Minimum Cuts and Shortest Cycles in Directed Planar Graphs via Noncrossing Shortest Paths

Hung-Chun Liang† Hsueh-I Lu‡
March 24, 2017

Abstract

Let $G$ be an $n$-node simple directed planar graph with nonnegative edge weights. We study the fundamental problems of computing (1) a global cut of $G$ with minimum weight and (2) a cycle of $G$ with minimum weight. The best previously known algorithm for the former problem, running in $O(n \log^3 n)$ time, can be obtained from the algorithm of Łącki, Nussbaum, Sankowski, and Wulff-Nilsen for single-source all-sinks maximum flows. The best previously known result for the latter problem is the $O(n \log^3 n)$-time algorithm of Wulff-Nilsen. By exploiting duality between the two problems in planar graphs, we solve both problems in $O(n \log n \log \log n)$ time via a divide-and-conquer algorithm that finds a shortest non-degenerate cycle. The kernel of our result is an $O(n \log \log n)$-time algorithm for computing noncrossing shortest paths among nodes well ordered on a common face of a directed plane graph, which is extended from the algorithm of Italiano, Nussbaum, Sankowski, and Wulff-Nilsen for an undirected plane graph.

1 Introduction

Let $G$ be an $n$-node $m$-edge simple graph with nonnegative edge weights. $G$ is unweighted if the weights of all edges of $G$ are identical. Let $C$ be a subgraph of $G$. The weight $w(C)$ of $C$ is the sum of edge weights of $C$. Let $G \setminus C$ denote the graph obtained from $G$ by deleting the edges of $C$. Paths are allowed to repeat nodes throughout the paper. For nodes $s$ and $t$, an $st$-path of $G$ is a path of $G$ from $s$ to $t$ and an $st$-cut of $G$ is a subgraph $C$ of $G$ such that there are no $st$-paths in $G \setminus C$. A (global) cut of $G$ is an $st$-cut of $G$ for some nodes $s$ and $t$ of $G$. A cycle of $G$ is an $ss$-path of $G$ for some node $s$ of $G$.

- The minimum-cut problem on $G$ seeks a cut of $G$ with minimum weight. For instance, the $v_1v_3$-cut consisting of edge $v_2v_3$ is the minimum cut of the graph in Figure 1(a). The best known algorithm on directed $G$, due to Hao and Orlin [30], runs in $O(mn \log \frac{n^2}{m})$ time. On undirected $G$, Nagamochi and Ibaraki [54] and Stoer and Wagner [61] solved the problem in $O(mn + n^2 \log n)$ time and Karger [37] solved the problem in expected

---

* A preliminary version of this paper appeared as the master’s thesis of the first author [45]. The journal version appeared in SIAM Journal on Discrete Mathematics [46].
† Graduate Institute of Computer Science and Information Engineering, National Taiwan University. Email: sirbatostar@yahoo.com.tw.
‡ Corresponding author. Department of Computer Science and Information Engineering, National Taiwan University. This author also holds joint appointments in the Graduate Institute of Networking and Multimedia and the Graduate Institute of Biomedical Electronics and Bioinformatics, National Taiwan University. Address: 1 Roosevelt Road, Section 4, Taipei 106, Taiwan, ROC. Research of this author is supported in part by MOST grant 104-2221-E-002-044-MY3. Email: hil@csie.ntu.edu.tw. Web: www.csie.ntu.edu.tw/~hil
Figure 1: (a) A simple planar graph $G$. (b) A simple bidirected plane graph $G_\triangle$ obtained from $G$ by adding edges with weights $\star = 0$ (respectively, $\star = \infty$) if we are seeking a minimum cut (respectively, shortest cycle) of $G$. (c) The dual of $G_\triangle$.

$O(m \log^3 n)$ time. Kawarabayashi and Thorup [38] recently announced the first known $o(mn)$-time algorithm on undirected unweighted $G$, improving upon the algorithm of Gabow [24] designed twenty years ago.

- The shortest-cycle problem on $G$ seeks a cycle of $G$ with minimum weight. For instance, cycle $v_2v_3v_2$ with weight 6 is the shortest cycle of the graph in Figure 1(a). Since a shortest directed cycle containing edge $ts$ is obtainable from a shortest $st$-path, the problem on directed graphs can be reduced to computing all-pairs shortest paths in, e.g., $O(mn + n^2 \log n)$ time [10]. Vassilevska Williams and Williams [65] argued that finding a truly subcubic algorithm for the problem might be hard. For directed (respectively, undirected) unweighted $G$, Itai and Rodeh [32] solved the problem in $O(\mu(n) \log n)$ (respectively, $O(\mu(n) \log n)$) time, where $\mu(n) = O(n^{2.373})$ [64] is the time for multiplying two $n \times n$ matrices.

If $G$ is undirected and planar, Chalermsook, Fakcharoenphol, and Nanongkai [8] showed that the time complexity of both aforementioned problems on $G$ is $O(\log n)$ times that of finding an $st$-cut of $G$ with minimum weight for any given nodes $s$ and $t$. Plugging in the $O(n \log n)$-time algorithms, e.g., of Frederickson [23], Borraidaile and Klein [1], and Erickson [14], the reduction of Chalermsook et al. solved both problems in $O(\log n)$ time. Plugging in the $O(n \log n)$-time algorithm of Italiano, Nussbaum, Sankowski, and Wulff-Nilsen [33], the reduction of Chalermsook et al. solved both problems in $O(n \log \log n)$ time. The best known result for both problems on $G$ is the $O(n \log \log n)$-time algorithm of Łącki and Sankowski [44], relying upon the $st$-cut oracle of Italiano et al. [33].

This paper addresses both problems for the case that $G$ is directed and planar. While the minimum-cut problem has been thoroughly studied for undirected planar graphs, surprisingly no prior work is specifically for directed planar graphs. Djidjev [12] claimed that his technique for unweighted undirected planar graphs solves the shortest-cycle problem on unweighted directed planar $G$ in $O(n^{3/2})$ time and left open the problem of finding a shortest cycle in unweighted directed planar $G$ in $o(n^{3/2})$ time. Weimann and Yuster [67] gave an $O(n^{3/2})$-time algorithm for the shortest-cycle problem, which should be adjustable to solve the minimum-cut problem also in $O(n^{3/2})$ time (via similar techniques to our proof for Lemma 4.2 in §4 to handle degeneracy in shortest cycles). Wulff-Nilsen [65] reduced the time for the shortest-cycle problem on $G$ to $O(n \log^3 n)$, but it is unclear how to adjust his algorithm to solve the minimum-cut problem without increasing the required time by too much. The algorithm of Łącki, Nuss-
baum, Sankowski, and Wulff-Nilsen [43] for single-source all-sinks maximum flows solves the minimum-cut problem on directed planar \( G \) in \( O(n \log^3 n) \) time. Below is our result:

**Theorem 1.1.** It takes \( O(n \log n \log \log n) \) time to solve the minimum-cut and shortest-cycle problems on an \( n \)-node simple directed planar graph with nonnegative edge weights.

As pointed out by anonymous reviewers, Mozes, Nikolaev, Nussbaum, and Weimann [51] recently announced an \( O(n \log \log n) \)-time algorithms for the minimum-cut problem. However, unlike our Theorem 1.1, their algorithm requires the condition that there is a unique shortest path between any two nodes. For general directed planar graphs with nonnegative edge weights, they apply an isolation lemma [50, 53] to perturb the edge weights to meet the condition with high probability. Thus, their results are Monte Carlo randomized algorithms.

**Related work**

The only known nontrivial linear-time algorithm for the shortest-cycle problem, due to Chang and Lu [9], works on undirected unweighted planar graphs. For undirected \( G \), if \( G \) is embedded on an orientable surface of genus \( g \), Erickson, Fox, and Nayyeri [15] solved the problem in \( g^{O(g^2)} n \log n \) time, based on the algorithm of Lacki and Sankowski [44] for undirected planar graphs. If \( G \) is undirected and unweighted and is 2-cell embedded on an orientable surface of genus \( g = O(n^{ \alpha}) \) with \( 0 < \alpha < 1 \), Djidjev [12] solved the problem in \( O(g^{3/2} n^{3/2}) \) time. The shortest-cycle problem on \( G \) with negative edge weights can be reduced to one with nonnegative edge weights using the standard reweighting technique via a shortest-path tree in \( G \) (e.g., [20, 25, 27, 42, 52]). Cygan, Gabow, and Sankowski [11] studied the problem on graphs whose edge weights are bounded integers. Yuster [69] studied the version on undirected \( G \) asking for each node a shortest cycle containing the node. See e.g., [32, 47, 49, 59, 60, 70] for algorithms that compute shortest cycles with prescribed topological properties. See e.g., [32, 47, 49, 59, 60, 70] for approximation algorithms of the shortest-cycle problem.

The closely related problem that seeks a minimum \( st \)-cut for given nodes \( s \) and \( t \) and its dual problem that seeks a maximum \( st \)-flow has been extensively studied even for only planar graphs (see, e.g., [1, 14, 59, 68]). A minimum \( st \)-cut of \( G \) can be obtained in \( O(m + n) \) time from a maximum \( st \)-flow of \( G \) by identifying the edges from the nodes of \( G \) reachable from \( s \) to the nodes of \( G \) not reachable from \( s \) in the residual graph of \( G \) with respect to \( f \). No efficient reductions for the other direction are known. Orlin [55] gave the only known \( O(mn) \)-time algorithms for the maximum \( st \)-flow problem on general graphs with integral edge weights. For undirected planar \( G \), Reif [58] gave an \( O(n \log^2 n) \)-time algorithm for the minimum \( st \)-cut problem. Frederickson [23] improved the time complexity of Reif’s algorithm to \( O(n \log n) \). The best known algorithms for both problems, due to Italiano et al. [53], run in \( O(n \log \log n) \) time. The attempt of Janiga and Koubek [34] to generalize Reif’s algorithm to directed planar \( G \) turned out to be flawed [17, 55, 51]. Borradaile and Klein [11] and Erickson [13] gave \( O(n \log n) \)-time algorithms for both problems on directed planar graphs. On directed planar unweighted \( G \), Brandes and Wagner [4] and Eisenstat and Klein [12] solved both problems in \( O(n) \) time. The algorithm of Kaplan and Nussbaum [36] is capable of exploiting the condition that nodes \( s \) and \( t \) are close. For directed planar \( G \), the \( O(n \log^3 n) \)-time algorithm of Lacki et al. [43] obtains the minimum weights of \( st \)-cuts for any given \( s \) and all nodes \( t \) of \( G \). For any given node subsets \( S \) and \( T \) of directed planar \( G \), the \( O(n \log^3 n) \)-time algorithm of Borradaile, Klein, Mozes,
Nussbaum, and Wulff-Nilsen [2] computes a subgraph $C$ of $G$ with minimum weight such that there is no $st$-path in $G \setminus C$ for any $s \in S$ and $t \in T$. On undirected planar $G$, Borradaile, Sankowski, and Wulff-Nilsen [3] gave an $O(n \log^4 n)$-time algorithm to compute a Gomory-Hu cut-equivalent tree [28], a compact representation of $st$-cuts with minimum weights for all nodes $s$ and $t$.

The kernel of our result is an $O(n \log \log n)$-time algorithm for computing noncrossing shortest paths among nodes well ordered on a common face of a directed plane graph, which is extended from the algorithm of Italiano et al. [33] for an undirected plane graph. A closely related NP-hard shortest-noncrossing-paths problem seeks noncrossing paths between $k$ given terminal pairs on $h$ faces with minimum total weight in a plane graph. Takahashi, Suzuki, and Nishizeki [63] solved the problem for undirected plane graphs with $h \leq 2$ in $O(n \log k)$ time. Papadopoulou [56] addressed the geometric version of the problem, where the terminal pairs are on the boundaries of $h$ polygonal obstacles in the plane with complexity $n$ and gave an $O(n)$-time algorithm for the case $h \leq 2$. Erickson and Nayyeri [18] generalized the result of Takahashi et al., solving the problem for undirected planar graphs in $2^{O(h^2)} n \log k$ time. They also generalized the result of Papadopoulou to solve the geometric version in $2^{O(h^2)} n$ time. Each of these algorithms computes an implicit representation of the answers, which may have total size $\Omega(kn)$. Polishchuk and Mitchell [57] addressed the problem of finding noncrossing thick paths with minimum total weight. Takahashi, Suzuki, and Nishizeki [62] also considered the rectilinear version of the problem.

**Technical overview and outline**

Our proof for Theorem 1.1 consists of a series of reductions. Based upon the duality between simple cycles and minimal cuts in plane graphs, Section 2 gives an $O(n)$-time reduction from the minimum-cut and shortest-cycle problems in an $n$-node planar graph to the problem of finding a shortest non-degenerate cycle in an $n$-node $O(1)$-degree plane graph $G$ (Lemma 2.1). Let $C$ be a balanced separator of $G$ that corresponds to a fundamental cycle with respect to a shortest-path tree of $G$. A shortest non-degenerate cycle that does not cross $C$ can be recursively computed from the subgraphs of $G$ separated by $C$. Although we cannot afford to compute a shortest non-degenerate cycle that crosses $C$, Section 3 reduces the problem of finding a shortest non-degenerate cycle to finding a $C$-short cycle, i.e., a non-degenerate cycle that crosses $C$ with the property that if it is not shortest, then a shortest non-degenerate cycle that does not cross $C$ has to be a shortest non-degenerate cycle of $G$ (Lemma 3.1). This reduction is a divide-and-conquer recursive algorithm using the balanced separator $C$ and thus introduces an $O((\log n) \cdot k)$-factor overhead in the running time. A cycle of $G$ that crosses a shortest path $P$ of $G$ can be shortcutted into a non-degenerate cycle that crosses $P$ at most once. Section 4 further reduces the problem of finding a $C$-short cycle to that of finding a $(C, P)$-short cycle, i.e., a non-degenerate cycle whose weight is no more than that of any non-degenerate cycle that crosses a shortest path $P$ of $G$ in $C$ exactly once (Lemma 4.2). By the technique of Reif [58] that incises $G$ along $P$, Section 4 further reduces the problem of finding a $(C, P)$-short cycle to that of finding shortest noncrossing cycles among nodes well ordered on the boundary of external face (Lemma 4.1). As a matter of fact, this shortest-noncrossing-paths problem can be solved by the $O(n \log n)$-time algorithm of Klein [40], already yielding improved $O(n \log^2 n)$-time algorithms for the minimum-cut and shortest-cycle problems. (Mozes et al. [51] also mentioned that $O(n \log^2 n)$-time algorithms can be obtained by plugging in the $O(n \log n)$-time minimum $st$-cut algorithm of Borradaile and Klein [11] into a directed version of the reduction algorithm of Chalermsook et al. [53].) To achieve the time complexity of Theorem 1.1 Section 5 solves the problem in $O(n \log \log n)$ time by extending the algorithm of Italiano et al. [33] for an undi-
2 Reduction to finding shortest non-degenerate cycles

Directed graph \( G \) is bidirected if, for any two nodes \( s \) and \( t \) of \( G \), \( st \) is an edge of \( G \) if and only if \( ts \) is an edge of \( G \). The graph in Figure 1(a) is not bidirected. The degree of node \( v \) in bidirected \( G \) is the number of neighbors of \( v \) in \( G \). The degree of bidirected \( G \) is the maximum degree of the nodes in \( G \). A bidirected plane graph is a bidirected planar graph equipped with a plane embedding in which edges between two adjacent nodes are bundled together. Figures 1(b) and 1(c) show two bidirected plane graphs \( G_\Delta^* \) and \( G_\Delta^o \). A cycle passing each node at most once is simple. A cycle is degenerate if it is a node or passes both edges \( st \) and \( ts \) for two nodes \( s \) and \( t \). A cycle not simple (respectively, degenerate) is non-simple (respectively, non-degenerate). Cycle \( C_1 \) in Figure 2(a) is non-degenerate and non-simple. In the graph \( G \) of Figure 1(a), cycle \( v_2v_3v_2 \) is degenerate and simple, cycle \( v_2v_3v_4v_2 \) is non-degenerate and simple, and cycle \( v_1v_2v_4v_2v_1 \) is degenerate and non-simple. The shortest degenerate cycle of \( G \) is \( v_2v_3v_2 \) with weight 6. The shortest non-degenerate cycle of \( G \) is \( v_2v_3v_4v_2 \) with weight 16. Theorem 1.1 can be proved by the following lemma:

**Lemma 2.1.** It takes \( O(n \log n \log \log n) \) time to compute a shortest non-degenerate cycle in an \( n \)-node \( O(1) \)-degree simple bidirected plane graph with nonnegative edge weights.

**Proof of Theorem 1.1.** Adding edges with weights 0 (respectively, \( \infty \)) to the input graph does not affect the weight of minimum cuts (respectively, shortest cycles). Hence, we may assume without loss of generality that the input graph \( G_\Delta^* \) has at least four nodes and is a simple bidirected plane graph such that each face of \( G_\Delta^* \) is a triangle. See Figures 1(a) and 1(b) for examples. Let the dual \( G_\Delta^* \) of \( G_\Delta^* \) be the simple bidirected plane graph on the \( 2n - 4 \) faces of \( G_\Delta^* \) sharing the same set of \( 6n - 12 \) edges with \( G_\Delta \) that is obtainable in \( O(n) \) time from \( G_\Delta \) as follows: For any two adjacent nodes \( s \) and \( t \) of \( G_\Delta \), there are directed edges \( fg = st \) and \( gf = ts \) in \( G_\Delta \), where \( f \) and \( g \) are the two faces of \( G_\Delta \) incident with the bundled edges between \( s \) and \( t \) such that face \( g \) immediately succeeds face \( f \) in clockwise order around node \( s \) of \( G_\Delta \). See Figure 1(c) for an example. Observe that \( C \) is a minimal cut of \( G_\Delta \) if and only if \( C \) is a simple non-degenerate cycle of \( G_\Delta^* \). By nonnegativity of edge weights, a shortest non-degenerate cycle in \( G_\Delta^* \) is a minimum cut of \( G_\Delta \). For instance, the shortest non-degenerate cycle of the graph \( G_\Delta^* \) in Figure 1(c) is \( f_1f_4f_3f_1 \) with weight 5. It corresponds to the \( v_1v_2 \)-cut \( \{v_1v_3, v_2v_3, v_4v_3\} \) of \( G_\Delta \), which in turn corresponds to the minimum cut \( \{v_2v_3\} \) of \( G \). Although the degenerate cycle \( f_1f_4f_3f_1 \) is a shortest cycle of \( G_\Delta^* \), it does not correspond to a cut of \( G \) in the above manner. Since each node of \( G_\Delta^* \) has exactly three neighbors, the statement of the theorem for the minimum-cut problem follows from applying Lemma 2.1 on \( G_\Delta^* \).

By nonnegativity of edge weights, it takes \( O(n) \) time to obtain a shortest degenerate cycle of \( G_\Delta \) by examining the \( O(n) \) degenerate cycles of \( G_\Delta \) on exactly two nodes. By Lemma 2.1, the statement of the theorem for the shortest-cycle problem is immediate from the following claim:

*It takes \( O(n) \) time to obtain from \( G_\Delta \) an \( O(n) \)-node \( O(1) \)-degree simple bidirected plane graph \( G \) such that a shortest non-degenerate cycle of \( G_\Delta \) can be computed from a shortest non-degenerate cycle of \( G \) in \( O(n) \) time.*

Let \( G_1 \) and \( G_2 \) be simple bidirected plane graphs with nonnegative edge weights such that \( G_2 \) is obtained from \( G_1 \) by the following \( O(d) \)-time operation on a degree-\( d \) node \( v \) of \( G_1 \) with \( d \geq 4 \): If \( u_1, \ldots, u_d \) are the neighbors of \( v \) in \( G_1 \) in clockwise order around \( v \), then \text{SPLIT}(v)\).
See Figure 2 for an example of $G_1$ and $G_2$. An $O(n)$-node $O(1)$-degree simple bidirected plane graph $G$ can be obtained in $O(n)$ time from $G_\Delta$ by iteratively applying SPLIT on each node $v$ of $G_\Delta$ with degree $d \geq 4$. To prove the claim, it suffices to ensure the following statement:

A shortest non-degenerate cycle $C_1$ of $G_1$ is obtainable in $O(d)$ time from a shortest non-degenerate cycle $C_2$ of $G_2$.

For each $u_iv_j$-path $P$ of $C_2$ with $1 \leq i \neq j \leq d$ such that $P$ has at least two edges and all internal nodes of $P$ are in $\{v_1, \ldots, v_d\}$, we replace $P$ by path $u_i v_j$. By $w(P) = w(u_i v_j)$, we have $w(C_1) = w(C_2)$. Since $C_2$ is non-degenerate, so is the resulting $O(d)$-time obtainable cycle $C_1$ of $G_1$. Since $C_1$ may pass $v$ more than once, $C_1$ could be non-simple. See Figure 2 for an example of $C_2^* \land C_1$. It remains to show $w(C_1) = w(C_1^*)$ for any shortest simple non-degenerate cycle $C_1^*$ of $G_1$. By nonnegativity of edge weights, we have $w(C_1) \geq w(C_1^*)$ even if $C_1$ is non-simple. Let $C_2$ be the cycle of $G_2$ that is obtained from $C_1^*$ as follows: If there is a path $u_iv_j$ with $1 \leq i \neq j \leq d$ in $C_1^*$, then replace it by path $u_i v_j$. By $w(u_i v_j) = w(u_i v_j)$, we have $w(C_2) = w(C_1^*)$. Otherwise, let $C_2 = C_1^*$. Since $C_1^*$ is simple, there is at most one path $u_i v_j$ in $C_1^*$, so is $C_2$. See Figure 2 for an example of $C_1^*$ and $C_2$. By $w(C_1) = w(C_2^*) \leq w(C_2) = w(C_1^*)$, $C_1$ is a shortest non-degenerate cycle of $G_1$.

The rest of the paper proves Lemma 2.1.

3 Divide-and-conquer via balanced separating cycles

Let $C$ be a simple non-degenerate cycle of a bidirected plane graph $G$ with nonnegative edge weights. Let $int_G(C)$ (respectively, $ext_G(C)$) denote the subgraph of $G$ consisting of the nodes and edges on the boundary of the faces of $G$ inside (respectively, outside) of $C$. A non-degenerate cycle $C_3$ of $G$ is $C$-short if one of $C_1$, $C_2$, and $C_3$ is a shortest non-degenerate cycle.
of $G$, where $C_1$ (respectively, $C_2$) is a shortest non-degenerate cycle of $\text{int}_G(C)$ (respectively, $\text{ext}_G(C)$). We say that $C$ is segmented if it consists of the following three paths in order: (1) a shortest path $P_1$, (2) an edge, and (3) the reverse of a shortest path $P_2$, where one of $P_1$ and $P_2$ is allowed to be a node. Let shortest paths $P_1$ and $P_2$ be the segments of $C$. See Figure 3(a) for an example. This section proves Lemma 2.1 using Lemmas 3.1, 3.2, and 3.3. Section 4 proves Lemma 3.2 (Henzinger, Klein, Rao, and Subramanian [31]). It takes $O(n)$ time to compute a shortest-path tree rooted at any given node of an $n$-node connected simple directed planar graph with nonnegative edge weights.

Lemma 3.3 (Lipton and Tarjan [48], Goodrich [29], Klein, Mozes, and Sommer [41] Lemma 1). Let $G_0$ be an $n$-node simple undirected plane triangulation with nonnegative face weights summing to 1 such that the weight of each face of $G_0$ is at most $\frac{1}{4}$. Given any spanning tree $T$ of $G_0$, it takes $O(n)$ time to obtain an edge $e$ of $G_0 \setminus T$ such that the total weight of the faces of $G_0$ inside (respectively, outside) of the simple cycle in $T \cup \{e\}$ is no more than $\frac{1}{4}$.

Proof of Lemma 2.1 We give a divide-and-conquer recursive algorithm on the input graph $H$, which can be assumed to be connected without loss of generality. For each degree-2 node $y$ whose neighbors $x$ and $z$ are non-adjacent, we replace $y$ and its incident edges by edges $xz$ and $zx$ with weights $w(xy) + w(yz)$ and $w(zy) + w(xy)$, respectively. The resulting graph $G$ obtainable in $O(n)$ time from $H$ remains an $O(1)$-degree simple connected bidirected plane graph. See Figure 3 for an example of $H$ and $G$. Let $\ell$ be the number of faces in $G$. Since each maximal simple path on the degree-2 nodes of $G$ has $O(1)$ edges, $G$ has $O(\ell)$ nodes. A shortest non-degenerate cycle of $H$ can be obtained in $O(n)$ time from a shortest non-degenerate cycle of $G$, which can be found in $O(1)$ time for the case with $\ell \leq 4$. 

![Figure 3](image-url) (a) $G$, $C$, and $C^*$ (b) $H$
To obtain a shortest non-degenerate cycle of \( G \) for the case with \( \ell \geq 5 \), let \( T \) be an \( O(n) \)-time obtainable shortest-path tree of \( G \) rooted at an arbitrary node as ensured by Lemma 3.2. For each face \( f \) of the simple undirected unweighted version \( G_0 \) of \( G \) having size \( k \geq 3 \), (1) let \( f \) be triangulated into \( k - 2 \) faces via adding \( k - 3 \) edges without introducing multiple edges, (2) let an arbitrary one of the \( k - 2 \) faces be assigned weight \( \frac{1}{k} \), and (3) let the remaining \( k - 3 \) faces be assigned weights 0. Let \( G_\triangle \) be the resulting simple plane triangulation. The undirected version \( T_0 \) of \( T \) is a spanning tree of \( G_0 \) and \( G_\triangle \). See Figure 4 for an example. Lemma 3.3 ensures an edge \( xy \) of \( G_\triangle \setminus T_0 \) obtainable in \( O(n) \) time such that the face weights of \( G_\triangle \) inside (respectively, outside) of the simple cycle of \( T_0 \cup \{xy\} \) sum to at most \( \frac{3}{4} \). For instance, such an edge \( xy \) in the example in Figure 4 is \( v_3v_4 \). If \( x \) and \( y \) are adjacent in \( G \), then let \( E = \emptyset \); otherwise, let \( E \) consist of edges \( xy \) and \( yx \) with weights \( \infty \). We union \( G \) and \( E \) to obtain a simple bidirected plane graph \( G^* \). Let \( s \) be the least common ancestor of \( x \) and \( y \) in \( T \). Let \( C \) be the segmented simple non-degenerate cycle of \( G^* \) consisting of (1) the \( sx \)-path of \( T \), (2) edge \( xy \), and (3) the reverse of the \( sy \)-path of \( T \). By Lemma 3.1 it takes \( O(n \log \log n) \) time to compute a \( C \)-short cycle \( C_3^* \) of \( G^* \). Let \( H_1 = \text{int}_{G^*}(C) \setminus E \) and \( H_2 = \text{ext}_{G^*}(C) \setminus E \). No matter \( E = \emptyset \) or not, \( H_1 \) and \( H_2 \) are subgraphs of \( G \). We recursively compute a shortest non-degenerate cycle \( C_1 \) (respectively, \( C_2 \)) in \( H_1 \) (respectively, \( H_2 \)), which is also a shortest non-degenerate cycle of \( \text{int}_{G^*}(C) \) (respectively, \( \text{ext}_{G^*}(C) \)). By definition of \( C_3^* \), a cycle \( C^* \) in \( \{C_1, C_2, C_3\} \) with minimum weight is a shortest non-degenerate cycle of \( G^* \). If \( C^* \) passes an edge in \( E \neq \emptyset \), then the weight of each non-degenerate cycle of \( G^* \) and \( G \) is \( \infty \). Otherwise, we return \( C^* \) as a shortest non-degenerate cycle of \( G \). The algorithm runs in \( O(n \log \log n) \) time without accounting for the time for its subsequent recursive calls. By \( \ell \geq 5 \), the number \( \ell_1 \) (respectively, \( \ell_2 \)) of faces in \( H_1 \) (respectively, \( H_2 \)) is at most \( \frac{3}{4}\ell + 1 \leq \frac{19}{8}\ell \), implying that there are \( O(\log n) \) levels of recursion. By \( \ell_1 + \ell_2 \leq \ell + 2 \), the overall number of faces in each recursion level is \( O(n) \), implying that the overall number of nodes in each recursion level is \( O(n) \). The algorithm runs in \( O(n \log n \log \log n) \) time.
Figure 5: (a) With $P = su_1u_2t$ and $C = su_1u_2tvu$, the red dotted cycle $u_1u_2vu_1$ and the blue dashed cycle $u_1vu_2u_1$ are $(C, P)$-cycles of $G$ with minimum weight 2. (b) $H$ is obtained from incising $G$ along $P$.

4 Non-degenerate cycles that cross the separating cycle

This section proves Lemma 3.1 by Lemma 4.2 which is proved by Lemmas 3.2 and 4.1. Section 5 proves Lemma 4.1. If graph $G$ has $uv$-paths, then let $d_G(u, v)$ denote the weight of a shortest $uv$-path of $G$. If $G$ has no $uv$-paths, then let $d_G(u, v) = \infty$.

**Lemma 4.1.** Let $G$ be an $n$-node simple connected bidirected plane graph with nonnegative edge weights. Let $u_1, \ldots, u_t, v_1, \ldots, v_{t-1}$ be $O(n)$ nodes on the boundary of the external face of $G$ in order. It takes overall $O(n \log \log n)$ time to compute $d_G(u_i, v_i)$ for each $i = 1, \ldots, t$.

Let $G$ be a simple bidirected plane graph. A simple path $Q$ of $G$ aligns with subgraph $H$ of $G$ if $Q$ or the reverse of $Q$ is a path of $H$. A simple path $Q$ of $G$ passing at least one edge deviates from subgraph $H$ of $G$ if the edges and the internal nodes of $Q$ are not in $H$. For any simple path $P$ of $G$, a non-degenerate cycle of $G$ is a $P$-cycle if it consists of a path aligning with $P$ and a path deviating from $P$. For any non-degenerate cycle of $G$ and any path $P$ of $G$ aligning with $C$, a $P$-cycle is a $(C, P)$-cycle if the first edge of its path deviating from $P$ is in $int_G(C)$ and only if the last edge of its path deviating from $P$ is in $ext_G(C)$. For instance, the $C^*$ in Figure 5(a) is a $P_1$-cycle of $G$ whose path aligning with $P_1$ is node $v_2$. The first edge $v_2v_3$ of its path deviating from $P_1$ is in $ext_G(C)$ (respectively, $int_G(C)$), so $C^*$ is a $(C, P_1)$-cycle of $G$. $C^*$ is also a $(C, P_2)$-cycle. A non-degenerate cycle of $G$ is $(C, P)$-short if its weight is no more than that of any $(C, P)$-cycle of $G$.

**Lemma 4.2.** Let $G$ be an $n$-node $O(1)$-degree simple bidirected plane graph with nonnegative edge weights. Let $C$ be a simple non-degenerate cycle of $G$. Given a path $P$ of $G$ aligning with $C$, it takes $O(n \log \log n)$ time to compute a $(C, P)$-short cycle of $G$.

**Proof.** Let $C^*$ be a $(C, P)$-cycle of $G$ with minimum weight. For instance, the red and blue cycles in Figure 5(a) are two $(C, P)$-cycles with minimum weight 2. Let $C_0$ be a shortest non-degenerate cycle of $G$ passing at least one endpoint of $P$, which can be obtained in $O(n)$ time via examining shortest $uv$-paths in $G \setminus \{uv, vu\}$ by Lemma 5.2 for all $O(1)$ edges $uv$ of $G$ incident to at least one endpoint of $P$. If $C^*$ passes some endpoint of $P$, then $w(C_0) \leq w(C^*)$, implying
Figure 6: (a) The red dotted $P_1$-cycle not intersecting $P_2$ is in $\text{int}_G(C)$. The blue dashed $P_2$-cycle not intersecting $P_1$ is in $\text{ext}_G(C)$. (b) The red dotted cycle consists of $Q_1$, $R_1$, $Q_2$, and $R_2$ in order is a $(C, P_1)$-cycle. (c) The degenerate cycle $C'$ is obtained from the non-degenerate red dotted cycle $C^*$ by replacing the $u_iv_1$-path of $C^*$ with the blue dashed $u_1v_1$-path of $P_1$. The green dash-dotted cycle $C''$ is a non-degenerate cycle contained by $C'$.

that $C_0$ is a cycle ensured by the lemma. The rest of the proof assumes that $C^*$ does not pass any endpoint of $P$. Thus, $P$ has internal nodes. Let $H$ be an $O(n)$-node $O(1)$-degree simple bidirected plane graph obtainable in $O(n)$ time as follows: Suppose that $u_0, \ldots, u_{\ell+1}$ with $\ell \geq 1$ are the nodes of $P$ in order. Let $s = v_0 = u_0$ and $t = v_{\ell+1} = u_{\ell+1}$. We incise $G$ along $P$ by

- adding new nodes $v_1, \ldots, v_\ell$, a new path $P' = sv_1 \cdots v_\ell t$ and the reverse of $P'$,
- for each $i = 1, \ldots, \ell$, let each edge $vu_i$ (respectively, $u_i v$) incident to $u_i$ in $\text{int}_G(C) \setminus P$ be replaced by $vv_i$ (respectively, $v_i v$) with the same weight,
- letting the weight of each edge in $P'$ and the reverse of $P'$ be $\infty$, and
- embedding the resulting graph $H$ such that $P$ and $P'$ are on the external face.

See Figure 5 for an example. By Lemma 4.1, it takes overall $O(n \log \log n)$ time to compute $d_H(u_i, v_i)$ and $d_H(v_i, u_i)$ for each $i \in \{1, \ldots, \ell\}$. Let $i_1$ (respectively, $i_2$) be an $i \in \{1, \ldots, \ell\}$ that minimizes $d_H(u_i, v_i)$ (respectively, $d_H(v_i, u_i)$). By Lemma 3.2, it takes $O(n)$ time to obtain a simple shortest $u_1v_1$-path $P_1$ of $H$ and a simple shortest $v_1u_1$-path $P_2$ of $H$. The weight of $P_1$ (respectively, $P_2$) is minimum over all $u_1v_1$-paths (respectively, $v_1u_1$-paths) of $H$ with $1 \leq i \leq \ell$. Let $C_1$ (respectively, $C_2$) be the non-degenerate cycle of $G$ corresponding to $P_1$ (respectively, $P_2$). Let $Q$ be the path of $C^*$ that deviates from $P$. Let $u_i$ and $u_j$ with $1 \leq i, j \leq \ell$ be the first and last nodes of $Q$, respectively. If the first edge of $Q$ is in $\text{int}_G(C)$, then $C^*$ corresponds to a $v_1u_1$-path of $H$, implying $w(C_1) \leq w(C^*)$. If the last edge of $Q$ is in $\text{int}_G(C)$, then $C^*$ corresponds to a $u_jv_j$-path of $H$, implying $w(C_2) \leq w(C^*)$. For instance, the red (respectively, blue) cycle of $G$ in Figure 5(a) corresponds to the red $u_1v_1$-path (respectively, blue $v_1u_1$-path) of $H$ in Figure 5(b). Thus, one of $C_0$, $C_1$, and $C_2$ with minimum weight is a cycle ensured by the lemma.

Proof of Lemma 3.7 Let $G_1 = \text{int}_G(C)$ and $G_2 = \text{ext}_G(C)$. Let $P_1$ and $P_2$ be the given segments of $C$. Let $C^*$ be a shortest non-degenerate cycle of $G$ whose number of edges not in $P_1 \cup P_2$ is minimized over all shortest non-degenerate cycles of $G$. If $C^*$ is a cycle of $G_1$ or $G_2$, then
any cycle of $G$ is $C$-short, including the one ensured by Lemma 4.2. The rest of the proof assumes that neither $G_1$ nor $G_2$ contains $C^*$. By Lemma 4.2 it suffices to ensure that $C^*$ is a $(C, P_1)$-cycle. We need the following claim:

For each $i \in \{1, 2\}$, if $C^* \cap P_i \neq \emptyset$, then $C^*$ is a $P_i$-cycle of $G$.

By the claim, $C^*$ intersects both $P_1$ and $P_2$ or else $C^*$ would be a cycle of $G_1$ or $G_2$, as illustrated by Figure 6(a), contradicting the assumption. Since $C^*$ is a $P_1$-cycle and a $P_2$-cycle, $C^*$ consists of four paths $Q_1, R_1, Q_2$, and $R_2$ in order such that $Q_i$ aligns with $P_i$ and $R_i$ deviates from $P_1 \cup P_2$ for each $i \in \{1, 2\}$. By the assumption, if $R_1 \subseteq G_i$ and $R_2 \subseteq G_j$, then $\{i, j\} = \{1, 2\}$. Thus, $C^*$ is a $(C, P_1)$-cycle. See Figure 6(b) for an illustration. It remains to prove the claim. Assume for contradiction that $C^*$ intersects $P_i$ but is not a $P_i$-cycle for an index $i \in \{1, 2\}$. There are nodes $u_1, v_1, u_2, v_2$ of $P_i$ with $u_1 \neq v_1$ and $u_2 \neq v_2$ such that

- $u_1$ precedes $v_1$ in $P_i$,
- $u_2$ succeeds $v_2$ in $P_i$,
- the $u_1v_1$-path and the $u_2v_2$-path of $C^*$ deviate from $P_i$, and
- the $u_1v_1$-path of $C^*$ deviates from the $u_2v_2$-path of $C^*$.

Let $C'$ be the cycle of $G$ obtained from $C^*$ by replacing the $u_1v_1$-path of $C^*$ with the $u_1v_1$-path of $P_i$. Since $P_i$ is a shortest path of $G$, $w(C') \leq w(C^*)$. Since $C^*$ is non-degenerate, the reverse of each of the $u_2v_2$-path of $C^*$ is not in $C'$. Thus, even if $C'$ is degenerate, there is a non-degenerate cycle $C''$ in $C'$. See Figure 6(c) for an illustration. By nonnegativity of edge weights, $w(C'') \leq w(C')$. By $w(C'') \leq w(C^*)$, $C''$ is a shortest non-degenerate cycle of $G$ whose number of edges not in $P_1 \cup P_2$ is fewer than the number of edges of $C^*$ not in $P_1 \cup P_2$, contradicting the definition of $C^*$.

\section{Noncrossing shortest paths among nodes on external face}

This section proves Lemma 4.1 via extending techniques of Reif [58] and Italiano et al. [33] for undirected planar graphs. Algorithms for $r$-divisions (Lemma 5.2) and dense-distance graphs (Lemma 5.2) are reviewed in §5.1 Data structures for fast-Dijkstra algorithm (Lemma 5.5) are given in §5.2 Data structures that enables efficient partition of boundary nodes via non-crossing paths (Lemma 5.6) are given in §5.3 Tools involving noncrossing shortest paths (Lemma 5.8) are given in §5.4 Lemma 4.1 is proved by Lemmas 3.2, 5.1, 5.2, 5.5, 5.6 and 5.8 in §5.5.

\subsection{Dense-distance graph}

Let $G$ be a simple bidirected plane graph. A division $D$ of $G$ is an edge-disjoint partition of $G$ into bidirected plane subgraphs, each of which is a piece of $D$. The multiplicity of node $v$ of $G$ in $D$ is the number of pieces of $D$ containing $v$. A node of $G$ with multiplicity two or more in $D$ is a boundary node of $D$. A face of a piece of $D$ is a hole of the piece if it is not a face of $G$. For any $r > 0$, an $r$-division (see, e.g., [23, 31, 33, 41, 44]) of $G$ is a division of $G$ with $O(n/r)$ pieces, each having $O(r)$ nodes, $O(\sqrt{r})$ boundary nodes, and $O(1)$ holes.

\begin{lemma}[Klein, Mozes, and Sommer [41]]
For any $r > 0$, it takes $O(n)$ time to compute an $r$-division for an $n$-node simple bidirected plane graph each of whose faces contains at most three nodes.
\end{lemma}
Figure 7: (a) A piece $H$ in which $x_1$, $x_2$, and $x_3$ are the boundary nodes in one hole and $y_1$ and $y_2$ the boundary nodes in the other hole. (b) $K(H)$.

Let $D$ be an $r$-division of $G$. For any connected component $H$ of any piece of $D$, let $K(H)$ denote the complete directed graph on the boundary nodes of $D$ in $H$ in which $w(uv) = d_H(u, v)$. See Figure 7 for an example. The dense distance graph (see, e.g., [33]) $K(D)$ of $D$ is the $O(n)$-edge simple directed graph on the $O(n/\sqrt{r})$ boundary nodes of $D$ simplified from the union of $K(H)$ over all connected components $H$ of all pieces of $D$ by keeping exactly one copy of parallel edges with minimum weight. For any edge $uv$ of $K(D)$, an underlying $uv$-path is a $uv$-path in some connected component $H$ of some piece of $D$ with weight equal to $w(uv)$ in $K(D)$. For any path $\Pi$ of $K(D)$, an underlying path of $\Pi$ consists of an underlying $uv$-path for each edge $uv$ of $\Pi$.

**Lemma 5.2 (Klein [40]).** For any given $r$-division $D$ of an $n$-node simple bidirected plane graph with nonnegative edge weights, it takes $O(n \log r)$ time to compute $K(D)$ and a data structure from which, for any path $\Pi$ of $K(D)$, the first $c$ edges of an underlying path of $\Pi$ can be obtained in $O(c \log \log r)$ time.

### 5.2 Fast-Dijkstra algorithm

Consider the following equation

$$w(u_1v_1) + w(u_2v_2) \leq w(u_1v_2) + w(u_2v_1)$$

for distinct nodes $u_1, u_2, v_1, v_2$ of a simple directed graph $H$ with edge weights. A type-1 Monge unit is a complete $H$ equipped with a cyclic ordering for the nodes of $H$ such that Equation (1) holds for any distinct nodes $u_1, u_2, v_1, v_2$ of $H$ in order. A type-2 Monge unit is a complete bipartite $H$ equipped with an ordering for each of the two maximal independent sets of $H$ such that Equation (1) holds for any distinct nodes $u_1$ and $u_2$ of one independent set in order and any distinct nodes $v_1$ and $v_2$ of the other independent set in order.

A Monge decomposition of a simple directed graph $K$ with edge weights is a set $M$ of Monge units on node subsets of $K$ such that $K$ is the graph simplified from the union of the Monge units in $M$. The multiplicity of a node $v$ of $K$ in $M$ is the number of Monge units in $M$ that contain $v$. The size of $M$ is the sum of the multiplicities of all nodes of $K$ in $M$. An equivalent
Lemma 5.3. For any given r-division $D$ of an $n$-node simple bidirected plane graph with nonnegative edge weights, it takes $O(n \log r)$ time to obtain a Monge decomposition $M(D)$ of $K(D)$ such that the multiplicity of a node of $K(D)$ in $M(D)$ is $O(1)$ times its multiplicity in $D$.

As summarized in the following lemma, given a size-$m$ Monge decomposition of graph $K$, there are $O(m \log_2^2 m)$-time obtainable data structures for range minimum queries (see, e.g., Kaplan et al. [35] and Gawrychowski, Mozes, and Weimann [26]) with which the fast-Dijkstra algorithm of Fakcharoenphol and Rao [20] outputs a shortest-path tree of $K$ in $O(m \log^2 m)$ time.

Lemma 5.4. Given a size-$m$ Monge decomposition of a simple strongly connected directed graph $K$ with nonnegative edge weights, it takes $O(m \log^2 m)$ time to compute a shortest-path tree of $K$ rooted at any given node.

Lemma 5.5. Let $D$ be a given r-division of an $n$-node simple plane graph with nonnegative edge weights. It takes $O(n \log r)$ time to compute a data structure from which, for any subset $X$ of the boundary nodes of $D$ such that the subgraph $K$ of $K(D)$ induced by $X$ is strongly connected, it takes $O(m \log^2 m)$ time to compute a shortest-path tree of $K$ rooted at any given node, where $m$ is the sum of the multiplicities of the nodes of $X$ in $D$.

Proof. Let $M(D)$ be a Monge decomposition of $K(D)$ as ensured by Lemma 5.3. Let $M$ consist of the subgraph $H[X]$ of $H$ induced by $X$ for each Monge unit $H$ in $M(D)$. Each $H[X]$ remains

Figure 8: Each of the two graphs can be simplified from the union of two type-2 Monge units.
and \( v \) disjoint simple such that the following property: For distinct nodes \( u \) and \( v \) in \( K \), let \( D \) be an induced cyclic ordering of \( K \) preserving the property that the multiplicity of a node of \( K \) in \( M \) is \( O(1) \) times its multiplicity in \( D \), implying that the size of \( M \) is \( O(m) \). It takes overall \( O(m) \) time to obtain the induced cyclic ordering or the two induced orderings of the nodes of \( H[X] \) from \( H \) for each Monge unit \( H \) in \( M(D) \).

Since the weight of each edge of \( H[X] \) can be obtained in \( O(1) \) time from its weight in \( H \), we have an implicit representation of \( M \) in \( O(m) \) time. The lemma follows from Lemma 5.4.

### 5.3 Noncrossing paths

Let \( G \) be a simple connected bidirected plane graph. Let \( u_1, u_2, v_2, v_1 \) be distinct nodes on the boundary of the external face of connected plane graph \( G \) in order. A simple \( u_1v_1 \)-path \( P_1 \) and a simple \( u_2v_2 \)-path \( P_2 \) of \( G \) are noncrossing if \( P_1 \cap P_2 \) is empty or is a path. For instance, in Figure 9, \( P_1 \) in red and \( P_2 \) in blue are noncrossing. For noncrossing \( P_1 \) and \( P_2 \), let \( G[P_1, P_2] \) denote the connected bidirected plane subgraph of \( G \) enclosed by \( P_1, P_2 \), and the \( u_1u_2 \)-path and \( v_2v_1 \)-path on the boundary of the external face of \( G \) following the order of \( u_1, u_2, v_2, v_1 \). See Figure 9 for an example.

Let \( D \) be an \( r \)-division of \( G \). Our proof of Lemma 4.1 needs a data structure \( B(D) \) with the following property: For distinct nodes \( u_1, u_2, v_3, v_2, v_1 \) on the external face of \( G \) in order, any disjoint simple \( u_1v_1 \)-path \( P_1 \) and \( u_3v_3 \)-path \( P_3 \) of \( G \), and any simple \( u_2v_2 \)-path \( P_2 \) of \( G \) such that \( P_1 \) and \( P_2 \) are noncrossing, given \( X(1, 3) \) and \( P_2 \setminus P_1 \), it takes \( O((m_1 + m_2) \log r) \) time to obtain \( X(1, 2) \) and \( X(2, 3) \), where \( X(i, j) \) with \( 1 \leq i < j \leq 3 \) consists of the boundary nodes of \( D \) in \( G[P_1, P_3] \), \( m_1 \) is the sum of multiplicities of the nodes of \( X(1, 3) \) in \( D \), and \( m_2 \) is the number of edges in \( P_2 \setminus P_1 \). See Figure 10 for an illustration.

**Lemma 5.6.** It takes \( O(n) \) time to compute a data structure \( B(D) \) for any given \( r \)-division \( D \) of any \( n \)-node simple connected bidirected plane graph.

**Proof.** Given \( X(1, 3) \) and the edge set \( E \) of \( P_2 \setminus P_1 \), it takes \( O(m_1 + m_2) \) time to obtain the nodes of \( X(1, 3) \) in \( E \), which belongs to \( X(1, 2) \cap X(2, 3) \). Let \( X \) consist of the nodes of \( X(1, 3) \) not in \( E \). If \( X = \emptyset \), then \( X(1, 2) = X(2, 3) = X(1, 3) \). The rest of the proof assumes \( X \neq \emptyset \). Let \( \mathbb{H}_0 \) (respectively, \( \mathbb{H}_1 \)) consist of the pieces \( H \) of \( D \) such that \( H \) contains nodes of \( X \) and no (respectively, some) edges of \( E \). We have \( \mathbb{H}_1 \neq \emptyset \), since \( G[P_1, P_3] \) is connected and \( E \neq \emptyset \). Let \( A \) be the \( O(m_1 + m_2) \)-time obtainable undirected bipartite graph on the nodes \( x \) in \( X \) and the pieces \( H \) of \( D \) in \( \mathbb{H}_0 \) such that \( H \) contains nodes of \( X \) and \( H \) contains \( x \) in \( G \). The nodes of \( X \) in the same connected component of \( A \) either all belong to \( X(1, 2) \) or all belong to \( X(2, 3) \). Since \( G[P_1, P_3] \) is connected, each connected component of \( A \) contains a node of \( X \).
that belongs to a piece of $H \in \mathcal{H}_1$ in $G$. It takes overall $O(m_1 + m_2)$ time to obtain $H \cap E$, $C \cap E$, and $C \cap X$ for each hole $C$ of each piece $H$ of $D$ in $\mathcal{H}_1$. Since each piece of $D$ has $O(1)$ holes, it remains to show that with the $B(D)$ defined below, for each hole $C$ of each piece $H$ of $D$ in $\mathcal{H}_1$, it takes $O(m \log r)$ time to determine the nodes of $C \cap X$ in $X(1, 2)$, where $m$ is the number of nodes in $H \cap X$ plus the number of edges in $H \cap E$.

Assume without loss of generality that the external face of each piece $H$ of $D$ is a hole of $H$. The $O(n)$-time obtainable data structure $B(D)$ consists of (1) the cyclic ordering of the incident edges around each node of $G$ and (2) the following items for each hole $C$ of each piece $H$ of $D$:

- An arbitrary simple path $Q$ of $H$ from a node of $C$ to a node $q$ on the external face of $H$.
- The ordering indices of the nodes on $Q$.
- The cyclic ordering indices of the nodes on $C$.

It takes overall $O(m_1 + m_2)$ time to obtain $Q \cap E$ for each hole $C$ of each piece $H$ of $D$ in $\mathcal{H}_1$. With the first part of $B(D)$, if $uv$ is an edge of $G[P_1, P_3]$ with $u \in P_2$ and $v \notin P_2$, then it takes $O(1)$ time to determine whether $v \in G[P_1, P_2]$. With the second part of $B(D)$, for any $k$-node subset $U$ of any piece $H$ of $D$ and any hole $C$ of $H$, it takes $O(k)$ time to determine the ordering indices of the nodes of $U \cap Q$ in $Q$ and the cyclic ordering indices of the nodes of $U \cap C$ in $C$.

Case 1: $C \cap E \neq \emptyset$. As illustrated by Figure 11(a), it takes overall $O(m \log r)$ time via sorting their ordering indices to compute, for each node $x$ of $C \cap X$, the first node $u \in E$ in the traversal of $C$ starting from $x$ following the order of $u_1, u_3, v_3, v_1$ and the node $v$ of $C$ preceding $u$ in the traversal. We have $x \in X(1, 2)$ if and only if $v \in G[P_1, P_2]$, which can be determined in $O(1)$ time.

Case 2: $C \cap E = \emptyset$. As illustrated by Figure 11(b), if $Q \cap E \neq \emptyset$, then let $v$ be the node preceding the first node $u$ of $Q$ in $E$. Let $C'$ be the boundary of the external face of $H$. As illustrated by Figure 11(c), if $Q \cap E = \emptyset$, then let $v$ be the node of $C'$ preceding the first node $u$ of $C'$ in $E$ on the traversal of $C'$ starting from $q$ following the order of $u_1, u_3, v_3, v_1$. Either way, it takes $O(m)$

---

Figure 10: An illustration for the definition of $B(D)$, where $P_1$ is the blue solid $u_1v_1$-path, $P_3$ is the green dash-dotted $u_3v_3$-path, and $P_2$ is the red dotted $u_2v_2$-path. $P_1$ and $P_3$ are disjoint. $P_1$ and $P_2$ are noncrossing. (a) $G[P_1, P_3]$, in which the boundary nodes of $D$ form $X(1, 3)$. (b) $G[P_1, P_2]$, in which the boundary nodes of $D$ form $X(1, 2)$. (c) $G[P_2, P_3]$, in which the boundary nodes of $D$ form $X(2, 3)$. 

---

Figure 11: An illustration for the definition of $B(D)$, where $P_1$ is the blue solid $u_1v_1$-path, $P_3$ is the green dash-dotted $u_3v_3$-path, and $P_2$ is the red dotted $u_2v_2$-path. $P_1$ and $P_3$ are disjoint. $P_1$ and $P_2$ are noncrossing. (a) $G[P_1, P_3]$, in which the boundary nodes of $D$ form $X(1, 3)$. (b) $G[P_1, P_2]$, in which the boundary nodes of $D$ form $X(1, 2)$. (c) $G[P_2, P_3]$, in which the boundary nodes of $D$ form $X(2, 3)$. 

---

Figure 12: An illustration for the definition of $B(D)$, where $P_1$ is the blue solid $u_1v_1$-path, $P_3$ is the green dash-dotted $u_3v_3$-path, and $P_2$ is the red dotted $u_2v_2$-path. $P_1$ and $P_3$ are disjoint. $P_1$ and $P_2$ are noncrossing. (a) $G[P_1, P_3]$, in which the boundary nodes of $D$ form $X(1, 3)$. (b) $G[P_1, P_2]$, in which the boundary nodes of $D$ form $X(1, 2)$. (c) $G[P_2, P_3]$, in which the boundary nodes of $D$ form $X(2, 3)$.
time to obtain $v$ and determine whether $v \in G[P_1, P_2]$. If $v \in G[P_1, P_2]$, then $C \cap X \subseteq X(1,2)$. Otherwise, $C \cap X \subseteq X(2,3)$.

### 5.4 Noncrossing shortest paths

**Lemma 5.7.** Let $G$ be a simple connected bidirected plane graph with nonnegative edge weights. If nodes $u_1, u_2, v_2, v_1$ are on the boundary of the external face of $G$ in order, then for any shortest $u_1v_1$-path $P_1$ of $G$, there is a shortest $u_2v_2$-path $P_2$ of $G$ such that $P_1$ and $P_2$ are noncrossing.

**Proof.** As illustrated by Figure 12, suppose that $P'_2$ is a shortest $u_2v_2$-path of $G$ with $P_1 \cap P'_2 \neq \emptyset$. Let $u$ (respectively, $v$) be the first (respectively, last) node of $P'_2$ in $P_1$. Let $P_2$ be obtained from $P'_2$ by replacing its $uv$-path with the $uv$-path of $P_1$. By the order of $u_1, u_2, v_2, v_1$ on the boundary of the external face of $G$, $P_2$ is well defined and is a shortest $u_2v_2$-path of $G$ such that $P_1$ and $P_2$ are noncrossing.

**Lemma 5.8.** Let $G$ be an $n$-node simple connected bidirected plane graph with nonnegative edge weights. Let $u_1, \ldots, u_k, v_k, \ldots, v_1$ be $2k$ distinct nodes on the boundary of the external face of $G$ in
order. For each \( i \in \{1, k\} \), let \( P_i \) be a simple shortest \( u_i v_i \)-path of \( G \) such that \( P_1 \) and \( P_k \) are noncrossing. Let \( h \) be the number of nodes of \( G[P_1, P_k] \) not in \( P_1 \cap P_k \). Given \( P_1 \setminus P_k \) and \( P_k \setminus P_1 \), consider the problem of computing \( d_G(u_i, v_i) \) for all \( i = 1, \ldots, k \).

1. If \( P_1 \cap P_k = \emptyset \), then the problem can be solved in \( O(h \log k) \) time.
2. If \( P_1 \cap P_k = \emptyset \) and we are given a set \( Z \) of \( O(1) \) nodes such that for each \( i = 1, \ldots, k \) at least one shortest \( u_i v_i \)-path passes at least one node of \( Z \), then the problem can be solved in \( O(h) \) time.
3. If \( P_1 \cap P_k \neq \emptyset \) and we are given \( w(P_1 \cap P_k) \), then the problem can be solved in \( O(h) \) time.

Proof. Since \( P_1 \setminus P_k \) and \( P_k \setminus P_1 \) are given, it takes \( O(h) \) time to obtain \( G[P_1, P_k] \) excluding the edges and internal nodes of \( P_1 \cap P_k \). Statements 2 and 3 follow from Lemmas 3.2 and 5.7. As for Statement 1, under the assumption that a simple shortest \( u_a v_a \)-path \( P_a \) and a simple shortest \( u_b v_b \)-path \( P_b \) of \( G \) are given and disjoint, below is the recursive algorithm \( \text{MEASURE}(a, b) \) with \( 1 \leq a < b \leq k \) for solving the \((a, b)\)-subproblem of computing \( d_G(u_i, v_i) \) for all indices \( i \) with \( a < i < b \):

Let \( i = \lfloor (a + b)/2 \rfloor \). By Lemma 3.2 it takes time linear in the number of nodes in \( G[P_a, P_b] \) to obtain \( d_G(u_i, v_i) \) and a simple shortest \( u_i v_i \)-path \( P_i \) of \( G[P_a, P_b] \) that is noncrossing with both \( P_a \) and \( P_b \). For the \((a, i)\)-subproblem, if \( P_a \cap P_i = \emptyset \), then call \( \text{MEASURE}(a, i) \); otherwise, apply Statement 2 with \( Z \) consisting of an arbitrary node in \( P_a \setminus P_i \). For the \((i, b)\)-subproblem, if \( P_i \cap P_b = \emptyset \), then call \( \text{MEASURE}(i, b) \); otherwise, apply Statement 2 with \( Z \) consisting of an arbitrary node in \( P_i \setminus P_b \).

The algorithm for the statement obtains \( d_G(u_1, v_1) \) and \( d_G(u_k, v_k) \) from \( P_1 \) and \( P_k \) and calls \( \text{MEASURE}(1, k) \). Since each \( d_G(u_i, v_i) \) with \( 1 < i < k \) is computed by Lemma 3.2 or Statement 2 the correctness holds trivially. By the choice of \( i \), \( \text{MEASURE}(1, k) \) runs in \( O(\log k) \) levels of recursion. Since \( P_a \cap P_b = \emptyset \) holds for each call to \( \text{MEASURE}(a, b) \), each node of \( G[P_a, P_b] \) appears in at most two subgraphs \( G[P_a, P_b] \) in the same level of recursion. Thus, the overall running time for each level of recursion is \( O(h) \). The algorithm runs in \( O(h \log k) \) time.

5.5 Proving Lemma 4.1

Proof of Lemma 2.7. For each \( i = 1, \ldots, \ell \), let \( d_i = d_G(u_i, v_i) \). With the modification below, each \( d_i \) with \( 1 \leq i \leq \ell \) equals the weight of a shortest \( u_i' v_i' \)-path in the resulting \( G \), which remains an \( O(n) \)-node simple connected bidirected plane graph: (1) for each \( i = 1, \ldots, \ell \), add new nodes \( u_i' \) and \( v_i' \) in the external face, zero-weighted edges \( u_i' u_i \) and \( v_i' v_i \), and \( \infty \)-weighted edges \( u_i u_i' \) and \( v_i v_i' \); (2) contract each zero-weighted strongly connected subgraph into a single node; (3) delete all self-loops, and (4) delete all except one copy of each set of multiple edges with minimum weight. Thus, the rest of the proof assumes that \( u_1, u_{\ell}, v_{\ell}, \ldots, v_1 \) are distinct and \( G \) does not have any zero-weighted cycles, implying that all shortest paths of \( G \) are simple.

Let \( G_\Delta \) be an \( O(n) \)-node bidirected plane graph obtainable in \( O(n) \) time from \( G \) by identifying nodes \( u_i \) and \( v_i \) into a new node \( z_i \) for each \( i = 1, \ldots, \ell \) and then triangulating each face of size larger than 3. Let

\[
r = \max(1, \lfloor \log_2^3 n \rfloor).
\]

By Lemma 5.1 an \( r \)-division \( D_0 \) for \( G_\Delta \) can be computed in \( O(n) \) time. Let \( D_1 \) be the division of \( G \) induced by \( D_0 \): Each piece of \( D_1 \) is obtained from a piece of \( D_0 \) by deleting the edges added to triangulate faces of size larger than 3. Each piece of \( D_0 \) has \( O(r) \) nodes, \( O(\sqrt{r}) \) boundary nodes, and \( O(1) \) holes, so does each piece of \( D_1 \). Let \( I \) consist of indices 1 and \( \ell \) and the indices \( i \) such that at least one of \( u_i \) and \( v_i \) is a boundary node of \( D_1 \). Since each \( z_i \) with \( i \in I \setminus \{1, \ell\} \) is a boundary node in \( D_0 \), the cardinality of \( I \) is \( O(n/\sqrt{r}) \). To turn both of \( u_i \) and \( v_i \) with \( i \in I \)
Figure 13: Subroutine \textsc{Solve}(a, b).

into boundary nodes, we introduce \(O(n/r)\) new \(O(\sqrt{r})\)-node pieces, which form a partition of the nodes \(u_i\) and \(v_i\) with \(i \in I\). Let \(D\) be the resulting division of \(G\). Each new piece of \(D\) has \(O(\sqrt{r})\) nodes and no edges, so it has \(O(\sqrt{r})\) boundary nodes and \(O(1)\) holes. Thus, \(D\) is an \(r\)-division of \(G\) such that each \(u_i\) with \(1 \leq i \leq \ell\) is a boundary node in \(D\) if and only if so is \(v_i\). Let \(G'\) be the simple bidirected plane graph with edge weights obtained from \(G\) by reversing the direction of each edge. Let \(D'\) be the \(r\)-division of \(G'\) corresponding to \(D\). By Equation \((2)\), it takes \(O(n \log \log n)\) time to compute \(K(D)\) and \(K(D')\) and the data structures ensured by Lemmas \(\ref{lem:improvement}\) and \(\ref{lem:label}\).

For any nodes \(x\) and \(y\) in a shortest path \(P\) of \(G\), let \(P[x, y]\) denote the \(xy\)-path of \(P\). We need a subroutine \textsc{Label}(\(P\)) to compute label \(\phi(z)\) for each node \(z\) of a shortest path \(P\) of \(G\) under the assumption that \(\phi(z)\) for at most one node of \(P\) is pre-computed:

\[
\phi(z) = \begin{cases} 
\phi(z^*) & \text{if there is no such a node, then let} \\
0 & \text{otherwise}
\end{cases}
\]

Subroutine \textsc{Label}(\(P\)) runs in \(O(1)\) time per node of \(P\) and does not overwrite \(\phi(z)\) for any \(z\) with pre-computed \(\phi(z)\). After running \textsc{Label}(\(P\)), for any nodes \(x\) and \(y\) of \(P\), \(w(P[x, y])\) can be obtained from \(\phi(y) - \phi(x)\) in \(O(1)\) time.

For any indices \(a\) and \(b\), let set \(I(a, b)\) consist of the indices \(i \in I\) with \(a < i < b\). For each \(i \in \{1, \ell\}\), let \(P_i\) be a shortest \(u_i,v_i\)-path of \(G\) obtainable in \(O(n)\) time by Lemma \(\ref{lem:division}\). If \(P_1 \cap P_\ell \neq \emptyset\), then the lemma follows from Lemma \(\ref{lem:improvement}\) with \(Z = \{x\}\) for an arbitrary node \(x \in P_1 \cap P_\ell\). The rest of the proof assumes \(P_1 \cap P_\ell = \emptyset\). The algorithm proving the lemma calls \textsc{Label}(\(P_1\)), \textsc{Label}(\(P_\ell\)), and \textsc{Solve}(1, \ell), where the main subroutine \textsc{Solve}(\(a, b\), as defined in Figure \(\ref{fig:solve}\) and elaborated below, solves the \((a,b)\)-subproblem of computing \(d_i\) for all indices \(i\) with \(a \leq i \leq b\) under the condition that 

\begin{itemize}
  \item \textbf{Subroutine} \textsc{Solve}(\(a, b\))
  \item If \(I(a, b) = \emptyset\), then solve the \((a,b)\)-subproblem by Lemma \(\ref{lem:improvement}\) and return.
  \item If \(I(a, b) \neq \emptyset\), then let \(i\) be a median of \(I(a, b)\) and let \(P\) (respectively, \(P'\)) be a shortest \(u_i,v_i\)-path whose first (respectively, last) \(c\) edges can be obtained in \(O(c \log \log r)\) time.

  \begin{enumerate}
    \item Case 1: \(P \cap (P_a \cup P_b) = \emptyset\). Let \(P_i = P\), Call \textsc{Label}(\(P_i\)), \textsc{Solve}(\(a, i\)), and \textsc{Solve}(\(i, b\)). Return.
    \item Case 2: \(P \cap (P_a \cup P_b) \neq \emptyset\).
      \begin{itemize}
        \item Call \textsc{Label}(\(P[u_i,x]\)), where \(x\) is the first node of \(P\) in \(P_a \cup P_b\).
        \item Call \textsc{Label}(\(P'[y,v_i]\)), where \(y\) is the last node of \(P'\) in \(P[a,v_i]\) \(P_a \cup P_b\).
      \end{itemize}
  \end{enumerate}
\end{itemize}
Figure 14: Illustrations for the proof of Lemma 4.1. All $P_a$ and $P_b$ are in black. Each $P[u_i, x]$ is in red dots. Each $P'[y, v_i]$ is in blue dashes.

- shortest $u_a v_a$-path $P_a$ of $G$ and shortest $u_b v_b$-path $P_b$ of $G$ are disjoint,
- $\phi(z)$ is pre-computed for each node $z \in P_a \cup P_b$, and
- the set $X(a, b)$ of boundary nodes of $D$ in $G[P_a, P_b]$ is given.

By Equation (2), it remains to prove that $SOLVE(1, \ell)$ correctly solves the $(1, \ell)$-subproblem in $O(n \log r)$ time. If $I(a, b) = \emptyset$, then all $u_i$ with $a < i < b$ are not boundary nodes in $D$. Since these $u_i$ induce a connected subgraph of $G$, they belong to a common piece of $D$, implying $b - a = O(r)$. The $(a, b)$-subproblem can be solved by Lemma 5.8(1) in $O(h(a, b) \log r)$ time, where $h(a, b)$ is the number of nodes in $G[P_a, P_b]$ that are not in $P_a \cap P_b$.

For the case with $I(a, b) \neq \emptyset$, we cannot afford to directly compute a shortest $u_iv_i$-path $P_i$ of $G$ for a median $i$ of $I(a, b)$ by Lemma 5.2. Instead, in the subgraph of $K(D)$ induced by the given set $X(a, b)$ of boundary nodes of $D$ in $G[P_a, P_b]$ we compute a shortest $u_iv_i$-path $\Pi$ (respectively, $\Pi'$) of $K(D)$ (respectively, $K(D')$), the first (respectively, last) $c$ edges of whose underlying path $P$ (respectively, $P'$) can be obtained in $O(c \log \log r)$ time by Lemma 5.2. By Lemma 5.7 $G[P_a, P_b]$ contains at least one shortest $u_iv_i$-path of $G$, implying that the subgraph of $K(D)$ induced by $X(a, b)$ contains at least one shortest $u_iv_i$-path of $K(D)$. Therefore, $P$ and $P'$ are shortest $u_iv_i$-paths of $G$ in $G[P_a, P_b]$. If $P$ does not intersect $P_a \cup P_b$, then it takes $O(\log \log r)$ time per node to obtain $P$. As in Case 1 of Figure 13 the subroutine lets $P_i = P$ and calls $LABEL(P_i)$, $SOLVE(a, i)$, and $SOLVE(i, b)$. If $P$ intersects $P_a \cup P_b$ it takes $O(\log \log r)$ time per node to obtain $P[u_i, x]$ and $P'[y, v_i]$, where $x$ is the first node of $P$ in $P_a \cup P_b$ and $y$ is the last node of $P'$ in $P[u_i, x] \cup P_a \cup P_b$, as stated by the first two bullets in Case 2 of Figure 13. The subroutine calls $LABEL(P[u_i, x])$ and $LABEL(P'[y, v_i])$.

- As illustrated by Figure 14(a), if each of $P_a$ and $P_b$ contains exactly one of $x$ and $y$, then the $(a, b)$-subproblem is solved in $O(h(a, b))$ time by Lemma 5.8(2) with $Z = \{x, y\}$, as stated by the first bullet in Case 2(1) of Figure 13.
- As illustrated by Figure 14(b), if $x, y \in P_a$, then let $P_i = P[u_i, x]P_a[x, y]P'[y, v_i]$. The $(i, b)$-subproblem is solved by calling $SOLVE(i, b)$. The $(a, i)$-subproblem is solved by Lemma 5.8(3) with $w(P_a \cap P_i) = \phi(y) - \phi(x)$ in $O(h(a, b))$ time. The case with $x, y \in P_b$ is similar. The second bullet of Case 2(1) in Figure 13 states these two cases.
- As illustrated by Figure 14(c), if $x \in P_a$ and $y \notin P_a \cup P_b$ then the shortest $u_iv_i$-path $P_i = P[u_i, y]P'[y, v_i]$ is disjoint with $P_a \cup P_b$. The $(i, b)$-subproblem is solved by calling $SOLVE(i, b)$. Since at least one shortest $u_iv_i$-path of $G[P_a, P_i]$ passes $x$, the $(a, i)$-subproblem can be solved in $O(h(a, b))$ time by Lemma 5.8(2) with $Z = \{x\}$. The case with $x \in P_b$ and $y \notin P_a \cup P_b$ is similar. Case 2(2) in Figure 13 states these two cases.
The correctness holds trivially, since each $d_i$ with $1 \leq i \leq \ell$ is computed somewhere during the execution of $\text{SOLVE}(1, \ell)$ by Lemma 5.5. Since $i$ is chosen to be a median of $I(a, b)$ in each subroutine call to $\text{SOLVE}(a, b)$, there are $O(\log n)$ levels of recursion in executing $\text{SOLVE}(1, \ell)$. Let $m(a, b)$ be the sum of the multiplicities of the nodes of $X(a, b)$ in $D$. By Lemma 5.5, the time for computing $\Pi$ and $\Pi'$ is $O(m(a, b) \log^2 m(a, b))$. In order to maintain the condition that $X(a, b)$ is given whenever $\text{SOLVE}(a, b)$ is called, we apply Lemma 5.5 to obtain $X(a, i)$ and $X(i, b)$ in $O((m(a, b) + m_i) \log r)$ time before calling $\text{SOLVE}(a, i)$ or $\text{SOLVE}(i, b)$, where $m_i$ is the number of edges in $P_i \setminus (P_a \cup P_b)$. Since $P_a$ and $P_b$ are disjoint, each boundary node of $D$ is contained by one or two subgraphs $G[P_a, P_b]$ of the same recursion level. Since there are $O(n/r)$ pieces of $D$ and each piece of $D$ has $O(\sqrt{r})$ boundary nodes, the sum of $m(a, b)$ over all subgraphs $G[P_a, P_b]$ at the same recursion level is $O(n/\sqrt{r})$. Since each edge of $G$ appears in at most one $P_i \setminus (P_a \cup P_b)$ for all subroutine calls to $\text{SOLVE}(a, b)$, the sum of all $m_i$ throughout the execution of $\text{SOLVE}(1, \ell)$ is $O(n)$. By Equation (2), the overall time for computing $\Pi$ and $\Pi'$ is

$$O \left( \log n \cdot \frac{n}{\sqrt{r}} \log^2 n \right) = O(n).$$

The overall time of finding all paths $P$, $P[u_i, x]$, and $P[y, v_i]$ is $O(n \log log r)$, since their edges are disjoint and all of them are obtainable in $O(\log log r)$ time per node. Therefore, the running time of $\text{SOLVE}(1, \ell)$ is dominated by the sum of the $O(h(a, b) \log r)$ time for solving the $(a, b)$-subproblems by Lemmas 5.3, 5.4, and 5.5 at the bottom of recursion. Since the sum of $h(a, b)$ over all these $(a, b)$-subproblems is $O(n)$, the running time of $\text{SOLVE}(1, \ell)$ is $O(n \log r)$. The lemma is proved.

6 Concluding remarks

We give the first known $O(n \log n \log \log n)$-time algorithms for finding a minimum cut and a shortest cycle in an $n$-node simple directed planar graph $G$ with nonnegative edge weights. For the case that $G$ is restricted to be unweighted, our shortest-cycle algorithm remains the best known result for the shortest-cycle problem. The best algorithm for the minimum-cut problem, running in $O(n \log n)$ time, is obtained by plugging in the $O(n)$-time minimum st-cut algorithm of, e.g., Brandes and Wagner [4] and Eisenstat and Klein [13] to a directed version of the reduction algorithm of Chalamoom et al. [8]. Thus, an interesting future direction is to further reduce the running time of our algorithms on both problems for this special case. Extending our results to bounded-genus graphs is also of interest.

Acknowledgment

We thank the anonymous reviewers for helpful comments.

References

[1] G. Borradaile and P. N. Klein. An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. Journal of the ACM, 56(2):9.1–9.30, 2009.

[2] G. Borradaile, P. N. Klein, S. Mozes, Y. Nussbaum, and C. Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science, pages 170–179, 2011.
[3] G. Borradaile, P. Sankowski, and C. Wulff-Nilsen. Min $st$-cut oracle for planar graphs with near-linear preprocessing time. *ACM Transactions on Algorithms*, 11(3):16.1–16.29, 2015.

[4] U. Brandes and D. Wagner. A linear time algorithm for the arc disjoint Menger problem in planar directed graphs. *Algorithmica*, 28(1):16–36, 2000.

[5] S. Cabello. Finding shortest contractible and shortest separating cycles in embedded graphs. *ACM Transactions on Algorithms*, 6(2):24.1–24.18, 2010.

[6] S. Cabello, E. W. Chambers, and J. Erickson. Multiple-source shortest paths in embedded graphs. *SIAM Journal Computing*, 42(4):1542–1571, 2013.

[7] S. Cabello, É. Colin de Verdière, and F. Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. In *Proceedings of the 26th ACM Symposium on Computational Geometry*, pages 156–165, 2010.

[8] P. Chalermsook, J. Fakcharoenphol, and D. Nanongkai. A deterministic near-linear time algorithm for finding minimum cuts in planar graphs. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 828–829, 2004.

[9] H.-C. Chang and H.-I. Lu. Computing the girth of a planar graph in linear time. *SIAM Journal on Computing*, 42(3):1077–1094, 2013.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[11] M. Cygan, H. N. Gabow, and P. Sankowski. Algorithmic applications of Baur-Strassen’s theorem: Shortest cycles, diameter and matchings. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, 2012.

[12] H. Djidjev. A faster algorithm for computing the girth of planar and bounded genus graphs. *ACM Transactions on Algorithms*, 7(1):3.1–3.16, 2010.

[13] D. Eisenstat and P. N. Klein. Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In *Proceedings of the 45th ACM Symposium on Theory of Computing*, pages 735–744, 2013.

[14] J. Erickson. Maximum flows and parametric shortest paths in planar graphs. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 794–804, 2010.

[15] J. Erickson, K. Fox, and A. Nayyeri. Global minimum cuts in surface embedded graphs. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1309–1318, 2012.

[16] J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004.

[17] J. Erickson and A. Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1166–1176, 2011.

[18] J. Erickson and A. Nayyeri. Shortest non-crossing walks in the plane. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 297–208, 2011.
[19] J. Erickson and P. Worah. Computing the shortest essential cycle. *Discrete & Computational Geometry*, 44(4):912–930, 2010.

[20] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006.

[21] K. Fox. Shortest non-trivial cycles in directed and undirected surface graphs. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 352–364, 2013.

[22] K. Fox. *Fast algorithms for surface embedded graphs via homology*. PhD thesis, University of Illinois at Urbana-Champaign, 2014.

[23] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987.

[24] H. N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences*, 50(2):259–273, 1995.

[25] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989.

[26] P. Gawrychowski, S. Mozes, and O. Weimann. Submatrix maximum queries in Monge matrices are equivalent to predecessor search. In B. Speckmann, editor, *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming*, pages 580–592, 2015.

[27] A. V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing*, 24(3):494–504, 1995.

[28] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the SIAM*, 9(4):551–570, 1961.

[29] M. T. Goodrich. Planar separators and parallel polygon triangulation. *Journal of Computer and System Sciences*, 51(3):374–389, 1995.

[30] J. Hao and J. B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *Journal of Algorithms*, 17(3):424–446, 1994.

[31] M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997.

[32] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978.

[33] G. F. Italiano, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 313–322, 2011.

[34] L. Janiga and V. Koubek. Minimum cut in directed planar networks. *Kybernetika*, 28(1):37–49, 1992.

[35] H. Kaplan, S. Mozes, Y. Nussbaum, and M. Sharir. Submatrix maximum queries in Monge matrices and Monge partial matrices, and their applications. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 338–355, 2012.
[36] H. Kaplan and Y. Nussbaum. Minimum s-t cut in undirected planar graphs when the source and the sink are close. In T. Schwentick and C. Dürr, editors, Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, pages 117–128, 2011.

[37] D. R. Karger. Minimum cuts in near-linear time. Journal of the ACM, 47(1):46–76, 2000.

[38] K.-i. Kawarabayashi and M. Thorup. Deterministic global minimum cut of a simple graph in near-linear time. In Proceedings of the 47th ACM Symposium on Theory of Computing, pages 665–674, 2015.

[39] S. Khuller and J. Naor. Flow in planar graphs: A survey of recent results. In Planar Graphs, DIMACS Series in Discrete Math and Theoretical Computer Science 9, pages 59–84. AMS, 1993.

[40] P. N. Klein. Multiple-source shortest paths in planar graphs. In Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 146–155, 2005.

[41] P. N. Klein, S. Mozes, and C. Sommer. Structured recursive separator decompositions for planar graphs in linear time. In Proceedings of the 45th ACM Symposium on Theory of Computing, pages 505–514, 2013.

[42] P. N. Klein, S. Mozes, and O. Weimann. Shortest paths in directed planar graphs with negative lengths: A linear-space $O(n \log^2 n)$-time algorithm. ACM Transactions on Algorithms, 6(2):30.1–30.18, 2010.

[43] J. Łącki, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Single source - all sinks max flows in planar digraphs. In Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, pages 599–608, 2012.

[44] J. Łącki and P. Sankowski. Min-cuts and shortest cycles in planar graphs in $O(n \log \log n)$ time. In Proceedings of the 19th Annual European Symposium on Algorithms, pages 155–166, 2011.

[45] H.-C. Liang. Minimum cuts and shortest cycles in directed planar graphs via shortest non-crossing paths. Master’s thesis, National Taiwan University, July 2015.

[46] H.-C. Liang and H.-I. Lu. Minimum cuts and shortest cycles in directed planar graphs via noncrossing shortest paths. SIAM Journal on Discrete Mathematics, 31(1):454–476, 2017.

[47] A. Lingas and E.-M. Lundell. Efficient approximation algorithms for shortest cycles in undirected graphs. Information Processing Letters, 109(10):493–498, 2009.

[48] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. SIAM Journal on Applied Mathematics, 36:177–189, 1979.

[49] B. Monien. The complexity of determining a shortest cycle of even length. Computing, 31(4):355–369, 1983.

[50] R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, 1995.

[51] S. Mozes, C. Nikolaev, Y. Nussbaum, and O. Weimann. Minimum cut of directed planar graphs in $O(n \log \log n)$ time. Computing Research Repository, December 2015. http://arxiv.org/abs/1512.02068
[52] S. Mozes and C. Wulff-Nilsen. Shortest paths in planar graphs with real lengths in $O(n \log^2 n / \log \log n)$ time. In M. de Berg and U. Meyer, editors, Proceedings of the 18th Annual European Symposium on Algorithms, Lecture Notes in Computer Science 6347, pages 206–217. Springer, 2010.

[53] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. Combinatorica, 7(1):105–113, 1987.

[54] H. Nagamochi and T. Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. SIAM Journal on Discrete Mathematics, 5(1):54–66, 1992.

[55] J. B. Orlin. Max flows in $O(nm)$ time, or better. In Proceedings of the 45th ACM Symposium on Theory of Computing, pages 765–774, 2013.

[56] E. Papadopoulou. $k$-pairs non-crossing shortest paths in a simple polygon. International Journal of Computational Geometry and Applications, 9(6):533–552, 1999.

[57] V. Polishchuk and J. S. B. Mitchell. Thick non-crossing paths and minimum-cost flows in polygonal domains. In Proceedings of the 23rd ACM Symposium on Computational Geometry, pages 56–65, 2007.

[58] J. H. Reif. Minimum $s$-$t$ cut of a planar undirected network in $O(n \log^2 n)$ time. SIAM Journal on Computing, 12(1):71–81, 1983.

[59] L. Roditty and R. Tov. Approximating the girth. ACM Transactions on Algorithms, 9(2):15.1–15.13, 2013.

[60] L. Roditty and V. Vassilevska Williams. Subquadratic time approximation algorithms for the girth. In Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, pages 833–845, 2012.

[61] M. Stoer and F. Wagner. A simple min-cut algorithm. Journal of the ACM, 44(4):585–591, 1997.

[62] J. Takahashi, H. Suzuki, and T. Nishizeki. Finding shortest non-crossing rectilinear paths in plane regions. In Proceedings of the 4th International Symposium on Algorithms and Computation, pages 98–107, 1993.

[63] J.-y. Takahashi, H. Suzuki, and T. Nishizeki. Shortest noncrossing paths in plane graphs. Algorithmica, 16(3):339–357, 1996.

[64] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In Proceedings of the 44th ACM Symposium on Theory of Computing, pages 887–898, 2012.

[65] V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, pages 645–654, 2010.

[66] K. Weihe. Edge-disjoint $(s, t)$-paths in undirected planar graphs in linear time. Journal of Algorithms, 23(1):121–138, 1997.

[67] O. Weimann and R. Yuster. Computing the girth of a planar graph in $O(n \log n)$ time. SIAM Journal on Discrete Mathematics, 24(2):609–616, 2010.
[68] C. Wulff-Nilsen. *Algorithms for planar graphs and graphs in metric spaces*. PhD thesis, University of Copenhagen, 2010.

[69] R. Yuster. A shortest cycle for each vertex of a graph. *Information Processing Letters*, 111(21-22):1057–1061, 2011.

[70] R. Yuster and U. Zwick. Finding even cycles even faster. *SIAM Journal on Discrete Mathematics*, 10(2):209–222, 1997.