Planarity Testing and Optimal Edge Insertion with Embedding Constraints

Carsten Gutwenger, Karsten Klein, and Petra Mutzel

University of Dortmund, Germany
{carsten.gutwenger,karsten.klein,petra.mutzel}@cs.uni-dortmund.de

Abstract. Many practical applications demand additional restrictions on an admissible planar embedding. In particular, constraints on the permitted (clockwise) order of the edges around a vertex, like so-called *side constraints*, abound. In this paper, we introduce a set of hierarchical embedding constraints that also comprises side constraints. We present linear time algorithms for testing if a graph is *ec-planar*, i.e., admits a planar embedding satisfying the given embedding constraints, as well as for computing such an embedding. Moreover, we characterize the set of all possible ec-planar embeddings and consider the problem of finding a planar combinatorial embedding of a planar graph such that an additional edge can be inserted with the minimum number of crossings; we show that this problem can still be solved in linear time under the additional restrictions of embedding constraints.

1 Introduction

Graphs are used in numerous application domains for visualizing information. Examples include software engineering, data bases, business process modeling, VLSI-design, and bioinformatics. In many cases, application specific layout rules have to be observed which impose restrictions on an admissible graph layout. Consequently, automatic layout systems have to respect these restrictions 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.

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. Such constraints occur, e.g., in 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. 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.

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, 15, 4, 3]), there is only few previous work concerning constraints on the admissible embeddings of a graph. Di Battista et al. [6] consider embedding constraints that appear in database schemas, and Dornheim [8] studies the problem of computing embeddings satisfying topological constraints, where prescribed edges have to be embedded inside or outside of a cycle, repsectively. On the other hand, linear time complexity for planarity testing and embedding has been shown in [14, 5].

Our contribution is a linear time algorithm for testing if a graph with a set of embedding constraints is ec-planar; see Sect. 5. The main challenge is to incorporate oriented constraints, where a given clockwise order 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.

An important optimization goal for laying out graphs is the minimization of crossings. The problem of minimizing the number of crossings in a drawing is NP-hard [9] and no practically efficient method exists so far. The *planarization* approach for crossing minimization first deletes a number of edges until the remaining graph is planar and then carefully reinserts these edges so that the number of crossings is minimized; see [12]. In [13], the problem of optimally inserting an additional edge between vertices v and w into a planar graph G is considered and an algorithm to solve the problem in linear time is given. The algorithm first computes the SPQR-tree $\mathcal{T}$ of G and a shortest path $\varPsi$ between nodes in $\mathcal{T}$ whose skeletons contain v and w, respectively. The optimal insertion path is constructed by simply concatenating locally optimal insertion paths of the tree nodes on $\varPsi$. When embedding constraints have to be considered, locally optimal solutions need not lead to globally optimal solutions and the greedy approach cannot be applied anymore. In Sect. 6, we give a linear time algorithm to solve the optimal edge insertion problem under the presence of embedding constraints. Given an ec-planar graph G with embedding constraints C and an additional edge e, our algorithm computes an ec-planar embedding of G with respect to C, together with a crossing minimal insertion path for e.

All the proofs omitted in this paper can be found in [10].

2 Preliminaries

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. If G

is 2-connected, its *SPQR-tree* $\mathcal{T}$ represents the decomposition of G into its 3-connected components comprising serial, parallel, and 3-connected structures; see [7] for a formal definition. The respective structure is given by a skeleton graph associated with each tree node, which is either a cycle (S-node), a bundle of parallel edges (P-node), or a 3-connected simple graph (R-node). We denote with $skeleton(\mu)$ the skeleton graph associated with node μ. In addition, Q-nodes serve as representatives for the edges of G. For each vertex v of G, the nodes in $\mathcal{T}$ whose skeletons contain v are called the *allocation nodes* 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 incident edges of 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 incident edges of v. We consider this order as a cyclic order and represent all *admissible* cyclic, clockwise orders of the incident edges of 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.

Fig. 1 gives an example. A c-node with a single child is obviously redundant, therefore we demand that each c-node has at least two children.

Let C be a set of embedding constraints at distinct vertices of G. A combinatorial embedding Γ 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 Γ 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

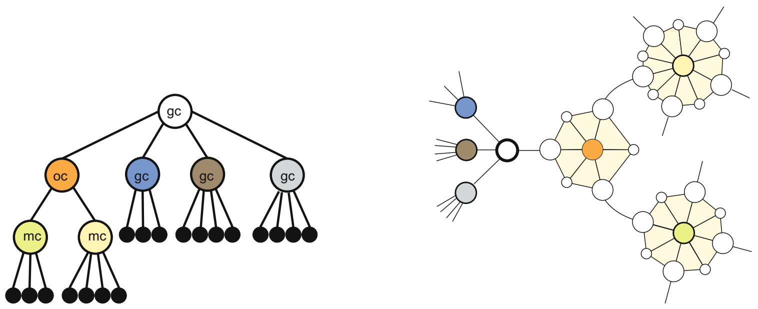

Fig. 1. Embedding constraint T_v (left) and the corresponding expansion (right)

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 to 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 $2d$ spokes, were $e_1, \ldots, e_d$ are the edges incident to u. Then, the respective wheel gadget consists of a cycle $x_1, y_1, \ldots, x_d, y_d$ of length $2d$ and a vertex, called *hub*, incident to every vertex on the cycle. 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*. Fig. 1 shows a constraint tree and the corresponding expansion of the vertex. $E(G, C)$ can be constructed in linear time and its size is also linear in the size of G.

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 incident to the hub is a triangle; we call these faces *inner wheel gadget faces*.

4.2 ec-Expansion and ec-Planar Embeddings

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 Γ of $E(G, C)$ *ec-planar* if

1. the external face of Γ 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 Γ corresponds to the order specified by the corresponding oc-node.

Let Γ be an ec-planar embedding of $E(G, C)$. Then, we obtain an ec-planar embedding of (G, C) as follows. For each vertex $v \in G$, 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 rest of the graph, i.e., the graph induced by the vertices not contained in G_v. The conditions above assure that the planar embedding Γ_v of G_v induced by Γ is such that $\bar{G}_v$ lies in the external face of Γ_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 1. *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

Though it is sufficient to test each block of a graph separately for planarity, this is not the case for ec-planarity. However, it is sufficient to test the blocks of the ec-expansion separately as the following lemma shows.

Lemma 2. *$E(G, C)$ is ec-planar iff 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. Since a wheel gadget $\mathscr{G}$ in $E(G, C)$ is 3-connected, $\mathscr{G}$ is completely contained in a single block B of $E(G, C)$ and therefore also the hub of $\mathscr{G}$ is not a cut vertex of $E(G, C)$. For each edge $(u, v) \in \mathscr{G}$, the pair $\{u, v\}$ is not a separation pair in B by construction, hence every inner wheel face of $\mathscr{G}$ is also a face in every planar embedding of B.

We construct an ec-planar embedding of $E(G, C)$ starting with an arbitrary block B of $E(G, C)$. Let Π be an ec-planar embedding of B. We add the remaining blocks successively to Π. Let B' be another block of $E(G, C)$ that shares a vertex c with B, and let Π' be an ec-embedding of B'. Since c cannot be an O- or M-hub, we can pick faces $f \in \Pi$ and $f' \in \Pi'$ that are incident to c and not inner wheel gadget faces. We insert Π' with f' as external face into the face f of Π. This results in an ec-planar embedding of $B \cup B'$. We add the remaining blocks in the same way, resulting in an ec-planar embedding of $E(G, C)$. □

If we can characterize all ec-planar embeddings of the blocks of $E(G, C)$, the construction in the proof of Lemma 2 also shows, 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.*

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 iff 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 iff 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 choose arbitrary embeddings for the remaining skeletons. This holds, since every such embedding Π 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 gadget faces are preserved. We can pick any face of Π as external face which is not an inner wheel face (such a face always exists) and obtain an ec-planar embedding of B. □

Function IsEcPlanar depicted in Alg. 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 IsEcPlanar(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 $skeleton(\mu)$ 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, algorithm* IsEcPlanar *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 1 and 2, it is sufficient to test every block of $E(G, C)$ for ec-planarity. Hence, the correctness of Alg. 1 follows from Theorem 1.

Constructing the ec-expansion and testing planarity [14] 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 [11]. 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., [5]), we need linear running time for each block. Hence, the total running time is linear in the size of $E(G, C)$, which is $O(|V| + |E|)$.

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 2. □

6 ec-Edge Insertion

We first generalize the terms insertion path and traversing costs introduced in [13]. 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 Π of G is a sequence of edges $e_1, \ldots, e_k$ 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 Π 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 Π 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 1 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′. $e_1, \ldots, e_k$ 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 [13] to ec-planarity. Let e be a skeleton edge, and let Π be an arbitrary ec-embedding of the graph *expansion*$^+(e)$ (which is the expansion graph of e plus the edge e) with dual graph Π^*, in which all edges corresponding to gadget edges have length ∞ and the other edges have length 1. Let f_1 and f_2 be the two faces in Π separated by e. We denote with $P(\Pi^*, e)$ the length of the shortest path in Π^* that connects f_1 and f_2 and does not use the dual edge of e. Hence, we have $P(\Pi, e) \in \mathbb{N} \cup \{\infty\}$.

Lemma 3. *Let μ be a node in $\mathcal{T}$ and let e be an edge in skeleton(μ). Then, the length of the path $P(\Pi^*, e)$ is independent of the ec-embedding Π of expansion$^+$ (e).*

Proof. Let m be the number of edges in $G_e :=$ *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 Π of G_e corresponds to an embedding Π' of G'_e, and the length of the path $P(\Pi, e)$ is ∞ if and only if the corresponding path in Π' is longer than m. Applying Lemma 1 in [13] 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 the length of the path $P(\Pi^*, e)$ for an arbitrary ec-embedding Π of *expansion*$^+(e)$.

The hard part of the algorithm 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. In particular, we are not allowed to cross expansion edges of B. The function OptimalEcBlockInserter shown in Alg. 2 and 3 solves this problem.

It starts by computing the SPQR-tree $\mathcal{T}$ of B and embeds the skeletons such that they imply an ec-embedding of B, i.e, the R-node skeletons are embedded

```
procedure 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 $k > 1$).
    $\lambda_\ell := \lambda_r := 0$        ▷ length of shortest insertion path leaving to the left/ right
    for $i = 1, \dots, k$ do
        let $S_i = skeleton(\mu_i)$
        let $G_i$ be the graph obtained from $S_i$ by replacing each edge not repre-
        senting $v$ or $w$ with its expansion graph, and let $\Pi_i$ be the embedding of
        $G_i$ induced by the embeddings of the skeletons of $\mathcal{T}$.
        if $\mu_i$ is a P-node then
            $(\phi^i_\ell, \Delta^i_\ell) := (\ell, \epsilon)$; $(\phi^i_r, \Delta^i_r) := (r, \epsilon)$        ▷ no crossings required
        else                                                    ▷ S- or R-node
            if $i = 1$ then
                $L_v := R_v :=$ the set of incident faces of the copy of $v$ in $S_i$
            else
                let $e_v$ be the representative of $v$ in $S_i$
                $L_v := \{$ the left face of $e_v\}$
                $R_v := \{$ the right face of $e_v\}$
            end if
            if $i = k$ then
                $L_w := R_w :=$ the set of incident faces of the copy of $w$ in $S_i$
            else
                let $e_w$ be the representative of $w$ in $S_i$
                $L_w := \{$ the left face of $e_w\}$
                $R_w := \{$ the right face of $e_w\}$
            end if                                      ▷ continued on next page...
```

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

correctly. Then, the shortest path $\Upsilon := \mu_1, \dots, \mu_k$ between an allocation node of v and of w is identified. In order to achieve a consistent orientation, we root $\mathcal{T}$ such that Υ is a descending path in the tree, i.e., μ_i is the parent of μ_{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., [7]. This direction is necessary in order to identify the left and the right face of an edge.

The algorithm traverses the path Υ from μ_1 to μ_{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 μ_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 μ_i. These lengths are maintained in the variables λ_ℓ and λ_r. Finally, when node μ_k is considered, this information is used to determine a shortest insertion path ending at w.

```
            ▷ Compute shortest ec-insertion paths (from l/r to l/r) within $G_i$.
            ▷ Note: $p_{\ell r} = p_{\ell\ell}$ and $p_{rr} = p_{r\ell}$ if $i \in \{1, k\}$.
            $p_{\ell r} :=$ ShortestEcInsPath($\Pi_i, L_v, L_w$)
            $p_{\ell\ell} :=$ ShortestEcInsPath($\Pi_i, L_v, R_w$)
            $p_{rr} :=$ ShortestEcInsPath($\Pi_i, R_v, L_w$)
            $p_{r\ell} :=$ ShortestEcInsPath($\Pi_i, R_v, R_w$)
            ▷ 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.
            $(\lambda_\ell, \phi^i_\ell, \Delta^i_\ell) := \min_{1,3} \Lambda_\ell$
            $(\lambda_r, \phi^i_r, \Delta^i_r) := \min_{1,3} \Lambda_r$
        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.
        $p_i := \Delta^i_{s_i}$;  $s_{i-1} := \phi^i_{s_i}$
    end for
    return $p_1 + \cdots + p_k$
end procedure
```

Algorithm 3. Procedure OptimalEcBlockInserter (part 2)

For each node μ_i, the following information is computed:

- ϕ^i_ℓ (resp. ϕ^i_r) 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 ℓ) or to the right (the value is r).
- Δ^i_ℓ (resp. Δ^i_r) 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 procedure. 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 for-loop starts by expanding all edges of the skeleton S_i of μ_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}.

If μ_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 incident to v; if $i = k$ then G_i contains w and the ec-insertion path may end in any face incident to w. We compute the (at most) four possible shortest ec-insertion paths using the function ShortestEcInsPath(Π, F_s, F_t). Here Π 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 BFS in the dual graph of Π, 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 μ_i:

– We leave P_{i-1} to the left (or start at v if $i = 1$) and end in a face in R_w (e.g., 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 (e.g., 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 μ_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 Π_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 μ_i, it is easy to reconstruct the best ec-insertion path from v to w using $\phi^i_{\ell/r}$ and $\Delta^i_{\ell/r}$. Notice that $\lambda_\ell = \lambda_r$ holds at the end, since $L^k_w = R^k_w$.

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 $O(|E|)$.*

The edge insertion algorithm can easily be generalized to connected graphs by using the same technique as in [13] for the unconstrained edge insertion.

7 Conclusion and Future Work

We introduced a flexible concept of embedding constraints which allows to model a wide range of constraints on the order of incident edges. 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:

- Incorporate the concept of embedding constraints into the planarization approach [1, 12] so that also non-ec-planar graphs can be handled. In particular, algorithms for finding ec-planar subgraphs are required; this problem can, e.g., be solved in quadratic time using successive ec-planarity testing.
- Solve the so-called *orientation problem* for orthogonal graph drawing, e.g., allow to fix some edges to attach only at the top side of a rectangular vertex.
- 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, 1990.

[3] U. Brandes, M. Eiglsperger, M. Kaufmann, and D. Wagner. Sketch-driven orthogonal graph drawing. In *Proc. GD 2002*, volume 2528 of *LNCS*, pages 1–11, 2002.

[4] Ulrik Brandes and Dorothea Wagner. A bayesian paradigm for dynamic graph layout. In *Proc. GD '97*, volume 1353 of *LNCS*, pages 236–247, 1997.

[5] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. *J. Computer and System Sciences*, 30:54–76, 1985.

[6] G. Di Battista, W. Didimo, M. Patrignani, and M. Pizzonia. Drawing database schemas. *Softw. Pract. Exper.*, 32(11):1065–1098, 2002.

[7] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5):956–997, 1996.

[8] C. Dornheim. Planar graphs with topological constraints. *J. Graph Algorithms Appl*, 6(1):27–66, 2002.

[9] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.

[10] C. Gutwenger, K. Klein, and P. Mutzel. Planarity testing and optimal edge insertion with embedding constraints. Technical Report TR06-1-005, Chair of Algorithm Engineering, University of Dortmund, 2006.

[11] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR trees. In J. Marks, editor, *Proc. GD 2000*, volume 1984 of *LNCS*, pages 77–90, 2001.

[12] C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In G. Liotta, editor, *Proc. GD 2003*, volume 2912 of *LNCS*, pages 13–24, 2004.

[13] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.

[14] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[15] Stephen C. North. Incremental layout in DynaDAG. In *Proc. GD '95*, volume 1027 of *LNCS*, pages 409–418, 1996.