other hand, linear time algorithms for planarity testing and embedding are long since known; see [18, 3, 8, 20, 4].

This paper is organized as follows. After recalling some known results on planar embeddings in Section 2, Section 3 formally defines the embedding constraints considered in this paper. The first part of the ec-planarity test consists of transforming the input graph into an *ec-expansion* which is described in Section 4; the characterization of ec-planar embeddings and the ec-planarity test itself is then presented in Section 5. Section 6 covers the linear time algorithm for solving the ec-edge insertion problem. Finally, we conclude the paper with remarks on open problems.

## 2 Preliminaries

For basic graph terminology, we refer the reader to [11]. A *combinatorial embedding* of a planar graph  $G$  is defined as a clockwise ordering of the incident

edges for each vertex with respect to a crossing-free drawing of  $G$  in the plane. A *planar embedding* is a combinatorial embedding together with a fixed *external face*.

A *block* is a maximal 2-connected subgraph. The relationship between blocks and cut vertices is given by the *block-cutvertex tree*, or *BC-tree* for short. A slight generalization of the BC-tree is the *block-vertex tree*  $\mathcal{B}$  of a connected graph  $G$ . It represents the relation between the blocks and the vertices of  $G$  and contains a B-node for each block of  $G$  and a V-node for each vertex of  $G$ ; a V-node  $v$  and a B-node  $B$  are connected by an edge if and only if  $v \in B$ . The *representative* of a vertex  $v$  of  $G$  in block  $B$  is either  $v$  itself if  $v \in B$ , or the first vertex on the unique path from  $B$  to  $v$  in  $\mathcal{B}$ .

If  $G$  is 2-connected, its *SPQR-tree*  $\mathcal{T}$  represents the decomposition of  $G$  into its 3-connected components [23] comprising serial, parallel, and 3-connected structures. SPQR-trees [10] are formally defined as follows.

A *split pair* of  $G$  is either a separation pair or a pair of adjacent vertices. A *split component* of a split pair  $\{u, v\}$  is either an edge  $(u, v)$  or a maximal subgraph  $C$  of  $G$  such that  $\{u, v\}$  is not a split pair of  $C$ . Let  $\{s, t\}$  be a split pair of  $G$ . A *maximal split pair*  $\{u, v\}$  of  $G$  with respect to  $\{s, t\}$  is such that, for any other split pair  $\{u', v'\}$ , vertices  $u, v, s$ , and  $t$  are in the same split component.

Let  $e = (s, t)$  be an edge of  $G$ , called the *reference edge*. The SPQR-tree  $\mathcal{T}$  of  $G$  with respect to  $e$  is a rooted ordered tree whose nodes are of four types: S, P, Q, and R. Each node  $\mu$  of  $\mathcal{T}$  has an associated biconnected multi-graph, called the *skeleton* of  $\mu$  (denoted with *skeleton*( $\mu$ )). The tree  $\mathcal{T}$  is recursively defined as follows:

*Trivial Case:* If  $G$  consists of exactly two parallel edges between  $s$  and  $t$ , then  $\mathcal{T}$  consists of a single Q-node whose skeleton is  $G$  itself.

*Parallel Case:* If the split pair  $\{s, t\}$  has at least three split components  $G_1, \dots, G_k$ , the root of  $\mathcal{T}$  is a P-node  $\mu$ , whose skeleton consists of  $k$  parallel edges  $e = e_1, \dots, e_k$  between  $s$  and  $t$ .

*Series Case:* Otherwise, the split pair  $\{s, t\}$  has exactly two split components, one of them is  $e$ , and the other one is denoted with  $G'$ . If  $G'$  has cut-vertices  $c_1, \dots, c_{k-1}$  ( $k \geq 2$ ) that partition  $G$  into its blocks  $G_1, \dots, G_k$ , in this order from  $s$  to  $t$ , the root of  $\mathcal{T}$  is an S-node  $\mu$ , whose skeleton is the cycle  $e_0, e_1, \dots, e_k$ , where  $e_0 = e$ ,  $c_0 = s$ ,  $c_k = t$ , and  $e_i = (c_{i-1}, c_i)$  ( $i = 1, \dots, k$ ).

*Rigid Case:* If none of the above cases applies, let  $\{s_1, t_1\}, \dots, \{s_k, t_k\}$  be the maximal split pairs of  $G$  with respect to  $\{s, t\}$  ( $k \geq 1$ ), and, for  $i = 1, \dots, k$ , let  $G_i$  be the union of all the split components of  $\{s_i, t_i\}$  but the one containing  $e$ . The root of  $\mathcal{T}$  is an R-node, whose skeleton is obtained from  $G$  by replacing each subgraph  $G_i$  with the edge  $e_i = (s_i, t_i)$ .

Except for the trivial case,  $\mu$  has children  $\mu_1, \dots, \mu_k$ , such that  $\mu_i$  is the root of the SPQR-tree of  $G_i \cup e_i$  with respect to  $e_i$  ( $i = 1, \dots, k$ ). The endpoints of

edge  $e_i$  are called the *poles* of node  $\mu_i$ . Edge  $e_i$  is said to be the *virtual edge* of node  $\mu_i$  in skeleton of  $\mu$  and of node  $\mu$  in skeleton of  $\mu_i$ . We call node  $\mu$  the *pertinent node* of  $e_i$  in skeleton of  $\mu_i$ , and  $\mu_i$  the *pertinent node* of  $e_i$  in skeleton of  $\mu$ . The virtual edge of  $\mu$  in skeleton of  $\mu_i$  is called the reference edge of  $\mu_i$ .

Let  $\mu_r$  be the root of  $\mathcal{T}$  in the decomposition given above. We add a Q-node representing the reference edge  $e$  and make it the parent of  $\mu_r$  so that it becomes the new root.

Let  $e$  be an edge in  $skeleton(\mu)$  and  $\nu$  the pertinent node of  $e$ . Deleting edge  $(\mu, \nu)$  in  $\mathcal{T}$  splits  $\mathcal{T}$  into two connected components. Let  $\mathcal{T}_\nu$  be the connected component containing  $\nu$ . The *expansion graph* of  $e$  (denoted with  $expansion(e)$ ) is the graph induced by the edges that are represented by the Q-nodes in  $\mathcal{T}_\nu$ . We further introduce the notation  $expansion^+(e)$  for the graph  $expansion(e) + e$ .

Replacing a skeleton edge  $e$  by its expansion graph is called *expanding  $e$* . The *pertinent graph* of a tree node  $\mu$  results from expanding all edges in  $skeleton(\mu)$  except for the reference edge of  $\mu$  and is denoted with  $pertinent(\mu)$ . Hence, if  $e$  is a skeleton edge and  $\nu$  its pertinent node, then  $expansion^+(e)$  equals  $pertinent(\nu)$ . If  $v$  is a vertex in  $G$ , a node in  $\mathcal{T}$  whose skeleton contains  $v$  is called an *allocation node* of  $v$ .

If  $G$  is 2-connected and planar, its SPQR-tree  $\mathcal{T}$  represents all combinatorial embeddings of  $G$ . In particular, a combinatorial embedding of  $G$  uniquely defines a combinatorial embedding of each skeleton in  $\mathcal{T}$ , and fixing the combinatorial embedding of each skeleton uniquely defines a combinatorial embedding of  $G$ .

### 3 Embedding constraints

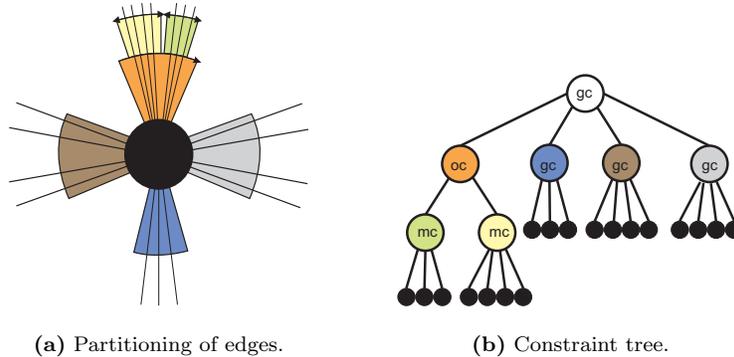
Let  $G = (V, E)$  be a graph. An embedding constraint specifies the admissible clockwise order of the edges incident to a vertex in a combinatorial embedding of  $G$ . In this paper, we consider the case where a vertex has at most one embedding constraint and either all or none of the edges incident to a vertex are subject to embedding constraints.

An *embedding constraint* at a vertex  $v \in V$  is a rooted, ordered tree  $T_v$  such that its leaves are exactly the edges incident to  $v$ . The inner nodes of  $T_v$ , also called *constraint-nodes* or *c-nodes* for short, are of three types: *oc-nodes* (oriented constraint-nodes), *mc-nodes* (mirror constraint-nodes), and *gc-nodes* (grouping constraint-nodes). Since  $T_v$  is an ordered tree, it imposes an order on its leaves and thus on the edges incident to  $v$ . We consider this order as a cyclic order and represent all *admissible* cyclic, clockwise orders of the edges incident to  $v$  by defining how the order of the children of c-nodes in  $T_v$  can be changed:

**gc-node:** The order of children may be arbitrarily permuted.

**mc-node:** The order of children may be reversed.

**oc-node:** The order of children is fixed.



**Figure 1:** The hierarchical partitioning of edges imposed by an embedding constraint (a) and the corresponding constraint tree (b).

Figure 1 shows an example for an embedding constraint. A c-node with a single child is obviously redundant, therefore we demand that each c-node has at least two children. While gc- and mc-nodes alone resemble the concept of PQ-trees [3], the additional concept of oc-nodes is necessary to model constraints that arise in many practical applications, and that complicate planarity testing.

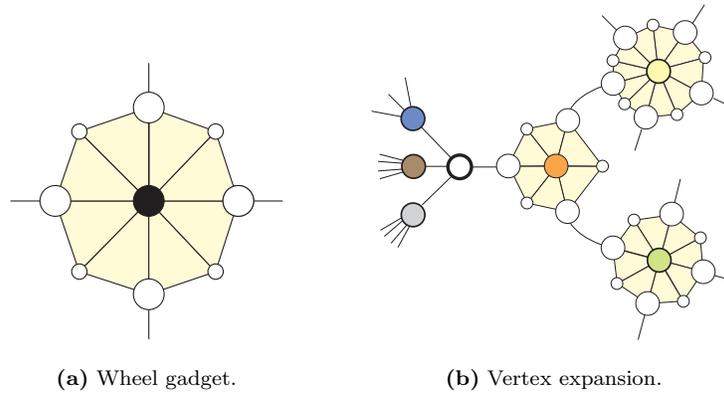
Let  $C$  be a set of embedding constraints at distinct vertices of  $G$ . A combinatorial embedding  $\Gamma$  of  $G$  *observes* the embedding constraints in  $C$ , if for each embedding constraint  $T_v \in C$ , the cyclic clockwise order of the edges around  $v$  in  $\Gamma$  is admissible with respect to  $T_v$ . A planar embedding observing the embedding constraints in  $C$  is an *ec-planar embedding* with respect to  $C$ , and  $(G, C)$  is *ec-planar*, if there exists an ec-planar embedding of  $G$  with respect to  $C$ .

## 4 ec-Expansion

A basic building block of the ec-planarity test is a structural transformation applied to a given graph  $G$  with embedding constraints  $C$ . For each embedding constraint  $T_v$  at vertex  $v$ , this transformation expands  $v$  according to the structure of  $T_v$ . We call the resulting graph the *ec-expansion*  $E(G, C)$  of  $G$  with respect to  $C$ . The details of this transformation are given below.

### 4.1 Construction of the ec-Expansion

The *ec-expansion*  $E(G, C)$  of  $G$  with respect to  $C$  is constructed as follows. Let  $T_v \in C$  be an embedding constraint and  $T'_v$  the subgraph obtained from  $T_v$  by omitting its leaves. Recall that the leaves of  $T_v$  are exactly the edges incident to  $v$ . We replace  $v$  in  $G$  by the tree  $T'_v$  and connect the edges incident with  $v$  with the parents of the corresponding leaves. This transformation introduces a vertex in  $G$  for every c-node in  $T_v$ . Each vertex  $u$  corresponding to an oc- or mc-node is further replaced by a *wheel gadget* which is a wheel graph with  $2d$



**Figure 2:** Expansion gadgets: (a) a wheel gadget replacing a vertex with degree 4; (b) vertex expansion according to the constraint tree in Figure 1(b) (the thick hollow vertex is the root).

spokes, were  $e_1, \dots, e_d$  are the edges incident to  $u$ . Then, the respective wheel gadget consists of a cycle  $x_1, y_1, \dots, x_d, y_d$  of length  $2d$  and a vertex, called *hub*, incident to every vertex on the cycle; see Figure 2(a). The vertex  $u$  is replaced by this wheel gadget, such that  $e_i$  is connected to  $x_i$  for  $1 \leq i \leq d$ . According to the type of the expanded c-node, we distinguish between *O-hubs* (oc-nodes) and *M-hubs* (mc-nodes). We refer to the edges introduced during the ec-expansion as *expansion edges*. Figure 2(b) shows the expansion of a vertex according to the constraint tree shown in Fig 1(b).

The purpose of the wheel gadgets is to model the fixed order of the children of the corresponding c-node. Since a wheel gadget is a 3-connected graph, it admits only two combinatorial embeddings that are mirror images of each other. The order in which non-gadget edges are attached to the wheel cycle is either the order given by the corresponding c-node, or the reverse order. Every face adjacent to the hub is a triangle. We call these faces *inner wheel gadget faces*.

**Lemma 1.** *Let  $G = (V, E)$  be a graph with embedding constraints  $C$ . Then, its ec-expansion  $E(G, C)$  has size  $O(|V| + |E|)$  and can be constructed in time  $O(|V| + |E|)$ .*

*Proof.* Consider an embedding constraint  $T_v \in C$ . Since the leaves of  $T_v$  are in one-to-one correspondence to the edges incident to  $v$  and each c-node has at least two children, the size of  $T_v$  is linear in  $\deg(v)$ . We replace each oc- and mc-node  $\mu$  by a wheel gadget with  $4 \deg(\mu)$  edges. Thus, the expansion of vertex  $v$  creates  $O(\deg(v))$  edges, and the total number of additional edges in  $E(G, C)$  is bounded by  $\sum_{v \in V} O(\deg(v)) = O(|E|)$ . Therefore, the size of the expansion graph is  $O(|V| + |E|)$ , and the expansion can obviously be computed in  $O(|E(G, C)|) = O(|V| + |E|)$  time.  $\square$

## 4.2 ec-Expansion and ec-Planar Embeddings

In this section we discuss the relationship between planar embeddings of the ec-expansion  $E(G, C)$  and ec-planar embeddings of  $(G, C)$ . Though the ec-expansion serves as a tool for modeling the embedding constraints in  $C$ , a planar embedding of  $E(G, C)$  needs to fulfill certain conditions in order to induce an ec-planar embedding of  $G$  with respect to  $C$ . We call a planar embedding  $\Gamma$  of  $E(G, C)$  *ec-planar* if

1. the external face of  $\Gamma$  does not contain a hub;
2. every face incident to a hub is a triangle consisting solely of edges of the corresponding wheel gadget; and
3. each O-hub  $h$  is *oriented correctly*, i.e., the cyclic, clockwise order of the edges around  $h$  in  $\Gamma$  corresponds to the order specified by the corresponding oc-node.

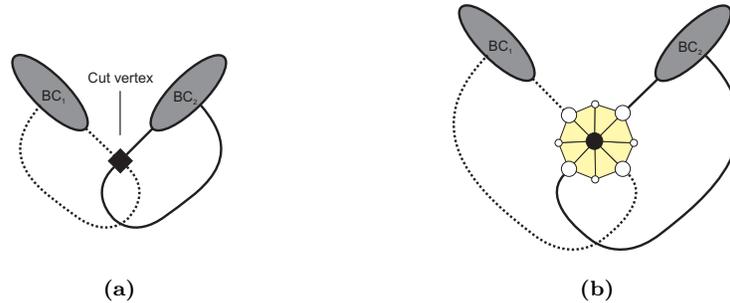
Let  $\Gamma$  be an ec-planar embedding of  $E(G, C)$ . We obtain an ec-planar embedding of  $(G, C)$  as follows. For each vertex  $v$  with corresponding embedding constraint in  $C$ , there is a connected subgraph  $G_v$  in  $E(G, C)$  resulting from expanding  $v$ . Let  $\bar{G}_v \subset E(G, C)$  be the graph induced by the vertices not contained in  $G_v$ . The conditions above assure that the planar embedding  $\Gamma_v$  of  $G_v$  induced by  $\Gamma$  is such that  $\bar{G}_v$  lies in the external face of  $\Gamma_v$ . The edges that connect  $G_v$  to  $\bar{G}_v$  correspond to the edges incident to  $v$  in  $G$ . Their cyclic clockwise order around  $G_v$  is admissible with respect to  $T_v$ , since the wheel gadgets fix the order of the edges specified by oc- and mc-nodes, and O-hubs are oriented correctly. We shrink  $G_v$  to a single vertex by contracting all edges in  $G_v$  while preserving the embedding, thus resulting in an admissible order of the edges around  $v$ .

If we have an ec-planar embedding of  $(G, C)$ , then the edges around each vertex  $v$  are ordered such that the constraints in  $T_v$  are fulfilled. It is easy to see that we can replace each such vertex  $v$  by the expansion graph corresponding to  $T_v$  in such a way that we obtain an ec-planar embedding of  $E(G, C)$ . Thus, we get the following result:

**Lemma 2.** *Let  $G$  be a graph with embedding constraints  $C$ . Then,  $(G, C)$  is ec-planar if and only if  $E(G, C)$  is ec-planar. Moreover, every ec-planar embedding of  $E(G, C)$  induces an ec-planar embedding of  $(G, C)$ .*

## 5 ec-Planarity Testing

It is well-known that planarity testing can be reduced to 2-connected graphs, i.e., it is sufficient to test the blocks of a graph independently. However, adding embedding constraints complicates this task. Let  $G$  be a graph with embedding constraints  $C$ . Consider a cut vertex  $c$  in  $G$  that connects two blocks  $BC_1$  and  $BC_2$  via the edge sets  $S_1$  and  $S_2$ , respectively; see Figure 3(a). If these edge



**Figure 3:** A crossing is needed between edges of two blocks due to embedding constraints (a). The expansion using a wheel gadget merges the two blocks into a single one (b).

sets are subject to embedding constraints that force the edges in  $S_1$  and  $S_2$  to be intermixed as in Figure 3(a), then the given graph is not ec-planar even if its blocks are ec-planar. We solve this problem by first applying the ec-expansion to the graph. This replaces the cut vertex  $c$  by a wheel gadget so that  $c$  does not separate  $BC_1$  and  $BC_2$  anymore; see Figure 3(b).

By Lemma 2, we know that it is sufficient to test the ec-expansion  $E(G, C)$  for ec-planarity. In contrast to the graph  $G$  itself, the following lemma shows that we can test the blocks of  $E(G, C)$  separately.

**Lemma 3.**  *$E(G, C)$  is ec-planar if and only if every block of  $E(G, C)$  is ec-planar.*

*Proof.* If  $E(G, C)$  is ec-planar, then there is an ec-planar embedding of  $E(G, C)$ , and this embedding implies an ec-planar embedding for each block of  $E(G, C)$ .

Suppose now that each block of  $E(G, C)$  is ec-planar. Consider a wheel gadget  $\mathcal{G}$  in  $E(G, C)$ . Since  $\mathcal{G}$  is 3-connected,  $\mathcal{G}$  is completely contained in a single block  $B$  of  $E(G, C)$ . For each edge  $(u, v) \in \mathcal{G}$ , the pair  $\{u, v\}$  is not a separation pair in  $B$  by construction, hence every inner wheel face of  $\mathcal{G}$  is also a face in every planar embedding of  $B$ . Moreover, the hub of  $\mathcal{G}$  is not a cut vertex of  $E(G, C)$ , since all its incident edges are in  $B$ .

We construct an ec-planar embedding of  $E(G, C)$  as follows. We start with an arbitrary block  $B$  of  $E(G, C)$ . Let  $\Pi$  be an ec-planar embedding of  $B$ . In particular, the external face of  $\Pi$  is not an inner wheel face of a wheel gadget. We add the remaining blocks successively to  $\Pi$ . Let  $B'$  be another block of  $E(G, C)$  that shares a vertex  $c$  with  $B$ , and let  $\Pi'$  be an ec-embedding of  $B'$ . We pick faces  $f \in \Pi$  and  $f' \in \Pi'$  that are adjacent to  $c$  and not inner wheel faces of a wheel gadget. This is possible, since the only vertices adjacent solely to inner wheel faces are the O- and M-hubs. Then, we insert  $\Pi'$  with  $f'$  as external face into the face  $f$  of  $\Pi$ . This results in an ec-planar embedding of  $B \cup B'$ . We can add the remaining blocks (if any) in the same way, resulting in an ec-planar embedding of  $E(G, C)$ .  $\square$

If we can characterize all ec-planar embeddings of the blocks of  $E(G, C)$ ,

the construction in the proof of Lemma 3 also shows us how to enumerate all ec-planar embeddings of  $E(G, C)$  by traversing its BC-tree. In the following, we devise such a characterization. Let  $B$  be a block of  $E(G, C)$  and  $\mathcal{T}$  its SPQR-tree.

**Observation 1.** *Every wheel gadget  $\mathcal{G}$  is completely contained within the skeleton of an R-node. In particular, the hub of  $\mathcal{G}$  occurs only in the skeleton of a single R-node.*

*Proof.*  $\mathcal{G}$  is 3-connected, and for each edge  $(u, v) \in \mathcal{G}$ , the pair  $\{u, v\}$  is not a separation pair in  $B$  by construction. Therefore, all edges of  $\mathcal{G}$  occur in the same skeleton graph, which must be the skeleton of an R-node  $\mu$ . The hub  $h$  of  $\mathcal{G}$  is only incident to edges of  $\mathcal{G}$  and no other edge of  $B$ , hence  $h$  occurs only in  $\text{skeleton}(\mu)$ .  $\square$

If  $B$  is planar, then the skeleton of an R-node is a 3-connected planar graph, thus having exactly two planar embeddings which are mirror images of each other. We call two O-hubs contained in the same skeleton  $S$  *conflicting* if none of the two planar embeddings of  $S$  orients both O-hubs correctly. The following theorem gives us an easy to check condition for ec-planarity and characterizes all possible ec-planar embeddings:

**Theorem 1.** *Let  $G$  be a graph with embedding constraints  $C$ . Let  $B$  be a block of  $E(G, C)$  and  $\mathcal{T}$  its SPQR-tree. Then, the following holds:*

1.  *$B$  is ec-planar if and only if  $B$  is planar and no skeleton of an R-node of  $\mathcal{T}$  contains conflicting O-hubs.*
2. *If  $B$  is ec-planar, then the embeddings of the skeletons of  $\mathcal{T}$  induce an ec-planar embedding of  $B$  if and only if each O-hub in the skeleton of an R-node is oriented correctly.*

*Proof.* If  $B$  admits an ec-planar embedding, then this embedding induces embeddings of the skeletons of  $\mathcal{T}$  such that every O-hub in the skeleton of an R-node is oriented correctly. In particular, no R-node skeleton contains conflicting O-hubs.

Suppose now that  $B$  is planar and no R-node skeleton contains conflicting O-hubs. For each R-node skeleton containing at least one O-hub, we can choose planar embeddings such that all O-hubs are oriented correctly within the skeletons. We have to show that the embeddings of the skeletons induce an ec-planar embedding of  $B$ , even if we chose arbitrary embeddings for the remaining skeletons. This holds, since every such embedding  $\Pi$  has the property that each O-hub is oriented correctly because wheel gadgets are completely contained within R-node skeletons by Observation 1, and inner wheel faces are preserved. We can pick any face of  $\Pi$  as external face which is not an inner wheel face (such a face always exists) and obtain an ec-planar embedding of  $B$ .  $\square$

Function `ISECPLANAR` depicted in Algorithm 1 applies Theorem 1 and devises a linear time ec-planarity test, which can easily be extended so that it computes an ec-planar embedding as well.

```

1: function ISECPANAR(Graph  $G$ , Constraints  $C$ ) : bool
2:   Construct ec-expansion  $E$  of  $(G, C)$ .
3:   if  $E$  is not planar then return false
4:   for each block  $B$  of  $E$  do
5:     Construct SPQR-tree  $\mathcal{T}$  of  $B$ .
6:     for each R-node  $\mu \in \mathcal{T}$  do
7:       if  $\text{skeleton}(\mu)$  contains two conflicting O-hubs then
8:         return false
9:       end if
10:    end for
11:  end for
12:  return true
13: end function

```

**Algorithm 1:** Ec-planarity testing.

**Theorem 2.** *Let  $G = (V, E)$  be a graph with embedding constraints  $C$ . Then, the function ISECPANAR tests  $(G, C)$  for ec-planarity in time  $O(|V| + |E|)$ . Moreover, if  $(G, C)$  is ec-planar, an ec-planar embedding of  $(G, C)$  can also be computed in time  $O(|V| + |E|)$ .*

*Proof.* By Lemma 2 and 3, it is sufficient to test every block of  $E(G, C)$  for ec-planarity. Hence, the correctness of ISECPANAR follows from Theorem 1.

Constructing the ec-expansion (Lemma 1) and testing planarity [18] can be done in linear time. For each block  $B$  of  $E(G, C)$ , we construct its SPQR-tree, which requires linear time in the size of  $B$ ; see [15]. The check for conflicting O-hubs is easy to implement: For each R-node skeleton  $S$ , we compute a planar embedding of  $S$ . If this embedding contains both correctly as well as not correctly oriented O-hubs, then there is a conflict, otherwise not. Since the total size of skeleton graphs is linear in the size of  $B$  and a planar embedding can be found in linear time (see, e.g., [8]), we need linear running time for each block. Hence, the total running time is linear in the size of  $E(G, C)$  which is  $O(|V| + |E|)$  by Lemma 1.

In order to find an ec-planar embedding of  $G$ , we just have to compute embeddings of the skeleton graphs for each block as described in Theorem 1 and combine the embeddings as described in the proof of Lemma 3.  $\square$

## 6 ec-Edge Insertion

### 6.1 ec-Edge Insertion Paths and ec-Traversing Costs

We first generalize the terms insertion path and traversing costs introduced in [17]. Intuitively, the edges in an insertion path are the edges we need to cross when inserting an edge  $(x, y)$  into an embedding. Let  $G + (x, y)$  be a graph with embedding constraints  $C$ . An *ec-edge insertion path* for  $(x, y)$  in an ec-planar

embedding  $\Pi$  of  $G$  is a sequence of edges  $e_1, \dots, e_k$  of  $G$  satisfying the following conditions:

1. There is a face  $f_x \in \Pi$  with  $x, e_1 \in f_x$ , a face  $f_y \in \Pi$  with  $e_k, y \in f_y$ , and faces  $f_i \in \Pi$  with  $e_i, e_{i+1} \in f_i$  for  $1 \leq i < k$ .
2. The edge order around  $x$  and  $y$  is admissible with respect to  $C$  if  $(x, y)$  leaves  $x$  via face  $f_x$  and enters  $y$  via face  $f_y$ .

Finding a shortest ec-insertion path in a fixed embedding  $\Pi$  is easy: We only need to identify the set of faces  $F_x$  incident to  $x$  where the insertion path may start, and  $F_y$  incident to  $y$  where it may end, and then find a shortest path in the dual graph of  $\Pi$  connecting a face in  $F_x$  with a face in  $F_y$ .

We are interested in the shortest possible ec-insertion path among all ec-planar embeddings of  $G$ , which we also call an *optimal ec-insertion path* in  $G$ . In particular, we need to identify the required ec-planar embedding of  $G$ . In order to represent all ec-planar embeddings of  $G$ , we apply Lemma 2 and use its ec-expansion instead. More precisely, we use the subgraph  $K = E(G + (x, y), C) \setminus e$ , where  $e = (v, w)$  is the edge of  $E(G + (x, y), C)$  connecting the expansion of  $x$  with the expansion of  $y$ . An ec-insertion path in an ec-planar embedding of  $K$  is defined as before with the only difference that we replace the second condition with

- 2'.  $e_1, \dots, e_k$  contains no expansion edge of  $K$ .

It is easy to see that we can also use this definition for a subgraph  $B$  of  $K$  and two distinct vertices of  $B$  that are not hubs.

We adapt the notion of traversing costs defined in [17] to ec-planarity. Let  $e$  be a skeleton edge, and let  $\Pi$  be an arbitrary ec-embedding of the graph  $\text{expansion}^+(e)$  with dual graph  $\Pi^*$ , in which all edges corresponding to gadget edges have length  $\infty$  and the other edges have length 1. Let  $f_1$  and  $f_2$  be the two faces in  $\Pi$  separated by  $e$ . We denote with  $P(\Pi^*, e)$  the length of the shortest path in  $\Pi^*$  that connects  $f_1$  and  $f_2$  and does not use the dual edge of  $e$ . Hence, we have  $P(\Pi^*, e) \in \mathbb{N} \cup \{\infty\}$ .

The following lemma follows analogously to the result shown in [17].

**Lemma 4.** *Let  $\mu$  be a node in  $\mathcal{T}$  and  $e$  an edge in  $\text{skeleton}(\mu)$ . Then,  $P(\Pi^*, e)$  is independent of the ec-embedding  $\Pi$  of  $\text{expansion}^+(e)$ .*

*Proof.* Let  $m$  be the number of edges in  $G_e := \text{expansion}^+(e)$  and  $G'_e$  be the graph obtained from  $G_e$  by replacing each gadget edge with  $m + 1$  parallel edges. Then, each embedding  $\Pi$  of  $G_e$  corresponds to an embedding  $\Pi'$  of  $G'_e$ , and  $P(\Pi^*, e)$  is  $\infty$  if and only if the corresponding path in  $\Pi'$  is longer than  $m$ . Lemma 1 in [17] shows that for the general case, i.e., without embedding constraints,  $P(\Pi^*, e)$  is independent of the embedding  $\Pi$ . Applying this lemma and observing that the ec-embeddings of  $G_e$  are a non-empty subset of the embeddings of  $G_e$  yields the lemma.  $\square$

Thus, we define the *ec-traversing costs*  $c(e)$  of a skeleton edge  $e$  as  $P(\Pi^*, e)$  for an arbitrary ec-embedding  $\Pi$  of  $\text{expansion}^+(e)$ .

## 6.2 The Algorithm for 2-connected Graphs

The hard part is to find an ec-insertion path in a block  $B$  of  $K$ . Our task is to compute an optimal ec-insertion path between two nodes  $v, w$  of  $B$ . The function OPTIMALECBLOCKINSERTER shown in Algorithm 2 and 3 solves this problem. In this algorithms, we use the notation  $\min_{i,n} \Lambda$  which returns a tuple in the set  $\Lambda$  of  $n$ -tuples whose  $i$ -th component is minimal among all tuples in  $\Lambda$ .

The function OPTIMALECBLOCKINSERTER is called with a block  $B$  of an ec-planar ec-expansion and two distinct vertices  $v$  and  $w$  of  $B$ . Since we assume that  $B$  contains all gadget edges, we do not need to pass further constraint information for the edge  $(v, w)$ . In particular, using any insertion path in any ec-planar embedding of  $B$  that connects  $v$  and  $w$  and does not cross a gadget edge yields an ec-embedded planarization of  $B \cup (v, w)$ . Hence, we look for an ec-embedding of  $B$  that allows the insertion of the edge  $(v, w)$  with the minimum number of crossings.

First, we compute the SPQR-tree  $\mathcal{T}$  of  $B$  and embed the skeletons such that they imply an ec-embedding of  $B$ , i.e. the R-node skeletons are embedded correctly. Then, the shortest path  $\Upsilon := \mu_1, \dots, \mu_k$  between an allocation node  $\mu_1$  of  $v$  and  $\mu_k$  of  $w$  is identified. In order to achieve a consistent orientation, we root  $\mathcal{T}$  such that  $\Upsilon$  is a descending path in the tree, i.e.,  $\mu_i$  is the parent of  $\mu_{i-1}$  for  $i = 2, \dots, k$ . Note that the rooting of the SPQR-tree implies a direction of the skeleton edges: the edges in a skeleton with reference edge  $e_r = (s, t)$  are directed such that the skeleton is a planar  $st$ -graph; see, e.g., [10]. This direction is necessary in order to identify the left and the right face of an edge.

The algorithm traverses the path  $\Upsilon$  from  $\mu_1$  to  $\mu_{k-1}$  and iteratively computes the lengths of the shortest ec-insertion paths that start from  $v$  and leave the pertinent graph  $P_i$  of  $\mu_i$  to the left or to the right, respectively, where all ec-embeddings of  $P_i$  are considered. Here, left and right refer to the direction of the reference edge of  $\mu_i$ . These lengths are maintained in the variables  $\lambda_\ell$  and  $\lambda_r$ . Finally, when node  $\mu_k$  is considered, this information is used to determine a shortest insertion path ending at  $w$ .

For each node  $\mu_i$ , the following information is computed:

- $\phi_\ell^i$  (resp.  $\phi_r^i$ ) indicates if the shortest ec-insertion path leaving  $P_i$  to the left (right) uses the shortest ec-insertion path that leaves  $P_{i-1}$  to the left (in this case the value is  $\ell$ ) or to the right (the value is  $r$ ).
- $\Delta_\ell^i$  (resp.  $\Delta_r^i$ ) is the subpath that is appended to the path leaving  $P_{i-1}$  when leaving  $P_i$  to the left (right).

These values are solely used for the purpose of creating the optimal ec-insertion path at the end of the function. If  $s \in \{\ell, r\}$  denotes a side, we denote with  $\bar{s}$  the other side, i.e.,  $\bar{\ell} = r$  and vice versa.

The body of the for-loop starts by expanding all edges of the skeleton  $S_i$  of  $\mu_i$  except for edges representing  $v$  or  $w$ . The resulting graph is called  $G_i$ . If  $1 < i < k$ , then  $G_i$  will contain two virtual edges  $e_v$  (representing  $v$ ) and  $e_w$  (representing  $w$ ). Note that we obtain  $P_i$  (plus reference edge) by replacing  $e_v$  with  $P_{i-1}$ .

**function** OPTIMALECBLOCKINSERTER(Block  $B$  of  $K$ , vertex  $v$ , vertex  $w$ )  
 Construct SPQR-tree  $\mathcal{T}$  of  $B$  such that the embeddings of the skeletons imply a feasible embedding of  $B$ .

Find the shortest path  $\mu_1, \dots, \mu_k$  in  $\mathcal{T}$  between an allocation node  $\mu_1$  of  $v$  and  $\mu_k$  of  $w$ .  
 Root  $\mathcal{T}$  such that  $\mu_k$  becomes the parent of  $\mu_{k-1}$  (if  $k > 1$ ).

$\lambda_\ell := \lambda_r := 0$   $\triangleright$  length of shortest insertion path leaving to the left/right

**for**  $i = 1, \dots, k$  **do**  
   **let**  $S_i = \text{skeleton}(\mu_i)$   
   **let**  $G_i$  be the graph obtained from  $S_i$  by replacing each edge not representing  $v$  or  $w$  with its expansion graph, and let  $\Pi_i$  be the embedding of  $G_i$  induced by the embeddings of the skeletons of  $\mathcal{T}$ .

$\triangleright \phi_{l/r}^i$  indicates which insertion path of  $\mu_{i-1}$  is chosen;  
    $\triangleright \Delta_{l/r}^i$  denotes the subpath within  $S_i$  when leaving left/right.

**if**  $\mu_i$  is a P-node **then**  
      $(\phi_\ell^i, \Delta_\ell^i) := (\ell, \epsilon); (\phi_r^i, \Delta_r^i) := (r, \epsilon)$   $\triangleright$  No crossings required.  
   **else**  $\triangleright$  S- or R-node.  
     **if**  $i = 1$  **then**  
        $L_v := R_v :=$  the set of adjacent faces of the copy of  $v$  in  $S_i$   
     **else**  
       **let**  $e_v$  be the representative of  $v$  in  $S_i$   
        $L_v := \{ \text{the left face of } e_v \}$   
        $R_v := \{ \text{the right face of } e_v \}$   
     **end if**  
     **if**  $i = k$  **then**  
        $L_w := R_w :=$  the set of adjacent faces of the copy of  $w$  in  $S_i$   
     **else**  
       **let**  $e_w$  be the representative of  $w$  in  $S_i$   
        $L_w := \{ \text{the left face of } e_w \}$   
        $R_w := \{ \text{the right face of } e_w \}$   
     **end if**  
      $\triangleright$  Compute shortest ec-insertion paths (from l/r to l/r) within  $G_i$ .  
      $\triangleright$  Note:  $p_{\ell r} = p_{\ell \ell}$  and  $p_{r r} = p_{r \ell}$  if  $i \in \{1, k\}$ .  
      $p_{\ell r} := \text{SHORTESTECINSPATH}(\Pi_i, L_v, L_w)$   
      $p_{\ell \ell} := \text{SHORTESTECINSPATH}(\Pi_i, L_v, R_w)$   
      $p_{r r} := \text{SHORTESTECINSPATH}(\Pi_i, R_v, L_w)$   
      $p_{r \ell} := \text{SHORTESTECINSPATH}(\Pi_i, R_v, R_w)$

$\triangleright$  Continued on next page...

**Algorithm 2:** Computation of an optimal ec-insertion path (2-connected case).

```

    ▷ Collect possible solutions.
     $\Lambda_\ell := \{(\lambda_\ell + |p_{\ell\ell}|, \ell, p_{\ell\ell}), (\lambda_r + |p_{r\ell}|, r, p_{r\ell})\}$ 
     $\Lambda_r := \{(\lambda_\ell + |p_{\ell r}|, \ell, p_{\ell r}), (\lambda_r + |p_{rr}|, r, p_{rr})\}$ 
    if  $\mu_i$  is an R-node that can be mirrored then
         $\Lambda_\ell := \Lambda_\ell \cup \{(\lambda_\ell + |p_{rr}|, \ell, p_{rr}^*), (\lambda_r + |p_{\ell r}|, r, p_{\ell r}^*)\}$ 
         $\Lambda_r := \Lambda_r \cup \{(\lambda_\ell + |p_{r\ell}|, \ell, p_{r\ell}^*), (\lambda_r + |p_{\ell\ell}|, r, p_{\ell\ell}^*)\}$ 
    end if

    ▷ Pick best solution.
     $(\lambda_\ell, \phi_\ell^i, \Delta_\ell^i) := \min_{1,3} \Lambda_\ell$ 
     $(\lambda_r, \phi_r^i, \Delta_r^i) := \min_{1,3} \Lambda_r$ 
end if
end for

    ▷ Build final ec-insertion path. Note:  $\lambda_\ell = \lambda_r$  always holds here!
     $s_k := \ell$  ▷ Start with empty path.
for  $i := k$  downto 1 do ▷ Collect path backward.
     $p_i := \Delta_{s_i}^i$ ;  $s_{i-1} := \phi_{s_i}^i$ 
end for

return  $p_1 + \dots + p_k$ 
end function

```

**Algorithm 3:** Function OPTIMALECBLOCKINSERTER (part 2).

We distinguish according to the type of  $\mu_i$ . If  $\mu_i$  is a P-node, then the optimal ec-insertion path leaving  $P_{i-1}$  to the left (right) is also an optimal ec-insertion path leaving  $P_i$  to the left (right); we just need to permute the parallel edges in  $S_i$  such that  $e_v$  is the leftmost (rightmost) edge. Otherwise, we have four possibilities for extending an ec-insertion path leaving  $P_i$ . Such a path may start in a face left or right of  $e_v$ , and may end in a face left or right of  $e_w$ . In addition, we have to consider two special cases: if  $i = 1$  then  $G_i$  contains  $v$  and the ec-insertion path may start in any face adjacent to  $v$ ; if  $i = k$  then  $G_i$  contains  $w$  and the ec-insertion path may end in any face adjacent to  $w$ . We compute the (at most) four possible shortest ec-insertion paths using the function SHORTESTECINSPATH( $\Pi, F_s, F_t$ ). Here  $\Pi$  is an ec-embedding of an ec-expansion,  $F_s$  are the faces where the insertion path may start, and  $F_t$  are the faces where it may end. The ec-insertion path is found using a breadth-first search (BFS) in the dual graph of  $\Pi$ , where edges corresponding to gadget edges are removed (which means that it is forbidden to cross their primal counterparts). We call these shortest ec-insertion paths  $p_{\ell\ell}, p_{\ell r}, p_{r\ell}, p_{rr}$ , where  $p_{\ell\ell}$  stands for the path starting in a face in  $L_v$  and ending in a face in  $R_w$  etc. We have two choices for a shortest ec-insertion path leaving  $P_i$  to the left if we consider only the given embedding of the skeleton of  $\mu_i$ :

- We leave  $P_{i-1}$  to the left (or start at  $v$  if  $i = 1$ ) and end in a face in  $R_w$  (i.e., we enter  $e_w$  from right). This path has length  $\lambda_\ell + |p_{\ell\ell}|$ .
- We leave  $P_{i-1}$  to the right (or start at  $v$  if  $i = 1$ ) and end in a face in  $R_w$

(i.e., we enter  $e_w$  from left). This path has length  $\lambda_r + |p_{r\ell}|$ .

For the shortest ec-insertion path leaving  $P_i$  to the right, we have two similar cases. Further choices are possible if  $\mu_i$  is an R-node that can be mirrored. We could mirror the embedding of  $S_i$ , expand the skeleton edges as before such that we obtain an embedding  $\tilde{\Pi}_i$ , and compute the four paths in  $\tilde{\Pi}_i$  again. Notice that  $\tilde{\Pi}_i$  is not simply the mirror image of  $\Pi_i$ . However, this is not necessary. We observe that, e.g., the path  $\tilde{p}_{\ell\ell}$  is obtained from  $p_{rr}$  by reversing the subsequences of edges that have been created by expanding a common skeleton edge of  $S_i$ . We call this path  $p_{rr}^*$ . A similar argumentation holds for  $\tilde{p}_{\ell r}, \tilde{p}_{r\ell}, \tilde{p}_{rr}$ . It follows that we have at most four possible choices for leaving  $P_i$  to the left and to the right, respectively. Among all possible choices, we pick the shortest one.

After processing all nodes  $\mu_i$ , it is easy to reconstruct the best ec-insertion path from  $v$  to  $w$  using  $\phi_{\ell/r}^i$  and  $\Delta_{\ell/r}^i$ . Notice that  $\lambda_\ell = \lambda_r$  holds at the end, since  $L_w^k = R_w^k$ .

### 6.3 Correctness and Optimality

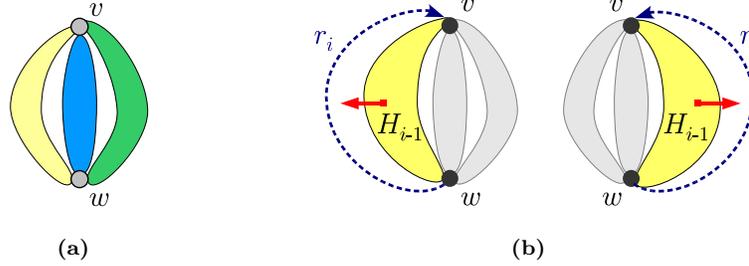
**Lemma 5.** *There exists an ec-embedding  $\Pi$  of  $B$  such that  $p_1 + \dots + p_k$  is an ec-insertion path for  $v$  and  $w$  in  $B$  with respect to  $\Pi$ .*

*Proof.* Consider the path  $\Upsilon = \mu_1, \dots, \mu_k$  computed by the algorithm. By construction of  $\Upsilon$ , the skeleton of  $\mu_1$  contains  $v$ , the skeleton of  $\mu_k$  contains  $w$ , and, for each  $j = 2, \dots, k-1$ , the skeleton of  $\mu_j$  contains neither  $v$  nor  $w$ . Moreover,  $\Upsilon$  does not contain a Q-node.

First, we prove the lemma for the case where  $\Upsilon$  consists of a single node  $\mu_1$ . In this case, the skeleton of  $\mu_1$  contains both  $v$  and  $w$ . We distinguish two cases according to the type of  $\mu_1$ :

1.  **$\mu_1$  is a P-node.** Let  $\Pi$  be an arbitrary ec-embedding of  $B$ . Since  $v$  and  $w$  share a common face in  $\Pi$ , the empty path returned by the algorithm is an ec-insertion path for  $v$  and  $w$  in  $B$  with respect to  $\Pi$ ; see Figure 4(a).
2.  **$\mu_1$  is an S- or an R-node.** In this case the graph  $G_1$  constructed by the algorithm is the original block  $B$ , since all skeleton edges are expanded. Moreover,  $\Pi_1$  is an ec-embedding of  $B$ , and  $p_{\ell r} = p_{\ell\ell}$  and  $p_{rr} = p_{r\ell}$  are ec-insertion paths in  $B$  with respect to  $\Pi_1$ . We do not need to consider the case where the embedding of the skeleton can be mirrored, since this will not yield a shorter path. Hence,  $p_1$  is either  $p_{\ell r}$  or  $p_{rr}$  and thus an ec-insertion path in  $B$  with respect to  $\Pi_1$ .

Assume now that  $k > 1$ . For  $i = 1, \dots, k$ , we denote with  $H_i$  the pertinent graph of  $\mu_i$ , with  $r_i$  the reference edge of  $\mu_i$  in  $H_i$ , and, for  $1 < i$ , with  $e_i$  the edge in  $\text{skeleton}(\mu_i)$  whose pertinent node is  $\mu_{i-1}$ . Recall that  $s_i \in \{\ell, r\}$  is the side of  $H_i$  where the computed insertion path shall leave. We show by induction over  $i$  that, for  $1 \leq i < k$ , there is an embedding  $\Gamma_i$  of  $H_i$  such that  $p_1 + \dots + p_i$  is an ec-insertion path leaving  $H_i$  at side  $s_i$ . The embeddings  $\Gamma_1, \dots, \Gamma_{k-1}$  are



**Figure 4:** Proof of Lemma 5;  $k = 1$  and  $\mu_1$  is a P-node (a), and  $\mu_i$  is a P-node (b).

iteratively constructed during the proof. For our convenience, we denote with  $\Gamma_i^-$  the embedding of  $H_i - r_i$  induced by  $\Gamma_i$ .

$i = 1$ . Consider the different types for node  $\mu_1$ :

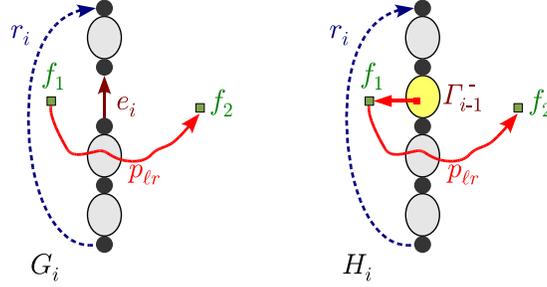
1.  $\mu_1$  is a **P-node**. This case does not apply, since  $\mu_2$  is not an allocation node of  $v$ .
2.  $\mu_1$  is an **S- or an R-node**. In this case,  $G_1 = H_1$ , and  $p_1$  is a path leaving either  $\Pi_1$  or the mirror image of  $\Pi_1$  to side  $s_1$ . Hence, we set  $\Gamma_1$  to  $\Pi_1$  or its mirror image, respectively.

$1 < i < k$ . We distinguish again between the types of  $\mu_i$ .

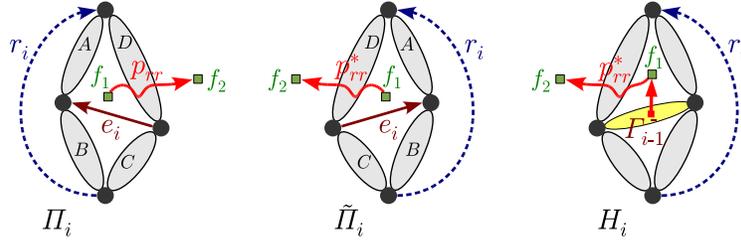
1.  $\mu_i$  is a **P-node**. In this case,  $p_i = \epsilon$ , i.e., no further edges need to be crossed. The embedding  $\Gamma_i$  is obtained as follows. If  $s_i = \ell$ , we permute the edges in  $skeleton(\mu_i)$  such that  $e_i$  is to the right of  $r_i$ ; otherwise, we permute the edges such that  $e_i$  is to the left of  $r_i$ . Then, we replace  $e_i$  by  $\Gamma_{i-1}^-$ , and the remaining edges  $e \neq r_i$  in  $skeleton(\mu_i)$  by an arbitrary embedding of  $expansion(e)$ ; see Figure 4(b).
2.  $\mu_i$  is an **S- or an R-node**. In this case,  $p_i$  is either  $p_{s_{i-1}s_i}$  or  $p_{\bar{s}_{i-1}\bar{s}_i}$ ; the latter case corresponds to mirroring the embedding of  $skeleton(\mu_i)$  before.

We first restrict us to the case in which  $p_i$  is set to  $p_{s_{i-1}s_i}$ , i.e., an ec-insertion path in the embedding  $\Pi_i$  that starts in a face at side  $s_{i-1}$  of  $e_i$  and ends in a face at side  $\bar{s}_i$  of edge  $r_i$ . We obtain  $\Gamma_i$  by replacing  $e_i$  by  $\Gamma_i^-$  in  $\Pi_i$ ; see Figure 5. Since the ec-insertion path  $p_1 + \dots + p_{i-1}$  leaves  $\Gamma_i^-$  to the side  $s_{i-1}$ ,  $p_1 + \dots + p_i$  is an ec-insertion path leaving  $\Gamma_i$  to the side  $s_i$ .

Finally, assume that  $p_i = p_{\bar{s}_{i-1}\bar{s}_i}$ . Let  $\tilde{\Pi}_i$  be the embedding that we obtain by first mirroring the embedding of  $skeleton(\mu_i)$  and then expanding and embedding each skeleton edge not representing  $v$  or  $w$  as before. We observe that  $p_i$  is an ec-insertion path in  $\tilde{\Pi}_i$  that starts in a face at side  $s_{i-1}$  of  $e_i$  and ends in a face at side  $\bar{s}_i$  of edge  $r_i$ ; see Figure 6. With the same argumentation as above, we obtain  $\Gamma_i$  by replacing  $e_i$  with  $\Gamma_{i-1}^-$  in  $\tilde{\Pi}_i$ .



**Figure 5:** Proof of Lemma 5;  $\mu_i$  is an S-node,  $L_v = R_w = \{f_1\}$ ,  $R_v = L_w = \{f_2\}$ .



**Figure 6:** Proof of Lemma 5;  $\mu_i$  is an R-node,  $R_v = \{f_1\}$ ,  $L_w = \{f_2\}$ .

To conclude the proof, we consider the node  $\mu_k$ . We know that  $\mu_k$  is either an S- or an R-node, and we may assume that  $p_k = p_{s_{i-1}s_i}$ , since  $p_{\ell r} = p_{\ell \ell}$  and  $p_{rr} = p_{r \ell}$  holds for  $i = k$ . Hence,  $p_k$  is an ec-insertion path in  $\Pi_k$  that starts in a face at side  $s_{i-1}$  of  $e_i$  and ends in a face adjacent to the copy of  $w$  in  $G_k$ . We obtain  $\Pi$  by replacing  $e_k$  with  $\Gamma_{k-1}^-$  in  $\Pi_k$ , and thus  $p_1 + \dots + p_k$  is an ec-insertion path for  $v$  and  $w$  in  $\Pi$ .  $\square$

**Lemma 6.** *Let  $\Pi'$  be an arbitrary ec-embedding of  $B$  and let  $p'$  be a shortest ec-insertion path for  $v$  and  $w$  in  $B$  with respect to  $\Pi'$ . Then  $|p'| \geq |p_1 + \dots + p_k|$ .*

*Proof.* Let  $G_i$ ,  $S_i$ , and  $s_i$  be as defined in OPTIMALECBLOCKINSERTER, and let  $\lambda_\ell^i$  and  $\lambda_r^i$  be the value of  $\lambda_\ell$  and  $\lambda_r$ , respectively, after the  $i$ -th iteration of the for-loop. For  $i = 1, \dots, k$ , we denote with  $H_i$  the pertinent graph of  $\mu_i$ . Observe, that  $\Pi'$  induces embeddings of  $G_i$  and  $S_i$ . Accordingly, we denote the induced embedding of  $G_i$  with  $\Pi'_i$ , and of  $S_i$  with  $\Sigma'_i$ .

Since  $p'$  is a shortest ec-insertion path, it does not visit a face twice. Therefore, we can subdivide  $p'$  into  $p' = p'_1 + \dots + p'_k$  such that  $p'_i$  contains exactly the edges of  $p'$  that are in  $G_i$ , for  $1 \leq i \leq k$ . This follows from the fact that  $H_i$  shares only two vertices with the rest of the graph and  $p'$  does not visit a face twice. For  $1 \leq i < k$ , we denote with  $s'_i \in \{\ell, r\}$  the side at which the ec-insertion path  $p'_1 + \dots + p'_i$  leaves  $H_i$  in  $\Pi'$ .

We show by induction over  $i$  that  $\lambda_{s'_i}^i \leq |p'_1 + \dots + p'_i|$ .

$i = 1$ . If  $k = 1$ , then  $G_1 = B$  and the proposition follows immediately, so assume  $k > 1$ . If  $\mu_1$  is not an R-node, then  $\lambda_{s'_1} = 0$  and the proposition follows immediately. Otherwise, the algorithm also computes the shortest ec-insertion path leaving at side  $s'_1$  in  $\Sigma'_1$ , where the costs of the edges are their traversing costs. Since the traversing costs are independent of the embedding by Lemma 4, we get  $\lambda_{s'_1}^1 \leq |p'_1|$ .

$1 < i < k$ . Assume now that  $\lambda_{s'_j}^j \leq |p'_1 + \dots + p'_j|$  for  $1 \leq j < i$ . We distinguish two cases:

1.  $\mu_i$  is a **P-node**. In this case, we have  $s'_{i-1} = s'_i$ , since  $p_i + \dots + p_k$  does not contain an edge of  $H_{i-1}$ . This yields

$$\lambda_{s'_i}^i = \lambda_{s'_{i-1}}^{i-1} \leq |p'_1 + \dots + p'_{i-1}| \leq |p'_1 + \dots + p'_i|.$$

2.  $\mu_i$  is an **S- or an R-node**. Observe that  $p'_i$  is an ec-insertion path in  $\Pi'_i$  starting in the face at side  $s'_i$  of the edge representing  $v$  and ending in a face at side  $s'_{i+1}$  of the edge representing  $w$  if  $i < k$ , or a face adjacent to  $w$  otherwise. This implies an ec-insertion path in  $\Sigma'_i$ , where the costs of a skeleton edge are its traversing costs. On the other hand, the algorithm computes a shortest ec-insertion path in  $\Sigma'_i$ , since the traversing costs of a skeleton edge are independent of the embedding by Lemma 4. Thus, we get  $\lambda_{s'_i} - \lambda_{s'_{i-1}} \leq |p'_i|$ , and hence

$$\lambda_{s'_i} \leq \lambda_{s'_{i-1}} + |p'_i| \leq |p'_1 + \dots + p'_i|.$$

Finally, we get  $|p_1 + \dots + p_k| = \lambda_{s'_i}^i \leq |p'|$  and the lemma holds.  $\square$

**Theorem 3.** *Let  $B = (V, E)$  be a block of  $K$  and let  $v$  and  $w$  be two distinct vertices of  $B$ . Then, function OPTIMALECBLOCKINSERTER computes an optimal ec-insertion path for  $v$  and  $w$  in  $B$  in time  $O(|E|)$ .*

*Proof.* The correctness and optimality of the algorithm follows from Lemma 5 and Lemma 6. Constructing the SPQR-tree and embedding the skeleton graphs takes time  $O(|E|)$ ; see [15, 19, 8]. Let  $G_i = (V_i, E_i)$  be the graph considered in each iteration of the for-loop. Then, each iteration takes time  $O(|E_i|)$ , since SHORTESTECINSPATH takes only time linear in the size of  $G_i$  by applying BFS. Moreover, the set  $E_i$  consists of some edges  $E'_i$  of  $G$  plus at most two virtual edges (the representatives of  $v$  and  $w$ ). Thus,  $|E_1| + \dots + |E_k| = O(|E|)$ , and hence we get a total running time of  $O(|E|)$ .  $\square$

## 6.4 Generalization to Connected Graphs

The edge insertion algorithm can easily be generalized to connected graphs by using the same technique as in [17] for the unconstrained case; see Alg. 4. For each block  $B_i$  on the path from  $v$  to  $w$  in the block-vertex tree  $\mathcal{B}$  of  $G$ , we compute the optimal ec-edge insertion path  $p_i$  between the representatives of

```

function OPTIMALECINSERTER(ec-expansion  $G$ , vertex  $v$ , vertex  $w$ )
  Compute the block-vertex tree  $\mathcal{B}$  of  $G$ 
  Find the path  $v, B_1, c_1, \dots, B_{k-1}, c_{k-1}, B_k, w$  from  $v$  to  $w$  in  $\mathcal{B}$ .
  for  $i := 1, \dots, k$  do
    let  $x_i$  and  $y_i$  be the representatives of  $v$  and  $w$  in  $B_i$ 
     $p_i :=$  OPTIMALECBLOCKINSERTER( $B_i, x_i, y_i$ )
  end for
  return  $p_1 + \dots + p_k$ 
end function

```

**Algorithm 4:** Computation of an optimal ec-insertion path.

$v$  and  $w$  with a corresponding ec-planar embedding  $\Pi_i$ . Then, we concatenate these ec-edge insertion paths building the optimal ec-edge insertion path for  $v$  and  $w$ .

The correctness proof in [17] uses induction over the number of blocks on the path from  $v$  to  $w$  in  $\mathcal{B}$ . We briefly recall this proof. Let  $B_1, \dots, B_k$  be the blocks on this path and let  $H_i$  be the union of the blocks  $B_1$  to  $B_i$ . Let  $\Pi_i$  be an embedding of  $B_i$  such that  $p_i$  is an optimal edge insertion path for the representatives  $x_i$  and  $y_i$  in  $B_i$  with respect to  $\Pi_i$ . Let  $\Psi_i$  denote the concatenation  $p_1 + \dots + p_i$ .

An embedding  $\Gamma_i$  for  $H_i$  with an optimal edge insertion path  $\Psi_i$  can be iteratively constructed by combining the embedding  $\Gamma_{i-1}$  for  $H_{i-1}$  and the embedding  $\Pi_i$  for block  $B_i$ . Both  $y_{i-1}$  and  $x_i$  denote the same vertex in  $G$  and there exist optimal edge insertion paths  $\Psi_{i-1}$  for  $v_1$  and  $y_{i-1}$  as well as  $p_i$  for  $x_i$  and  $y_i$ . Therefore there is a face  $f \in \Gamma_{i-1}$  that contains  $y_{i-1}$  and either  $v_1$  if  $\Psi_{i-1}$  is empty or the last edge in  $\Psi_{i-1}$ . Similarly, there is a face  $f' \in \Pi_i$  that contains  $x_i$  and either  $y_i$  if  $p_i$  is empty or the first edge in  $p_i$ . We can directly concatenate the two paths if both faces coincide. This can be achieved by choosing  $f$  as the external face of  $\Gamma_{i-1}$  and placing this embedding of  $H_{i-1}$  into face  $f'$  of  $\Pi_i$ . Then  $\Psi_i$  is an optimal ec-insertion path for  $v_1$  and  $y_i$  in  $H_i$  with respect to  $\Gamma_i$ .

We need to show that—under the presence of embedding constraints—ec-planarity is preserved, i.e.,  $\Gamma_k$  is still an ec-planar embedding. The only critical aspect in each step is the selection of  $f$  as the external face; but this does not change the clockwise order of the edges around the vertices of  $G$ . Furthermore, we ensure in the computation of the ec-edge insertion paths  $p_i$  that we do not cross any expansion edges. Hence, we know that the paths  $\Psi_{i-1}$  and  $p_i$  do not start or end in a face containing a hub. Therefore, the ec-planarity conditions are still fulfilled and  $\Gamma_k$  is ec-planar.

It is obvious that  $p_1 + \dots + p_k$  is an ec-edge insertion path for  $v$  and  $w$  with respect to an embedding  $\Pi$  that results from inserting the remaining blocks not contained in  $H_k$  (as shown in the proof of Lemma 3) into  $\Gamma_k$ . The length of the computed ec-edge insertion path is obviously minimal, since a shorter path would imply that there exists a shorter path within a block. The block-

vertex tree of a graph can be constructed in linear time and the running time of  $\text{OPTIMALECBLOCKINSERTER}(B_i, x_i, y_i)$  is linear in the size of the block  $B_i$  (Theorem 3), thus yielding linear running time for  $\text{OPTIMALECINSERTER}$ .

Together with Lemma 1, we obtain the following result:

**Theorem 4.** *Let  $G = (V, E)$  be a graph with embedding constraints  $C$  and  $e = (v, w) \in E$  such that  $G - e$  is ec-planar. Then, we can compute an optimal ec-edge insertion path for  $(v, w)$  in  $G - e$  in  $O(|V| + |E|)$  time.*

## 7 Conclusion and Future Work

We introduced a flexible concept of embedding constraints which allows us to model a wide range of constraints on the order of edges incident to a vertex. We presented a linear time algorithm for testing ec-planarity, as well as a characterization of all possible ec-embeddings. The latter is in particular important for developing algorithms that optimize over the set of all ec-planar embeddings. We showed that optimal edge insertion can still be performed in linear time when embedding constraints have to be respected. In order to devise practically successful graph drawing algorithms, the following problems should be considered:

- Develop faster algorithms for finding ec-planar subgraphs.
- Solve the so-called *orientation problem* for orthogonal graph drawing, e.g., allow us to fix some edges to attach only at the top side of a rectangular vertex. The problem arises when angles are assigned at each vertex between adjacent edges to fix the assignment to the vertex's sides, e.g., in network flow based drawing approaches. The vertices then need to be oriented such that the edges that are assigned to the same sides at different vertices are aligned.
- In some applications, only a subset of the edges is subject to embedding constraints at a vertex  $v$ , i.e., some edges can attach at arbitrary positions. Hence, we wish to extend the concept of embedding constraints for so-called *free edges* that are not contained in the tree  $T_v$ .

## References

- [1] C. Batini, M. Talamo, and R. Tamassia. Computer aided layout of entity relationship diagrams. *Journal of Systems and Software*, 4:163–173, 1984.
- [2] K.-F. Böhringer and F. N. Paulisch. Using constraints to achieve stability in automatic graph layout algorithms. In *Proc. of CHI-90*, pages 43–51, Seattle, WA, 1990.
- [3] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using P-Q tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.
- [4] J. Boyer and W. Myrvold. On the cutting edge: Simplified  $O(n)$  planarity by edge addition. *Journal on Graph Algorithms and Applications*, 8(3):241–273, 2004.
- [5] U. Brandes, M. Eiglsperger, M. Kaufmann, and D. Wagner. Sketch-driven orthogonal graph drawing. In *Proc. 10th Intl. Symp. Graph Drawing (GD 2002)*, volume 2528 of *LNCS*, pages 1–11, 2002.
- [6] U. Brandes and D. Wagner. A bayesian paradigm for dynamic graph layout. In *Graph Drawing*, pages 236–247, 1997.
- [7] C. Buchheim, M. Jünger, A. Menze, and M. Percan. Bimodal crossing minimization. In D. Z. Chen and D. T. Lee, editors, *COCOON*, volume 4112 of *LNCS*, pages 497–506. Springer, 2006.
- [8] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. *Journal of Computer and System Sciences*, 30:54–76, 1985.
- [9] G. Di Battista, W. Didimo, M. Patrignani, and M. Pizzonia. Drawing database schemas. *Software: Practice and Experience*, 32(11):1065–1098, 2002.
- [10] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997, 1996.
- [11] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, third edition, 2005.
- [12] C. Dornheim. Planar graphs with topological constraints. *Journal on Graph Algorithms and Applications*, 6(1):27–66, 2002.
- [13] M. Eiglsperger, U. Fößmeier, and M. Kaufmann. Orthogonal graph drawing with constraints. In *Proc. of the 11th ACM-SIAM Symp. on Discr. Alg.*, pages 3–11, 2000.
- [14] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.

- [15] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR trees. In J. Marks, editor, *Proc. 8th Intl. Symp. Graph Drawing (GD 2000)*, volume 1984 of *LNCS*, pages 77–90. Springer-Verlag, 2001.
- [16] C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In G. Liotta, editor, *Proc. 11th Intl. Symp. Graph Drawing (GD 2003)*, volume 2912 of *LNCS*, pages 13–24. Springer, 2004.
- [17] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.
- [18] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.
- [19] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973.
- [20] K. Mehlhorn and P. Mutzel. On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. *Algorithmica*, 16:233–242, 1996.
- [21] S. C. North. Incremental layout in dynadag. In *Proc. 3rd Intl. Symp. Graph Drawing (GD '95)*, volume 1027 of *LNCS*, pages 409–418. Springer-Verlag, 1996.
- [22] R. Tamassia. Constraints in graph drawing algorithms. *Constraints*, 3(1):87–120, 1998.
- [23] W. T. Tutte. *Connectivity in graphs*, volume 15 of *Mathematical Expositions*. University of Toronto Press, 1966.