One-Bend Drawings of Outerplanar Graphs Inside Simple Polygons

Patrizio Angelini[1], Philipp Kindermann[2], Andre Löffler[2], Lena Schlipf[3], and Antonios Symvonis[4]

1 **John Cabot University, Rome, Italy**
pangelini@johncabot.edu
2 **Universität Würzburg, Würzburg, Germany**
philipp.kindermann@uni-wuerzburg.de, andre.loeffler@uni-wuerzburg.de
3 **Universität Tübingen, Tübingen, Germany**
schlipf@informatik.uni-tuebingen.de
4 **National Technical University of Athens, Athens, Greece**
symvonis@math.ntua.gr

Abstract

We consider the problem of drawing an outerplanar graph with n vertices with at most one bend per edge if the outer face is already drawn as a simple polygon with m corners. We prove that it can be decided in $O(mn)$ time if such a drawing exists. In the positive case, our algorithm also outputs such a drawing.

1 Introduction

One of the fundamental problems in graph drawing is to draw a planar graph crossing-free under certain geometric or topological constraints. Many classical algorithms draw planar graphs under the constraint that all edges have to be straight-line segments [4, 14, 15]. But we do not always have the freedom of drawing the whole graph from scratch. In practical applications, parts of the graph may already be drawn and we want to extend it to a planar drawing of the whole graph. For example, in visualizations of large networks, certain patterns may be required to be drawn in a standard way, or a social network may be updated as new people enter a social circle or as new links emerge between already existing persons.

This problem is known as the Partial Drawing Extensibility problem. Formally, given a planar graph $G = (V, E)$, and subgraph $H = (V', E')$ with $V' \subseteq V$ and $E' \subsetneq E$, and a planar drawing Γ_H of H, the problem asks for a planar drawing Γ_G of G such that the drawing of H in Γ_G coincides with Γ_H. This problem was first proposed by Brandenburg et al. [2] and has received a lot of attention in the previous years.

Related work. For the case of straight-line drawings, Patrignani showed the problem to be NP-hard [12], but he could not prove membership in NP, as a solution may require coordinates not representable with a polynomial number of bits. Recently, Lubiw et al. [8] proved that a generalization of the problem where overlaps (but not proper crossings) between edges of $E \setminus E'$ and E' are allowed is hard for the existential theory of the reals ($\exists\mathbb{R}$-hard).

These results motivate allowing bends in the drawing. Angelini et al. [1] presented a linear-time algorithm to test whether there exists any topological planar drawing of G, and Jelínek et al. [7] gave a characterization via forbidden substructures. Chan et al. [3] showed that a linear number of bends ($72|V'|$) per edge suffices, which is also worst-case optimal as shown by Pach and Wenger [11] for the special case that $E' = \emptyset$.

Special attention has been given to the case that H is exactly the outer face of G. Already Tutte's seminal paper [15] showed how to obtain a straight-line convex drawing of a

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

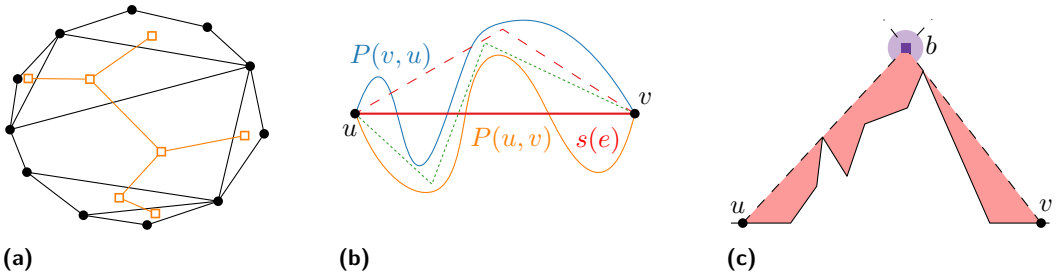


Figure 1 (a) A biconnected outerplanar graph (in black) and the dual tree (in orange); (b) for an edge $e = (u, v)$, the straight line $s(e)$ intersects both $P(u, v)$ and $P(v, u)$, the dashed red 1-bend drawing of e only avoids crossing $P(u, v)$, possible 2-bend drawing in green; (c) the edge connecting u and v has to cut away at least the red region, b is the minimal bend point, and the modified polygon has a reflex angle at b.

triconnected planar graph with its outer face drawn as a prescribed convex polygon. This result has been extended by Hong and Nagamochi [6] to the case that the outer face is drawn as a star-shaped polygon without chords (that is, interior edges between outer vertices). Mchedlidze et al. [9] gave a linear-time testing algorithm for the existence of a straight-line drawing of G in the case that H is an arbitrary cycle of G and is drawn as a convex polygon, while Mchedlidze and Urhausen [10] study the number of bends required based on the shape of the drawing of H and show that 1 bend suffices if H is drawn as a star-shaped polygon.

Our contribution. In this paper, we consider the case that G is an outerplanar graph and H is exactly the outer face of G, which is drawn as a simple polygon P with arbitrarily many bends between its vertices. Note that G has to be biconnected for its outer face to be a simple cycle. For any constant number k of bends, there exists some instance such that G has a k-bend drawing but no $(k-1)$-bend drawing; see, e.g., Fig. 1b for $k = 2$. Hence, it is of interest to test for a given k whether a k-bend drawing of G exists. This is trivial for $k = 0$. In this paper, we prove that for $k = 1$ the problem can be solved in time $O(mn)$, where n is the number of vertices in G and m is the number of corners of P.

Notation. We assume that we are given a biconnected outerplanar graph $G = (V, E)$, a simple polygon P with boundary ∂P, and an injective mapping of V to ∂P such that ∂P coincides with a plane drawing of the outer face H of G with arbitrarily many bends per edge. We say that G *can be drawn in* P if there is a crossing-free drawing of G with its vertices on ∂P as defined by the mapping, its outer face drawn as ∂P, and its interior edges drawn with at most one bend per edge. Considering two vertices u and v, following ∂P in counterclockwise order from u to v gives an open interval $P(u, v)$ of ∂P.

For a pair of vertices u, v, denote by $\overline{uv}$ the straight-line segment between u and v and by $\pi(u, v)$ the shortest path between u and v in P.

The faces $f_1, \ldots, f_n$ of G induce a unique dual tree T [13] with edges $e_1^*, \ldots, e_{n-1}^*$, where e_i^* is the dual of the interior edge e_i; see Fig. 1a. Our algorithm will use T, incrementally processing and pruning T and P. Consider T to be rooted at some degree-1 node f_n. Denote by $p(f_i)$ the parent of f_i in T, and by e_i the edge between f_i and its parent. We say that f_i and f_j are *siblings* if $p(f_i) = p(f_j)$. For an edge $e_i = (u, v)$, let $\pi(u, v) \circ P(v, u)$ be the part of P containing the root f_n (where $\circ$ denotes the concatenation). Then for e_i with $P(u, v)$ containing no other vertices of V, the face f_i is a leaf in T.

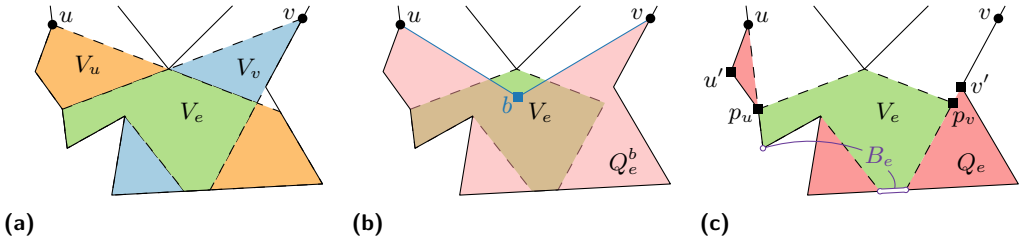


Figure 2 Illustrations for Lemma 2.2: (a) Visibility regions V_u, V_v and intersection V_e; (b) the region cut off by drawing e_i with its bend at point b; (c) construction of p_u, p_v and obstructed region Q_e.

2 Algorithm

An interior edge $e = (u, v)$ divides the polygon into two parts. During the algorithm, we will use these edges to cut some parts of the polygon off. We will make clear in each step which part of the polygon is the remaining one.

We say that any edge $e = (u, v)$ is a *reflex* edge if it has to be drawn with a bend that defines a reflex angle inside the remaining polygon—see Fig. 1c—and a *convex* edge if it can be drawn with a convex bend or as a straight line. Note that an edge $e = (u, v)$ is reflex if and only if $\overline{uv}$ intersects $P(u, v)$ or is completely outside of P.

In the following analysis, we fix some leaf f' of T as the root and will consider the faces of G in some order $f_1, \ldots, f_n$ with $f_n = f'$. Let G_i be the subgraph of G induced by the vertices incident to the faces $f_i, \ldots f_n$, hence $G = G_1$. Symmetrically, define T_i to be the dual tree of G_i with $T = T_1$. In step i of our algorithm, we consider T_i and pick f_i to be a leaf, process the interior edge e_i between f_i and its parent, and refine the polygon to P_{i+1} such that G_{i+1} can be drawn in P_{i+1} if and only if G_i can be drawn in P_i. So, $P_1 = P$ and, in each step, the remaining part P_i of the polygon is the one containing the root f_n (i.e., e_n.)

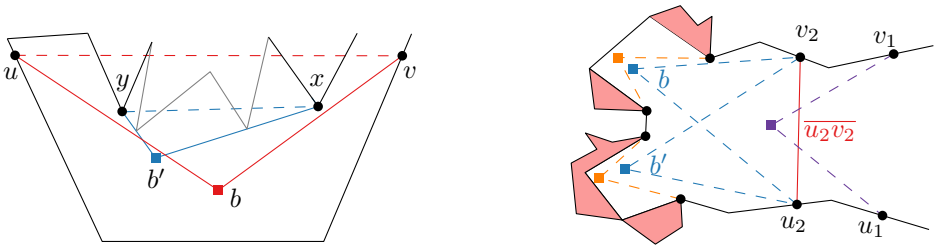
The algorithm chooses the next face f_i to process as follows: If T_i has a leaf corresponding to a reflex edge, we choose that face as f_i. Otherwise, all leaves in T_i correspond to convex edges, and we choose one of the lowest leaves, that is, a leaf with the largest distance (in the graph-theoretic sense) to the root. This way, we can make sure that a convex edge is only chosen if all its siblings that correspond to reflex edges have already been processed.

Let u and v be the end-vertices of the edge e_i separating f_i from its parent in T_i. Let V_u and V_v be the regions of P_i visible from u and v, respectively, and let $V_{e_i} = V_u \cap V_v$ be their intersection. For any point $b \in V_{e_i}$, let $Q^b_{e_i}$ be the subpolygon of P_i bounded by $P_i(u, v) \circ \overline{vb} \circ \overline{bu}$, that is, the part of P_i that is "cut off" by drawing e_i with its bend at b; see Fig. 2b. A point $b \in V_{e_i}$ is called a *minimal bend point* for e_i if there is no other point $b' \in V_{e_i}$ with $Q^{b'}_{e_i} \subsetneq Q^b_{e_i}$. Let B_{e_i} be the set of all minimal bend points for e_i. Further, let (u, u') be the segment of $P_i(u, v)$ incident to u, and let (v', v) be the segment of $P_i(u, v)$ incident to v.

We define $Q_{e_i} = \bigcap_{b \in B_{e_i}} Q^b_{e_i}$ to be the region of P_i *obstructed* by e_i: Wherever we place the bend point of e_i, all the points of Q_{e_i} will be cut off; see Fig. 2c. Conversely, for every point $p \in P_i \setminus Q_{e_i}$, there is a placement of the bend point of e_i such that p is not cut off.

To construct Q_{e_i}, proceed as follows: Rotate a ray around u starting from (u, u') in counterclockwise order until we hit V_{e_i}, call this point p_u. Rotate a ray around v starting from (v', v) in clockwise order until we hit V_{e_i}, call this point p_v. Then Q_{e_i} is the (not necessarily simple) subpolygon of P_i bounded by $\overline{vp_v} \circ V_e(p_u, p_v) \circ \overline{p_u u} \circ P_i(u, v)$; see Fig. 2c.

Similarly, we define $R_{e_i} = \left(\bigcup_{b \in B_{e_i}} Q^b_{e_i}\right) \setminus Q_{e_i}$ to be the region of P_i *restricted* by e_i: For



(a) (b)

Figure 3 (a) The gray part of $P(x, y)$ forces b' to be placed inside $R^b_{(u,v)}$, inducing an intersection of (u, v) and (x, y). For that to be necessary, b' must create a reflex angle. (b) Two edges $e_1 = (u_1, v_1)$, $e_2 = (u_2, v_2)$ with $p(e_2) = e_1$. Fixing the 1-bend drawing of e_1 to intersect $\overline{u_2v_2}$ makes e_2 become a reflex edge, eliminating any choice.

each point $r \in R_{e_i}$, there are two minimal bend points b and b' for e_i such that bending e_i at b cuts off r, whereas bending e_i at b' does not.

▶ **Lemma 2.1.** *Let e be a reflex edge. Then there is a unique minimal bend point b for e.*

Consider how b is constructed in Fig. 1c; note that b is a vertex of V_e. Any point b' above the dashed lines would not be minimal, and any point inside the red region is not visible by at least one of u and v.

▶ **Lemma 2.2.** *Let e be a convex edge. Then $B_e \subset \partial V_e$ and we can safely remove Q_e.*

In step i, our algorithm computes V_{e_i}. If $V_{e_i} = \emptyset$, then it is impossible to draw G_i in P_i, so by induction it is impossible to draw G in P and our algorithm stops. Otherwise, the algorithm computes Q_{e_i} and creates $P_{i+1} = P_i \setminus Q_{e_i}$. If an edge e_j $(j > i)$ had to place its bend point inside Q_{e_i}, then e_i and e_j would cross, independently of the choice of the bend point of e_i; in this case, our algorithm will conclude that it is impossible to draw G in P when it processes e_j. In the following, we will show that G_{i+1} can be drawn in P_{i+1} if and only if G_i can be drawn in P_i.

▶ **Lemma 2.3.** *Let $\mathcal{S}(f)$ be the set of all convex siblings with parent f in T. For any pair of edges $e_1, e_2 \in \mathcal{S}(f)$, the restricted regions R_{e_1} and R_{e_2} are interior-disjoint.*

Sketch of proof. For $f = (u', v')$ consider $P(u', v')$. Since e_1 and e_2 are siblings below f, they are "next to" each other along $P(u', v')$, not nested. Consider edge (u, v) in Fig. 3a: without the gray part of $P(x, y)$ intersecting $\overline{xy}$, the edge (x, y) would be convex. Since the restricted regions $R_{(u,v)}$ and $R_{(x,y)}$ are "hidden" behind the corresponding straight lines, they need to be disjoint. ◀

As a side note: If some parts of P would force two edges to cross, then at least one of these edges has to be reflex; see Fig. 3a with the gray parts in place.

While we can fix the bend points for any reflex edges, convex bends remain undecided until the root is reached, as only some of its minimal bend points might be compatible with the valid bend points of its parent edge. Hence, our algorithm will work bottom-up, fixing reflex edges and computing the obstructed regions for all edges, refining the polygon. When the root f_n is encountered, it is a leaf and the polygon is refined using the obstructed region of its (unique) child. If P_n is not empty, then a solution exists, and we find it by traversing the tree top-down. If e_{n-1} is reflex, then we already fixed its bend point in the bottom-up traversal; otherwise, we place the bend point of e_{n-1} at an arbitrary point inside

the restricted region of e_{n-1} in P_n. Then we obtain the subpolygon P^*_{n-1} that has to contain the drawing of all children of P^*_{n-1} by removing the subpolygon bounded by $P_n(v,u)$ and e_{n-1} from P_{n-1}. When processing face f_i, we again place the bend points of its convex children at arbitrary points inside their restricted region in P^*_i. Since these regions are interior-disjoint by Lemma 2.3, the edges will not intersect. Note that e_i might have been placed in the restricted region of one of its children e_j in P_i; see Fig. 3b. However, since this is the only bend point of the edges incident to f_i that can lie in the restricted region of e_j, by definition there is still a valid bend point for e_j. We again construct the subpolygons P^*_j by removing the subpolygon bounded by $P_i(v,u)$ and e_j from P_j.

This procedure allows us to limit correctness-considerations to consecutive decisions only, and we get the following lemma.

▶ **Lemma 2.4.** *Given the edge e_i in step i, if V_{e_i} is non-empty, then G_{i+1} can be drawn in P_{i+1} if and only if G_i can be drawn in P_i.*

Proof. Whenever we have $V_{e_i} = \emptyset$ for any edge e_i inside P_i, we stop. Further refining P_i will only make it smaller and thus cannot increase the size of any visibility regions.

Otherwise, we either compute the unique best bend point b (Lemma 2.1), or the obstructed region Q_{e_i} (as described above), refining P_i to P_{i+1} accordingly. Lemma 2.2 ensures that the latter is safe.

By construction, no point of the drawing of G_{i+1} can lie in the region Q_{e_i} obstructed by e_i, but the restricted region of e_i can overlap with other restricted regions. Since minimal bend points cannot lie in restricted regions of siblings (Lemma 2.3), only the bend point of the edge corresponding to $p(e_i)$ can possibly be placed in the restricted region R_{e_i} of e_i; any other bend point that lies inside $R(e_i)$ must lie on the opposite side of the drawing of $p(e_i)$, so it cannot influence the choice of the bend point for e_i.

With at most one other bend point in any restricted region, routing the corresponding edge is still possible by definition of R_{e_i}. ◀

We are now ready to state the main result of this paper.

▶ **Theorem 2.5.** *Given an outerplanar graph G with n vertices, a polygon P with m corners, and a mapping of the vertices of G to ∂P, we can decide in $O(mn)$ time whether G can be drawn in P with at most one bend per edge.*

Proof. We use the algorithm described above. The correctness follows immediately from Lemma 2.4. The most time consuming part of the algorithm is to compute the region V_e for each edge $e = (u,v)$, namely the one that is visible from both points u and v. Since V_e is a simple polygon with at most $2m$ edges, it can be computed in $O(m)$ time [5, page 15]. Hence, all these regions can be computed in $O(mn)$ total time. The remaining parts of the algorithm (computing the dual graph of G, choosing the order of the faces f_i in which we traverse the graph, computing Q_e, "cutting off" parts of P, and propagating the graph at the end to fix the presentation) can clearly be done within this time. Thus, the total running time is $O(mn)$. ◀

Acknowledgments. This work was initiated at the Workshop on Graph and Network Visualization 2019. We thank all the participants for helpful discussions and Anna Lubiw for bringing the problem to our attention.

References

1 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. *ACM Trans. Algorithms*, 11(4):32:1–32:42, 2015. doi:10.1145/2629341.

2 Franz-Josef Brandenburg, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, Giuseppe Liotta, and Petra Mutzel. Selected open problems in graph drawing. In Giuseppe Liotta, editor, *Proc. 11th Int. Symp. Graph Drawing (GD)*, volume 2912 of *Lecture Notes Comput. Sci.*, pages 515–539. Springer, 2003. doi:10.1007/978-3-540-24595-7_55.

3 Timothy M. Chan, Fabrizio Frati, Carsten Gutwenger, Anna Lubiw, Petra Mutzel, and Marcus Schaefer. Drawing partially embedded and simultaneously planar graphs. *J. Graph Algorithms Appl.*, 19(2):681–706, 2015. doi:10.7155/jgaa.00375.

4 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.

5 Alexander Gilbers. *Visibility Domains and Complexity.* PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2014.

6 Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of graphs with non-convex boundary constraints. *Discrete Appl. Math.*, 156(12):2368–2380, 2008. doi:10.1016/j.dam.2007.10.012.

7 Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. A Kuratowski-type theorem for planarity of partially embedded graphs. *Comput. Geom.*, 46(4):466–492, 2013. doi:10.1016/j.comgeo.2012.07.005.

8 Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. The complexity of drawing a graph in a polygonal region. In Therese C. Biedl and Andreas Kerren, editors, *Proc. 26th Int. Symp. Graph Drawing Netw. Vis.*, volume 11282 of *Lecture Notes Comput. Sci.*, pages 387–401. Springer, 2018. doi:10.1007/978-3-030-04414-5_28.

9 Tamara Mchedlidze, Martin Nöllenburg, and Ignaz Rutter. Drawing planar graphs with a prescribed inner face. In Stephen K. Wismath and Alexander Wolff, editors, *Proc. 21st Int. Symp. Graph Drawing*, volume 8242 of *Lecture Notes Comput. Sci.*, pages 316–327. Springer, 2013. doi:10.1007/978-3-319-03841-4_28.

10 Tamara Mchedlidze and Jérôme Urhausen. β-stars or on extending a drawing of a connected subgraph. In Therese C. Biedl and Andreas Kerren, editors, *Proc. 26th Int. Symp. Graph Drawing Netw. Vis.*, volume 11282 of *Lecture Notes Comput. Sci.*, pages 416–429. Springer, 2018. doi:10.1007/978-3-030-04414-5_30.

11 János Pach and Rephael Wenger. Embedding planar graphs at fixed vertex locations. *Graphs Comb.*, 17(4):717–728, 2001. doi:10.1007/PL00007258.

12 Maurizio Patrignani. On extending a partial straight-line drawing. *Int. J. Found. Comput. Sci.*, 17(5):1061–1070, 2006. doi:10.1142/S0129054106004261.

13 Andrzej Proskurowski and Maciej Syslo. Efficient Vertex- and Edge-Coloring of Outerplanar Graphs. *SIAM J. Alg. Disc. Meth.*, 7:131–136, 01 1986. doi:10.1137/0607016.

14 Walter Schnyder. Embedding planar graphs on the grid. In David S. Johnson, editor, *Proceedings 1st Ann. ACM-SIAM Symp. Discrete Alg. (SODA)*, pages 138–148. SIAM, 1990. URL: http://dl.acm.org/citation.cfm?id=320176.320191.

15 William Thomas Tutte. How to draw a graph. *Proc. London Math. Soc.*, 3(1):743–767, 1963.