SIAM J. COMPUT.
Vol. 42, No. 3, pp. 1077–1094

COMPUTING THE GIRTH OF A PLANAR GRAPH IN LINEAR TIME*

HSIEN-CHIH CHANG† AND HSUEH-I LU‡

Abstract. The *girth* of a graph is the minimum weight of all simple cycles of the graph. We study the problem of determining the girth of an n-node unweighted undirected planar graph. The first nontrivial algorithm for the problem, given by Djidjev, runs in $O(n^{5/4} \log n)$ time. Chalermsook, Fakcharoenphol, and Nanongkai reduced the running time to $O(n \log^2 n)$. Weimann and Yuster further reduced the running time to $O(n \log n)$. In this paper, we solve the problem in $O(n)$ time.

Key words. paths and cycles, planar graphs, graph algorithms, data structures

AMS subject classifications. 05C38, 05C10, 05C85, 68P05

DOI. 10.1137/110832033

1. Introduction.

Let G be an edge-weighted simple graph, i.e., G does not contain multiple edges and self-loops. We say that G is *unweighted* if the weight of each edge of G is one. A cycle of G is *simple* if each node and each edge of G is traversed at most once in the cycle. The *girth* of G, denoted *girth*(G), is the minimum weight of all simple cycles of G. For instance, the girth of each graph in Figure 1.1 is four. As shown by, e.g., Bollobás [4], Cook [12], Chandran and Subramanian [10], Diestel [14], Erdős [21], and Lovász [39], girth is a fundamental combinatorial characteristic of graphs related to many other graph properties, including degree, diameter, connectivity, treewidth, and maximum genus. We address the problem of computing the girth of an n-node graph. Itai and Rodeh [28] gave the best known algorithm for the problem, running in time $O(M(n) \log n)$, where $M(n)$ is the time for multiplying two $n \times n$ matrices [13]. In the present paper, we focus on the case that the input graph is undirected, unweighted, and planar. Djidjev [16, 17] gave the first nontrivial algorithm for the case, running in $O(n^{5/4} \log n)$ time. The min-cut algorithm of Chalermsook, Fakcharoenphol, and Nanongkai [9] reduced the time complexity to $O(n \log^2 n)$, using the maximum-flow algorithms of, e.g., Borradaile and Klein [5] or Erickson [22]. Weimann and Yuster [49] further reduced the running time to $O(n \log n)$. Linear-time algorithms for an undirected unweighted planar graph were known only when the girth of the input graph is bounded by a constant, as shown by Itai and Rodeh [28], Vazirani and Yannakakis [47], and Eppstein [20]. We give the first optimal algorithm for any undirected unweighted planar graph.

THEOREM 1.1. *The girth of an n-node undirected unweighted planar graph is computable in $O(n)$ time.*

Related work. The $O(M(n) \log n)$-time algorithm of Itai and Rodeh [28] also works for directed graphs. The best known algorithm for directed planar graphs, due

*Received by the editors April 26, 2011; accepted for publication (in revised form) April 22, 2013; published electronically May 30, 2013. A preliminary version appeared in *Proceedings of the 17th Annual International Computing and Combinations Conference*, Lecture Notes in Comput. Sci. 6842, Springer-Verlag, Berlin, 2011, pp. 225–360.

http://www.siam.org/journals/sicomp/42-3/83203.html

†Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan (hchang17@illinois.edu).

‡Corresponding author. Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan (hil@csie.ntu.edu.tw). This author's research was supported in part by NSC grants 98–2221–E-002–079–MY3 and 101–2221–E–002–062–MY3.

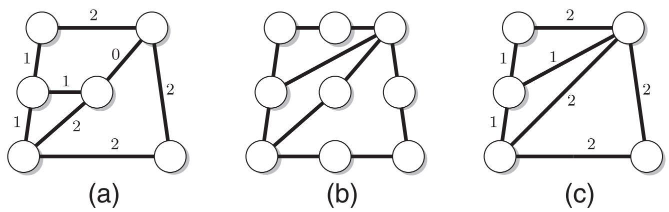

FIG. 1.1. (a) *A planar graph G with nonnegative integral edge weights.* (b) *The expanded version* EXPAND(G) *of G.* (c) *A contracted graph G' with* EXPAND(G') = EXPAND(G).

to Weimann and Yuster [49], runs in $O(n^{3/2})$ time. The $O(n \log^2 n)$-time algorithm of Chalermsook, Fakcharoenphol, and Nanongkai [9], using the maximum-flow algorithms of Borradaile and Klein [5] or Erickson [22], also works for undirected planar graphs with nonnegative weights. The recent max-flow algorithm of Italiano et al. [29] improved the running time of the algorithm in [9] to $O(n \log n \log\log n)$. For any given constant k, Alon, Yuster, and Zwick [1] showed that a k-edge cycle of any n-node general graph, if it exists, can be found in $O(M(n) \log n)$ time or expected $O(M(n))$ time. The time complexity was reduced to $O(n^2)$ by Yuster and Zwick [50] (respectively, $O(n)$ by Dorn [18]) if k is even (respectively, the input graph is planar). See, e.g., [24, 42, 26, 43, 31, 30, 36, 45, 2, 40, 15, 27, 37, 41, 19, 34, 35, 32, 7, 8, 23, 48, 29] for work related to girths and min-weight cycles in the literature.

Overview. The *degree* of a graph is the maximum degree of the nodes in the graph. For instance, the number of neighbors of each node in an $O(1)$-degree graph is bounded by an absolute constant. To compute *girth*(G_0) for the input n-node planar graph G_0, we turn G_0 into an m-node "contracted" (see section 2.1) graph G' with positive integral edge weights such that $m \le n$ and *girth*(G') = *girth*(G_0), as done by Weimann and Yuster [49]. If the "density" (see section 2.1) of G' is $\Omega(\log^2 m)$, we can afford to use the algorithm of [9] (see Theorem 2.1) to compute *girth*(G'). Otherwise, by *girth*(G') = $O(\log^2 m)$, as proved by Weimann and Yuster (see Lemma 2.4), and the fact G' has positive integral weights, we can further transform G' to a $\Theta(m)$-node $O(\log^2 m)$-outerplane graph G with $O(1)$ degree, $O(\log^2 m)$ density, and $O(\log^2 m)$ maximum weight such that *girth*(G) = *girth*(G'). The way we reduce the "outerplane radius" (see section 2.2) is similar to those of Djidjev [17] and Weimann and Yuster [49]. In order not to increase the outerplane radius, our degree-reduction operation (see section 2.2) is different from that of Djidjev [17]. Although G may have zero-weight edges and may no longer be contracted, it does not affect the correctness of the following approach for computing *girth*(G).

A cycle of a graph is *nondegenerate* if some edge of the graph is traversed exactly once in the cycle. Let u and v be two distinct nodes of G. Let $g(u, v)$ be the minimum weight of any simple cycle of G that contains u and v. Let $d(u, v)$ be the distance of u and v in G. For any edge e of G, let $d(u, v; e)$ be the distance of u and v in $G \setminus \{e\}$. If $e(u, v)$ is an edge of some min-weight path between u and v in G, then $d(u, v) + d(u, v; e(u, v))$ is the minimum weight of any nondegenerate cycle containing u and v that traverses $e(u, v)$ exactly once. In general, $d(u, v) + d(u, v; e(u, v))$ could be less than $g(u, v)$. However, if u and v belong to a min-weight simple cycle of G, then $d(u, v) + d(u, v; e(u, v)) = g(u, v) =$ *girth*(G).

Computing the minimum $d(u, v) + d(u, v; e(u, v))$ over all pairs of nodes u and v in G is too expensive. However, computing $d(u, v) + d(u, v; e(u, v))$ for all pairs of nodes u and v in a small node set S of G leads to a divide-and-conquer procedure for computing *girth*(G). Specifically, since G is an $O(\log^2 m)$-outerplane graph, there is an $O(\log^2 m)$-node set S of G partitioning $V(G) \setminus S$ into two nonadjacent sets V_1

and V_2 with roughly equal sizes. Let C be a min-weight simple cycle of G. Let G_1 (respectively, G_2) be the subgraph of G induced by $V_1 \cup S$ (respectively, $V_2 \cup S$). If $V(C) \cap S$ has at most one node, the weight of C is the minimum of $girth(G_1)$ and $girth(G_2)$. Otherwise, the weight of C is the minimum $d(u,v) + d(u,v;e(u,v))$ over all $O(\log^4 m)$ pairs of nodes u and v in S. Edges $e(u,v)$ and distances $d(u,v)$ and $d(u,v;e(u,v))$ in G can be obtained via dynamic programming from edges $e(u,v)$ and distances $d(u,v)$ and $d(u,v;e(u,v))$ in G_1 and G_2 for any two nodes u and v in an $O(\log^3 m)$-node superset $Border(S)$ (see section 4) of S. The above recursive procedure (see Lemma 5.4) is executed for two levels. The first level (see the proofs of Lemmas 3.3 and 5.4) reduces the girth problem of G to girth and distance problems of graphs with $O(\log^{30} m)$ nodes. The second level (see the proofs of Lemmas 5.6 and 6.1) further reduces the problems to girth and distance problems of graphs with $O((\log\log m)^{30})$ nodes, each of whose solutions can thus be obtained directly from an $O(m)$-time precomputable data structure (see Lemma 5.5). Just like Djidjev [17] and Chalermsook, Fakcharoenphol, and Nanongkai [9], we rely on dynamic data structures for planar graphs. Specifically, we use the dynamic data structure of Klein [33] (see Lemma 5.2) that supports point-to-point distance queries. We also use Goodrich's decomposition tree [25] (see Lemma 4.2), which is based on the link-cut tree of Sleator and Tarjan [46]. The interplay among the densities, outerplane radii, and maximum weights of subgraphs of G is crucial to our analysis. Although it seems unlikely to complete these two levels of reductions in $O(m)$ time, we can fortunately bound the overall time complexity by $O(n)$.

The rest of the paper is organized as follows. Section 2 gives the preliminaries and reduces the girth problem on a general planar graph to the girth problem on a graph with $O(1)$ degree and polylogarithmic maximum weight, outerplane radius, and density. Section 3 gives the framework of our algorithm, which consists of three tasks. Section 4 shows Task 1. Section 5 shows Task 2. Section 6 shows Task 3. Section 7 concludes the paper.

2. Preliminaries.

All logarithms throughout the paper are to the base of two. Unless clearly specified otherwise, all graphs are undirected simple planar graphs with nonnegative integral edge weights. Let $|S|$ denote the cardinality of set S. Let $V(G)$ consist of the nodes of graph G. Let $E(G)$ consist of the edges of graph G. Let $|G| = |V(G)| + |E(G)|$. By planarity of G, we have $|G| = \Theta(|V(G)|)$. Let $wmax(G)$ denote the maximum edge weight of G. For instance, if G is as shown in Figures 1.1(a) and 1.1(b), then $wmax(G) = 2$ and $wmax(G) = 1$, respectively. Let $w(G)$ denote the sum of edge weights of graph G. Therefore, $girth(G)$ is the minimum $w(C)$ over all simple cycles C of G.

THEOREM 2.1 (see [9]). *If G is an m-node planar graph with nonnegative weights, then it takes $O(m \log^2 m)$ time to compute $girth(G)$.*

2.1. Expanded version, density, weight decreasing, contracted graph.

The *expanded version* of graph G, denoted EXPAND(G), is the unweighted graph obtained from G by the following operations: (1) For each edge (u,v) with positive weight k, we replace edge (u,v) by an unweighted path $(u, u_1, u_2, \ldots, u_{k-1}, v)$; and (2) for each edge (u,v) with zero weight, we delete edge (u,v) and merge u and v into a new node. For instance, the graph in Figure 1.1(b) is the expanded version of the graphs in Figures 1.1(a) and 1.1(c). One can verify that the expanded version of G has $w(G) - |E(G)| + |V(G)|$ nodes. Define the *density* of G to be

$$density(G) = \frac{|V(\text{EXPAND}(G))|}{|V(G)|}.$$

For instance, the densities of the graphs in Figures 1.1(a) and 1.1(c) are $\frac{3}{2}$ and $\frac{9}{5}$, respectively.

LEMMA 2.2. *The following statements hold for any graph G:*

(1) *$girth(\text{EXPAND}(G)) = girth(G)$.*

(2) *$density(G)$ can be computed from G in $O(|G|)$ time.*

For any number w, let $\text{DECR}(G, w)$ be the graph obtainable in $O(|G|)$ time from G by decreasing the weight of each edge e with $w(e) > w$ down to w. The following lemma is straightforward.

LEMMA 2.3. *If G is a graph and w is a positive integer, $density(\text{DECR}(G, w)) \leq density(G)$. Moreover, if $w \geq girth(G)$, $girth(\text{DECR}(G, w)) = girth(G)$.*

A graph is *contracted* if the two neighbors of any degree-two node of the graph are adjacent in the graph. For instance, the graphs in Figures 1.1(a) and 1.1(b) are not contracted and the graph in Figure 1.1(c) is contracted.

LEMMA 2.4 (Weimann and Yuster [49, Lemma 3.3]).

(1) *Let G_0 be an n-node unweighted biconnected planar graph. It takes $O(n)$ time to compute an m-node biconnected contracted planar graph G with positive integral weights such that $m \leq n$ and $G_0 = \text{EXPAND}(G)$.*

(2) *If G is a biconnected contracted planar graph with positive integral weights, then we have that $girth(G) \leq 36 \cdot density(G)$.*

2.2. Outerplane radius and degree reduction. A *plane graph* is a planar graph equipped with a planar embedding. A node of a plane graph is *external* if it is on the outer face of the embedding. The *outerplane depth* of a node v in a plane graph G, denoted $depth_G(v)$, is the positive integer such that v becomes external after peeling $depth_G(v) - 1$ levels of external nodes from G. The *outerplane radius* of G, denoted $orad(G)$, is the maximum outerplane depth of any node in G. A plane graph G is *r-outerplane* if $orad(G) \leq r$. For instance, in the graph shown in Figure 1.1(a), the outerplane depth of the only internal node is two, and the outerplane depths of the other five nodes are all one. The outerplane radius of the graph in Figure 1.1(a) is two and the outerplane radius of the graph in Figure 1.1(c) is one. All three graphs in Figure 1.1 are 2-outerplane. The graph in Figure 1.1(c) is also 1-outerplane.

Let v be a node of plane graph G with degree $d \geq 4$. Let u_1 be a neighbor of v in G. For each $i = 2, 3, \ldots, d$, let u_i be the ith neighbor of v in G starting from u_1 in clockwise order around v. Let $\text{REDUCE}(G, v, u_1)$ be the plane graph obtained from G by the following steps, as illustrated by Figure 2.1: (1) adding a zero-weight path $(v_1, v_2, \ldots, v_d)$, (2) replacing each edge (u_i, v) by edge (u_i, v_i) with $w(u_i, v_i) = w(u_i, v)$, and (3) deleting node v.

LEMMA 2.5. *Let v be a node of plane graph G with degree four or more. If u_1 is a neighbor of v with the smallest outerplane depth in G, then*

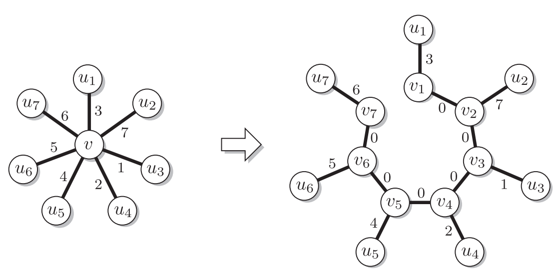

FIG. 2.1. *The operation that turns a plane graph G into $\text{REDUCE}(G, v, u_1)$.*

(1) $\textsc{reduce}(G, v, u_1)$ *can be obtained from* G *in time linear in the degree of* v *in* G,
(2) $\textsc{expand}(\textsc{reduce}(G, v, u_1)) = \textsc{expand}(G)$, *and*
(3) $\mathit{orad}(\textsc{reduce}(G, v, u_1)) = \mathit{orad}(G)$.

Proof. The first two statements are straightforward. To prove the third statement, let $j = \mathit{depth}_G(v)$ and $G' = \textsc{reduce}(G, v, u_1)$. Let G'' be the plane graph obtained from G' by peeling $j - 1$ levels of external nodes. By the choice of u_1, each v_i with $1 \le i \le d$ is an external node in G''. Therefore, for each $i = 1, 2, \ldots, d$, we have $\mathit{depth}_{G'}(v_i) = j$. Since the plane graphs obtained from G and $\textsc{reduce}(G, v, u_1)$ by peeling j levels of external nodes are identical, the lemma is proved. ∎

2.3. Proving the theorem by the main lemma. This subsection shows that, to prove Theorem 1.1, it suffices to ensure the following lemma.

LEMMA 2.6. *If* G *is an* $O(1)$*-degree plane graph satisfying the equation*

$$\mathit{wmax}(G) + \mathit{orad}(G) = O(\mathit{density}(G)) = O(\log^2 |G|), \tag{2.1}$$

then $\mathit{girth}(G)$ *can be computed from* G *in* $O(|G| + |\textsc{expand}(G)|)$ *time.*

Now we prove Theorem 1.1.

Proof of Theorem 1.1. Assume without loss of generality that the input n-node graph G_0 is biconnected. Let G be an m-node biconnected contracted planar graph with $\textsc{expand}(G) = G_0$ and $m \le n$ that can be computed from G_0 in $O(n)$ time, as ensured by Lemma 2.4(1). By Lemma 2.2(1), $\mathit{girth}(G) = \mathit{girth}(G_0)$. If $n > m \log^2 m$, by Theorem 2.1, it takes $O(m \log^2 m) = O(n)$ time to compute $\mathit{girth}(G)$. The theorem is proved. The rest of the proof assumes $m \le n \le m \log^2 m$.

We first equip the m-node graph G with a planar embedding, which is obtainable in $O(m)$ time (see, e.g., [6]). Initially, we have $|V(G)| = m$, $|V(\textsc{expand}(G))| = n$, and $\mathit{density}(G) = \frac{n}{m} = O(\log^2 m)$. We update G in three $O(m+n)$-time stages which maintain $|V(G)| = \Theta(m)$, $|V(\textsc{expand}(G))| = \Theta(n)$, $\mathit{girth}(G) = \mathit{girth}(G_0)$, and the planarity of G. At the end of the third stage, G may contain zero-weight edges and may no longer be biconnected and contracted. However, the resulting G is of degree at most three, has nonnegative weights, and satisfies (2.1). The theorem then follows from Lemma 2.6.

Stage 1. *Bounding the maximum weight of* G. We repeatedly replace G by $\textsc{decr}(G, \lceil 36 \cdot \mathit{density}(G) \rceil)$ until $\mathit{wmax}(G) \le \lceil 36 \cdot \mathit{density}(G) \rceil$. Although $\mathit{density}(G)$ may change in each iteration of the weight decreasing, by Lemmas 2.3 and 2.4(2) we know that $\mathit{girth}(G)$ remains the same and $\mathit{density}(G)$ does not increase. Since G remains biconnected and contracted and has positive weights, Lemma 2.4(2) ensures $\mathit{girth}(G) \le 36 \cdot \mathit{density}(G)$ throughout the stage. After the first iteration, $\mathit{wmax}(G) \le \lceil 36 \cdot \frac{n}{m} \rceil$. Each of the following iterations decreases $\mathit{wmax}(G)$ by at least one. Therefore, this stage has $O(\frac{n}{m})$ iterations, each of which takes $O(m)$ time, by Lemma 2.2(2). The overall running time is $O(n)$. The resulting m-node graph G satisfies $\mathit{wmax}(G) = O(\mathit{density}(G)) = O(\log^2 |G|)$.

Stage 2. *Bounding the outerplane radius of* G. For each positive integer j, let V_j consist of the nodes with outerplane depths j in G. For each integer $i \ge 0$, let G_i be the plane subgraph of G induced by the union of V_j with $36 \cdot i \cdot \mathit{density}(G) < j \le 36 \cdot (i+2) \cdot \mathit{density}(G)$. Let G' be the plane graph formed by the disjoint union of all the plane subgraphs G_i such that the external nodes of each G_i remain external in G'. We have $\mathit{orad}(G') = O(\mathit{density}(G))$. Each cycle of G' is a cycle of G, so $\mathit{girth}(G) \le \mathit{girth}(G')$. By Lemma 2.4(2), we have $\mathit{girth}(G) \le 36 \cdot \mathit{density}(G)$. Since the weight of each edge of G is at least one, the overlapping of the subgraphs G_i in G ensures that any cycle C of G with $w(C) = girth(G)$ lies in some subgraph G_i of G, implying $girth(G) \geq girth(G')$. Therefore, $girth(G') = girth(G)$. By $|V(G')| = \Theta(|V(G)|)$ and $|V(\text{EXPAND}(G'))| = \Theta(|V(\text{EXPAND}(G))|)$, we have $density(G') = \Theta(density(G))$. We replace G by G'. The resulting G satisfies $girth(G) = girth(G_0)$ and (2.1).

Stage 3. *Bounding the degree of* G. For each node v of G with degree four or more, we find a neighbor u of v in G whose outerplane depth in G is minimized and then replace G by $\text{REDUCE}(G, v, u)$. By Lemma 2.5(1), this stage takes $O(m)$ time. At the end, the degree of G is at most three. By Lemma 2.5(2), the expanded version of the resulting G is identical to that of G at the beginning of this stage. By Lemma 2.5(3), the outerplane radius remains the same. The number of nodes in G increases by at most a constant factor. The maximum weight remains the same. Therefore, the resulting G satisfies (2.1). By Lemma 2.2(1), we have $girth(G) = girth(G_0)$. ☐

The rest of the paper proves Lemma 2.6.

3. Framework: Dissection tree, nonleaf problem, and leaf problem. This section shows the framework of our proof for Lemma 2.6. Let $G[S]$ denote the subgraph of G induced by node set S. Let T be a rooted binary tree such that each member of $V(T)$ is a subset of $V(G)$. To avoid confusion, we use nodes to specify the members of $V(G)$ and vertices to specify the members of $V(T)$. Let $Root(T)$ denote the root vertex of T. Let $Leaf(T)$ consist of the leaf vertices of T. Let $Nonleaf(T)$ consist of the nonleaf vertices of T. For each vertex S of T, let $Below(S)$ denote the union of the vertices in the subtree of T rooted at S. Therefore, if S is a leaf vertex of T, then $Below(S) = S$. Also, $Below(Root(T))$ consists of the nodes of G that belong to some vertex of T. For each nonleaf vertex S of T, let $Lchild(S)$ and $Rchild(S)$ denote the two children of S in T. Therefore, if S is a nonleaf vertex of T, then $Below(S) = S \cup Below(Lchild(S)) \cup Below(Rchild(S))$. For instance, let T be the tree in Figure 3.1(b). We have $Root(T) = \{2, 7, 10\}$. Let $S = Rchild(Root(T))$. We have $S = \{7, 8\}$ and $Below(S) = \{2, 3, 4, 7, 8, 10, 11, 12\}$. Let $L = Lchild(S)$. We have $L = Below(L) = \{2, 3, 4, 7, 8\}$.

Node sets V_1 and V_2 are *dissected* by node set S in G if any node in $V_1 \setminus S$ and any node in $V_2 \setminus S$ are not adjacent in G. We say that T is a *dissection tree* of G if the following properties hold:

- Property 1. $Below(Root(T)) = V(G)$.
- Property 2. The following statements hold for each nonleaf vertex S of T:
  - (a) $S \subseteq Below(Lchild(S)) \cap Below(Rchild(S))$.
  - (b) $Below(Lchild(S))$ and $Below(Rchild(S))$ are dissected by S in G.

For instance, Figure 3.1(b) is a dissection tree of the graph in Figure 3.1(a).

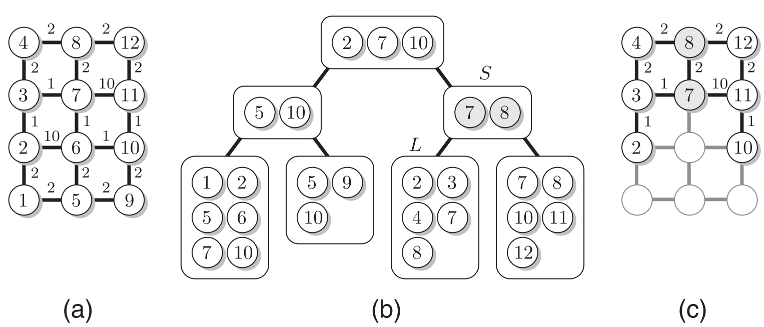

FIG. 3.1. (a) *A weighted plane graph* G. (b) *A dissection tree* T *of* G *with* $S = \{7, 8\}$ *and* $Border(S) = \{2, 7, 8, 10\}$. (c) *Graph* $G[Below(S)]$.

For any subset S of $V(G)$, any two distinct nodes u and v of S, and any edge e of G, let $d_S(u, v; e)$ denote the distance of u and v in $G[Below(S)] \setminus \{e\}$ and let $d_S(u, v)$ denote the distance of u and v in $G[Below(S)]$. Observe that if $e_S(u, v)$ is an edge in some min-weight path between u and v in $G[Below(S)]$, then $d_S(u, v) + d_S(u, v; e_S(u, v))$ is the minimum weight of any nondegenerate cycle in $G[Below(S)]$ containing u and v that traverses $e_S(u, v)$ exactly once. For instance, let G and T be shown in Figures 3.1(a) and 3.1(b). If $S = \{7, 8\}$, then $G[Below(S)]$ is as shown in Figure 3.1(c). We have $d_S(7, 10) = 7$ (e.g., path $(7, 8, 12, 11, 10)$ has weight 7) and $d_S(7, 10; (7, 8)) = 10$ (e.g., path $(7, 3, 4, 8, 12, 11, 10)$ has weight 10). Since $(7, 8)$ is an edge in a min-weight path $(7, 8, 12, 11, 10)$ between nodes 7 and 10, the minimum weight of any nondegenerate cycle in $G[Below(S)]$ containing nodes 7 and 10 that traverses $(7, 8)$ exactly once is 17 (e.g., nondegenerate cycle $(7, 8, 12, 11, 10, 11, 12, 8, 4, 3, 7)$ has weight 17 and traverses $(7, 8)$ exactly once).

DEFINITION 3.1. *For any dissection tree T of graph G, the* nonleaf problem *of (G, T) is to compute the following information for each nonleaf vertex S of T and each pair of distinct nodes u and v of S:* (1) *an edge $e_S(u, v)$ in a min-weight path between u and v in $G[Below(S)]$ and* (2) *distances $d_S(u, v)$ and $d_S(u, v; e_S(u, v))$.*

DEFINITION 3.2. *For any dissection tree T of graph G, the* leaf problem *of (G, T) is to compute the minimum girth$(G[L])$ over all leaf vertices L of T.*

Define the *sum of squares* of a dissection tree T as

$$squares(T) = \sum_{S \in Nonleaf(T)} |S|^2.$$

Our proof for Lemma 2.6 consists of the following three tasks:

- Task 1. Compute a dissection tree T of G with $squares(T) = O(|G|)$.
- Task 2. Solve the nonleaf problem of (G, T).
- Task 3. Solve the leaf problem of (G, T).

The following lemma ensures that to prove Lemma 2.6, it suffices to complete all three tasks in $O(|G| + |\text{EXPAND}(G)|)$ time for any $O(1)$-degree plane graph G satisfying (2.1).

LEMMA 3.3. *Given a dissection tree T of graph G and solutions to the leaf and nonleaf problems of (G, T), it takes $O(squares(T))$ time to compute girth(G).*

Proof. Let g_{leaf} be the given solution to the leaf problem of (G, T). It takes $O(squares(T))$ time to obtain the minimum value $g_{nonleaf}$ of $d_S(u, v) + d_S(u, v; e_S(u, v))$ over all pairs of distinct nodes u and v of S, where $e_S(u, v)$ is the edge in the given solution to the nonleaf problem of (G, T). Let C be a simple cycle of G with $w(C) = girth(G)$. It suffices to show $w(C) = \min\{g_{leaf}, g_{nonleaf}\}$. By Property 1 of T, there is a lowest vertex S of T with $V(C) \subseteq Below(S)$. If S is a leaf vertex of T, then $w(C) = g_{leaf}$. If S is a nonleaf vertex of T, then $w(C) = girth(G[Below(S)])$. We know $|S \cap V(C)| \geq 2$: Assume $|S \cap V(C)| \leq 1$ for contradiction. By Property 2(b) and simplicity of C, we have $V(C) \subseteq S \cup Lchild(S)$ or $V(C) \subseteq S \cup Rchild(S)$. By Property 2(a), either $V(C) \subseteq Lchild(S)$ or $V(C) \subseteq Rchild(S)$ holds, contradicting the choice of S. Let u and v be two distinct nodes in $S \cap V(C)$. Since C is a min-weight nondegenerate cycle of $G[Below(S)]$, we have $w(C) = d_S(u, v) + d_S(u, v; e_S(u, v))$. Therefore, $w(C) = g_{nonleaf}$. The lemma is proved. ∎

4. Task 1: Computing a dissection tree. Let T be a dissection tree of graph G. For each vertex S of T, let $Above(S)$ be the union of the ancestors of S in T and let $Inherit(S) = Above(S) \cap Below(S)$. If S is a leaf vertex of T, then let $Border(S) = Inherit(S)$. If S is a nonleaf vertex of T, then let $Border(S) = S \cup Inherit(S)$. For instance, let T be as shown in Figure 3.1(b). Let $S = Rchild(Root(T))$. We have

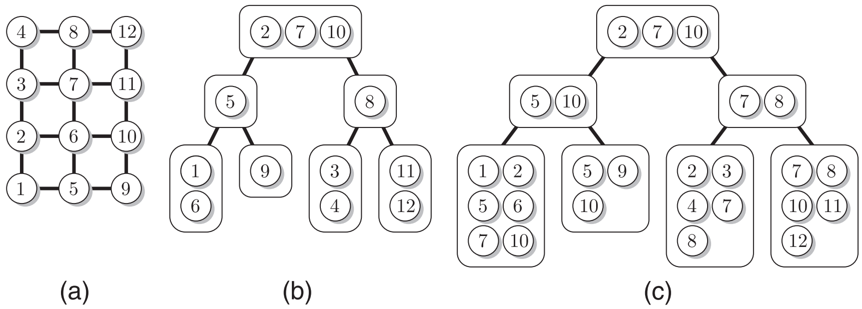

FIG. 4.1. (a) *A plane graph G.* (b) *A decomposition tree T' of G.* (c) *A dissection tree T of G.*

$\mathit{Above}(S) = \mathit{Inherit}(S) = \{2, 7, 10\}$ and $\mathit{Border}(S) = \{2, 7, 8, 10\}$. Let $L = \mathit{Lchild}(S)$. We have $\mathit{Above}(L) = \{2, 7, 8, 10\}$ and $\mathit{Inherit}(L) = \mathit{Border}(L) = \{2, 7, 8\}$. Define

$$\ell(m) = \lceil \log^{30} m \rceil.$$

For any positive integer r, a dissection tree T of an m-node graph G is an r-*dissection tree* of G if the following conditions hold:

- Condition 1. $|V(T)| = O(m/\ell(m))$ and $\sum_{L \in \mathit{Leaf}(T)} |\mathit{Border}(L)| = O(mr/\ell(m))$.
- Condition 2. $|L| = \Theta(\ell(m))$ and $|\mathit{Border}(L)| = O(r \log m)$ holds for each leaf vertex L of T.
- Condition 3. $|S| + |\mathit{Border}(S)| = O(r \log m)$ holds for each nonleaf vertex S of T.

For any r-outerplane G, it takes $O(m)$ time to compute an $O(r)$-node set S of G such that the node subsets V_1 and V_2 of G dissected by S satisfy $|V_1|/|V_2| = \Theta(1)$ (see, e.g., [44, 3]). By recursively applying this linear-time procedure, an r-dissection tree can be obtained in $O(m \log m)$ time, which is too expensive for our algorithm. Instead, based upon Goodrich's $O(m)$-time separator decomposition [25], we prove the following lemma.

LEMMA 4.1. *Let G be an m-node r-outerplane $O(1)$-degree graph with $r = O(\log^2 m)$. It takes $O(m)$ time to compute an r-dissection tree of G.*

Let T' be a rooted binary tree such that each vertex of T' is a subset of $V(G)$. We say that T' is a *decomposition tree* of G if Properties 1 and 2b hold for T'. For instance, Figure 4.1(b) shows a decomposition tree of the graph in Figure 4.1(a). For any m-node triangulated plane graph Δ and for any positive integer $\ell \leq m$, Goodrich [25] showed that it takes $O(m)$ time to compute an $O(m/\ell)$-vertex $O(\log m)$-height decomposition tree T' of Δ such that $|L| = \Theta(\ell)$ holds for each leaf vertex L of T' and $|S| = O(|\mathit{Below}(S)|^{0.5})$ holds for nonleaf vertex S of T'. As a matter of fact, Goodrich's techniques directly imply that if an $O(r)$-diameter spanning tree of Δ is given, then a decomposition tree T' of Δ satisfying the following four conditions can also be obtained efficiently:

- Condition 1′. $|V(T')| = O(m/\ell(m))$.
- Condition 2′. $|L| = \Theta(\ell(m))$ and $|\mathit{Border}(L)| = 0$ hold for each leaf vertex L of T'.
- Condition 3′. $|S| = |\mathit{Border}(S)| = O(r)$ holds for each nonleaf vertex S of T'.
- Condition 4′. The height of T' is $O(\log m)$.

LEMMA 4.2. *Given an $O(r)$-diameter spanning tree of an m-node simple triangulated plane graph Δ with $r = O(\log^2 m)$, it takes $O(m)$ time to compute a decomposition tree T' of Δ that satisfies Properties* 1 *and* 2(b) *and Conditions* 1′, 2′, 3′, *and* 4′.

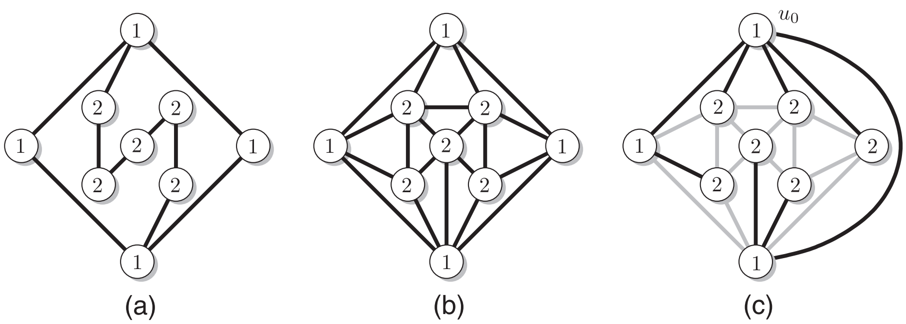

FIG. 4.2. (a) *A plane graph G. Each node is labeled by its outerplane depth.* (b) *A biconnected internally triangulated plane graph G' obtained from G.* (c) *A triangulated plane graph Δ obtained from G' with a spanning tree of Δ rooted at u_0.*

Proof. The lemma can be proved by following what Goodrich did in [25], so we give only a proof sketch here. Goodrich [25, section 2.4] showed that with some $O(m)$-time precomputable dynamic data structures for the given $O(r)$-diameter spanning tree and Δ, it takes $O(r \log^{O(1)} m)$ time to find a fundamental cycle C of Δ with respect to the given spanning tree such that the maximum number of nodes either inside or outside C is minimized. Since the diameter of the given spanning tree is $O(r)$, we have $|C| = O(r)$. Let V_1 (respectively, V_2) consist of the nodes of Δ inside (respectively, outside) C. We have $|V_1|/|V_2| = \Theta(1)$, as shown by Lipton and Tarjan [38]. With the precomputed data structures, it also takes $O(r \log^{O(1)} m)$ time to (1) split Δ into $\Delta[V_1]$ and $\Delta[V_2]$ and (2) split the given $O(r)$-diameter spanning tree of Δ into an $O(r)$-diameter spanning tree of $\Delta[V_1]$ and an $O(r)$-diameter spanning tree of $\Delta[V_2]$. Let T' be obtained by recursively computing $O(r)$-node sets $\mathit{Lchild}(S)$ and $\mathit{Rchild}(S)$ of $\Delta[V_1]$ and $\Delta[V_2]$ until $|S| \leq \ell(m)$. As long as $r = O(m^{1-\epsilon})$ holds for some constant $\epsilon > 0$, the overall running time is $O(m)$. One can verify that the resulting tree T' indeed satisfies Properties 1 and 2(b) and Conditions $1'$, $2'$, $3'$, and $4'$. ∎

We prove Lemma 4.1 using Lemma 4.2.

Proof of Lemma 4.1. It takes $O(m)$ time to triangulate the m-node r-outerplane graph G into an m-node simple triangulated plane graph Δ that admits a spanning tree with diameter $O(r)$. Specifically, we first triangulate each connected component of G into a simple biconnected internally triangulated plane graph G' such that the outerplane depth of each node remains the same after the triangulation. Let u_0 be an arbitrary external node of G'. We then add an edge (u_0, u) for each external node u of G' that is not adjacent to u_0. The resulting graph Δ is an m-node $O(r)$-outerplane simple triangulated plane graph. An $O(r)$-diameter spanning tree of Δ can be obtained in $O(m)$ time as follows. Let u_0 be the parent of all its neighbors in Δ. For each node u other than u_0 and the neighbors of u_0, we arbitrary choose a neighbor v of u in Δ with $\mathit{depth}_\Delta(v) = \mathit{depth}_\Delta(u) - 1$ and let v be the parent of u in the spanning tree. The diameter of the resulting spanning tree of Δ is $O(r)$. For instance, let G be as shown in Figure 4.2(a). An example of G' is shown in Figure 4.2(b). An example of Δ together with its spanning tree rooted at u_0 is shown in Figure 4.2(c).

Let T' be a decomposition tree of Δ as ensured by Lemma 4.2. Since Δ is obtained from G by adding edges, T' is also a decomposition tree of G that satisfies Properties 1 and 2(b) and Conditions $1'$, $2'$, $3'$, and $4'$. We prove the lemma by showing that T' can be modified in $O(m)$ time into an r-dissection tree T of G by calling DESCEND($Root(T')$), where the recursive procedure DESCEND(S) is defined as follows. If S is a leaf vertex of T', then we return. If S is a nonleaf vertex of T', we first (1) run the following steps for each node u of the current S, and then (2) recursively call DESCEND($Lchild(S)$) and DESCEND($Rchild(S)$).

Step 1. If u is not adjacent to any node in the current $Below(Lchild(S))$ in G, then we delete u from S and insert u into the current $Rchild(S)$.

Step 2. If u is adjacent to some node in the current $Below(Lchild(S))$ in G and is not adjacent to any node in the current $Below(Rchild(S))$ in G, then we delete u from S and insert u into the current $Lchild(S)$.

Step 3. If u is adjacent to some node in the current $Below(Lchild(S))$ and some node in the current $Below(Rchild(S))$ in G, then we leave u in S and insert u into the current $Lchild(S)$ and $Rchild(S)$.

For instance, if the decomposition tree T' is as shown in Figure 4.1(b), then the resulting tree T of running DESCEND($Root(T')$) is as shown in Figure 4.1(c).

We show that T is indeed an r-dissection tree of G. By definition of DESCEND, one can verify that a node u belongs to a nonleaf vertex S of T if and only if u belongs to both $Below(Lchild(S))$ and $Below(Rchild(S))$ in T. Property 2(a) holds for T and, thereby, Properties 1 and 2 of T follow from Properties 1 and 2(b) of T'. Moreover, if u belongs to a nonleaf vertex S of T, then the degrees of u in $G[Below(Lchild(S))]$ and $G[Below(Rchild(S))]$ are strictly less than the degree of u in $G[Below(S)]$. Since the degree of G is $O(1)$, each node u of G belongs to $O(1)$ vertices of T. By Conditions 1′ and 3′ of T', we have $\sum_{L\in Leaf(T)} |Border(L)| = \sum_{S\in Nonleaf(T')} O(|S|) = O(mr/\ell(m))$ and $|V(T)| = |V(T')| = O(m/\ell(m))$. Condition 1 of T holds. By Conditions 3′ and 4′ of T', the procedure increases $|S|$ and $|Border(S)|$ for each vertex S of T' by $O(r \log m)$. Therefore, Conditions 2 and 3 of T follow from Conditions 2′ and 3′ of T'.

We show that T can be obtained from T' in $O(m)$ time. We first spend $O(m)$ time to compute for each node v of G a list of $O(1)$ vertices of the original T' that contain v. Consider the case that S is a nonleaf vertex of the current T'. Let S' be a child vertex of S in the current T'. To determine whether a node u of S is adjacent to some node in the current $Below(S')$, for all $O(1)$ neighbors v of u in G, we traverse upward in T' from the $O(1)$ vertices of T' that currently contain v. The traversal passes S' if and only if u is adjacent to some node in the current $Below(S')$. By Condition 4′ of T', it takes $O(\log m)$ time to determine whether u is adjacent to the current $Below(S')$. Each update to the list of vertices of T' that contains u takes $O(1)$ time. By Conditions 1′, 3′, and 4′ of T', the overall running time of DESCEND($Root(T')$) is $O(mr \log^2 m/\ell(m)) = O(m)$. The lemma is proved. ∎

5. Task 2: Solving the nonleaf problems. This section proves the following lemma.

LEMMA 5.1. *Let G be an m-node $O(1)$-degree r-outerplane graph with $wmax(G)+r = O(\log^2 m)$. Given an r-dissection tree T of G, the nonleaf problem of (G,T) can be solved in $O(mr)$ time.*

DEFINITION 5.2. *Let T be a dissection tree of G. Let S be a vertex of T. The border problem of (G,T) for S is to compute the following information for any two distinct nodes u and v of $Border(S)$:* (1) $d_S(u,v)$, (2) *an edge $e_S(u,v)$ on some min-weight path between u and v in $G[Below(S)]$ that is incident to u, and* (3) $d_S(u,v;e)$ *for each edge e of G incident to u.*

Since $S \subseteq \mathit{Border}(S)$ holds for each nonleaf vertex S of T, any collection of solutions to the border problems of (G, T) for all nonleaf vertices of T yields a solution to the nonleaf problem of (G, T). We prove Lemma 5.1 by solving the border problems of (G, T) for all vertices of T in $O(mr)$ time. A leaf vertex L in an r-dissection tree T of an m-node graph G is *special* if

$$|\mathit{Border}(L)| + r \leq \lceil \log^2 \ell(m) \rceil.$$

Section 5.1 shows that the border problems of (G, T) for all vertices of T can be reduced in $O(mr)$ time to the border problems of (G, T) for all special leaf vertices of T, as summarized by Lemma 5.4. Section 5.2 shows that the border problems of (G, T) for all special leaf vertices of T can be solved in $O(mr)$ time, as summarized by Lemma 5.6. Lemma 5.1 follows immediately from Lemmas 5.4 and 5.6.

5.1. A reduction to the border problems for the special leaf vertices. Our reduction uses the following dynamic data structure that supports distance queries.

LEMMA 5.2 (Klein [33]). *Let G be an ℓ-node planar graph. It takes $O(\ell \log^2 \ell)$ time to compute a data structure $\mathit{Oracle}(G)$ such that each update to the weight of an edge and each query to the distance between any two nodes in G can be supported by $\mathit{Oracle}(G)$ in time $O(\ell^{2/3} \log^{5/3} \ell) = O(\ell^{7/10})$.*

The following lemma is needed to ensure the correctness of our reduction via dynamic programming.

LEMMA 5.3. *For each nonleaf vertex S of T, we have $S \subseteq \mathit{Border}(\mathit{Lchild}(S)) \cap \mathit{Border}(\mathit{Rchild}(S))$ and $\mathit{Border}(S) \subseteq \mathit{Border}(\mathit{Lchild}(S)) \cup \mathit{Border}(\mathit{Rchild}(S))$.*

Proof. Let $S' = \mathit{Lchild}(S)$ and $S'' = \mathit{Rchild}(S)$. By Property 2(a) of T, $S \subseteq \mathit{Below}(S') \cap \mathit{Below}(S'')$. By $S \subseteq \mathit{Above}(S') \cap \mathit{Above}(S'')$, we have $S \subseteq \mathit{Inherit}(S') \cap \mathit{Inherit}(S'')$. By $\mathit{Inherit}(S') \subseteq \mathit{Border}(S')$ and $\mathit{Inherit}(S'') \subseteq \mathit{Border}(S'')$, we have $S \subseteq \mathit{Border}(S') \cap \mathit{Border}(S'')$. We also have

$$\begin{aligned}
\mathit{Inherit}(S) \setminus S &= ((\mathit{Below}(S') \cup \mathit{Below}(S'') \cup S) \cap \mathit{Above}(S)) \setminus S \\
&\subseteq (\mathit{Below}(S') \cup \mathit{Below}(S'')) \cap \mathit{Above}(S) \\
&= (\mathit{Below}(S') \cap \mathit{Above}(S)) \cup (\mathit{Below}(S'') \cap \mathit{Above}(S)) \\
&\subseteq (\mathit{Below}(S') \cap \mathit{Above}(S')) \cup (\mathit{Below}(S'') \cap \mathit{Above}(S'')) \\
&= \mathit{Inherit}(S') \cup \mathit{Inherit}(S'') \\
&\subseteq \mathit{Border}(S') \cup \mathit{Border}(S'').
\end{aligned}$$

Thus, $\mathit{Border}(S) = S \cup (\mathit{Inherit}(S) \setminus S) \subseteq \mathit{Border}(S') \cup \mathit{Border}(S'')$. The lemma is proved. ◻

The following lemma shows the reduction.

LEMMA 5.4. *Let G be an m-node $O(1)$-degree graph. Given (1) an r-dissection tree T of G with $r = O(\log^2 m)$ and (2) solutions to the border problems of (G, T) for all special leaf vertices of T, it takes $O(mr)$ time to solve the border problems of (G, T) for all vertices of T.*

Proof. Solutions for special leaf vertices are given. We first show that it takes $O(mr)$ time to compute solutions for all nonspecial leaf vertices L of T. Let $\ell = \ell(m)$. By Condition 1 of T, we have $\sum_{L \in \mathit{Leaf}(T)} (|\mathit{Border}(L)| + r) = O(mr/\ell)$, implying that T has $O(\frac{mr}{\ell \log^2 \ell})$ nonspecial leaf vertices. For each nonspecial leaf vertex L of T, we run the following $O(\ell \log^2 \ell)$-time steps.

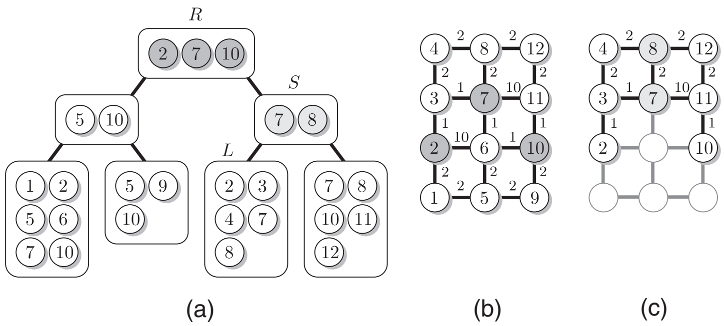

FIG. 5.1. (a) *A dissection tree* T *of the graph in* (b) *with* $R = Border(R) = \{2, 7, 10\}$, $S = \{7, 8\}$, *and* $Border(S) = \{2, 7, 8, 10\}$. (b) *Graph* $G = G[Below(R)]$. (c) *Graph* $G[Below(S)]$.

Step 1. By Condition 2 of T, we have $|L| = \Theta(\ell)$. We compute a data structure $Oracle(G[L])$ in $O(\ell \log^2 \ell)$ time as ensured by Lemma 5.2.

Step 2. For any two nodes u and v in $Border(L)$, we first obtain $d_L(u, v)$ from $Oracle$ in $O(\ell^{7/10})$ time. We then find a neighbor x of u in $G[L]$ with $d_L(u, v) = w(u, x) + d_L(x, v)$ and let $e_L(u, v) = (u, x)$, which can be obtained from $Oracle$ in $O(\ell^{7/10})$ time, since the degree of G is $O(1)$. By Lemma 5.2 and Condition 2 of T, the overall time complexity for this step is $O(\ell^{7/10} \cdot |Border(L)|^2) = O(\ell^{7/10} \cdot r^2 \log^2 m) = O(\ell^{9/10})$.

Step 3. For each edge e that is incident to $Border(L)$, we compute $d_L(u, v; e)$ from $Oracle$ for all nodes u and v of $Border(L)$ as follows: (1) temporarily setting $w(e) = \infty$; (2) for each pair of distinct nodes u and v in $Border(L)$, obtaining $d_L(u, v; e)$ from the distance of u and v in the current $G[L]$; and (3) restoring the original weight of e. Since the degree of G is $O(1)$, there are $O(|Border(L)|)$ choices of e. By Lemma 5.2 and Condition 2 of T, the running time of this step is $O(\ell^{7/10} \cdot |Border(L)|^3) = O(\ell^{7/10} \cdot r^3 \log^3 m) = O(\ell)$.

We now show that the solutions for all nonleaf vertices S of T can be computed in $O(m)$ time. By definition of $\ell(m)$ and Condition 1 of T, we have $|Nonleaf(T)| = O(m/\log^{30} m)$. By $r = O(\log^2 m)$ and Condition 3 of T, we have $|S| + |Border(S)| = O(\log^3 m)$. It suffices to prove the following claim for each nonleaf vertex S of T: "Given solutions for $S' = Lchild(S)$ and $S'' = Rchild(S)$, a solution for S can be computed in $O(|Border(S)|^3 \cdot |S|^2)$ time." By Property 2(b) of T, $Below(S')$ and $Below(S'')$ are dissected by S in G. We use (S, k)-path to denote a path of $G[Below(S)]$ that switches to a different side of S at most k times: Precisely, an $(S, 0)$-*path* is a path that completely lies in $G[Below(S')]$ or completely lies in $G[Below(S'')]$. For any positive integer k, we say that $(u_1, u_2, \ldots, u_t)$ is an (S, k)-*path* if $(u_1, u_2, \ldots, u_{t'})$ is an $(S, k-1)$-path, where t' is the smallest integer such that $(u_{t'}, u_{t'+1}, \ldots, u_t)$ is an $(S, 0)$-path. For instance, let T and G be as shown in Figures 5.1(a) and 5.1(b). Let $S = \{7, 8\}$. Note that $(8, 7, 11, 10)$ is both an $(S, 0)$-path and an $(S, 1)$-path with $u_{t'} = 8$. However, $(2, 3, 7, 11, 10)$ is an $(S, 1)$-path with $u_{t'} = 7$ but not an $(S, 0)$-path. Based upon the facts $Border(S) \subseteq Border(S') \cup Border(S'')$ and $S \subseteq Border(S') \cap Border(S'')$ as ensured by Lemma 5.3, we prove the above claim in the following three stages, each of which is also illustrated by Figure 5.1:

Stage 1. For any nodes u and v in $Border(S)$, let $d_{S,i}(u, v)$ denote the minimum weight of any (S, i)-path of $G[Below(S)]$ between u and v. Any simple path of $G[Below(S)]$ is an $(S, |S|)$-path, so $d_S(u, v) = d_{S,|S|}(u, v)$. As illustrated by Figure 5.1(b), we have $d_{R,0}(10, 2) = 7$ and $d_{R,1}(10, 2) = 4$. As illustrated by Figure 5.1(c), we have

$d_{S,0}(10,2) = \infty$ and $d_{S,1}(10,2) = 9$. One can verify the following recurrence relation:

$$d_{S,i}(u,v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } u = v; \\ \min\{d_{S'}(u,v), d_{S''}(u,v)\} & \text{if } i = 0 \text{ and } u \neq v; \\ \min\limits_{y \in S \cup \{v\}} d_{S,i-1}(u,y) + d_{S,0}(y,v) & \text{if } i \geq 1. \end{cases}$$

This stage takes $O(|Border(S)|^2 \cdot |S|^2)$ time via dynamic programming.

Stage 2. For any distinct nodes u and v in $Border(S)$, let $e_{S,i}(u,v)$ denote an incident edge of u in a min-weight (S,i)-path of $G[Below(S)]$ between u and v. If no (S,i)-path of $G[Below(S)]$ between u and v exists, let $e_{S,i}(u,v) = \emptyset$. As illustrated by Figure 5.1(b), edge $(10,6)$ is the only choice for $e_{R,0}(10,2)$ and $e_{R,1}(10,2)$. As illustrated by Figure 5.1(c), we have $e_{S,0}(10,2) = \emptyset$, and edge $(10,11)$ is the only choice for $e_{S,1}(10,2)$. Let

$$e_{S,i}(u,v) = \begin{cases} e_{S'}(u,v) & \text{if } i = 0 \text{ and } d_{S'}(u,v) \leq d_{S''}(u,v); \\ e_{S''}(u,v) & \text{if } i = 0 \text{ and } d_{S'}(u,v) > d_{S''}(u,v); \\ e_{S,i-1}(u,y) & \text{if } i \geq 1, \end{cases}$$

where y can be any node in $S \cup \{v\} \setminus \{u\}$ with $d_{S,i}(u,v) = d_{S,i-1}(u,y) + d_{S,0}(y,v)$. Since both $e_{S'}(u,v)$ and $e_{S''}(u,v)$ are incident to u in $G[Below(S)]$, each $e_{S,i}(u,v)$ is incident to u in $G[Below(S)]$. Therefore, $e_{S,|S|}(u,v)$ is a valid choice of $e_S(u,v)$. This stage takes $O(|Border(S)|^2 \cdot |S|^2)$ time via dynamic programming.

Stage 3. For any nodes u and v in $Border(S)$ and any edge e of $G[Below(S)]$ that is incident to $Border(S)$, let $d_{S,i}(u,v;e)$ be the minimum weight of any (S,i)-path in $G[Below(S)] \setminus \{e\}$ between u and v. We have $d_S(u,v;e) = d_{S,|S|}(u,v;e)$. As illustrated by Figure 5.1(b), we have $d_{R,0}(10,2;(10,6)) = d_{R,1}(10,2;(10,6)) = 8$. As illustrated by Figure 5.1(c), $d_{S,0}(10,2;(10,11)) = d_{S,1}(10,2;(10,11)) = \infty$. One can verify the following recurrence relation:

$$d_{S,i}(u,v;e) = \begin{cases} 0 & \text{if } i = 0 \text{ and } u = v; \\ \min\{d_{S'}(u,v;e), d_{S''}(u,v;e)\} & \text{if } i = 0 \text{ and } u \neq v; \\ \min\limits_{y \in S \cup \{v\}} d_{S,i-1}(u,y;e) + d_{S,0}(y,v;e) & \text{if } i \geq 1. \end{cases}$$

Since the degree of G is $O(1)$, the number of choices of e is $O(|Border(S)|)$. This stage takes $O(|Border(S)|^3 \cdot |S|^2)$ time via dynamic programming.

The lemma is proved. $\square$

5.2. Solving the border problems for the special leaf vertices. We need the following linear-time precomputable data structure in the proof of Lemma 5.6 to solve the border problems of (G,T) for all special leaf vertices of T as well as in the proof of Lemma 6.1 to solve the leaf problem of (G,T).

LEMMA 5.5. *For any given positive integers $k = O(\log\log m)^{O(1)}$ and $w = O(\log m)^{O(1)}$, it takes $O(m)$ time to compute a data structure $Table(k,w)$ such that the following statements hold for any $O(1)$-degree graph H with at most k nodes whose edge weights are at most w:*

1. *It takes $O(|H|)$ time to obtain a reference pointer $ref(H)$ from $Table(k,w)$ such that each of the following queries for any two distinct nodes u and v of H can be answered from $ref(H)$ and $Table(k,w)$ in $O(1)$ time:* (1) *the distance of u and v in H,* (2) *an edge incident to u that belongs to at least one min-weight path between u and v in H, and* (3) *the distance of u and v in $H \setminus \{e\}$ for each edge e of H incident to u.*

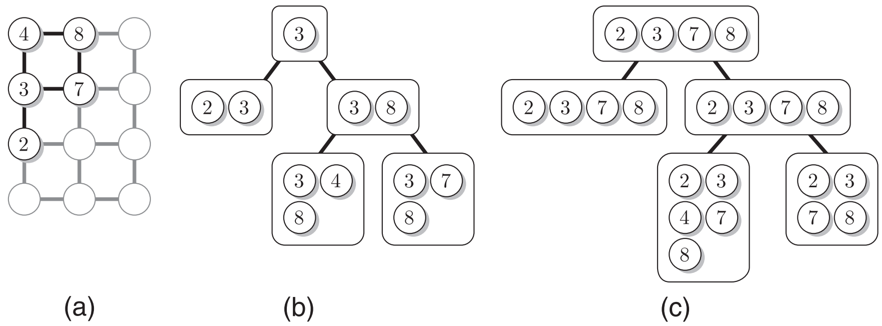

FIG. 5.2. (a) *Graph* $G_L = G[L]$ *with* $L = \{2, 3, 4, 7, 8\}$. (b) *A dissection tree* T'_L *of* G_L. (c) *A dissection tree* T_L *of* G_L *obtained from* T'_L.

2. *It takes* $O(|H|)$ *time to obtain* $girth(H)$ *from* $Table(k, w)$.

Proof. Let $\mathcal{H}$ consist of all graphs of at most k nodes whose maximum weight is at most w. It takes $O(w)^{O(k^2)}$ time to list all graphs H in $\mathcal{H}$. It takes $O(k^{O(1)})$ time to precompute the information in Statements 1 and 2 for each graph H in $\mathcal{H}$. The lemma follows from

$$\left(O(\log m)^{O(1)}\right)^{\left(O(\log\log m)^{O(1)}\right)} \cdot O\left((\log\log m)^{O(1)}\right)^{O(1)} = O(m). \qquad \square$$

LEMMA 5.6. *Let* G *be an* m*-node* $O(1)$*-degree* r*-outerplane graph with* $wmax(G) = O(\log^2 m)$. *Given an* r*-dissection tree* T *of* G*, the border problems of* (G, T) *for all special leaf vertices of* T *can be solved in* $O(mr)$ *time.*

Proof. We assume that T does have special leaf vertices, since otherwise the lemma holds trivially. By the assumption, we know $r \le \lceil \log^2 \ell(m) \rceil$. Let L be a special leaf vertex of T. Let $G_L = G[L]$. Let $m_L = |L|$. By Condition 2 of T, we know $m_L = \Theta(\ell(m))$. Let $r_L = r + |Border(L)|$. Clearly, G_L is an m_L-node $O(1)$-degree r_L-outerplane graph with $r_L = O(\log^2 m_L)$. By Lemma 4.1, it takes $O(m_L)$ time to obtain an r_L-dissection tree T'_L of G_L. Let T_L be obtained from T'_L by replacing each vertex S' of T'_L by $S' \cup Border(L)$. For instance, let T and G be as shown in Figures 5.1(a) and 5.1(b). If $L = \{2, 3, 4, 7, 8\}$ is a special leaf vertex of T, then G_L is as shown in Figure 5.2(a). We have $Border(L) = \{2, 7, 8\}$. If T'_L is as shown in Figure 5.2(b), then T_L is as shown in Figure 5.2(c). Clearly, $Border(L) \subseteq Root(T_L)$. We show that T_L is also an r_L-dissection tree of G_L. Since L is a leaf vertex of T, we have $Border(L) \subseteq L$. Therefore, Properties 1 and 2 of T_L follow from Properties 1 and 2 of T'_L. Let $\ell_L = \ell(m_L)$. By Condition 1 of T'_L and $|Border(L)| = O(r_L)$, we have $|V(T_L)| = |V(T'_L)| = O(m_L/\ell_L)$ and

$$\sum_{\hat{L} \in Leaf(T_L)} |Border(\hat{L})| \le |V(T'_L)| \cdot |Border(L)| + \sum_{L' \in Leaf(T'_L)} |Border(L')|$$
$$= O\left(\frac{m_L \cdot r_L}{\ell_L}\right).$$

Condition 1 holds for T_L. Adding $Border(L)$ to vertex S' of T'_L increases $|S'|$ and $|Border(S')|$ by no more than r_L, so Conditions 2 and 3 for T_L follow from Conditions 2 and 3 for T'_L. Therefore, T_L is an r_L-dissection tree of G_L with $\mathit{Border}(L) \subseteq \mathit{Root}(T_L)$. It follows that a solution to the border problem of (G_L, T_L) for $\mathit{Root}(T_L)$ yields a solution to the border problem of (G, T) for L.

Let k be the maximum $|\hat{L}|$ over all leaf vertices $\hat{L}$ of T_L and all special leaf vertices L of T. We have $k = \Theta(\ell_L) = O((\log\log m)^{30})$. By $\mathit{wmax}(G) = O(\log^2 m)$, it takes $O(m)$ time to compute a data structure $\mathit{Table}(k, \mathit{wmax}(G))$, as ensured by Lemma 5.5. By Lemma 5.5, it takes $O(|\hat{L}| + |\mathit{Border}(\hat{L})|^2) = O(\ell_L)$ time to obtain from the precomputed data structure $\mathit{Table}(k, \mathit{wmax}(G))$ a solution to the border problem of (G_L, T_L) for each special leaf vertex $\hat{L}$ of T_L. By Condition 1 of T_L, the border problems of (G_L, T_L) for all special leaf vertices of T_L can be solved in overall $O(m_L/\ell_L) \cdot O(\ell_L) = O(m_L)$ time. By Lemma 5.4, it takes $O(m_L \cdot r_L)$ time to obtain a collection of solutions to the border problems of (G_L, T_L) for all vertices of T_L, including $\mathit{Root}(T_L)$, which yields a solution to the border problem of (G, T) for the special leaf vertex L of T. By Condition 1 of T and $O(m_L \cdot r_L) = O(\ell(m) \cdot (r + |\mathit{Border}(L)|))$, the overall running time to solve the border problems of (G, T) for all special leaf vertices of T is $O(\ell(m)) \cdot \sum_{L \in \mathit{Leaf}(T)} O(r + |\mathit{Border}(L)|) = O(mr)$. The lemma is proved. $\square$

6. Task 3: Solving the leaf problem.

LEMMA 6.1. *Let G be an m-node $O(1)$-degree r-outerplane graph satisfying that $\mathit{wmax}(G) + r = O(\mathit{density}(G))$. Given an r-dissection tree T of G, the leaf problem of (G, T) can be solved in $O(m \cdot \mathit{density}(G))$ time.*

Proof. If $\mathit{density}(G) \geq \log^2 \ell(m)$, by Condition 1 of T and Theorem 2.1, the problem can be solved in $O(\ell(m) \log^2 \ell(m)) \cdot O(m/\ell(m)) = O(m \cdot \mathit{density}(G))$ time. The rest of the proof assumes $\mathit{wmax}(G) + r = O(\mathit{density}(G)) = O(\log^2 \ell(m))$. Let L be a leaf vertex of T. Let $m_L = |L|$. Let $G_L = G[L]$. By Condition 2 of T, we have $m_L = \Theta(\ell(m))$. Therefore, G_L is an m_L-node $O(1)$-degree r-outerplane graph with $\mathit{wmax}(G_L) + r = O(\log^2 m_L)$. By Lemma 4.1, an r-dissection tree T_L of G_L can be obtained from G_L in $O(m_L)$ time. Let k be the maximum $|\hat{L}|$ over all leaf vertices $\hat{L}$ of T_L and all leaf vertices L of T. We have $k = \Theta(\ell(m_L)) = O((\log\log m)^{30})$. Let $\mathit{Table}(k, \mathit{wmax}(G))$ be a data structure computable in $O(m)$ time as ensured by Lemma 5.5. By Lemma 5.5, $\mathit{girth}(G_L[\hat{L}])$ for any leaf vertex $\hat{L}$ of T_L can be obtained from $\mathit{Table}(k, \mathit{wmax}(G))$ in $O(|\hat{L}|)$ time. By Conditions 1 and 2 of T_L, the solution to the leaf problem of (G_L, T_L) can be obtained from $\mathit{Table}(k, \mathit{wmax}(G))$ in $O(m_L/\ell(m_L)) \cdot O(\ell(m_L)) = O(m_L)$ time. By Lemma 5.1, the nonleaf problem of (G_L, T_L) can be solved in $O(m_L \cdot r)$ time. By Conditions 1 and 3 of T_L, we have $\mathit{squares}(T_L) = O(m_L \cdot r^2 \log^2 m_L/\ell(m_L)) = O(m_L)$. By Lemma 3.3, it takes $O(m_L)$ time to compute $\mathit{girth}(G_L)$ from the solutions to the leaf and nonleaf problems of (G_L, T_L). Therefore, $\mathit{girth}(G[L])$ can be computed in $O(m_L \cdot r) = O(\ell(m) \cdot r)$ time. By Condition 1 of T, it takes $O(m/\ell(m)) \cdot O(\ell(m) \cdot r) = O(m \cdot \mathit{density}(G))$ time to solve the leaf problem of (G, T). The lemma is proved. $\square$

It remains to prove the main lemma of the paper, which implies Theorem 1.1, as already shown in section 2.3.

Proof of Lemma 2.6. Let $m = |V(G)|$ and $n = |V(\text{EXPAND}(G))|$. Let $r = \mathit{orad}(G)$. That is, G is an m-node $O(1)$-degree r-outerplane graph with $\mathit{wmax}(G) + r = O(\mathit{density}(G)) = O(\log^2 m)$. By Lemma 4.1, an r-dissection tree T of G can be obtained from G in $O(m)$ time. By Lemma 5.1, the nonleaf problem of (G, T) can be solved in $O(mr) = O(n)$ time. By Lemma 6.1, it takes $O(m \cdot \mathit{density}(G)) = O(n)$ time to solve the leaf problem of (G, T). By Conditions 1 and 3 of T, we have $\mathit{squares}(T) = O(mr^2 \log^2 m/\ell(m)) = O(m)$. The lemma follows from Lemma 3.3. $\square$

7. Concluding remarks. We give the first linear-time algorithm for computing the girth of any undirected unweighted planar graph. Our algorithm can be modified into one that finds a simple min-weight cycle. Specifically, when we solve each girth problem or each distance problem in our algorithm, we additionally let the algorithm output a node on a corresponding min-weight cycle or min-weight path. As a result, our algorithm not only computes the girth of the input graph but also outputs a node u on a min-weight cycle of the input graph. We can then use the breadth-first search algorithm of Itai and Rodeh [28] to output a min-weight cycle containing u in linear time.

The $O(n \log n)$-time algorithm of Weimann and Yuster [49] works on $O(1)$-genus graphs. It would be of interest to see if our algorithm can be extended to work for $O(1)$-genus graphs by, e.g., extending our black-box tools (the decomposition tree of Goodrich [25] and the distance oracle of Klein [33]) to work for $O(1)$-genus graphs.

Acknowledgments. The authors thank the anonymous reviewers for their helpful comments. We also thank Hsueh-Yi Chen, Chia-Ching Lin, and Cheng-Hsun Weng for discussion at an early stage of this research.

REFERENCES

[1] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. ACM, 42 (1995), pp. 844–856.
[2] N. Alon, R. Yuster, and U. Zwick, *Finding and counting given length cycles*, Algorithmica, 17 (1997), pp. 209–223.
[3] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoret. Comput. Sci., 209 (1998), pp. 1–45.
[4] B. Bollobás, *Chromatic number, girth and maximal degree*, Discrete Math., 24 (1978), pp. 311–314.
[5] G. Borradaile and P. N. Klein, *An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph*, J. ACM, 56 (2009), pp. 9.1–9.30.
[6] J. M. Boyer and W. J. Myrvold, *On the cutting edge: Simplified $O(n)$ planarity by edge addition*, J. Graph Algorithms Appl., 8 (2004), pp. 241–273.
[7] S. Cabello, *Finding shortest contractible and shortest separating cycles in embedded graphs*, ACM Trans. Algorithms, 6 (2010), pp. 24.1–24.18.
[8] S. Cabello, É. C. 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, 2010, pp. 156–165.
[9] 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, 2004, pp. 828–829.
[10] L. S. Chandran and C. R. Subramanian, *Girth and treewidth*, J. Combin. Theory Ser. B, 93 (2005), pp. 23–32.
[11] H.-C. Chang and H.-I. Lu, *Computing the girth of a planar graph in linear time*, in Proceedings of the 17th Annual International Computing and Combinatorics Conference, D.-Z. Du and B. Fu, eds., Lecture Notes in Comput. Sci. 6842, Springer-Verlag, Berlin, 2011, pp. 225–236.
[12] R. J. Cook, *Chromatic number and girth*, Period. Math. Hungar., 6 (1975), pp. 103–107.
[13] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Comput., 9 (1990), pp. 251–280.
[14] R. Diestel, *Graph Theory*, 2nd ed., Springer, Heidelberg, 2000.
[15] R. Diestel and C. Rempel, *Dense minors in graphs of large girth*, Combinatorica, 25 (2004), pp. 111–116.
[16] H. N. Djidjev, *Computing the girth of a planar graph*, in Proceedings of the 27th International Colloquium on Automata, Languages and Programming, 2000, pp. 821–831.
[17] H. N. Djidjev, *A faster algorithm for computing the girth of planar and bounded genus graphs*, ACM Trans. Algorithms, 7 (2010), pp. 3.1–3.16.

[18] F. Dorn, *Planar subgraph isomorphism revisited*, in Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science, J.-Y. Marion and T. Schwentick, eds., 2010, pp. 263–274.

[19] M. E. Dyer and A. M. Frieze, *Randomly colouring graphs with lower bounds on girth and maximum degree*, in Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, 2001, pp. 579–587.

[20] D. Eppstein, *Subgraph isomorphism in planar graphs and related problems*, J. Graph Algorithms Appl., 3 (1999), pp. 1–27.

[21] P. Erdős, *Graph theory and probability*, Canad. J. Math., 11 (1959), pp. 34–38.

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

[23] J. Erickson and P. Worah, *Computing the shortest essential cycle*, Discrete Comput. Geom., 44 (2010), pp. 912–930.

[24] R. E. Gomory and T. C. Hu, *Multi-terminal network flows*, J. SIAM, 9 (1961), pp. 551–570.

[25] M. T. Goodrich, *Planar separators and parallel polygon triangulation*, J. Comput. System Sci., 51 (1995), pp. 374–389.

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

[27] T. P. Hayes, *Randomly coloring graphs of girth at least five*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 269–278.

[28] A. Itai and M. Rodeh, *Finding a minimum circuit in a graph*, SIAM J. Comput., 7 (1978), pp. 413–423.

[29] G. F. Italiano, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen, *Improved minimum cuts and maximum flows in undirected planar graphs*, in Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, 2011, pp. 313–322.

[30] D. R. Karger, *Minimum cuts in near-linear time*, J. ACM, 47 (2000), pp. 46–76.

[31] D. R. Karger and C. Stein, *A new approach to the minimum cut problem*, J. ACM, 43 (1996), pp. 601–640.

[32] K.-I. Kawarabayashi and C. Thomassen, *Decomposing a planar graph of girth* 5 *into an independent set and a forest*, J. Combin. Theory Ser. B, 99 (2009), pp. 674–684.

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

[34] M. Kochol, *Girth restrictions for the* 5*-flow conjecture*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 705–707.

[35] A. V. Kostochka, D. Král', J.-S. Sereni, and M. Stiebitz, *Graphs with bounded tree-width and large odd-girth are almost bipartite*, J. Combin. Theory Ser. B, 100 (2010), pp. 554–559.

[36] A. Lingas and E.-M. Lundell, *Efficient approximation algorithms for shortest cycles in undirected graphs*, Inform. Process. Lett., 109 (2009), pp. 493–498.

[37] N. Linial, A. Magen, and A. Naor, *Girth and Euclidean distortion*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 705–711.

[38] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[39] L. Lovász, *On chromatic number of finite set systems*, Acta Math. Acad. Sci. Hungar., 19 (1968), pp. 59–67.

[40] W. Mader, *Topological subgraphs in graphs of large girth*, Combinatorica, 18 (1998), pp. 405–412.

[41] M. Molloy, *The Glauber dynamics on colourings of a graph with high girth and maximum degree*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 91–98.

[42] B. Monien, *The complexity of determining a shortest cycle of even length*, Computing, 31 (1983), pp. 355–369.

[43] H. Nagamochi and T. Ibaraki, *Computing edge-connectivity in multigraphs and capacitated graphs*, SIAM J. Discrete Math., 5 (1992), pp. 54–66.

[44] N. Robertson and P. D. Seymour, *Graph minors.* III. *Planar tree-width*, J. Combin. Theory Ser. B, 36 (1984), pp. 49–64.

[45] W.-K. Shih, S. Wu, and Y.-S. Kuo, *Unifying maximum cut and minimum cut of a planar graph*, IEEE Trans. Comput., 39 (1990), pp. 694–697.

[46] D. D. Sleator and R. E. Tarjan, *A data structure for dynamic trees*, J. Comput. System Sci., 26 (1983), pp. 362–391.

[47] V. V. Vazirani and M. Yannakakis, *Pfaffian orientations,* 0/1 *permanents, and even cycles in directed graphs*, in Proceedings of the 15th International Colloquium on Automata, Languages and Programming, T. Lepistö and A. Salomaa, ed., 1988, pp. 667–681.

[48] W.-F. Wang and K.-W. Lih, *Labeling planar graphs with conditions on girth and distance two*, SIAM J. Discrete Math., 17 (2003), pp. 264–275.

[49] O. Weimann and R. Yuster, *Computing the girth of a planar graph in* $O(n \log n)$ *time*, SIAM J. Discrete Math., 24 (2010), pp. 609–616.

[50] R. Yuster and U. Zwick, *Finding even cycles even faster*, SIAM J. Discrete Math., 10 (1997), pp. 209–222.