Accelerated Bend Minimization

Sabine Cornelsen and Andreas Karrenbauer

Department of Computer & Information Science, University of Konstanz
`firstname.lastname@uni-konstanz.de`

Abstract. We present an $\mathcal{O}(n^{3/2})$ algorithm for minimizing the number of bends in an orthogonal drawing of a plane graph. It has been posed as a long standing open problem at *Graph Drawing 2003*, whether the bound of $\mathcal{O}(n^{7/4}\sqrt{\log n})$ shown by Garg and Tamassia in 1996 could be improved. To answer this question, we show how to solve the uncapacitated min-cost flow problem on a planar bidirected graph with bounded costs and face sizes in $\mathcal{O}(n^{3/2})$ time.

1 Introduction

A drawing of a planar graph is called orthogonal if all edges are non-crossing axis-parallel polylines, i.e. sequences of finitely many horizontal and vertical line segments. The intersection point of a vertical and a horizontal line segment of an edge is a bend.

If a graph has an orthogonal drawing such that the vertices are drawn as points then the degree of any vertex is at most four. Biedl and Kant [1] gave a linear-time algorithm for constructing an orthogonal drawing with at most two bends per edge of a graph with degree at most four (except for the octahedron). The problem of minimizing the number of bends in an orthogonal drawing of a planar graph with maximum degree four is $\mathcal{NP}$-complete [2] if the *embedding* of the graph, i.e. the cyclic ordering of the incident edges around each vertex, is not fixed.

Tamassia [3] considered the bend-minimization problem on *plane* graphs, i.e., on planar graphs with a fixed embedding and a fixed outer face. He showed that the problem of minimizing the total number of bends in an orthogonal drawing of a plane graph with degree at most four can be modeled by a min-cost flow problem. There are also variations of the flow-based bend minimization approach which include a restricted number of bends, vertices of degree higher than four [4,5,6,7], drawing clustered graphs [8,9], or interactive and dynamic graph drawing [10,11].

Network flows are an important topic in combinatorial optimization and we refer the interested reader to [12] and [13] for a general overview. Instead, we concentrate on the special case of planar networks in this paper. To the best of the authors knowledge, there have not been many direct contributions to compute planar min-cost flows in the past decades. One exception is a dedicated analysis of an *interior point method* [14] restricted to linear programs arising from min cost flow problems on planar graphs by Imai and Iwano [15]. In 1990, they proved a running time bound of $O(n^{1.594}\sqrt{\log n}\log(n\gamma))$, at which γ is an upper bound on the absolute values of costs and capacities. Much more progress has been made on important special cases such as the *shortest path* problem and the *max flow* problem, which may be used in general flow algorithms as subroutines to obtain a better running time when the input is restricted to planar graphs. This includes the famous linear-time algorithm for planar shortest path with non-negative lengths [16], near linear-time algorithms for shortest path with real lengths [17,18,19], and for max s-t-flow [20,21]. The latter problem can be solved in linear time when s and t are on the same face because of its equivalence to a shortest path problem with non-negative lengths in the dual graph shown by Hassin [22]. This result has been extended to multiple sources and sinks on the same face by Miller and Naor [23].

Garg and Tamassia [24] proved that a min-cost flow problem on a flow network with n nodes, m arcs, and the minimum cost χ of a flow can be solved in $\mathcal{O}(\chi^{3/4}m\sqrt{\log n})$ time and concluded that the bend minimization problem of an embedded planar graph with degree at most four can be solved in $\mathcal{O}(n^{7/4}\sqrt{\log n})$ time. It was posed as an important open problem in graph drawing, whether this run time could be improved [25, Problem 14].[1]

Our Contribution

In this paper, we especially exploit the fact that the flow network is planar and show how to solve the problem in $\mathcal{O}(n^{3/2})$ time. Our algorithm splits the flow network using a cycle separator. To this end, the edges on the cycle are contracted, which maintains planarity. The separator thereby shrinks to a cut node that joins two biconnected components on which the min-cost flow problem can be solved independently. The recursive solutions of the two parts are combined by expanding the separator edge by edge and adjusting the flow between the endpoints of the corresponding edge in each step.

In particular, we show that the uncapacitated min-cost flow problem on a planar bidirected graph with bounded costs and face sizes can be solved in $\mathcal{O}(n^{3/2})$ time. This result only relies on linear-time algorithms for finding cycle separators [26], and for computing max s-t-flows in (s,t)-planar graphs ([22] combined with [16]). Note that our approach combined with a result on multiple-source multiple-sink max-flow in planar graphs [27] solves the bend-minimization problem in $\mathcal{O}(n^{3/2}\log n)$ time if we additionally wish to constrain the number of bends on some edges and it yields an $\mathcal{O}(\sqrt{\chi}n\log^3 n)$ algorithm for computing a flow of minimum-cost χ on a planar flow network with n nodes and $\mathcal{O}(n)$ arcs.

The paper is organized as follows. In Section 2, we define the min-cost flow problem and briefly describe the flow model of Tamassia for bend-minimization. In Section 3, we describe the primal-dual algorithm that generally solves the min-cost flow problem. Our main result, based on the divide and conquer approach, that yields the $\mathcal{O}(n^{3/2})$ time algorithm is described in Section 4.

[1] The result of [15] provides a better bound, but the algorithm is not combinatorial and its correctness is hard to verify since not all details have been presented in the extended abstract. In any case, we improve w.r.t. both.

2 Bend Minimization and Flow Networks

Throughout this paper let $G = (V, E)$ be a simple undirected connected plane graph with n vertices of degree at most four and let $\mathcal{F}$ be the set of faces of a planar embedding. We consider the *vertex-face-incidence multi-graph* with node set $W_G = V \cup \mathcal{F}$ and arcs whenever the geometric intersection of two elements of W_G is non-empty in the planar embedding. Let $D_G = (W_G, A_G)$ denote the bidirected version of this graph. Let $D_{\mathcal{F}}$ be the subgraph of D_G that is induced by the face nodes only.

A *min-cost flow network* $\mathcal{N}$ consists of a directed (multi-)graph $D = (W, A)$, capacities $u : A \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$, node demands $b : W \to \mathbb{Z}$, and arc costs $c : A \to \mathbb{Z}_{\geq 0}$. A map $f : A \to \mathbb{Z}_{\geq 0}$ is a *pseudo-flow* on $\mathcal{N}$ if $f(a) \leq u(a)$ for $a \in A$. A pseudo-flow f is a *flow* if the *deficiency*

$$b_f(v) = b(v) + \sum_{(w,v)\in A} f(w, v) - \sum_{(v,w)\in A} f(v, w)$$

of each node $v \in W$ is zero. The cost of a flow is $c(f) = \sum_{a\in A} c(a)f(a)$. We say that a flow problem is *uncapacitated* and with *unit costs*, respectively, if $u(a) = \infty$ and $c(a) = 1$, respectively, for all arcs $a \in A$.

The bend-minimization problem can be modeled by a min-cost flow network $\mathcal{N}_G = (D_G, u, b, c)$ [3] with the following properties.

1. $D_{\mathcal{F}}$ is planar, bidirected with infinite capacity and unit cost.
2. The degree of a face of $D_{\mathcal{F}}$ is at most 4.
3. A cycle separator of $D_{\mathcal{F}}$ is a cycle separator of D_G.
4. D_G is planar and triangulated.
5. The minimum cost of a flow in $\mathcal{N}_G$ is at most $2n + 4$ [1].

Readers to whom these properties sound familiar may safely skip the next subsection, which contains a brief presentation of Tamassia's approach [3].

Bend Minimization as Min Cost Flow

In this section, we briefly describe the approach of Tamassia [3] for constructing an orthogonal drawing of a plane graph with the minimum total number of bends. The approach consists of two phases. In the first phase an *orthogonal representation* is computed, which fixes the angle at each vertex between two consecutive adjacent edges on one hand and the number of right and left turns on an edge on the other hand. In a second step an area efficient orthogonal grid drawing is constructed from a feasible orthogonal representation. The second step can be done in linear time using topological sorting [28, page 155].

The orthogonal representation associates four labels with each edge $\{v, w\} \in E$, two for each direction. The label $1 \leq \alpha(v, w) \leq 4$ is such that $\alpha(v, w) \cdot \pi/2$ denotes the angle at vertex v between $\{v, w\}$ and the next incident edge of v in counter-clockwise direction. The label $\tau(v, w) \geq 0$ denotes the number of left-turns on $\{v, w\}$ traversed from v to w. See Fig. 1(b), for an illustration.

Let the degree $\deg f$ of a face f be the number of its incident edges where bridges count twice. Elementary geometry implies that there is an orthogonal drawing that corresponds to some given labels α and τ if and only if they imply that the sum of angles around a vertex is 2π and that the sum of angles around an inner/outer face f is $\pi \cdot (\deg(f) + \text{number of bends} \mp 2)$. The latter can be reformulated as

$$\sum_{(v,w)\in E(f)} (\alpha(v,w) + \tau(w,v) - \tau(v,w)) = 2\deg(f) \mp 4$$

where $E(f)$ denotes the arcs incident to the face f directed in counter-clockwise direction. This yields a min-cost flow formulation for finding a feasible orthogonal representation with the minimum number of bends.

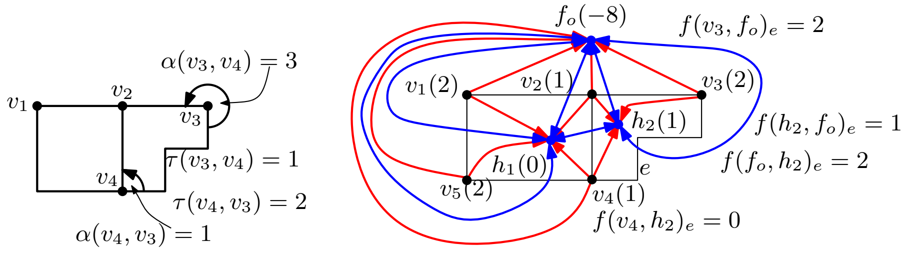


Fig. 1. Illustration of the approach of Tamassia [3] for solving the bend-minimization problem. In b) the directed arcs indicate the flow network. All arcs have infinite capacity, the bidirected blue arcs have cost one and the unidirected red arcs have costs zero. The node demands are indicated in brackets.

The bend minimization problem on G can be solved by the following min-cost flow network $\mathcal{N}_G$. The node set of the directed graph D_G is $W_G = V \cup \mathcal{F}$ with $b(v) = 4 - \deg(v), v \in V$, $b(h) = 4 - \deg(h)$ if $h \in \mathcal{F}$ is an inner face and $b(f_o) = -4 - \deg(f_o)$ for the outer face f_o. For each edge $e = \{v, w\} \in E$ with $(v, w) \in E(h)$ and $(w, v) \in E(g)$ the arc set A_G contains the arcs $(v, h)_e, (w, g)_e$ with costs zero and $(h, g)_e, (g, h)_e$ with costs one. All arcs have infinite capacities. Note that the index e is only used to distinguish possible multiple arcs. See Fig. 1(b), for an illustration.

Now a min-cost flow f on $\mathcal{N}_G$ corresponds to an orthogonal representation with the minimum number of bends as follows. For each edge $e = \{v, w\} \in E$ with $(v, w) \in E(h)$ and $(w, v) \in E(g)$ set $\alpha(v, w) = f(v, h)_e + 1$ and $\tau(v, w) = f(h, g)_e$.

3 The Primal Dual Algorithm

In this section, we briefly describe the primal-dual algorithm [29] for solving the min-cost flow problem.

Let $\mathcal{N} = (D = (W, A), u, b, c)$ be a min-cost flow network. An arc $a \in A$ is *saturated* by a pseudo-flow f if $f(a) = u(a)$. A *node potential* is a function $\pi : W \to \mathbb{Z}$. The *residual network* $\mathcal{N}_{f,\pi} = (D_f = (W, A_f), u_f, b_f, c_\pi)$ of the min-cost flow network $\mathcal{N}$, a pseudo-flow $f : A \to \mathbb{Z}_{\geq 0}$, and a node potential $\pi : W \to \mathbb{Z}$ is defined as follows. For each arc $a \in A$ with tail v and head w the arc set A_f contains a with $c_\pi(a) := c(a) + \pi(v) - \pi(w)$ if $u_f(a) := u(a) - f(a) > 0$. Further, if $f(a) > 0$ then A_f contains a reversed copy $-a$ from w to v with $c_\pi(-a) = -(c(a) + \pi(v) - \pi(w))$ and $u_f(-a) := f(a)$. The costs c_π are called the *reduced costs* and u_f are the *residual capacities*. The node potential is *valid* if $c_\pi(a) \geq 0$ for all $a \in A_f$. The primal-dual algorithm solves a min-cost flow problem utilizing the reduced cost optimality condition.

Lemma 1 ([12, Theorem 9.3]). *A flow has minimum cost if and only if it admits a valid node potential.*

The *primal-dual algorithm* works as follows on a min-cost flow network $\mathcal{N} = (D = (W, A), u, b, c)$. First, the equivalent min-cost max-flow network $\mathcal{N}^{st} = (D^{st} = (W \cup \{s, t\}, A^{st}), u, c, s, t)$ is constructed, i.e. a super source s and a super sink t is added to W. Note that in general this construction does not preserve planarity. However, this is not relevant for the following lemmas. For each node $v \in W$ with $b(v) > 0$ an arc (s, v) with $u(s, v) = b(v)$ and cost zero is added to A. Further, for each node $v \in W$ with $b(v) < 0$ an arc (v, t) with $u(v, t) = -b(v)$ and zero costs is added to A. The value of a flow in $\mathcal{N}^{st}$ is the sum of all flow values on the arcs incident to s. Note that $\mathcal{N}$ has a feasible flow if and only if a maximum s-t-flow of $\mathcal{N}^{st}$ saturates all arcs incident to s. Further, let f be a maximum flow with minimum costs on $\mathcal{N}^{st}$. Restricting f to A yields a min-cost flow on $\mathcal{N}$.

The primal-dual algorithm now basically augments as much flow as possible on shortest s-t-paths in the residual network. More precisely, the algorithm starts with the node potential $\pi = 0$ and the pseudo flow $f = 0$. As long as not all arcs incident to s are saturated, the algorithm adds the shortest-path distances $\text{dist}_{f,\pi}(s, v)$ in (D_f, c_π) to $\pi(v)$. Then it considers the *admissible network* $D_f^o = (W \cup \{s, t\}, A^o)$ with $A^o = \{a \in A_f^{st};\ c_\pi(a) = 0\}$ and augments f by a maximum s-t-flow in (D^o, u_f). See Algorithm 1 for a pseudocode.

To analyze the number of iterations, let f_i and π_i, respectively, be the flow and potential, respectively, after the ith iteration of the primal-dual algorithm. Further, let $f_0 = 0$, $\pi_0 = 0$ be the initial flow and potential. Recall that we consider integer costs and capacities.

Lemma 2. *We have the following properties.*

1. $\pi_i(v) = \mathit{dist}_{f_{i-1},\pi_0}(s, v), v \in W, i \geq 1.$
2. $\pi_i(t) < \pi_{i+1}(t), i \geq 1.$
3. $\pi_i(t) \geq i - 1.$
4. $i \leq \mathit{dist}_{f_i,\pi_0}(s, t).$

Proof. 1. Let $v \in W$. If there is no $s - v$-path in $D_{f_{i-1}}$, then $\text{dist}_{f_{i-1},\pi_0}(s, v) = \text{dist}_{f_{i-1},\pi_{i-1}}(s, v) = \infty$, and, hence, $\pi_i(v) = \pi_{i-1}(v) + \text{dist}_{f_{i-1},\pi_{i-1}}(s, v) = \infty$.

Algorithm 1. Primal-Dual Algorithm

```
Input  : min-cost flow network $\mathcal{N} = (D = (W, A), u, b, c)$.
Output : min-cost max-flow $f$ of $\mathcal{N}^{st}$ with valid node potential $\pi$,
                                                   both initialized to 0
Primal-Dual$(D, u, b, c)$
    while there is an $s$-$t$-path in $D_f^{st}$ do
        $\text{dist}(s, .) \leftarrow$ Single-Source-Shortest-Path$(D_f^{st}, c_\pi, s)$;
        for $v \in W \cup \{t\}$ do
            $\pi(v) \leftarrow \pi(v) + \text{dist}(s, v)$;
        $f^o \leftarrow$ Max-Flow$(D_f^o, u_f, s, t)$;
        $f \leftarrow f + f^o$;
    return $(f, \pi)$;
```

Let now $s = v_0, \ldots, v_\ell = v$ be the nodes on a shortest $s - v$-path in $(D_{f_{i-1}}, c_{\pi_i})$. Then we have that $0 = \text{dist}_{f_{i-1},\pi_i}(s, v) = \sum_{k=1}^{\ell} c_{\pi_i}(v_{k-1}, v_k) = \sum_{k=1}^{\ell} c(v_{k-1}, v_k) + \pi_i(s) - \pi_i(v) = \text{dist}_{f_{i-1},\pi_0}(s, v) - \pi_i(v)$, where the latter equality holds since $\pi_i(s) = 0$.

2. By definition, $\pi_{i+1}(t) = \pi_i(t) + \text{dist}_{f_i,\pi_i}(s, t)$. After augmenting a maximum s-t-flow on the arcs with zero reduced costs there is an s-t-cut on which all arcs with zero reduced costs are saturated. Hence, the residual network contains no s-t-path with zero reduced costs. Hence, $\text{dist}_{f_i,\pi_i}(s, t) > 0$.
3. $\pi_i(t) \geq i - 1$ follows immediately from $\pi_1(t) \geq 0$ and the previous item.
4. If there is no $i + 1$st iteration then $i < \infty = \text{dist}_{f_i,\pi_0}(s, t)$. Otherwise, combining the previous items, we obtain $i \leq \pi_i(t) + 1 \leq \pi_{i+1}(t) = \text{dist}_{f_i,\pi_0}(s, t)$. □

Lemma 3. *Let there be a feasible flow on $\mathcal{N}$, let χ be the minimum cost of a flow on $\mathcal{N}$, and let $i \geq 1$. Then the primal-dual algorithm terminates after at most $i + \chi/i$ iterations.*

Moreover, a min-cost flow can be computed by performing at most i max-flow computations and at most $i + \chi/i$ shortest path computations.

Proof. Let $i \geq 1$. The statement is trivially true if the algorithm performs at most i iterations. So assume that the algorithm performs more than i iterations. Let $r := b_{f_i}(s)$ be the sum of the residual capacities of the arcs leaving s after iteration i. Since in each of the following iterations at least one unit of flow is sent to t it follows that the primal-dual algorithm will finish within at most $i + r$ iterations even if in the last r iterations only one unit of flow is sent from s to t along a shortest path in the residual network and thus without the need to compute any further maximum flow.

On the other hand, since there is a feasible flow on $\mathcal{N}$, all arcs incident to s have to be saturated at the end. Augmenting one unit of flow augments the total cost of a flow by at least the original cost of a shortest s-t-path in the residual network. Since $\text{dist}_{f_i,\pi_0}(s, t) = \pi_{i+1}(t) < \pi_{i+2}(t) = \text{dist}_{f_{i+1},\pi_0}(s, t)$ it follows that the length of a shortest s-t-path increases with every step. Hence,

$\chi \geq r \cdot \mathrm{dist}_{f_i,\pi_0}(s,t) \geq r \cdot i$. Thus, at most $r \leq \chi/i$ shortest-path computations have to be performed after the ith iteration. □

Corollary 1. *Let there be a feasible flow on $\mathcal{N}$ and let χ be the minimum cost of a flow on $\mathcal{N}$. Then the primal-dual algorithm terminates after at most $2 \cdot \sqrt{\chi} + 1$ iterations.*

Proof. If $\chi = 0$ then the algorithm terminates after at most 1 iteration. Otherwise, let i be such that $i - 1 < \sqrt{\chi} \leq i$. Then the total number of iterations is bounded by $i + \chi/i < \sqrt{\chi} + 1 + \chi/\sqrt{\chi} = 2\sqrt{\chi} + 1$ iterations. □

In a network with n vertices and $\mathcal{O}(n)$ arcs the shortest-path problem can be solved in $\mathcal{O}(n \log n)$ time using the algorithm of Dijkstra [30], while the max-flow problem can be solved in $\mathcal{O}(n \log^3 n)$ time if the network is planar [27].

Remark 1. Hence, the primal-dual algorithm computes a flow with minimum cost χ on a planar min-cost flow network with n nodes and with $\mathcal{O}(n)$ arcs in $\mathcal{O}(\sqrt{\chi} n \log^3 n)$ time.

Since the number of bends in an orthogonal drawing and, hence, the cost of the flow in the corresponding min-cost flow network is in $\mathcal{O}(n)$ [1], it follows that the bend-minimization problem can be solved in $\mathcal{O}(n^{3/2} \log^3 n)$ time, even if the number of bends on some edges is restricted. In the next section, we give a divide and conquer approach that directly solves the uncapacitated bend minimization problem utilizing only less recent results.

4 A Recursive Approach

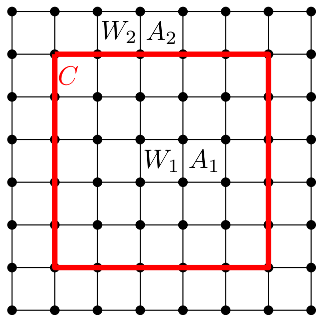
In this section, we show how to utilize a planar separator theorem to recursively solve the min-cost flow problem.

Let an assignment of non-negative weights to the vertices, faces, and edges of a plane graph G be given that sum to one. A simple cycle C of G is a *weighted cycle separator* of G if both, the weight of the interior of C and the weight of the exterior of C do not exceed 2/3.

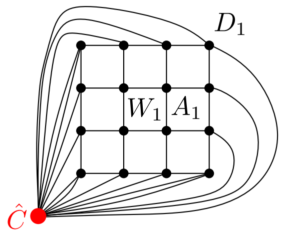
Miller [26] showed that every biconnected planar graph with n vertices and face degree at most d has a simple cycle separator with at most $2\sqrt{d \cdot n}$ vertices unless there is a face with weight higher than 2/3. Moreover, such a cycle separator can be constructed in linear time. Note that the min-cost flow problem decomposes into independent subproblems for each biconnected component.

This yields the following recursive algorithm for constructing a min-cost flow on a flow-network $\mathcal{N} = (D = (W, A), u, b, c)$ where D is a plane digraph with $\mathcal{O}(n)$ nodes and arcs.

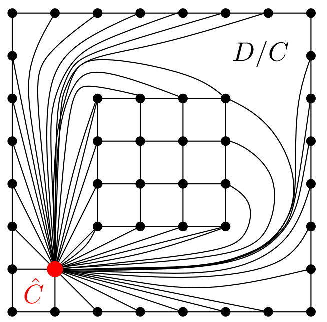
First, we find a small cycle separator $C : v_1, \ldots, v_\ell$ of D. Let W_1 be the set of nodes in the interior of C and let W_2 be the set of nodes in the exterior of C. Let A_i be the set of arcs of A that are incident to at least one node of W_i. See Fig. 2(a) for an illustration. Let $D_i = (W_i \cup \{\hat{C}\}, A_i), i = 1, 2$ be obtained from the subgraph of D induced by $W_i \cup C$ by shrinking C to a single node $\hat{C}$



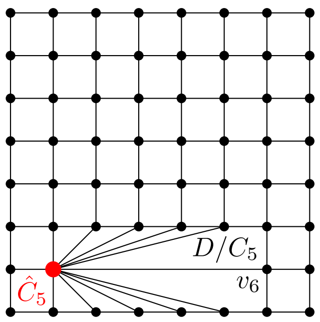
(a) Cycle Separator



(b) Recursive Solution: $b(\hat{C}) = -b(W_1)$



(c) Merging: $b(\hat{C}) = b(C) = -b(W_1) - b(W_2)$



(d) Expanding: $b(\hat{C}_5) = \sum_{i=1}^{5} b(v_i)$

Fig. 2. Illustration of Algorithm 2

maintaining all arcs between W_i and C with their respective costs and capacities. See Fig. 2(b) for an illustration. For a subset $W' \subset W$ let $b(W') = \sum_{v \in W'} b(v)$.

We now recursively solve the two min-cost flow problems

$$\mathcal{N}_i = (D_i, u|_{A_i}, \{b|_{W_i}, b(\hat{C}) = -b(W_i)\}, c|_{A_i}), i = 1, 2$$

obtaining a flow $f|_{A_i}$ with a valid node potential π_i.

Note that $\mathcal{N}_i, i = 1, 2$ has a feasible flow if $\mathcal{N}$ has a feasible flow: Let f be a feasible flow on $\mathcal{N}$. Clearly, f induces a flow on the graph D/C obtained from D by shrinking C to a single node $\hat{C}$ with demand $b(C)$. Note that D_1 is obtained from D/C by deleting W_2 and all its incident arcs. Let $f(C, W_2)$ be the amount of flow on the arcs from C to W_2 minus the amount of flow from W_2 to C. Then $f(C, W_2) = b(W_1) + b(C)$. So if we set $b(\hat{C}) = b(C) - f(C, W_2) = -b(W_1)$ then f induces a flow on D_1.

To merge the two solutions, we first set $\pi(\hat{C}) = \max\{\pi_1(\hat{C}), \pi_2(\hat{C})\}$ adjusting the potential in the respective components. Now we have a feasible flow with a valid node potential on D/C. See Fig. 2(c) for an illustration. We now expand C edge by edge assigning the nodes on C the current potential of $\hat{C}$. More precisely,

Algorithm 2. Recursive Min-Cost Flow

```
Input  : min-cost flow network $\mathcal{N} = (D = (W, A), u, b, c)$ admitting a flow.
Output : min-cost flow $f$ on $\mathcal{N}$ and valid node potential $\pi$, both init. to 0.

Min-Cost-Flow($D, u, b, c$)
    $(W_1, C, W_2) \leftarrow$ CycleSeparator($D$);
    $(f|_{A_i}, \pi_i) \leftarrow$ Min-Cost-Flow($(W_i \cup \{\hat{C}\}, A_i), u|_{A_i}, \{b|_{W_i}, -b(W_i)\}, c|_{A_i}$);
    $\pi(\hat{C}) \leftarrow \max\{\pi_1(\hat{C}), \pi_2(\hat{C})\}$;
    for $v \in W_i, i = 1, 2$ do
        $\pi(v) \leftarrow \pi_i(v) - \pi_i(\hat{C}) + \pi(\hat{C})$;
    Let $C : v_1, \ldots, v_\ell$;
    for $i = \ell, \ldots, 2$ do
        Expand $v_i$ setting $\pi(v_i) \leftarrow \pi(\hat{C})$;
        $(f, \pi) \leftarrow (f, \pi) +$ Primal-Dual($(D/\{v_1, \ldots, v_{i-1}\})_f, u_f, b_f, c_\pi$);
    return $(f, \pi)$;
```

for $2 < i \leq \ell$ let D/C_i be obtained from D by shrinking $C_i = \{v_1, \ldots, v_i\}$ to a single node $\hat{C}_i$ with demand $b(C_i)$. Assume that we have computed a flow f with a valid node potential π of D/C_i. *Expanding* v_i means extending f and π to D/C_{i-1} by setting the flow on the arcs between v_i and C_{i-1} to be zero and $\pi(v_i) = \pi(\hat{C}_{i-1}) = \pi(\hat{C}_i)$. See Fig. 2(d) for an illustration. This yields a pseudo-flow with a valid node potential, however, the deficiencies on v_i and $\hat{C}_{i-1}$ might be different from zero. To adjust the deficiencies, we run the primal-dual algorithm on the residual network. This yields a flow on $\mathcal{N}$ with a valid node potential and, hence, a min-cost flow on D. The algorithm is summarized in Algorithm 2.

Note that the max-flow within the primal-dual algorithm does only have to be performed between v_i and $\hat{C}_{i-1}$. Hence, there is no need for neither super source nor super sink and thus planarity is preserved. Moreover, v_i and $\hat{C}_{i-1}$ lie on the same face. Such a max flow computation can be done in linear time [22,16]. The same holds for the shortest path computation [16] because we maintain a valid node potential, i.e. non-negative reduced cost.

Theorem 1. *The recursive min-cost flow algorithm indicated in Algorithm 2 computes a min-cost flow on a planar bidirected uncapacitated min-cost flow network with n nodes, $\mathcal{O}(n)$ arcs, arc costs at most $c_{\max}$, and face degrees at most d in $\mathcal{O}(c_{\max}\sqrt{d}n^{3/2})$ time.*

Proof. Let m be the number of arcs in the flow network. We may assume that the network is connected and, hence, that $m \in \Theta(n)$. If the network is not biconnected, we first use the cut nodes as separators in the recursive algorithm. Since there is no expansion step, the combination of the recursive solutions of the biconnected components takes only constant time.

So assume now that the network is biconnected. Let all arcs have weight $1/m$ and let all faces and arcs have weight zero. Then the algorithm of Miller [26] constructs in linear time a cycle separator C with $\mathcal{O}(\sqrt{d \cdot n})$ nodes such that both, the interior and the exterior of C contain at most $2/3 \cdot m$ arcs. Let $c_{\max}$

be the maximum cost of an arc. Note that when expanding v_i then the only sources and sinks are v_i and $\hat{C}_{i-1}$ and there is an arc between the two of them in both directions with infinite capacity. Hence, the equivalent min-cost max-flow network remains planar and in all residual networks the length of a shortest path with respect to the original costs is at most $c_{\max}$. Hence, the primal-dual algorithm has to perform at most $c_{\max}$ max-flow operations (Lemma 2) before pushing the remaining deficiency directly over the arc incident to v_i and $\hat{C}_{i-1}$. It follows that $\mathcal{O}(c_{\max}\sqrt{d \cdot n})$ max-flow computations between two adjacent nodes of a planar graph have to be performed. Hence, each recursive step can be performed in $\mathcal{O}(c_{\max}\sqrt{d}n^{3/2}) = \mathcal{O}(c_{\max}\sqrt{d}m^{3/2})$. Hence, the run time $T(m)$ fulfills the recursion

$$T(m) \leq T(m_1) + T(m_2) + c \cdot c_{\max}\sqrt{d}m^{3/2}, \text{ with } m_1 + m_2 \leq m,\ m_1, m_2 \leq 2/3 \cdot n$$

Thus, the total running time is in $\mathcal{O}(c_{\max}\sqrt{d}m^{3/2}) = \mathcal{O}(c_{\max}\sqrt{d}n^{3/2})$. □

Note Theorem 1 remains true if the arc costs are not bounded in general and Algorithm 2 chooses separators that are not necessarily cycles but induce connected subgraphs with arc costs at most $c_{\max}$.

Corollary 2. *The bend-minimization problem on a plane graph with degree at most four and n vertices can be solved in $\mathcal{O}(n^{3/2})$ time.*

Proof. Let $G = (V, E)$ be a plane graph with n vertices and with degree at most four and let $\mathcal{N}_G = (D_G = (V \cup \mathcal{F}, A_G), u, b, c)$ be the min-cost flow network for the bend-minimization problem. Let $m = |A_G|$. Note that $m \in \Theta(n)$. For Computing the cycle separator in the recursive min-cost flow algorithm, we only consider the subgraph $D_{\mathcal{F}}$ induced by the face nodes. We assign each arc of $D_{\mathcal{F}}$ the weight $1/m$ and each face h of $D_{\mathcal{F}}$ the weight $\deg f/m$ while the nodes obtain zero weight. Now the cycle separator of $D_{\mathcal{F}}$ constructed by the algorithm of Miller [26] is a cycle separator C of the whole graph with $O(\sqrt{n})$ nodes such that both, the interior and the exterior of C contain at most $2/3 \cdot m$ arcs. Moreover the arcs on C are bidirected uncapacitated and have unit cost. Hence, each call of the primal-dual algorithm within Algorithm 2 performs one max-flow operation on two adjacent nodes and pushes the remaining deficiency over the corresponding cycle arc. Hence, each recursive step and thus, the whole algorithm can be performed in $\mathcal{O}(n^{3/2})$ time. □

If we wish to constrain the number of bends on an edge artificially, we may sacrifice a log-factor and use the result in [27] to obtain the following.

Remark 2. The bend-minimization problem on a plane graph with degree at most four and n vertices can be solved in $\mathcal{O}(n^{3/2} \log n)$ time even if the number of bends per edge is bounded by some upper bounds $u : A \to \mathbb{Z}_{\geq 0}$, provided that the bounds still admit an orthogonal drawing with a linear number of bends.

Proof. Instead of expanding the cycle separator node after node, we expand it at once. Now the nodes with deficiency other than zero are all on a path. Hence, the max-flow problem within the primal-dual algorithm is solvable in $\mathcal{O}(n \log^2 n)$ time [27]. Assume now that we perform $\sqrt{n}/\log n$ times an ordinary iteration of the primal dual algorithm. Then, by Lemma 3, at most $\mathcal{O}(n/(\sqrt{n}/\log n))$ additional shortest path computations have to be performed, each of which can be done in linear time [31]. Hence, one recursive step and thus the whole algorithm can be performed in $\mathcal{O}(\sqrt{n}/\log n \cdot n \log^2 n + \sqrt{n} \log n \cdot n) = \mathcal{O}(n^{3/2} \log n)$ time. □

Acknowledgments. We are grateful to Ulrik Brandes for bringing our attention to this problem and for fruitful discussions.

References

1. Biedl, T.C., Kant, G.: A better heuristic for orthogonal graph drawings. Computational Geometry 9(3), 159–180 (1998)
2. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. SIAM Journal on Computing 31(2), 601–625 (2001)
3. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM Journal on Computing 16, 421–444 (1987)
4. Fößmeier, U., Kaufmann, M.: Drawing High Degree Graphs with Low Bend Numbers. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 254–266. Springer, Heidelberg (1996)
5. Klau, G.W., Mutzel, P.: Quasi orthogonal drawing of planar graphs. Technical Report MPI-I-98-1-013, Max-Planck-Institut für Informatik, Saarbrücken, Germany (1998), `http://data.mpi-sb.mpg.de/internet/reports.nsf`
6. Tamassia, R., Di Battista, G., Batini, C.: Automatic graph drawing and readability of diagrams. IEEE Transactions on Systems, Man and Cybernetics 18(1), 61–79 (1988)
7. Bertolazzi, P., Di Battista, G., Didimo, W.: Computing orthogonal drawings with the minimum number of bends. IEEE Transactions on Computers 49(8), 826–840 (2000)
8. Brandes, U., Cornelsen, S., Fieß, C., Wagner, D.: How to draw the minimum cuts of a planar graph. Computational Geometry: Theory and Applications 29(2), 117–133 (2004)
9. Lütke-Hüttmann, D.: Knickminimales Zeichnen 4-planarer Clustergraphen. Master's thesis, Universität des Saarlandes (1999) (Diplomarbeit)
10. Brandes, U., Wagner, D.: Dynamic Grid Embedding with Few Bends and Changes. In: Chwa, K.-Y., Ibarra, O.H. (eds.) ISAAC 1998. LNCS, vol. 1533, pp. 89–98. Springer, Heidelberg (1998)
11. Brandes, U., Eiglsperger, M., Kaufmann, M., Wagner, D.: Sketch-Driven Orthogonal Graph Drawing. In: Goodrich, M.T., Kobourov, S.G. (eds.) GD 2002. LNCS, vol. 2528, pp. 1–11. Springer, Heidelberg (2002)
12. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice-Hall (1993)
13. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Heidelberg (2003)
14. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica 4(4), 373–395 (1984)

15. Imai, H., Iwano, K.: Efficient Sequential and Parallel Algorithms for Planar Minimum Cost Flow. In: Asano, T., Imai, H., Ibaraki, T., Nishizeki, T. (eds.) SIGAL 1990. LNCS, vol. 450, pp. 21–30. Springer, Heidelberg (1990)
16. Henzinger, M.R., Klein, P., Rao, S., Subramanian, S.: Faster shortest-path algorithms for planar graphs. Journal of Computer and System Sciences 55, 3–23 (1997); Special Issue on Selected Papers from STOC 1994
17. Fakcharoenphol, J., Rao, S.: Planar graphs, negative weight edges, shortest paths, and near linear time. J. Comput. Syst. Sci. 72, 868–889 (2006)
18. Klein, P., Mozes, S., Weimann, O.: Shortest paths in directed planar graphs with negative lengths: a linear-space $O(n \log^2 n)$-time algorithm. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, pp. 236–245. SIAM, Philadelphia (2009)
19. Mozes, S., Wulff-Nilsen, C.: Shortest Paths in Planar Graphs with Real Lengths in $O(n \log^2 n/ \log \log n)$ Time. In: de Berg, M., Meyer, U. (eds.) ESA 2010. LNCS, vol. 6347, pp. 206–217. Springer, Heidelberg (2010)
20. Weihe, K.: Maximum (s,t)-flows in planar networks in O(V log V) time. J. Comput. Syst. Sci. 55, 454–475 (1997)
21. Borradaile, G., Klein, P.: An O(n log n) algorithm for maximum st-flow in a directed planar graph. J. ACM 56, 9:1–9:30 (2009)
22. Hassin, R.: Maximum flow in (s, t) planar networks. Information Processing Letters 13(3), 107 (1981)
23. Miller, G.L., Naor, J.: Flow in planar graphs with multiple sources and sinks. SIAM J. Comput. 24, 1002–1017 (1995)
24. Garg, A., Tamassia, R.: A New Minimum Cost Flow Algorithm with Applications to Graph Drawing. In: North, S.C. (ed.) GD 1996. LNCS, vol. 1190, pp. 201–213. Springer, Heidelberg (1997)
25. Brandenburg, F.J., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Liotta, G., Mutzel, P.: Selected Open Problems in Graph Drawing. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 515–539. Springer, Heidelberg (2004)
26. Miller, G.L.: Finding small simple cycle separators for 2-connected planar graphs. Journal of Computer and System Sciences 32(4), 265–279 (1986)
27. Borradaile, G., Klein, P., Mozes, S., Nussbaum, Y., Wulff-Nilsen, C.: Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In: Proceedings of the 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011 (to appear, 2011)
28. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall (1999)
29. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press (1962)
30. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
31. Tazari, S., Müller-Hannemann, M.: Shortest paths in linear time on minor-closed graph classes, with an application to steiner tree approximation. Discrete Applied Mathematics 157(4), 673–684 (2009)