On Bend-Minimized Orthogonal Drawings of Planar 3-Graphs

Yi-Jun Chang[*1] **and Hsu-Chun Yen**[†2]

1 **Department of EECS, University of Michigan, Ann Arbor, MI, USA**
2 **Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan**

Abstract

An *orthogonal drawing* of a graph is a planar drawing where each edge is drawn as a sequence of horizontal and vertical line segments. Finding a bend-minimized orthogonal drawing of a planar graph of maximum degree 4 is NP-hard. The problem becomes tractable for planar graphs of maximum degree 3, and the fastest known algorithm takes $O(n^5 \log n)$ time. Whether a faster algorithm exists has been a long-standing open problem in graph drawing. In this paper we present an algorithm that takes only $\tilde{O}(n^{17/7})$ time, which is a significant improvement over the previous state of the art.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Bend minimization, graph drawing, orthogonal drawing, planar graph

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.29

1 Introduction

An *orthogonal drawing* of a graph is a planar drawing where each edge is composed of a sequence of horizontal and vertical line segments with no crossings. Orthogonal drawings appear in many applications such as the automation of VLSI circuit layout and the drawing of diagrams in information systems. Variants of orthogonal drawings have been introduced in the literature to cope with different constraints or to improve the readability and aesthetic feel: smoothing the drawing [2, 1], requiring orthogonal convexity [5], accommodating vertices of more than 4 incident edges [16, 8], and restricting directions of vertices [11, 13]. Refer to [12] for a survey on orthogonal drawings.

Bend-minimization is one of the most classical optimization problems on orthogonal drawings. Given a planar (or plane) graph, the problem asks for an orthogonal drawing with the total number of bends minimized. However, the problem is NP-hard for planar 4-graphs [15].[1] To obtain polynomial time algorithms, one has to relax the problem one way or another. For example, it is known that a polynomial time algorithm exists when the first bend on an edge does not incur any cost [3].

Much research effort has been made on bend-minimization for subclasses of planar 4-graphs [9, 7, 14, 18, 17]. The two most *natural* subclasses are planar 3-graphs (reducing

* Yi-Jun Chang was supported in part by NSF grant CCF-1514383.
† Hsu-Chun Yen was supported in part by Ministry of Science and Technology, Taiwan, under grant MOST 103-2221-E-002-154-MY3.
[1] We write k-graphs to denote graphs of maximum degree k. Note that the degree of each vertex cannot exceed 4 in an orthogonal drawing, and hence planar 4-graphs are the most general graph class that can be drawn orthogonally.

the maximum degree by 1) and plane 4-graphs (fixing the planar embedding). For plane 4-graphs, a seminal work by Tamassia [20] demonstrates a reduction from bend-minimization to a computation of a min-cost flow. Following this approach, the runtime has been reduced to $O(n^{7/4}\sqrt{\log n})$ [14], and subsequently to $O(n^{1.5})$ [7].[2]

For planar 3-graphs, the fastest known algorithm for bend-minimization is the $O(n^5 \log n)$ time algorithm designed by Di Battista, Liotta, and Vargiu, which dates back to 1998 [9]. As stated as an open problem in [4], further improving the time complexity is identified as an important issue in the field of graph drawing.

Problem 15: *Let G be a planar graph whose vertices have degree at most three. Is there an algorithm to compute a planar bend-minimum orthogonal drawing of G in $o(n^5 \log n)$ time?*

In this paper, we answer the problem affirmatively by demonstrating an $\tilde{O}(n^{17/7})$ time[3] algorithm. Precisely, our algorithm takes $O(n \cdot T(n))$ time, where $T(n)$ is the time complexity of constructing a bend-minimized orthogonal drawing of a plane 3-graph subject to the constraint that some designated edges have no bend. We will later see that bend-minimization of a plane 3-graph can be reduced to min-cost flow of *constant* capacity, and a recent breakthrough on unit-capacity min-cost flow in sparse graphs [6] implies $T(n) = \tilde{O}(n^{10/7})$.

The main challenge of designing a bend-minimization algorithm for planar 3-graphs is to handle the transition from the *variable* embedding setting to the *fixed* embedding setting. The naïve approach of testing *all* possible planar embeddings is very inefficient, as there can be an exponential number of different planar embeddings. A natural way to approach this problem is to devise a dynamic programming procedure on an SPQR-tree, which is a tree structure capable of storing all possible embeddings of a planar graph using linear space. The approach is briefly sketched as follows. By "contracting" all subgraphs of the planar graph G that can be flipped, a graph G' that has a fixed combinatorial embedding is obtained. An optimal drawing of G' can be computed quickly using a bend-minimization algorithm for plane 3-graphs. The optimal drawings of the contracted subgraphs are computed recursively. Merging these drawings yields a drawing of G. Using the terminology of SPQR-trees, if G is the *pertinent graph* of a node μ in the SPQR-tree, then the graph G' is (a subdivision of) the *skeleton* of μ, and the contracted subgraphs are the pertinent graphs of some descendants of μ. See Fig. 1 for a conceptual example: (1) $G' = H_2$ is resulting from contracting the subgraph H_1 into a vertex v in G. (2) Merging the drawings of H_1 and H_2 yields an orthogonal drawing of G (treating each white dot in the figure as a bend).

The above strategy does not immediately give us a bend-minimization algorithm. Observe that the outer boundary of G needs to have 4 convex corners. Thus we need an additional constraint which requires that the drawing of G' and the drawings computed by recursive calls jointly supply 4 convex corners in the outer boundary. To ensure that the final drawing of G is optimal, one has to compute *multiple* drawings of each contracted subgraph H subject to different *constraints* on the number of convex corners in the outer boundary of G that H can supply, and to examine all possible combinations of these constraints. In [9] the notion of *spirality*, which measures how an orthogonal drawing is "rolled up", is developed to serve as the aforementioned constraints.

In this paper, we utilize some tools developed by Rahman et al. in [19]. They characterized the condition for the existence of a no-bend orthogonal drawing based on the number of

[2] Throughout the paper, we define $n := |V(G)|$ as the number of vertices in the graph. Note that we have $|E(G)| = O(n)$ for planar graphs.
[3] The $\tilde{O}(\cdot)$ notation suppresses any poly $\log n$ factor.

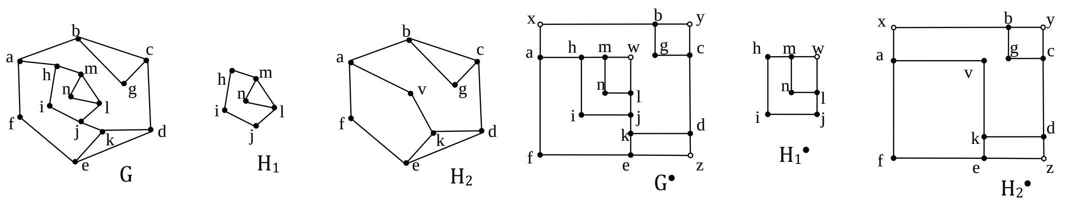

Figure 1 Contracting subgraphs and merging sub-drawings.

2-vertices[4] on some cycles, and they gave a linear time algorithm to construct such a drawing if one exists. Note that bend-minimization can be equated with finding a minimum number of subdivisions and a planar embedding to meet the condition for a no-bend orthogonal drawing to exist.

We first reduce the bend-minimization problem to a constrained version, and then we use the SPQR-tree dynamic programming to solve the constrained version of the bend-minimization problem recursively. In our algorithm, for each contracted subgraph, only a *constant* number of recursive calls is needed, and the merging of the subdrawings can be performed in time *linear* to the number of contracted subgraphs. Our algorithm is presented in a top-down manner. The main algorithm (for biconnected planar 3-graphs) is described in Sec. 3, and the details of some subroutines are left in Sec. 4 and 5. The SPQR-tree implementation is described in Sec. 6. Our algorithm can also be extended to planar 3-graphs that are not biconnected based on block-cutvertex tree. The detail is omitted due to space limit.

It is expected that our approach be applicable to some other variants of orthogonal drawings, such as the orthogonally convex drawing [5], as its no-bend version can be characterized analogously along the line of the work by Rahman at el. [19].

2 Preliminaries

Given a graph $G = (V, E)$, we write $\Delta(G)$ to denote the maximum degree of G. We write $V(G)$ and $E(G)$ to denote the sets of vertices and edges of G, respectively. We call G a d-graph if $\Delta(G) \leq d$. A vertex of degree d is called a d-vertex. A *multi-graph* is a graph where self-loops are disallowed while multi-edges are allowed. Throughout the paper, all graphs under consideration are planar 3-graphs with possibly multi-edges. Unless otherwise stated, all cycles and paths are assumed to be *simple* in the sense that they do not have repeated vertices.

A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We write $H \subseteq G$ to denote the subgraph relation. We write $H \subsetneq G$ if $H \subseteq G$ and $H \neq G$. For $H \subsetneq G$, we write $G - H$ to denote the graph resulting from removing from G all vertices in H and the edges incident to these vertices. A graph is *planar* if it can be drawn on a plane without any edge crossing. A planar drawing partitions the plane into several disconnected regions called *faces*. The face corresponding to the region of unbounded size is called the *outer face*. The remaining ones are called *inner faces*. A *facial cycle* is a cycle surrounding a face. All facial cycles are simple in a biconnected plane graph. A *plane graph* is a planar graph with a fixed combinatorial embedding (which specifies a cyclic order of the edges incident to each vertex in the planar embedding) and a designated *outer cycle* C_O that surrounds the outer face.

[4] We write a k-vertex to denote a vertex of degree k.

We call a cycle *inner* if it is not the outer one. For any vertex and edge, we call it *boundary* if it is located in C_O; otherwise, it is *non-boundary.* A cycle $C \neq C_O$ is called *boundary* if it contains some edges in C_O. For a cycle C in a plane graph G, we write $G[C]$ to denote the subgraph of G that contains exactly C and vertices and edges residing in its interior region. Note that $G[C] = G$ iff $C = C_O$; and $G[C] = C$ iff C is an inner facial cycle.

With respect to a cycle C of a plane graph G, an edge $e = \{u, v\} \notin E(G[C])$ is called a *leg* of C if at least one of u and v belongs to $V(C)$. A vertex in $V(C)$ incident to some leg of C is called a *legged-vertex* of C. In a 3-graph, each legged-vertex of C is incident to exactly one leg of C. A cycle C is *k-legged* if C has exactly k legs.

Consider the plane graph G in Fig. 1. The cycle $C_1 = (h, m, l, j, i)$ is a non-boundary 2-legged cycle of which the two legged-vertices are h and j, and the two legs are $\{a, h\}$ and $\{j, k\}$. The facial cycle $C_2 = (d, e, k)$ is a boundary 3-legged cycle in G. A *subdivision* is a process of adding a new 2-vertex w to an edge $e = \{u, v\}$ by replacing e with two edges $\{u, w\}$ and $\{w, v\}$. A *smoothing* is the reverse process of a subdivision which removes a 2-vertex w by replacing two edges $\{u, w\}$ and $\{w, v\}$ with a new edge $\{u, v\}$, assuming $\{u, w\}, \{w, v\} \in E(G)$.

2.1 Theorems of Rahman et al.

Rahman st al. [19] gave a characterization of those biconnected plane 3-graphs that admit orthogonal drawings without bends. Based on the characterization, they presented a linear time algorithm to construct such a drawing, if one exists.

▶ **Theorem 1** ([19]). *A biconnected plane 3-graph G admits a no-bend orthogonal drawing if and only if the following conditions are met:*
1. *The outer cycle C_O contains at least four 2-vertices.*
2. *Each 2-legged cycle contains at least two 2-vertices.*
3. *Each 3-legged cycle contains at least one 2-vertex.*

The conditions in Theorem 1 can be reformulated as a single condition requiring the number of 2-vertices plus the number of legged-vertices to be at least 4 in each cycle. Note that a cycle is drawn as an orthogonal polygon in an orthogonal drawing. Since an orthogonal polygon must have at least four 90° corners, the necessity of the conditions in Theorem 1 follows from the fact that only 2-vertices and legged-vertices can be drawn as 90° corners of a cycle. Corollary 2 is an immediate consequence of Theorem 1.

▶ **Corollary 2.** *A biconnected planar 3-graph G admits an orthogonal drawing using x bends if and only if there is a plane graph $G^\bullet$ which is a subdivision of G with $|V(G^\bullet)| - |V(G)| = x$ meeting the three conditions in Theorem 1.*

To better understand Theorem 1 and Corollary 2, consider the plane graph G in Fig. 1, the non-boundary 2-legged cycle $C_1 = (h, m, l, j, i)$, and the boundary 3-legged cycle $C_2 = (d, e, k)$. As $C_O(G)$ contains only one 2-vertex f, C_1 contains only one 2-vertex i, and $V(C_2)$ contains no 2-vertex, all three conditions in Theorem 1 are violated. The plane graph $G^\bullet$ in Fig. 1, which results from making 3 subdivisions in G, fulfills the three conditions. A no-bend orthogonal drawing of $G^\bullet$ can be seen as an orthogonal drawing of G with 3 bends.

Observe that in the linear time drawing algorithm of Rahman et al. [19], it is possible to choose *any* set of four 2-vertices in C_O as four convex corners in the outer boundary.[5] Hence we have the following theorem.

[5] The outer boundary refers to the orthogonal polygon corresponding to C_O in the drawing. Note that a convex corner in the outer boundary is also a concave corner of the outer face.

▶ **Theorem 3** ([19]). *Let G be a biconnected plane 3-graph that admits a no-bend orthogonal drawing, and let S be a set of at most four 2-vertices in C_O. Then there exists a no-bend orthogonal drawing of G in which all vertices in S are convex corners in the outer boundary.*

2.2 Orthogonal Representations and Min Cost Flow Formulation.

Let G be a biconnected plane 3-graph that admits a no-bend orthogonal drawing. A naïve way of describing a no-bend orthogonal drawing of G is to specify the actual coordinates of all vertices. The concept of an *orthogonal representation* [20] allows us to describe the shape of an orthogonal drawing, expressed in terms of angles around the vertices without reporting any information about the actual coordinates of the vertices.[6] Formally speaking, an orthogonal representation consists of assigning an angle $\theta \in \{90°, 180°, 270°\}$ to each pair (v, F) such that the vertex v belongs to the cycle surrounding face F. This indicates that v is a degree θ corner in face F. It is required that the summation of all angles around a vertex is 360°, and the difference between the number of convex corners and the number of concave corners is 4 in each inner face, and is -4 in the outer face. Given an orthogonal representation meeting the above requirements, in $O(n)$ time a no-bend orthogonal drawing realizing the angle assigned to each corner can be constructed [20].

In view of the above, the task of bend-minimization of a biconnected plane 3-graph reduces to applying a minimum number of subdivisions to make the graph to have an orthogonal representation. This can be formulated as a min-cost flow problem [20]. Each vertex v of degree k supplies $4 - k$ units of flow. Each inner face F consumes $p - 4$ units of flow, where p is the number of vertices surrounding F. The outer face F_O consumes $p + 4$ units of flow, where p is the number of vertices in C_O. For each pair (v, F) such that v belongs to the cycle surrounding F, add an arc (v, F) with capacity 2 and cost 0. Flowing k units of flow from v to F indicates that v is a $90(k + 1)°$ corner in F. For each edge e which borders the two faces F and F', add two arcs (F, F') and (F', F) with capacity 4 and cost 1. Flowing k units of flow from F to F' along the arc associated with e indicates that the edge e is subdivided k times, and these k new 2-vertices are convex corners in F and concave corners in F'. Since Theorem 1 implies that subdividing an edge more than 4 times makes no use, it is fine to set the capacity of these arcs to 4. It is straightforward to verify that any feasible flow corresponds to an orthogonal representation, and the cost of the flow equals the number of 2-vertices introduced by subdivisions. The flow network can be made unit-capacity by emulating an arc of capacity k by k arcs with capacity 1. Since the total number of arcs in the flow network is $m = O(n)$, and since the maximum cost of an arc is $W = 1$, the min-cost flow algorithm of [6] solves the bend-minimization problem of a biconnected plane 3-graph in $\tilde{O}(m^{10/7} \log W) = \tilde{O}(n^{10/7})$ time.

▶ **Theorem 4.** *A bend-minimized orthogonal drawing of a biconnected plane 3-graph can be constructed in $T(n) = \tilde{O}(n^{10/7})$ time.*

We are not aware of any unit-capacity min-cost flow formulation of bend-minimization of plane 4-graphs, so the $O(n^{1.5})$ time algorithm of [7] remains state-of-the-art.

[6] Here we only describe a simplified version of orthogonal representations that works for no-bend drawings of biconnected plane 3-graphs. See [20] for a complete definition that handles bends and general plane graphs.

```
Algorithm 1: Min-Bend-Draw($G, s$).
  Input: (1) a biconnected planar 3-graph $G$ that is not a cycle, (2) a 2-vertex $s \in V(G)$
  Output: a bend-minimized orthogonal drawing of $G$ subject to the condition that $s$
          belongs to the outer boundary
1 Let $u$ and $v$ be the two neighbors of $s$.
2 for each $b \in \{1, 2, 3\}$ do
3     $\tilde{G} \leftarrow$ the graph resulting from replacing $(u, s, v)$ with a path $P = (u, s_1, \ldots, s_b, v)$.
4     $G^\bullet \leftarrow$ Subdiv-Embed$(\tilde{G}, P)$.
5     Compute a no-bend orthogonal drawing of $G^\bullet$ (which can be seen as an orthogonal
       drawing of $G$).
6 Among all drawings computed, return the one that uses the minimum number of bends.
```

3 The Main Algorithm

In this section we present our main algorithm, which applies $O(n)$ calls to a subroutine that solves a *constrained* version of the bend-minimization problem. Let G be a biconnected planar 3-graph, and P be a u–v path such that $V(P) \setminus \{u, v\}$ contains only 2-vertices. A *P-orthogonal drawing* of G is an orthogonal drawing of G such that the outer cycle contains P as a subpath, and no bend is imposed on P. A P-orthogonal drawing for a biconnected plane 3-graph G, with $P \subseteq C_O(G)$, is defined analogously.

The procedure Subdiv-Embed(G, P), which is described in the next section, computes a plane graph $G^\bullet$ such that any no-bend orthogonal drawing of $G^\bullet$ is a bend-minimized P-orthogonal drawing of G. More precisely, the plane graph $G^\bullet$ is required to meet the following conditions:

- $G^\bullet$ is a subdivision of G, and no subdivision is made on the path P.
- P belongs to the outer cycle of $G^\bullet$.
- $G^\bullet$ admits a no-bend orthogonal drawing.
- Among all plane graphs meeting the above 3 criteria, $G^\bullet$ is chosen such that $|V(G^\bullet)| - |V(G)|$ is minimized.

To describe our main algorithm, we first note the following result.

▶ **Lemma 5.** *For any orthogonal drawing of a planar graph G, at least one of the following is true: (1) the outer boundary contains a 2-vertex $v \in V(G)$; (2) the outer boundary contains an edge $e \in E(G)$ that has at least one bend.*

Proof. If no 2-vertex of G belongs to C_O, there must be at least 4 bends in the outer cycle to serve as 4 convex corners of the orthogonal polygon corresponding to C_O. ◀

Using Subdiv-Embed as a blackbox, Algorithm 1 and Algorithm 2 compute bend-minimized orthogonal drawings subject to the two cases of Lemma 5. We remark that the reason that we do not need to consider the case of $b > 3$ in these two algorithms is due to Lemma 7. Based on these two algorithms, Algorithm 3, which is the main algorithm of the paper, computes a bend-minimized orthogonal drawing of a biconnected planar 3-graph G that is not a cycle. See Fig. 2 for an illustration of Min-Bend-Draw. The subdivisions introduced in the procedure Subdiv-Embed are drawn as white dots.

Let G be a plane graph, and let $H = G[C]$ be a subgraph of G such that C is a 2-legged cycle. Define *flipping* H as the operation that reverses the cyclic order of the edges incident to v in the combinatorial embedding of G, for each $v \in V(H)$. As long as C is 2-legged, the

Algorithm 2: Min-Bend-Draw(G, e).

```
Input: (1) a biconnected planar 3-graph $G$ that is not a cycle, (2) an edge
       $e = \{u, v\} \in E(G)$
Output: a bend-minimized orthogonal drawing of $G$ subject to the condition that $e$
        has at least one bend and belongs to the outer boundary
1 for each $b \in \{1, 2, 3\}$ do
2     $\tilde{G} \leftarrow$ the graph resulting from replacing $(u, v)$ with a path $P = (u, s_1, \ldots, s_b, v)$.
3     $G^\bullet \leftarrow$ Subdiv-Embed$(\tilde{G}, P)$.
4     Compute a no-bend orthogonal drawing of $G^\bullet$ (which can be seen as an orthogonal
       drawing of $G$).
5 Among all drawings computed, return the one that uses the minimum number of bends.
```

Algorithm 3: Min-Bend-Draw(G).

```
Input: a biconnected planar 3-graph $G$ that is not a cycle
Output: a bend-minimized orthogonal drawing of $G$
1 For each $e \in E(G)$, run Min-Bend-Draw$(G, e)$.
2 For each 2-vertex $s \in V(G)$, run Min-Bend-Draw$(G, s)$.
3 Among all drawings computed, return the one that uses the minimum number of bends.
```

flipping operation preserves the planarity of G. For example, in Fig. 3 the plane graph G_2 is resulting from flipping $H = G[C]$ in the plane graph G_1, where $C = (c, k, l, h, i, e, d)$. To prove the correctness of Min-Bend-Draw(G), we need the following lemmas.

▶ **Lemma 6.** *Let G be a biconnected plane 3-graph admitting a no-bend orthogonal drawing. Let C be a boundary 2-legged cycle that has no boundary 2-vertex. Then flipping $G[C]$ preserves the property of having a no-bend orthogonal drawing.*

▶ **Lemma 7.** *Let G be a planar graph that is not a cycle, and P be a path of four consecutive 2-vertices in G. Then smoothing one 2-vertex in P to make it a path of three consecutive 2-vertices does not increase the minimum number of bends needed to have an orthogonal drawing.*

Due to the space limit, the proof of Lemma 6 and Lemma 7 are omitted (The proof of Lemma 7 utilizes Lemma 6). Refer to Fig. 3 for an illustration:

- The plane graph G_2 is resulting from flipping $H = G[C]$ in the plane graph G_1, where $C = (c, k, l, h, i, e, d)$. The property of having no-bend orthogonal drawing is preserved.
- If we treat G_1 and G_3 as planar graphs, the planar graph G_3 is resulting from smoothing f in G_1; the graph G_3 still has a no-bend orthogonal drawing (with a different embedding than G_1).

▶ **Theorem 8.** *Min-Bend-Draw(G), Min-Bend-Draw(G, e), and Min-Bend-Draw(G, s) are correct.*

Proof. The correctness of Min-Bend-Draw(G, e) and Min-Bend-Draw(G, s) follows from the correctness of Subdiv-Embed. Note that Lemma 7 allows us not to consider the case of $b > 3$.

Lemma 5 ensures that there exists a bend-minimized orthogonal drawing of G such that either (1) the outer boundary contains a 2-vertex $s \in V(G)$, or (2) the outer boundary

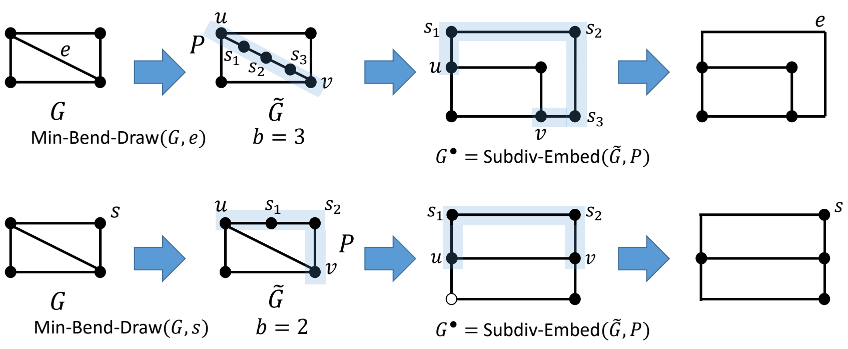

Figure 2 Illustration of Min-Bend-Draw.

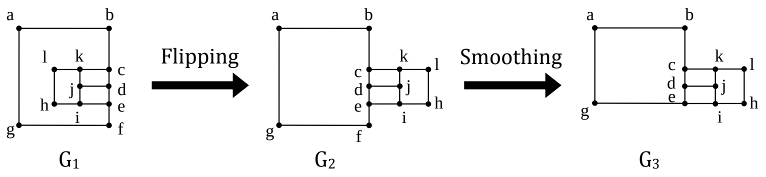

Figure 3 Flipping and smoothing.

contains an edge $e \in E(G)$ whose number of bends is at least 1. Therefore, among all edges $e \in E(G)$ and all 2-vertices $s \in V(G)$, one of Min-Bend-Draw(G, e) or Min-Bend-Draw(G, s) returns a bend-minimized orthogonal drawing. Thus, Min-Bend-Draw(G) is correct. ◀

4 Constrained Orthogonal Drawing

In this section we describe the procedure Subdiv-Embed(G, P) and prove its correctness. Recall that we need Subdiv-Embed(G, P) to return a plane graph $G^\bullet$ such that its no-bend orthogonal drawing is also a bend-minimized P-orthogonal drawing of G.

To better understand how Subdiv-Embed(G, P) works, the reader is encouraged to consult Fig. 4 as our discussion proceeds. Let G be a biconnected planar 3-graph, and P be a u–v path in G such that $V(P) \setminus \{u, v\}$ contains only 2-vertices (see G in the leftmost figure of Fig. 4). With respect to the given G and P, a biconnected subgraph B with $E(P) \subsetneq E(G) \setminus E(B)$ is said to be *essential* if there exists an embedding of G where $P \subseteq C_O$ such that in this particular embedding the unique cycle C with $B = G[C]$ is 2-legged. Note that changing "there exists" to "for all" does not alter the definition. That is, for any essential subgraph B, in *any* embedding of G with $P \subseteq C_O$, the unique cycle C with $B = G[C]$ must be 2-legged. Moreover, all 2-legged cycles C resulting from different embeddings of G with $P \subseteq C_O$ have the *same* two 2-vertices s and t. We define $\{s, t\}$ as the *poles* of B.[7]

With respect to G and P, an essential subgraph B is said to be *maximal* if for all essential subgraphs B', either $E(B) \cap E(B') = \emptyset$ or $B' \subseteq B$. Fig. 4 specifies two maximal essential subgraphs B_x and B_y. We write $\mathbb{B}(G, P)$ to denote the set of maximal essential subgraphs with respect to G and P. Given a fixed planar embedding of G with $P \subseteq C_O$, $\mathbb{B}_{\text{in}}(G, P)$ is

[7] These terms are related to SPQR tree, as described below. In an SPQR tree of G with any reference edge e in P, an essential subgraph B is a pertinent graph (with poles $\{s, t\}$) associated with either a P-node or an R-node. The reason that B is not associated with an S-node is that B is biconnected. See Section 6 for more details.

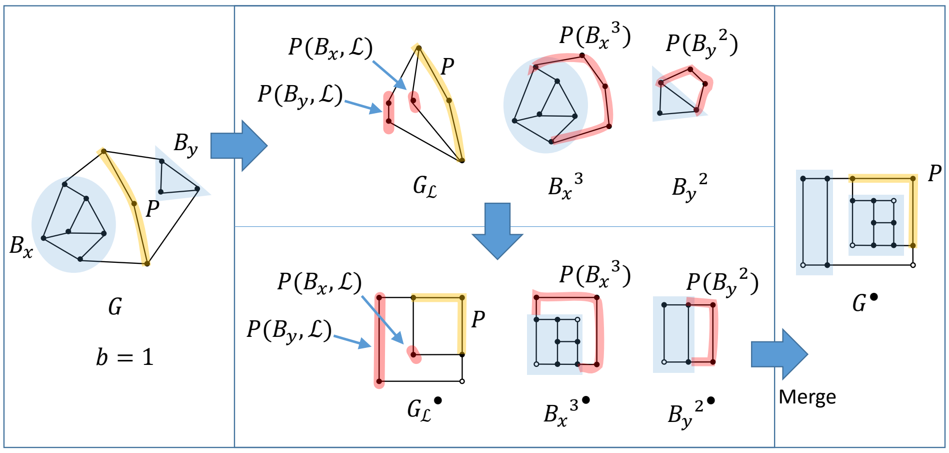

Figure 4 Illustration of Subdiv-Embed.

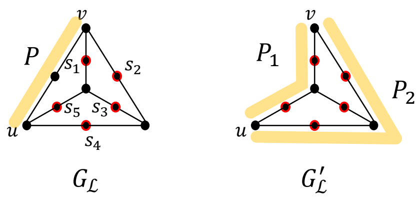

Figure 5 An example illustrating the proof of Lemma 10.

defined as the set of maximal essential subgraphs that does not have any edge in C_O, and $\mathbb{B}_{\text{out}}(G,P)$ is defined as $\mathbb{B}(G,P) \setminus \mathbb{B}_{\text{in}}(G,P)$.

For any essential subgraph B, we write B^b to denote the planar graph resulting from adding a path $(s, r_1, \dots, r_b, t)$ to B, where $\{s,t\}$ are the poles of B. We write $P(B^b)$ to denote the path $(s, r_1, \dots, r_b, t)$. The upper figure of Fig. 4 shows B_x^3, $P(B_x^3)$, B_y^2 and $P(B_y^2)$. Intuitively, the path $(s, r_1, \dots, r_b, t)$ is an abstraction for the sub-drawing (with b convex corners) to which the drawing of B will eventually be attached, and $4 - b$ represents the number of convex corners that the drawing of B is capable to contribute in forming the final orthogonal drawing.

Given a function $\mathcal{L}$ that maps $\mathbb{B}_{\text{out}}(G,P)$ to $\{1,2,3\}$, we write $G_{\mathcal{L}}$ to denote the plane graph resulting from replacing each $B \in \mathbb{B}_{\text{out}}(G,P)$ with a path of $4 - \mathcal{L}(B)$ 2-vertices $(r_1, \dots, r_{4-\mathcal{L}(B)})$ and replacing each $B \in \mathbb{B}_{\text{in}}(G,P)$ with a 2-vertex r. See $G_{\mathcal{L}}$ in the upper figure of Fig. 4. Let $\{s,s'\}$ and $\{t,t'\}$ be the two (uniquely defined) edges in $E(G) \setminus E(B)$, where $\{s,t\}$ are the poles of B. We write $P(B,\mathcal{L})$ to denote the path of $4 - \mathcal{L}(B)$ 2-vertices or the single 2-vertex in $G_{\mathcal{L}}$ that replaces B, depending on whether $B \in \mathbb{B}_{\text{out}}(G,P)$ or $B \in \mathbb{B}_{\text{in}}(G,P)$.

▶ **Definition 9.** A function $\mathcal{L}$ that maps $\mathbb{B}_{\text{out}}(G,P)$ to $\{1,2,3\}$ is a *valid labeling* if and only if $|\mathbb{B}_{\text{out}}(G,P)| \geq 3$ implies $\mathcal{L}(B) = 3$ for all $B \in \mathbb{B}_{\text{out}}(G,P)$.

The intuition behind Definition 9 is as follows. Recall that 4 convex corners are needed in the outer boundary of an orthogonal drawing. The path P can supply at least 1 convex corner (P has at least one 2-vertex). Thus, if $|\mathbb{B}_{\text{out}}(G,P)| \geq 3$, it suffices that each $B \in \mathbb{B}_{\text{out}}(G,P)$ supplies 1 convex corner.

▶ **Lemma 10.** *Among all embeddings of G with $P \subseteq C_O$, there are at most two ways of partitioning $\mathbb{B}(G,P)$ into $\mathbb{B}_{in}(G,P)$ and $\mathbb{B}_{out}(G,P)$. Moreover, given (1) a labeling $\mathcal{L}$ that maps $\mathbb{B}_{out}(G,P)$ to $\{1,2,3\}$, and (2) a set $\mathbb{B}_{out}(G,P)$, the plane graph $G_{\mathcal{L}}$ (if it exists) is uniquely determined.*

Proof. First of all, we can restrict our consideration to the function $\mathcal{L}$ such that $\mathcal{L}(B) = 1$ for all B, as taking subdivisions does not affect the number of planar embeddings.

To understand the proof better, the reader is referred to Fig. 5 for an illustrating example, in which $s_1, \dots, s_5$ are 2-vertices corresponding to essential subgraphs $B_1, \dots, B_5$, respectively. Let $G'_{\mathcal{L}}$ be the planar graph resulting from removing the intermediate vertices of the u–v path P in $G_{\mathcal{L}}$ and neglecting the planar embedding of $G_{\mathcal{L}}$. It is clear that in any planar embedding of $G'_{\mathcal{L}}$ such that $u, v \in V(C_O)$, there is no 2-legged cycle C in $G'_{\mathcal{L}}$ such that C contains both u and v. The existence of such a cycle C violates the definition of $\mathbb{B}(G,P)$, as C would have been contracted into a 2-vertex in $G_{\mathcal{L}}$.

Therefore, under the constraint that $u, v \in V(C_O)$ in an embedding of $G'_{\mathcal{L}}$, there is no way to flip $G'_{\mathcal{L}}[C]$ in $G'_{\mathcal{L}}$, where C is any 2-legged cycle. With respect to any embedding of $G'_{\mathcal{L}}$ such that $u, v \in V(C_O)$, let P_1 and P_2 be the two u–v paths along the outer boundary, and define $S_i = \{B | P(B, \mathcal{L}) \subseteq P_i\}$, where $i \in \{1,2\}$. The unordered pair $\{S_1, S_2\}$ is independent of the chosen embedding (since no flipping operation can be performed).

The outer boundary of $G_{\mathcal{L}}$ is either P together with P_1 (i.e., $\mathbb{B}_{\text{out}}(G,P)$=$\{B_1, B_5\}$) or P together with P_2 (i.e., $\mathbb{B}_{\text{out}}(G,P)$=$\{B_2, B_4\}$). Therefore, $\mathbb{B}_{\text{out}}(G,P)$ can only be S_1 or S_2, and once $\mathbb{B}_{\text{out}}(G,P)$ is fixed, the planar embedding of $G_{\mathcal{L}}$ is fixed. ◀

The procedure Subdiv-Embed(G,P) is defined as Algorithm 4. The procedure uses a subroutine Merge which constructs a plane graph $G^\bullet$ from the following plane graphs:

- ${G_{\mathcal{L}}}^\bullet \leftarrow$ a subdivision of $G_{\mathcal{L}}$ that admits a no-bend orthogonal drawing.
- ${B^3}^\bullet \leftarrow$ Subdiv-Embed$(B^3, P(B^3))$, for each $B \in \mathbb{B}_{\text{in}}(G,P)$.
- ${B^{\mathcal{L}(B)}}^\bullet \leftarrow$ Subdiv-Embed$(B^{\mathcal{L}(B)}, P(B^{\mathcal{L}(B)}))$, for each $B \in \mathbb{B}_{\text{out}}(G,P)$.

Recall that the plane graph $G^\bullet$ is a planar embedding of a subdivision of G that admits a no-bend orthogonal drawing. In addition, we require $|V(G^\bullet)| - |V(G)|$ to be $|V({G_{\mathcal{L}}}^\bullet)| - |V(G_{\mathcal{L}})| + \left(\sum_{B \in \mathbb{B}_{\text{in}}(G,P)} |V({B^3}^\bullet)| - |V(B^3)|\right) + \left(\sum_{B \in \mathbb{B}_{\text{out}}(G,P)} |V({B^{\mathcal{L}(B)}}^\bullet)| - |V(B^{\mathcal{L}(B)})|\right)$. Intuitively, $G^\bullet$ is constructed by merging these plane graphs in such a way that maintains the property of having a no-bend orthogonal drawing without making any new subdivision (see Fig. 4). As guaranteed by Lemma 10, there are two ways of partitioning $\mathbb{B}(G,P)$, and the upper middle figure shows one of the two in which $\mathbb{B}_{\text{out}}(G,P) = \{B_y\}$ and $\mathbb{B}_{\text{in}}(G,P) = \{B_x\}$. Procedure Subdiv-Embed$(G,P)$ produces ${G_{\mathcal{L}}}^\bullet$ (which is a subdivision of $G_{\mathcal{L}}$), ${B_x^3}^\bullet$ (i.e., Subdiv-Embed$(B_x^3, P(B_x^3))$ which is a subdivision of B_x^3), and ${B_y^2}^\bullet$ (i.e., Subdiv-Embed$(B_y^2, P(B_y^2))$, which is a subdivision of B_y^2). Note that the white dots in the drawing indicate 2-vertices created by subdivisions. Also note that displaying the orthogonal drawings of ${B_x^3}^\bullet$, ${B_y^2}^\bullet$, and $G^\bullet$ in Fig. 4 are simply for illustrating purposes. We only compute their planar embeddings and subdivisions, and no specific drawing is fixed. The description of the procedure Merge is left to the next section.

The P-orthogonal drawing of the plane graph $G_{\mathcal{L}}$ in Algorithm 4 can be computed using any orthogonal drawing algorithm for plane 3-graphs that allows us to restrict some edges to have no bend. This can be done in time $T(|V(G_{\mathcal{L}})|) = \tilde{O}(|V(G_{\mathcal{L}})|^{10/7})$ using the min-cost flow described in Sec. 2.2. It is straightforward to modify the min-cost flow to restrict some edges to have no bend. The proof of the correctness of Subdiv-Embed(G,P) is omitted due to the space limit.

Algorithm 4: Subdiv-Embed(G, P).

```
Input: (1) a biconnected planar 3-graph $G$ that is not a cycle, and (2) a $u$–$v$ path $P$
       in $G$ such that $V(P) \setminus \{u, v\}$ consists of exactly $b$ 2-vertices, where $b \in \{1, 2, 3\}$
Output: a plane graph $G^{\bullet}$ which is a planar embedding of a subdivision of $G$ such
        that no subdivision is made on $P$, and any no-bend orthogonal drawing of
        $G^{\bullet}$ is also a bend-minimized $P$-orthogonal drawing of $G$
1 for the at most 2 possibilities of partitioning $\mathbb{B}(G, P)$ into $\mathbb{B}_{in}(G, P)$ and $\mathbb{B}_{out}(G, P)$,
    and the at most 9 possibilities of valid labelings $\mathcal{L}$ do
2     Construct a bend-minimized $P$-orthogonal drawing $D$ of the plane graph $G_{\mathcal{L}}$
        subject to the constraint that no bend is made on the path $(r_1, \ldots, r_{4-\mathcal{L}(B)})$ that
        replaces $B$, for each $B \in \mathbb{B}_{\text{out}}(G, P)$. Let $G_{\mathcal{L}}{}^{\bullet}$ be the subdivision of $G_{\mathcal{L}}$ resulting
        from replacing each bend in $D$ by a 2-vertex.
3     For each $B \in \mathbb{B}_{\text{in}}(G, P)$, let $B^{3^{\bullet}} \leftarrow$ Subdiv-Embed$(B^3, P(B^3))$.
4     For each $B \in \mathbb{B}_{\text{out}}(G, P)$, let $B^{\mathcal{L}(B)^{\bullet}} \leftarrow$ Subdiv-Embed$(B^{\mathcal{L}(B)}, P(B^{\mathcal{L}(B)}))$.
5     Use Merge to construct a planar embedding of a subdivision of $G$ from $G_{\mathcal{L}}{}^{\bullet}$,
        $\{B^{3^{\bullet}}\}_{B \in \mathbb{B}_{\text{in}}(G,P)}$, and $\{B^{\mathcal{L}(B)^{\bullet}}\}_{B \in \mathbb{B}_{\text{out}}(G,P)}$.
6 Among all planar embeddings of subdivisions of $G$ computed, return the one that uses
    the minimum number of subdivisions.
```

▶ **Theorem 11.** *Procedure Subdiv-Embed(G, P) returns a plane graph $G^{\bullet}$ such that any no-bend orthogonal drawing of $G^{\bullet}$ is a bend-minimized P-orthogonal drawing of G.*

We will later see that the procedure Merge admits an implementation that runs in $|\mathbb{B}(G, P)|$ time. This leads to the following lemma.

▶ **Lemma 12.** *Suppose that Subdiv-Embed$(B^i, P(B^i))$ for each $B \in \mathbb{B}(G, P)$, $i \in \{1, 2, 3\}$ are precomputed. Subdiv-Embed(G, P) is in $O(T(|E(G)| - \sum_{B \in \mathbb{B}(G,P)} |E(B)|))$ time.*

Proof. The drawing D of $G_{\mathcal{L}}$ can be computed in $O(T(|V(G_{\mathcal{L}})|))$ time. The procedure Merge takes $O(|\mathbb{B}(G, P)|) \leq O(|V(G_{\mathcal{L}})|)$ time. The lemma follows from the fact that $|V(G_{\mathcal{L}})|$ and $|E(G)| - \sum_{B \in \mathbb{B}(G,P)} |E(B)|$ differs by at most a constant factor. ◀

5 Merging Subgraphs

In this section we describe the procedure Merge. For each essential subgraph B, we define $B^{\bullet}$ as its corresponding subgraph in $B^{3^{\bullet}}$ or $B^{\mathcal{L}(B)^{\bullet}}$, depending on whether $B \in \mathbb{B}_{\text{in}}(G, P)$ or $B \in \mathbb{B}_{\text{out}}(G, P)$. Here $B^{\bullet}$ is defined as a plane graph instead of a planar graph. Observe that simply replacing each $P(B, \mathcal{L})$ with $B^{\bullet}$ (i.e., an expansion) yields a planar embedding of a subdivision of G that meets the constraint on the number of subdivisions made (i.e., $|V(G^{\bullet})| - |V(G)|$). But this does not guarantee that the resulting plane graph $G^{\bullet}$ has a no-bend P-orthogonal drawing. A key in our merging procedure is that after replacing $P(B, \mathcal{L})$ with $B^{\bullet}$, $B^{\bullet}$ becomes a subgraph of $G^{\bullet}$ that can be flipped. We will see that, by appropriately flipping some $B^{\bullet}$, we obtain a planar embedding that has a no-bend P-orthogonal drawing.

Let B be an essential subgraph of G with poles $\{s, t\}$. We let $e_s = \{s, s'\}$ and $e_t = \{t, t'\}$ be the two edges incident to s and t, respectively, from the outside of B. The two edges e_s and e_t are well-defined since both s and t already have two incident edges in B as B is required to be biconnected. Recall that in Subdiv-Embed(G, P) an orthogonal drawing D of $G_{\mathcal{L}}$ is computed. By definition, D is also a no-bend orthogonal drawing of the plane

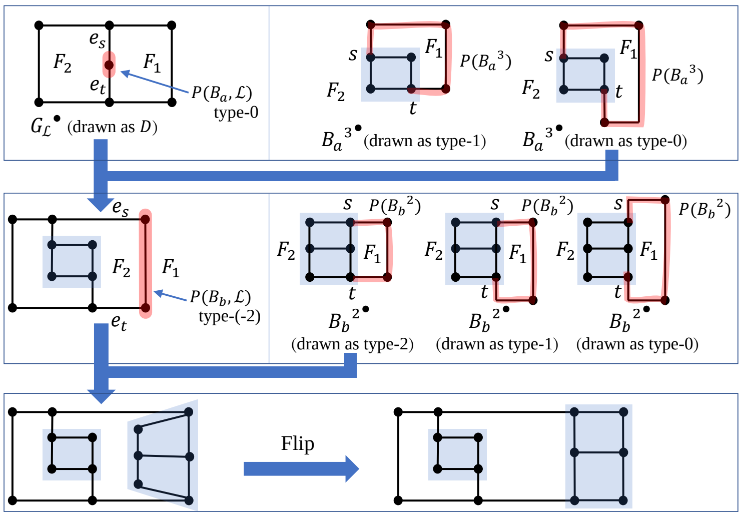

Figure 6 Illustration of Merge.

graph $G_{\mathcal{L}}{}^{\bullet}$. We let F_1 (resp., F_2) be the face in $G_{\mathcal{L}}{}^{\bullet}$ such that s' is followed by s (resp., s is followed by s') in the counter-clockwise ordering of vertices of its facial cycle. We say that $P(B,\mathcal{L})$ is of *type-k* if the number of convex corners minus the number of concave corners of F_1 in $P(B,\mathcal{L})$ is k in D. Note that $P(B,\mathcal{L})$ is of *type-k* if and only if the number of convex corners minus the number of concave corners of F_2 in $P(B,\mathcal{L})$ is $-k$, since a vertex in $P(B,\mathcal{L})$ is convex in F_1 if and only if it is concave in F_2. See Fig. 6 for examples of types of $P(B,\mathcal{L})$.

Next, we define the type of $B^{\bullet}$ with respect to a specific orthogonal drawing D' of $B^{3\bullet}$ or $B^{\mathcal{L}(B)\bullet}$. For notational simplicity, we write $i = 3$ if $B \in \mathbb{B}_{\text{in}}(G,P)$, and $i = \mathcal{L}(B)$ otherwise. Let $(s, x_1, \dots, x_a, t, y_1, \dots, y_b)$ be the clockwise ordering of vertices in the cycle surrounding $B^{\bullet}$, and let $P_1 = (s, x_1, \dots, x_a, t)$ and $P_2 = (s, y_b, \dots, y_1, t)$. Among the two faces in $B^{i\bullet}$ that are not within the subgraph $B^{\bullet}$, we let F_1 (resp., F_2) be the face that has P_1 (resp., P_2) as a subpath in the facial cycle. Then we say $B^{\bullet}$ is of *type-k* in a given orthogonal drawing of $B^{i\bullet}$ if the number of convex corners minus the number of concave corners of F_1 in P_1 is k. Note that $B^{\bullet}$ is of *type-k* if and only if the number of convex corners minus the number of concave corners of F_2 in P_2 is $-k$. See Fig. 6 for examples of $B^{\bullet}$ of different types and drawings of $B^{i\bullet}$ that realize these types.

Recall that our algorithm does not fix any specific orthogonal drawing of $B^{i\bullet}$. We define $\mathcal{T}(B^{\bullet})$ as the set of integers such that $k \in \mathcal{T}(B^{\bullet})$ if there exists an orthogonal drawing D' of $B^{i\bullet}$ such that (1) $B^{\bullet}$ is of type-k with respect to the drawing D', (2) in the drawing D', no bend is made in the subgraph $B^{\bullet}$ (but it is allowed to have bends in the path $P(B^i)$).

▶ **Lemma 13.** *The type of $P(B,\mathcal{L})$ is within $\{-4+i, \dots, 4-i\}$, where $i = 3$ if $B \in \mathbb{B}_{in}(G,P)$, and $i = \mathcal{L}(B)$ otherwise.*

Proof. As the path $P(B,\mathcal{L})$ contains exactly $4 - i$ 2-vertices, the result follows. ◀

▶ **Lemma 14.** *If s is followed by t (resp., t is followed by s) in $P(B^i)$ in the counter-clockwise ordering of vertices of the outer cycle of $B^{i\bullet}$, then $\mathcal{T}(B^{\bullet})$ contains all of $0, -1, \dots, -4+i$ (resp., $0, 1, \dots, 4-i$).*

Proof. We only focus on the case where t is followed by s in $P(B^i)$ in the counter-clockwise ordering of vertices of the outer cycle of $B^{i\bullet}$. In this case F_1 is an inner face. The proof

Algorithm 5: Merge.

```
  Input: $G_{\mathcal{L}}^{\bullet}$ and its no-bend orthogonal drawing $D$, $\{B^{3^{\bullet}}\}_{B\in\mathbb{B}_{\text{in}}(G,P)}$, and
         $\{B^{\mathcal{L}(B)^{\bullet}}\}_{B\in\mathbb{B}_{\text{out}}(G,P)}$
  Output: a plane graph $G^{\bullet}$ which is a planar embedding of a subdivision of $G$ that
          admits a no-bend orthogonal drawing
1 Initialize $\tilde{G} = G_{\mathcal{L}}^{\bullet}$.
2 for $B \in \mathbb{B}(G,P)$ do
3     Set $i = 3$ if $B \in \mathbb{B}_{\text{in}}(G,P)$, and $i = \mathcal{L}(B)$ otherwise.
4     Set $b_1$ as the sign of the type of $P(B,\mathcal{L})$ in $D$ (if the type is 0, then $b_1$ can be
        either $-1$ or 1).
5     Set $b_2 = 1$ if $t$ is followed by $s$ in $P(B^i)$ in the counter-clockwise ordering of
        vertices of the outer cycle of $B^{i^{\bullet}}$, and set $b_2 = -1$ otherwise.
6     Replace $P(B,\mathcal{L})$ in $\tilde{G}$ with $B^{\bullet}$.
7     Flip the subgraph $B^{\bullet}$ if $b_1 \cdot b_2 < 0$.
8 return $G^{\bullet} = \tilde{G}$.
```

of the other case is similar. To see that $4 - i \in \mathcal{T}(B^{\bullet})$, consider any no-bend orthogonal drawing D' of $B^{i^{\bullet}}$ where all the i 2-vertices on the path $P(B^i)$ are drawn as convex corners in F_1. Due to Theorem 3, such a drawing D' exists. The type of $B^{i^{\bullet}}$ with respect to D' is $4 - i$ since the number of convex corners minus the number of concave corners of F_1 in P_1 must be $4 - i$. In what follows, we prove that $\mathcal{T}(B^{\bullet})$ also contains $0, 1, \ldots, 4 - i - 1$. For each $x \in \{i + 1, \ldots, 4\}$, by adding $x - i$ new 2-vertices (which are treated as bends in a drawing) to the path $P(B^i)$, Theorem 3 allows us to construct an orthogonal drawing D' of $B^{i^{\bullet}}$ where the path $P(B^i)$, excluding the two endpoints, supplies x convex corners in F_1. The type of $B^{i^{\bullet}}$ with respect to D' is $4 - x$ since the number of convex corners minus the number of concave corners of F_1 in P_1 must be $4 - x$. ◀

Based on Lemma 13 and Lemma 14, we define the Merge procedure as Algorithm 5. In the iteration of the algorithm that processes B, let τ be the type of $P(B,\mathcal{L})$ in D. For the case where b_1 and b_2 have the same sign, there is an orthogonal drawing D' of $B^{i^{\bullet}}$ realizing the type τ (by Lemma 13 and Lemma 14). It is straightforward to see that replacing $P(B,\mathcal{L})$ with the drawing of $B^{\bullet}$ (taken from D') maintains the validity of the orthogonal representation D (since both $P(B,\mathcal{L})$ in D and $B^{\bullet}$ in D' have the same type τ).

For the case where b_1 and b_2 have opposite signs, there is an orthogonal drawing D' of $B^{i^{\bullet}}$ realizing the type $-\tau$, where τ is the type of $P(B,\mathcal{L})$ in D. Similarly, if the replacement is done with the drawing of $B^{\bullet}$ taken from D', then after flipping the subgraph $B^{\bullet}$, the validity of the orthogonal representation D is maintained (the flipping cancels the effect of opposite signs).

Though the correctness of Algorithm 5 is based on the existence of certain drawings of $B^{i^{\bullet}}$, there is no need to compute these drawings. Therefore, the Algorithm 5 takes only $|\mathbb{B}(G,P)|$ time. See Fig. 6 for an illustration of Merge.

6 SPQR-tree Implementation

In this section we show that the three procedures Min-Bend-Draw(G), Min-Bend-Draw(G, s), and Min-Bend-Draw(G, e) admit efficient implementations based on SPQR-trees [10].

We denote the SPQR-tree of G rooted at the edge e as $\mathbb{T}_{G,e}$. Each node in $\mathbb{T}_{G,e}$ is either an S-,P-,Q-, or R-node. Each node is associated with a subgraph of G which is called the *pertinent graph*. Each pertinent graph B is associated with two vertices $\{s,t\} \subseteq V(H)$, called *poles*, such that removal of s and t disconnects B from the rest of the graph. The root node of $\mathbb{T}_{G,e}$ is a Q-node whose pertinent graph is the subgraph of G resulting from removing the edge e. Let ν be a node in $\mathbb{T}_{G,e}$ that is a descendant of another node μ; then the pertinent graph of ν is a proper subgraph of the pertinent graph of μ.

For a given subgraph $H \subseteq G$ and two vertices $s, t \in V(H)$ such that H does not contain the reference edge e, the following two statements are equivalent:

- B is biconnected, and removing s and t disconnects B from the rest of the graph.
- B is the pertinent graph of a P-node or an R-node μ of $\mathbb{T}_{G,e}$, and the poles of μ are $\{s,t\}$.

For any node μ in $\mathbb{T}_{G,e}$, we define $\mathbb{B}(\mu)$ as the set of all pertinent graphs B meeting the following condition: B is associated with a P-node or an R-node ν which is a descendant of μ such that all intermediate nodes in the directed path $(\mu, \ldots, \nu)$ in the SPQR-tree $\mathbb{T}_{G,e}$ contains no P-node and R-node.

Consider the procedure Subdiv-Embed$(\tilde{G}, P)$ invoked in an execution of Min-Bend-Draw(G, e) or Min-Bend-Draw(G, s). With respect to the SPQR-tree $\mathbb{T}_{\tilde{G},e'}$ for any arbitrary choice of $e' \in E(P)$, it is clear that $\mathbb{B}(\tilde{G}, P)$ is exactly $\mathbb{B}(\mu)$, where μ is the root of $\mathbb{T}_{\tilde{G},e'}$. Therefore, any recursive call Subdiv-Embed$(B^i, P(B^i))$ invoked in the procedure Subdiv-Embed$(\tilde{G}, P)$ can be associated with a P-node or an R-node $\nu \in \mathbb{B}(\mu)$ of $\mathbb{T}_{\tilde{G},e}$ in the sense that B is the pertinent graph of ν. Similarly, it is straightforward to see that the set $\mathbb{B}(B^i, P(B^i))$ is exactly $\mathbb{B}(\nu)$, independent of i. Since the SPQR-tree $\mathbb{T}_{\tilde{G},e'}$ can be constructed in linear time, we have the following theorem (which is due to Lemma 12).

▶ **Theorem 15.** *Let $n = |V(G)|$. Both Min-Bend-Draw(G, e) and Min-Bend-Draw (G, s) can be implemented to run in $O(T(n))$ time, and Min-Bend-Draw(G) can be implemented to run in $O(n \cdot T(n))$ time, where $T(n) = \tilde{O}(n^{10/7})$.*

▶ Remark. We comment on the suggestion of an anonymous reviewer regarding the use of the terminology in [9] to derive our result. In the procedure Subdiv-Embed(G, P), the computation of Subdiv-Embed$(B^i, P(B^i))$ for $i \in \{1, 2, 3\}$ is analogous to the computation of *optimal set* of B in [9]. Theorem 1 actually implies that the *spirality* of a split component of a biconnected planar 3-graph is bounded by a *constant*. This also explains why it suffices to only consider $i \in \{1, 2, 3\}$. If one goes through the proof details in [9], tracks the dependence on spirality carefully, and incorporates the $\tilde{O}(n^{10/7})$ time algorithm for the fixed-embedding setting (Theorem 4, which is based on [6]) to the approach in [9], an $\tilde{O}(n^{17/7})$ time bend-minimization algorithm can also be obtained using the terminology in [9]. Nonetheless, we feel that our approach (directly based on tools in [19]) is more natural and simpler than [9].

Acknowledgements. We thank the anonymous reviewers for their thoughtful comments.

References

1 Michael A. Bekos, Michael Kaufmann, Stephen G. Kobourov, and Antonios Symvonis. Smooth orthogonal layouts. *JGAA*, 17(5):575–595, 2013.

2 Michael A. Bekos, Michael Kaufmann, Robert Krug, Thorsten Ludwig, Stefan Näher, and Vincenzo Roselli. Slanted orthogonal drawings: Model, algorithms and evaluations. *JGAA*, 18(3):459–489, 2014.

3 Thomas Bläsius, Ignaz Rutter, and Dorothea Wagner. Optimal orthogonal graph drawing with convex bend costs. *ACM Trans. Algorithms*, 12(3):33:1–33:32, 2016.

4 Franz Brandenburg, David Eppstein, Michael T. Goodrich, Stephen Kobourov, Giuseppe Liotta, and Petra Mutzel. Selected open problems in graph drawing. In *Proceedings of the 11th International Symposium on Graph Drawing (GD'03)*, pages 515–539. Springer Berlin Heidelberg, 2004.

5 Yi-Jun Chang and Hsu-Chun Yen. On orthogonally convex drawings of plane graphs. *Computational Geometry*, 62:34–51, 2017.

6 Michael B. Cohen, Aleksander Mądry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{O}(m^{10/7} \log W)$ time. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 752–771. Society for Industrial and Applied Mathematics, 2017.

7 Sabine Cornelsen and Andreas Karrenbauer. Accelerated bend minimization. *JGAA*, 16(3):635–650, 2012.

8 Giuseppe Di Battista, Walter Didimo, Maurizio Patrignani, and Maurizio Pizzonia. Orthogonal and quasi-upward drawings with vertices of prescribed size. In *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*, pages 297–310. Springer Berlin Heidelberg, 1999.

9 Giuseppe Di Battista, Giuseppe Liotta, and Francesco Vargiu. Spirality and optimal orthogonal drawings. *SIAM Journal on Computing*, 27(6):1764–1811, 1998.

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

11 Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. On the complexity of hv-rectilinear planarity testing. In *Proceedings of the 22nd International Symposium on Graph Drawing (GD'14)*, pages 343–354. Springer Berlin Heidelberg, 2014.

12 Christian A. Duncan and Michael T. Goodrich. Planar orthogonal and polyline drawing algorithms. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 8. CRC Press, 2013.

13 Stephane Durocher, Stefan Felsner, Saeed Mehrabi, and Debajyoti Mondal. Drawing hv-restricted planar graphs. In *Proceedings of the 11th Latin American Theoretical Informatics Symposium (LATIN'14)*, pages 156–167. Springer Berlin Heidelberg, 2014.

14 Ashim Garg and Roberto Tamassia. A new minimum cost flow algorithm with applications to graph drawing. In *Proceedings of the Symposium on Graph Drawing (GD'96)*, pages 201–216. Springer Berlin Heidelberg, 1997.

15 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001.

16 Gunnar W. Klau and Petra Mutzel. Quasi–orthogonal drawing of planar graphs. Technical Report MPI-I-98-1-013, Max-Planck-Institut für Informatik, Saarbrücken, 1998.

17 Md. Saidur Rahman, Shin-ichi Nakano, and Takao Nishizeki. A linear algorithm for bend-optimal orthogonal drawings of triconnected cubic plane graphs. *JGAA*, 3(4):31–62, 1999.

18 Md. Saidur Rahman and Takao Nishizeki. Bend-minimum orthogonal drawings of plane 3-graphs. In *Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'02)*, pages 367–378. Springer Berlin Heidelberg, 2002.

19 Md. Saidur Rahman, Takao Nishizeki, and Mahmuda Naznin. Orthogonal drawings of plane graphs without bends. *JGAA*, 7(4):335–362, 2003.

20 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.