Noname manuscript No.
(will be inserted by the editor)

Sparse Covers for Planar Graphs and Graphs that Exclude a Fixed Minor

Costas Busch · Ryan LaFortune · Srikanta Tirthapura

the date of receipt and acceptance should be inserted later

Abstract We consider the construction of sparse covers for planar graphs and other graphs that exclude a fixed minor. We present an algorithm that gives a cover for the γ-neighborhood of each node. For planar graphs, the cover has radius less than 16γ and degree no more than 18. For every n node graph that excludes a minor of a fixed size, we present an algorithm that yields a cover with radius no more than 4γ and degree $O(\log n)$.

This is a significant improvement over previous results for planar graphs and for graphs excluding a fixed minor; in order to obtain clusters with radius $O(\gamma)$, it was required to have the degree polynomial in n. Our algorithms are based on a recursive application of a basic routine called *shortest-path clustering*, which seems to be a novel approach to the construction of sparse covers.

Since sparse covers have many applications in distributed computing, including compact routing, distributed directories, synchronizers, and Universal TSP, our improved cover construction results in improved algorithms for all these problems, for the class of graphs that exclude a fixed minor.

A preliminary version of this paper appeared in the Proceedings of the ACM Symposium on Principles of Distributed Computing, pages 61-70, 2007 (PODC'07). This work is supported in part by the National Science Foundation through grants 0520102, 0520009, 0834743, 0831903.

This work was performed when Dr. LaFortune was a graduate student at Rensselaer Polytechnic Institute

Costas Busch
Department of Computer Science,
Louisiana State University,
Baton Rouge, LA 70803, USA,
E-mail: busch@csc.lsu.edu

Ryan LaFortune
MITRE Corporation,
202 Burlington Road
Bedford, MA 01730, USA
E-mail: rlafortune@mitre.org

Srikanta Tirthapura
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011, USA
E-mail: snt@iastate.edu

Keywords shortest path clustering

1 Introduction.

A cover Z of a graph G is a set of connected subgraphs called *clusters*, such that the union of all clusters is the vertex set of G. A cover is defined with respect to a locality parameter $\gamma > 0$. It is required that for each node $v \in G$, there is some cluster $C \in Z$ that contains the entire γ-neighborhood of v in G. Two locality metrics characterize the cover: (i) the *radius*, denoted $rad(Z)$, which is the maximum radius of any of its clusters. The radius of a cluster $C \in Z$ is the minimum, taken over all vertices $v \in C$, of the maximum distance from v to any other node in C, and (ii) the *degree*, denoted $deg(Z)$, which is the maximum number of clusters that a node in G is a part of.

Covers play a key role in the design of several locality preserving distributed data structures, including compact routing schemes [3,4,11,29,30,36], distance-dependent distributed directories [14,28,29], network synchronizers [9,12,27,29], transformers for certain classes of distributed algorithms [11], and universal TSP constructions [22,23]. In the design of these data structures, the degree of the cover often translates into the *load* on a vertex imposed by the data structure, and the radius of the cover translates into the *latency*. Thus, it is desirable to have a *sparse cover*, whose radius is close to its locality parameter γ, and whose degree is small.

Awerbuch and Peleg [13] present an algorithm for constructing a sparse cover of a general graph based on the idea of *coarsening*. Starting from an initial cover S consisting of the n clusters formed by taking the γ-neighborhoods of each of the n nodes in G, their algorithm constructs a coarsening cover Z by repeatedly merging clusters in S. For a parameter $k \geq 1$, their algorithm returns a cover Z with $rad(Z) = O(k\gamma)$ and $deg(Z) = O(kn^{1/k})$ (the average degree is $O(n^{1/k})$). By choosing $k = \log n$, the radius is $O(\gamma \log n)$ and the degree is $O(\log n)$. This is the best known result for general graphs. For these graphs, there exists an inherent tradeoff between the radius of a cover and its degree: a small degree may require a large radius, and vice versa. It is known ([29, Theorem 16.2.4]) that for every $k \geq 3$, there exist graphs and values of γ (e.g. $\gamma = 1$) such that for every cover Z, if $rad(Z) \leq k\gamma$, then $deg(Z) = \Omega(n^{1/k})$. Thus, in these graphs if $rad(Z) = O(\gamma)$, then $deg(Z)$ is polynomial in n.

In light of the above tradeoff for arbitrary graphs, it is natural to ask whether better sparse covers can be obtained for special classes of graphs. In this paper, we answer the question in the affirmative for the class of graphs that exclude a fixed minor. This includes many popular graph families, such as: *planar graphs*, which exclude K_5 and $K_{3,3}$, *outerplanar graphs*, which exclude K_4 and $K_{2,3}$, *series-parallel graphs*, which exclude K_4, and *trees*, which exclude K_3.

In particular we provide near-optimal results with small constant factors for the case of planar graphs. For a planar graph G, consider an embedding of G in the Euclidean plane. Let the external nodes of G be defined as those nodes that belong to the unbounded external face of the embedding. The depth of a vertex $v \in G$ is defined as the maximum graph distance from v to an external node of G. The depth of a planar graph G, denoted by $depth(G)$, is defined as the maximum depth of any vertex in G. For example, if G is an outerplanar graph, then its depth is 0 since there exists an embedding of G where every node is on the external face.

1.1 Contributions.

1. For any planar graph G, we present an algorithm for computing a sparse cover Z with $rad(Z) < 16\gamma$ and $deg(Z) \leq 18$. This cover is optimal (modulo constant factors) with respect to both the degree and the radius.
2. For planar graphs whose depth is small, we obtain even better results. For any planar graph G, we present a construction of a sparse cover Z with $rad(Z) \leq 4 \cdot \max\{2\gamma, \gamma + depth(G)\}$, and $deg(G) \leq 6$. For example, for an outerplanar graph, whose depth is 0, this yields a cover Z with $rad(Z) \leq 8\gamma$ and $deg(Z) \leq 6$.
3. For any graph G that excludes a fixed minor graph H, we present an algorithm for computing a sparse cover Z such that $rad(Z) \leq 4\gamma$ and $deg(Z) = O(\log n)$, where n is the number of nodes in G. The constants in the degree bound depend on the size of H.

In all cases above, the graphs are weighted and the algorithms run in polynomial time with respect to G. For the class of H-minor free graphs, our construction improves upon the previous work of Awerbuch and Peleg [13] by providing a smaller radius. For planar graphs, our construction simultaneously improves both the degree and the radius.

1.1.1 Techniques.

Our algorithms for cover construction are based on a recursive application of a basic routine called *shortest-path clustering*. We observe that it is easy to cluster the γ-neighborhood of all nodes along a shortest path in the graph using clusters of radius $O(\gamma)$ and degree $O(1)$. For a graph G, we first identify an appropriate set of shortest paths P in G. We cluster the $c\gamma$-neighborhood (for a constant c) of every path $p \in P$ using shortest-path clustering, and then remove P together with its $c'\gamma$-neighborhood from G, for some $c' < c$. This gives residual connected components $G'_1, G'_2, \ldots, G'_r$ that contain the remaining unclustered nodes as a subset. We apply the same procedure recursively to each G'_i component by identifying appropriate shortest paths in them. The algorithm terminates when there are no remaining nodes.

For H-minor free graphs, we use a result due to Abraham and Gavoille [1] that every H-minor free graph is *κ-path separable*, where κ is a constant that depends on H. The result in [1] is based on the structure theorems for graphs excluding minors of Robertson and Seymour [31,32]. With path separators, the size of each residual graph G'_i is at most half the size of G, and recursive application of this procedure on each G'_i results in a recursion tree of depth at most $\log n$. This results in a logarithmic degree cover, since a node may be clustered multiple times before it is removed from the graph. However, the radius of each cluster is still within a constant factor of γ, since every path is clustered independently. For planar graphs, we apply a similar technique but without using path separators. We show it is possible to choose the shortest paths so that each node is contained in the clusters of a constant number of shortest paths. This translates into covers with constant degree and a radius within a constant factor of γ.

We briefly contrast our techniques with those employed by Awerbuch and Peleg [13] for cover construction on general graphs. They start with a cover of optimal radius, but potentially high degree, and *coarsen* the cover by merging clusters together until the desired tradeoff between the radius and the degree is reached. In contrast, our algorithm does not merge clusters, and as a result, the radius of every cluster remains small. The degree of the cover is controlled through a careful partitioning of the graph through shortest paths, as described above.

1.2 Applications.

As a consequence of our improved sparse cover construction, we provide better data structures for the well-studied distributed computing problems of compact routing, distributed directories, synchronizers, and universal TSP.

1.2.1 Name-Independent Compact Routing.

Consider a distributed system where nodes have arbitrary identifiers. A *routing scheme* is a method that delivers a message to a destination given the identifier of the destination. A *name-independent* routing scheme does not alter the identifiers of the nodes, which are assumed to be in the range $1,2,\ldots,n$. The *stretch* of a routing scheme is the worst case ratio between the total cost of messages sent between a source and destination pair, and the length of the respective shortest path. The *memory overhead* is the number of bits (per node) used to store the routing table. A routing scheme is *compact* if its stretch and memory overhead are small.

There is a tradeoff between stretch and memory overhead. For example, a routing scheme that stores the next hop along the shortest path to every destination has stretch 1, but a very high memory overhead of $O(n\log n)$, and hence is not compact. The other extreme of flooding a message through the network has very little memory overhead, but is not compact either since the stretch can be as much as the total weight of all edges in the network. There has been much work on deriving interesting tradeoffs between the stretch and memory overhead of routing, including [3,4,8,25,26,30,36].

Sparse covers can be used to provide efficient name-independent routing schemes (for example, see [9]). A hierarchy of *regional* routing schemes is created based on a hierarchy of covers $Z_1,Z_2,\ldots,Z_\delta$, where the locality parameter of cover Z_i is $\gamma_i = 2^i$, and $\delta = \lceil \log D \rceil$ where D is the diameter of the graph[1]. Henceforth, we assume that $\log D = O(\log n)$, i.e. the diameter of the graph is polynomial in the number of nodes. Using the covers of Awerbuch and Peleg [13], the resulting routing scheme has stretch $O(k)$ and the average memory bits per node is $O(n^{1/k}\log^2 n)$, for some parameter k. When $k = \log n$, the stretch is $O(\log n)$ and the average memory overhead is $O(\log^2 n)$ bits per node.

On the other hand, using our covers we obtain routing schemes with optimal stretch (within constant factors) for planar and H-minor free graphs. For any planar graph G with n nodes, our covers give a name-independent routing scheme with $O(1)$ stretch and $O(\log^2 n)$ average memory overhead per node. For any graph that excludes a fixed minor, our covers give a name-independent routing scheme with $O(1)$ stretch and $O(\log^3 n)$ average memory overhead per node.

For planar graphs, to our knowledge, this is the first name-independent routing scheme that achieves constant stretch with $O(\log^2 n)$ space per node on average. For H-minor free graphs, Abraham, Gavoille, and Malkhi [3] present name-independent compact routing schemes with $O(1)$ stretch and $\widetilde{O}(1)$ maximum space per node (the $\widetilde{O}$ notation hides polylogarithmic factors). However, their paper does not provide the explicit power of $\log n$ inside the $\widetilde{O}$, hence, we cannot directly compare our results with those in [3].

There are also efficient routing schemes known for a weaker version of the routing problem called *labeled routing*, where the designer of the routing scheme is given the flexibility to assign names to nodes. Thorup [35] gives a labeled routing scheme for planar graphs with

[1] The diameter D of a graph G is the maximum shortest path distance between any two nodes in the graph. It also holds that $rad(G) \le D \le 2 \cdot rad(G)$, where $rad(G)$ denotes the radius of G.

stretch $(1+\varepsilon)$ and memory overhead of $O((1/\varepsilon)\log^2 n)$ maximum bits per node. Name-independent routing schemes are less restrictive to the user than labeled routing, and hence a harder problem.

1.2.2 Directories for Mobile Objects.

A directory is a basic service in a distributed system which, given an object's name, returns the location of the object (or any other information dependent on the object's current position). Very often, it is necessary to have directories that support mobile objects, such as an object being sensed and tracked by a wireless sensor network, or a mobile phone user in a large cellular phone network. A directory for mobile objects provides two operations: *find*, to locate an object given its name, and *move*, to move an object from one node to another. There is an inherent tradeoff between the cost of these operations. The greater the effort spent in updating the directory in response to a *move*, the easier is the implementation of the *find* operation, and vice versa. The performance of a directory is measured by the $Stretch_{find}$, the $Stretch_{move}$, and the memory overhead of the directory (formal definitions of these metrics can be found in [15]).

Awerbuch and Peleg [15,29] construct directories for mobile objects based on a hierarchy of *regional directories*, which are in turn constructed using sparse covers with appropriately defined locality parameters. Their directories are appropriate for general networks and have performance $Stretch_{find} = O(\log^2 n)$ and $Stretch_{move} = O(\log^2 n)$ ([15, Corollary 5.4.8])[2].

Our sparse cover construction yields improved directories for mobile objects for planar and H-minor free graphs with the following performance guarantees. For planar graphs, our covers give a distributed directory with $Stretch_{find} = O(1)$ and $Stretch_{move} = O(\log n)$. For any graph that excludes a fixed minor H, our covers give a distributed directory with $Stretch_{find} = O(\log n)$ and $Stretch_{move} = O(\log n)$. In both cases, we obtain improved bounds compared to those of previously known directories.

1.2.3 Synchronizers.

Many distributed algorithms are designed assuming a synchronous model where the processors execute and communicate in time synchronized rounds [9,27]. However, synchrony is not always feasible in real systems due to physical limitations such as different processing speeds or geographical dispersal. *Synchronizers* are distributed programs that enable the execution of synchronized algorithms in asynchronous systems [9,10,27,29]. A synchronizer uses logical rounds to simulate the time rounds of the synchronous algorithm.

One of the most efficient synchronizers is called ZETA [33]. This synchronizer is based on a sparse cover with locality parameter $\gamma = 1$, radius $O(\log_k n)$, and average degree $O(k)$, for some parameter k. ZETA simulates a round in $O(\log_k n)$ time steps and uses $O(k)$ messages per node on average. In contrast, using our covers, we obtain a better time to simulate a round. For planar graphs, our covers give a synchronizer with $O(1)$ time and average messages per node. For H-minor free graphs, the synchronizer has time $O(1)$ and uses $O(\log n)$ messages per node on average.

[2] We present all results assuming that the diameter of the graph is polynomial in the number of nodes.

1.2.4 Universal TSP.

In the *universal* TSP (traveling salesperson) problem, given a set of n nodes in a metric space, the task is to construct a single universal tour that connects all the points, such that the universal tour can provide a good approximation to any TSP tour for any subset of the n nodes. Jia *et al.* [23] give a universal TSP tour construction that approximates any TSP tour within a $O(\log^4 n/\log\log n)$ factor. The construction is based on transforming sparse covers with locality parameter γ, degree I, and diameter $\sigma \cdot \gamma$, to a parameterized (γ, σ, I)-partitioning of the nodes where the γ-neighborhood of every node belongs to at most I clusters of diameter at most $\sigma \cdot \gamma$ in the partition. The partition then gives a universal TSP tour with $O(\sigma^2 I \log_\sigma n)$ approximation. Using the Awerbuch and Peleg [13] sparse cover construction with $\sigma = O(\log n)$ and $I = O(\log n)$, this gives the resulting TSP tour.

For planar and minor-free metrics, where the distance between two nodes is evaluated as the distance between the nodes in the graph, Hajiaghayi *et al.* [22] provide a universal TSP tour with $O(\log^2 n)$ approximation. They use a randomized sparse cover construction for minor-free graphs which gives a cover of degree $O(\log n)$ and $\sigma = O(1)$. They also provide a lower bound of $\Omega(\sqrt[6]{\log n/\log\log n})$ for the best approximation of any algorithm in a $n \times n$ grid.

Our work improve the above results in the following ways:

- Using our sparse cover construction for planar graphs with $\sigma = O(1)$ and $I = O(1)$ we obtain a universal TSP tour with $O(\log n)$ approximation. The constants in the asymptotic notation are small, and our construction is deterministic while the one in [22] is randomized.
- Our result on minor-free graphs gives a deterministic construction of a universal TSP with $O(\log^2 n)$ approximation for any H-minor free graph.

1.3 Related Work.

Concurrent with our work, there is a closely related work by Abraham, Gavoille, Malkhi, and Wieder [6,5] that gives an algorithm for constructing a sparse cover of diameter $4(r+1)^2\gamma$ and degree $2^{O(r)}r!$ for any graph excluding $K_{r,r}$, for a fixed $r > 1$. While the goal of both works is the same, our work yields different tradeoffs than [6]. For graphs excluding a fixed minor H, our algorithm returns a cover with radius at most 4γ, which is independent of the size of the excluded minor, while their cover has a greater radius of $O(r^2\gamma)$. On the other hand, their degree of $O(1)$ is smaller than ours of $O(\log n)$.

For planar graphs, our algorithm improves on both degree and radius when compared with [5]. Our algorithm gives a radius less than 16γ, and a degree of no more than 18. Their result for planar graphs, by using $r = 3$, since a planar graph must exclude $K_{3,3}$, gives a diameter of 64γ (note that the upper bound for the radius is still 64γ) and the degree of the cover is 123 (this can be derived from analyzing the proof of Theorem 4 in [6]).

Klein, Plotkin, and Rao [24] obtain sparse covers for H-minor free graphs with degree $2^{O(r)}$ but with a *weak diameter* $O(r^2\gamma)$ where the shortest path between two nodes in the same cluster may not necessarily lie in the cluster itself. For many applications of covers, such as compact routing and distributed directories, this is not sufficient. In contrast, for constant r, our construction yields clusters with a *strong diameter* of $O(\gamma)$ where the shortest path lies completely within the cluster.

For graphs with doubling dimension α, Abraham, Gavoille, Goldberg, and Malkhi [2] present a sparse cover with degree 4^α and radius $O(\gamma)$. However, since planar graphs and

H-minor free graphs can have large doubling dimensions, this does not yield efficient sparse covers for these graphs.

Srinivasagopalan, Busch, and Iyengar [34] use the planar graph sparse cover construction of this paper to build an oblivious scheme for buy-at-bulk network design problems. By appropriately assigning colors to the clusters of a hierarchical sparse cover, they provide an algorithm that returns a fixed set of routing paths formed by connecting leaders chosen in the clusters. The routing scheme is oblivious to the set of source nodes and to the canonical aggregation function f, and achieves an $O(\log n)$ approximation to any buy-at-bulk aggregation problem, where n is the number of nodes in the graph. This is an asymptotically optimal result, since $\Omega(\log n)$ is a lower bound for the oblivious Steiner tree problem on the plane, which is a special case of the buy-at-bulk problem. The best previous result is $O(\log^2 n)$ approximation for general graphs [21].

Outline of the Paper: We give basic definitions and preliminaries for graphs and covers in Section 2. We present the algorithm for clustering shortest paths in Section 3. In Section 4 we give a clustering algorithm for k-path separable graphs, which is further applied to graphs excluding a fixed minor. The result for planar graphs is given in Section 5. We conclude in Section 6.

2 Definitions and Preliminaries.

Some of the following definitions are borrowed from Awerbuch and Peleg [13] and from Abraham and Gavoille [1].

2.1 Graph Basics.

All graphs in this paper are weighted. Consider a weighted graph $G = (V, E, \omega)$, where V is the set of nodes, E is the set of edges, and ω is a weight function $E \to \mathbb{R}^+$ that assigns a weight $\omega(e) > 0$ to every edge $e \in E$. For simplicity, we will also write $G = (V, E)$ and sometimes use the notation $v \in G$ to denote $v \in V$ and $e \in G$ to denote $e \in E$. For a graph H, we use the notation $V(H)$ and $E(H)$ to denote the nodes and edges of H respectively.

A *walk* q is a sequence of nodes $q = v_1, v_2, \ldots, v_k$ where nodes may be repeated. The length of q is defined as $length(q) = \sum_{i=1}^{k-1} \omega(v_i, v_{i+1})$. We also use walks with one node $q = v$, where $v \in V$, which has $length(q) = 0$. If $v_1 = v_k$, the walk is *closed*. A *path* is a walk with no repeated nodes.

Graph G is *connected* if there is a path between every pair of nodes. $G' = (V', E')$ is a *subgraph* of $G = (V, E)$, if $V' \subseteq V$, and $E' \subseteq E$. If $V' \neq V$ or $E' \neq E$, then G' is said to be a proper subgraph of G. In the case where graph G is not connected, it consists of *connected components* $G_1, G_2, \ldots, G_k$, where each G_i is a connected subgraph that is not a proper subgraph of any other connected subgraph of G. For any set of nodes $V' \subseteq V$, the *induced subgraph* by V' is $G(V') = (V', E')$ where $E' = \{(u, v) \in E : u, v \in V'\}$. Let $G - V' = G(V - V')$ denote the subgraph obtained by removing the vertex set V' from G. For any subgraph $G' = (V', E')$, $G - G' = G - V'$. For any two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their union graph is $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.

The distance between two nodes u, v in G, denoted $dist_G(u, v)$, is the length of the shortest path between u and v in G. If there is no path connecting the nodes, then $dist_G(u, v) = \infty$. The δ-*neighborhood* of a node v in G is $N_\delta(v, G) = \{w \in V | dist_G(v, w) \leq \delta\}$. For $V' \subseteq V$, the δ-*neighborhood* of V' in G is $N_\delta(V',G) = \bigcup_{v\in V'} N_\delta(v,G)$. If G is connected, the *radius* of a node $v \in V$ with respect to G is $rad(v,G) = \max_{w\in V}(dist_G(v,w))$. The radius of G is defined as $rad(G) = \min_{v\in V}(rad(v,G))$. If G is not connected, then $rad(G) = \infty$.

2.2 Covers.

Consider a set of vertices $C \subseteq V$ in graph $G = (V,E)$. The set C is called a *cluster* if the induced subgraph $G(C)$ is connected. When the context is clear, we will sometimes use C to refer to $G(C)$. Let $Z = \{C_1, C_2, \ldots, C_k\}$ be a set of clusters in G. For every node $v \in G$, let $Z(v) \subseteq Z$ denote the set of clusters that contain v. The *degree* of v in Z is defined as $deg(v,Z) = |Z(v)|$. The degree of Z is defined as $deg(Z) = \max_{v\in V} deg(v,Z)$. The radius of Z is defined as $rad(Z) = \max_{C\in Z}(rad(C))$.

For $\gamma > 0$, a set of clusters Z is said to γ-*satisfy* a node v in G, if there is a cluster $C \in Z$, such that $N_\gamma(v,G) \subseteq C$. A set of clusters Z is said to be a γ-*cover* for G, if every node of G is γ-satisfied by Z in G. We also say that Z γ-satisfies a set of nodes X in G, if every node in X is γ-satisfied by Z in G (note that the γ-neighborhood of each node in X is taken with respect to G).

2.3 Path Separators.

A graph G with n nodes is *k-path separable* [1] if there exists a subgraph S, called the *k-path separator*, such that:

(i) $S = P_1 \cup P_2 \cup \cdots \cup P_\ell$, where for each $1 \le i \le \ell$, subgraph P_i is the union of k_i paths where each path is shortest in $G - \bigcup_{1\le j<i} P_j$ with respect to its end points,
(ii) $\sum_i k_i \le k$, and
(iii) either $G - S$ is empty, or each connected component of $G - S$ is k-path separable and has at most $n/2$ nodes.

For instance, any rectangular grid of nodes (2-dimensional mesh) is 1-path separable by taking S to be the middle row path. Trees are also 1-path separable by taking S to be the *center node* whose subtrees have at most $n/2$ nodes. Thorup [35] shows how to compute in polynomial time a 3-path separator for planar graphs, in particular, the 3-path separator is $S = P_1$. That is, S consists of three paths each of which is a shortest path in the original graph.

2.4 Graph Minors.

The *contraction* of edge $e = (u,v)$ in G is the replacement of vertices u and v by a single vertex whose incident edges are all the edges incident to u or to v except for e. A graph H is said to be a *minor* of graph G, if H is a subgraph of a graph obtained by a series of edge contractions starting from G. Graph G is said to be *H-minor free*, if H is not a minor of G. Abraham and Gavoille [1] generalize the result of Thorup [35] for the class of H-minor free graphs:

Theorem 1 (Abraham and Gavoille [1]) *Every H-minor free connected graph is k-path separable, for some $k = k(H)$, and a k-path separator can be computed in polynomial time.*

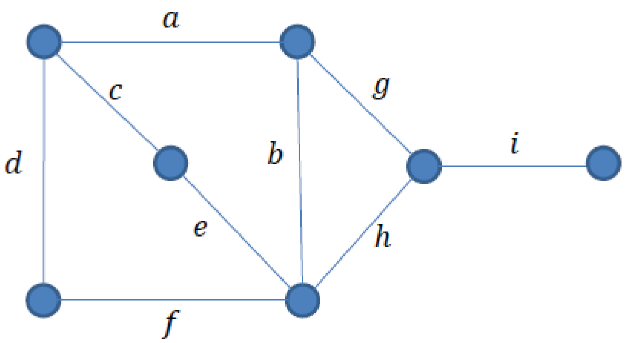

Fig. 1 An example planar graph with four faces (including the external face)

The proof of Theorem 1 is based on the structure theorems for graphs excluding minors of Robertson and Seymour [31,32]. We note that in Theorem 1, the parameter k is exponential in the size of the minor. Some interesting classes of H-minor free graphs are: planar graphs, which exclude K_5 and $K_{3,3}$, outerplanar graphs, which exclude K_4 and $K_{2,3}$, series-parallel graphs, which exclude K_4, and trees, which exclude K_3.

2.5 Planar Graphs.

Let G be a connected and weighted graph that has an embedding in the Euclidean plane where no two edges cross each other. Such an embedding is called a *plane graph*. In the following discussion, when we say "planar graph" G we actually refer to a plane graph that corresponds to G.

The edges of a planar graph G divide the Euclidean plane into closed geometric regions called *faces*. The *external face* is the face that surrounds the whole graph; the other faces are called *internal*. A node may belong to multiple faces, while an edge can belong to at most two faces. A node or edge that belongs to the external face will be called *external*. Figure 1 depicts a planar graph with 3 internal faces.

For any connected planar graph G, let f_G denote a walk that starts at some external vertex w, traverses the external edges of G in the clockwise order and returns to w. For example, for the graph G shown in Figure 1, $f_G = agiihfd$ (sequence of edges) or equivalently a sequence of vertices. This walk traverses every external edge of G exactly once, except for the cut-edges of G (that disconnect G) which are traversed twice, once along each direction. According to convenience, we treat f_G, as a sequence of vertices, or as a sequence of edges, or just as a subgraph of G (thus ignoring duplicate occurrences of an edge).

For any node $v \in G$, let $depth(v,G)$ denote the shortest distance between v and an external node of G. Let $depth(G) = \max_{v \in V} depth(v,G)$.

3 Shortest-Path Clustering.

Consider an arbitrary weighted graph G, and a shortest path p between a pair of nodes in G. For any $\beta > 0$, we construct a set of clusters R, which β-satisfies every node of p in G. The returned set R has a small radius, 2β, and a small degree, 3. Algorithm Shortest-Path-Cluster contains the details of the construction of R. Lemma 1 establishes the correctness of the algorithm.

Algorithm 1: Shortest-Path-Cluster(G, p, β)

```
Input: Graph $G$; shortest path $p \in G$; parameter $\beta > 0$;
Output: A set of clusters that $\beta$-satisfies $p$;
Suppose $p = v_1, v_2, \ldots, v_\ell$;
// partition $p$ into subpaths $p_1, p_2, \ldots, p_s$ of length at most $\beta$
$i \leftarrow 1$; $j \leftarrow 1$;
while $i \neq \ell + 1$ do
    Let $p_j$ consist of all nodes $v_k$ such that $i \leq k \leq \ell$ and $dist_G(v_i, v_k) \leq \beta$;
    $j \leftarrow j + 1$;
    Let $i$ be the smallest index such that $i \leq \ell$ and $v_i$ is not contained in any $p_k$ for $k < j$. If no such $i$
    exists, then $i = \ell + 1$;
end
Let $s$ denote the total number of subpaths $p_1, p_2, \ldots, p_s$ of $p$ generated;
// cluster the subpaths
for $i = 1$ to $s$ do
    $A_i \leftarrow N_\beta(p_i, G)$;
end
$R \leftarrow \bigcup_{1 \leq i \leq s} A_i$;
return $R$;
```

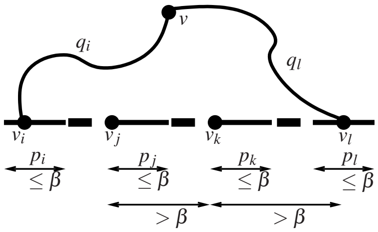

Fig. 2 A demonstration of the proof of property *iii* of Lemma 1.

Lemma 1 *For any graph G, shortest path $p \in G$, and $\beta > 0$, the set R returned by Algorithm* Shortest-Path-Cluster(G, p, β) *has the following properties: (i) R is a set of clusters that β-satisfies p in G; (ii) $rad(R) \leq 2\beta$; (iii) $deg(R) \leq 3$.*

Proof For property i, it is easy to see that R is a set of clusters, since each A_i is a connected subgraph of G consisting of the β-neighborhood of a subpath p_i of p. For each node $v \in p_i$, A_i β-satisfies v in G, since it contains $N_\beta(v, G)$. Thus, R β-satisfies p in G.

For property ii, we show that each cluster A_i has radius no more than 2β. Let v_i be an arbitrary vertex in p_i. By the construction, for any node $v \in p_i$, it must be true that

$dist_G(v_i, v) \leq \beta$. Since any node $u \in A_i$ is at a distance of no more than β from some node in p_i, there is a path of length at most 2β from v_i to u. Thus, $rad(R) \leq 2\beta$.

For property *iii*, suppose for the sake of contradiction that $deg(R) \geq 4$ (see Figure 2). Let v be a node with degree $deg(v, R) = deg(R)$. Then v belongs to at least 4 clusters, say: A_i, A_j, A_k, and A_l, with $i < j < k < l$. Since v belongs to A_i, there is a path q_i of length at most β between v and some node $v_i \in p_i$. Similarly, there exists a path q_l of length at most β between v and some node $v_l \in p_l$. By concatenating q_i and q_l, we obtain a path of length at most 2β connecting v_i and v_l. On the other hand, both v_i and v_l lie on p, which is a shortest path in G, and hence the path from v_i to v_l on p must be a shortest path from v_i to v_l. Let v_j and v_k denote the nodes on p_j and p_k respectively, that are closest to v_i. By the construction, $dist_G(v_j, v_k) > \beta$, since otherwise, v_k would have been included in p_j. Similarly, $dist_G(v_k, v_l) > \beta$. Since $dist_G(v_i, v_l) > dist_G(v_j, v_k) + dist_G(v_k, v_l)$, it follows that $dist_G(v_i, v_l) > 2\beta$, a contradiction. Thus, $deg(R) \leq 3$.

4 Cover for *k*-Path Separable Graphs.

We now present Algorithm Separator-Cover, which returns a cover with a small radius and degree for any graph that has a *k*-path separator. Theorem 2 establishes the correctness and properties of the algorithm, and uses Lemma 2, which gives some useful properties about clusters.

Algorithm 2: Separator-Cover(G, γ)

```
Input: Connected graph $G$ that is $k$-path separable; locality parameter $\gamma > 0$;
Output: $\gamma$-cover for $G$;
// base case
if $G$ consists of a single vertex $v$ then
    $Z \leftarrow \{v\}$;
    return $Z$;
end
// main case
Let $S = P_1 \cup P_2 \cup \cdots \cup P_l$ be a $k$-path separator of $G$;
for $i = 1$ to $l$ do
    foreach $p \in P_i$ do
        $A_i \leftarrow$ Shortest-Path-Cluster$(G - \bigcup_{1 \leq j < i} P_j, p, 2\gamma)$;
    end
end
$A \leftarrow \bigcup_{1 \leq i \leq l} A_i$;
$G' \leftarrow G - \bigcup_{1 \leq j \leq l} P_j$;
// recursively cluster each connected component
Let $G'_1, G'_2, \ldots, G'_r$ denote the connected components of $G'$;
$B \leftarrow \bigcup_{1 \leq i \leq r}$ Separator-Cover$(G'_i, \gamma)$;
$Z \leftarrow A \cup B$;
return $Z$;
```

Lemma 2 *Let C be a set of clusters that 2γ-satisfies a set of nodes W in graph G. If some set of clusters D is a γ-cover for $G - W$, then $C \cup D$ is a γ-cover for G.*

Proof Since C 2γ-satisfies W in G, C also γ-satisfies $N_\gamma(W, G)$ in G. Thus, C γ-satisfies $W \cup N_\gamma(W, G)$ in G. Next, consider a vertex $u \in G - (W \cup N_\gamma(W, G))$. For any vertex $u' \in W$, it must be true that $u' \notin N_\gamma(u,G)$, since $u \notin N_\gamma(W,G)$, implying that $u \notin N_\gamma(u',G)$. Thus, $N_\gamma(u,G)$ lies completely in $G-W$. Since D is a γ-cover for $G-W$, for every vertex $u \in (G-W)-N_\gamma(W,G)$, D γ-satisfies u in $G-W$, and hence in G. For any $u' \in W \cup N_\gamma(W,G)$, C γ-satisfies u' in G. Thus, for any $v \in G$, $C \cup D$ γ-satisfies v in G, and is therefore a γ-cover for G.

Theorem 2 *For any connected k-path separable graph G with n nodes, and locality parameter $\gamma > 0$, Algorithm* Separator-Cover(G,γ) *returns a set Z with the following properties: (i) Z is a γ-cover for G; (ii) $rad(Z) \leq 4\gamma$; (iii) $deg(Z) \leq 3k(\lg n+1)$.*

Proof For property *ii*, we note that each cluster is obtained from an invocation of Algorithm Shortest-Path-Cluster with input argument $\beta = 2\gamma$. From Lemma 1, the radius of each cluster is at most $2\beta = 4\gamma$. Thus, $rad(Z) \leq 4\gamma$.

For properties *i* and *iii*, the proof is by induction on the number of vertices in G. The base case is when G has only one vertex, in which case properties *i* and *iii* clearly hold. For the inductive case, suppose that for every k-path separable graph with number of vertices $n' < n$, the algorithm returns a γ-cover for the graph, and that the degree of the cover was no more than $3k(\lg n'+1)$. Let G be a k-path separable graph with n vertices.

We first prove the inductive case for property *i*. The last part of the algorithm recursively calls Separator-Cover on every connected component in G'. Since the number of vertices in G' is less than n, the number of vertices in each G'_i component is less than n. By the inductive assumption, for each $i = 1,2,\ldots,r$, Separator-Cover(G'_i,k,γ) returns a γ-cover for G'_i. The union of the γ-covers for the connected components of G' is clearly a γ-cover for G', hence B is a γ-cover for G'.

For $i = 1,2,\ldots,l+1$, define $G_i = G - \bigcup_{1\leq j<i} P_j$. Clearly, $G_1 = G$ and $G_{l+1} = G'$. We will prove that for all i such that $1 \leq i \leq l+1$, the set $\bigcup_{i\leq j\leq l} A_j \cup B$ is a γ-cover for G_i. The proof is through reverse induction on i starting from $i = l+1$ and going down until $i = 1$. The base case $i = l+1$ is clear since B is a γ-cover for $G' = G_{l+1}$. Suppose the above statement is true for $i = v$, i.e. $A_v \cup A_{v+1} \cup \ldots \cup A_l \cup B$ is a γ-cover for G_v. Consider $G_{v-1} = G_v \cup P_{v-1}$. From the correctness of Algorithm Shortest-Path-Cluster (proven in Lemma 1), we have that A_{v-1} 2γ-satisfies P_{v-1} in G_{v-1}. Since $A_v \cup A_{v+1} \cup \ldots \cup A_l \cup B$ is a γ-cover for $G_{v-1} - P_{v-1}$, using Lemma 2 we have $A_{v-1} \cup A_v \cup \ldots \cup A_l \cup B$ is a γ-cover for G_{v-1}, thereby proving the inductive step. Thus, we have $\bigcup_{1\leq j\leq l} A_j \cup B$ is a γ-cover for $G_1 = G$, proving the correctness of the algorithm for graph G with n vertices.

For property *iii*, consider any vertex v in G. Either v is included in the path separator S, or v is a part of G'. If $v \in S$, then the degree of v in Z is no more than $3k$, since it does not appear in any recursive invocation of Separator-Cover, and it is involved in no more than k calls to Shortest-Path-Cluster. Each such call causes the degree of v to increase by no more than 3, due to Lemma 1.

In case $v \in G'$, then we need to add the degree of v due to recursive invocations of Separator-Cover on the components of G'. Since v can belong to only one component of G', and the number of vertices in this component can be no more than $n/2$, using the inductive hypothesis, we have that the degree of v due to the recursive call is no more than $3k(\lg(n/2)+1)$. Note that the degree of v can further increase by no more than $3k$, leading to a total degree of no more than $3k(\lg(n/2)+1)+3k = 3k(\lg n+1)$.

Upon combining Theorem 2 with Theorem 1, we get the following.

Theorem 3 *For any graph G that excludes a fixed size minor H, given a parameter $\gamma > 0$, there is an algorithm that returns in polynomial time a set of clusters Z with the following* *properties: (i) Z is a γ-cover for G; (ii) $rad(Z) \le 4\gamma$; (iii) $deg(Z) \le 3k(\lg n + 1)$; where $k = k(H)$ is a parameter that depends on the size of the excluded minor H.*

According to [1] the parameter k for minor H in Theorem 3 is bounded by $k(H) = O(hg(h+g))$, where $g < |E(H)| = O(\rho^2)$, where ρ is the number of nodes in H, and the bound on $h = h(H)$ is determined by the structure theorems for graphs excluding minors of Robertson and Seymour [31,32].

5 Cover for Planar Graphs.

Since every planar graph is 3-path separable [35], Theorem 2 immediately yields a γ-cover for a planar graph with radius $O(\gamma)$ and degree $O(\log n)$. In this section, we present an improved cover for planar graphs whose radius is less than 16γ and degree no more than 18, both of which are optimal up to constant factors.

5.1 High Level Description of the Algorithm.

Without loss of generality, consider a connected and weighted planar graph G. If G is not connected, then it can be handled by clustering each connected component separately. Our algorithm breaks up the planar graph G into overlapping planar subgraphs called *zones*, such that: (i) the depth of each zone is less than 3γ, (ii) each node in G belongs to no more than three zones, and (iii) clustering each zone separately is sufficient to cluster the whole graph. This way, we can focus on clustering only planar graphs whose depth is $O(\gamma)$. This division of a planar graph into subgraphs based on the distance of nodes to an external node is not a new idea, and has been used before, notably by Brenda Baker [16] in her work on approximation algorithms for NP-complete problems on planar graphs. Thus, our algorithm is divided into two main parts.

- Algorithm Depth-Cover, which clusters a graph G, and is useful when $depth(G) = O(\gamma)$.
- Algorithm Planar-Cover, which clusters arbitrary planar graphs using Depth-Cover as a subroutine.

We now proceed to describe Algorithms Planar-Cover and Depth-Cover in Sections 5.2 and 5.4 respectively. In Section 5.3, we prove some basic results about planar graphs that are useful in analyzing Algorithm Depth-Cover.

5.2 General Planar Cover.

In this section we describe the main algorithm, Algorithm Planar-Cover, which, given a planar graph G, and locality parameter γ, constructs a γ-cover with radius $O(\gamma)$ and degree $O(1)$, for any $\gamma > 0$. At a high-level, Planar-Cover divides G into *zones*, as follows, and then clusters each zone using Algorithm Depth-Cover. The union of the clusters for the different zones is the cover for G.

The *bands* of G, denoted by W_j, where $0 \le j \le \kappa$, and $\kappa = \lfloor depth(G)/\gamma \rfloor$, are defined as follows.

$$W_j = \{v \in G \quad | \quad j\gamma \le depth(v, G) < (j+1)\gamma\}.$$

Our goal is to γ-satisfy the nodes in each band W_i. However, we cannot cluster each band in isolation, since in G, the γ-neighborhood of a node in W_i may not be completely contained within W_i. For this reason, we define the *zones* $S_i, 0 \leq i \leq \kappa$, which are vertex induced subgraphs of G, as follows. If κ is $0, 1$ or 2, then the entire graph G is a single zone, S_0. Otherwise:

- $S_0 = G(W_0 \cup W_1)$.
- For $1 \leq i < \kappa$, $S_i = G(W_{i-1} \cup W_i \cup W_{i+1})$.
- Finally, $S_\kappa = G(W_{\kappa-1} \cup W_\kappa)$.

Lemma 3 *The zones have the following properties.*

(i) For each vertex $v \in G$, v can belong to no more than three zones in $\{S_0, S_1, \ldots, S_\kappa\}$.
(ii) For each $i = 0, \ldots \kappa$, for each vertex $v \in W_i$, $N_\gamma(v, G) = N_\gamma(v, S_i)$.
(iii) For each $i = 0 \ldots \kappa$, $depth(S_i) < 3\gamma$.

Proof Proof of (i). Suppose vertex v is in band W_j. By the definition of the zones, v cannot belong to any zones outside of S_{j-1}, S_j, S_{j+1}. Thus it cannot belong to more than three zones.

Proof of (ii). Consider node $u \in N_\gamma(v, G)$, Suppose that u was in band W_j. We show that $(i-1) \leq j \leq (i+1)$. Since v is at a distance less than $(i+1)\gamma$ from some external node of G, and $dist_G(v, u) \leq \gamma$, it follows that u is at a distance less than $(i+2)\gamma$ from an external node of G. Thus, $j \leq (i+1)$. Similarly, we can show that $j \geq (i-1)$; if $j < (i-1)$, then v cannot be in W_i. Thus, for every $u \in N_\gamma(v, G)$, u must be in $\{W_{i-1} \cup W_i \cup W_{i+1}\}$. Since S_i is a vertex induced subgraph that contains all vertices in $N_\gamma(v, G)$, $N_\gamma(v, G) = N_\gamma(v, S_i)$.

Proof of (iii). Consider the case $0 < i < \kappa$. A similar proof applies for $i = 0$ and $i = \kappa$. Let B_i denote the nodes in S_i that are adjacent to nodes in W_j, for $j \leq (i-2)$ (if $i < 2$, then B_i is the set of external nodes in G). We show that every node in B_i is an external node of S_i. Consider the graph $G_i = G - \bigcup_{j=0}^{i-2} W_i$. Clearly, S_i is a subgraph of G_i. Let vertex $v \in B_i$. Let p be a path in G from v to an external node of G, that does not use any vertex of G_i (other than v itself). Such a path must exist since v is adjacent to some node in a band W_j, for $j \leq (i-2)$. In transforming G to G_i, every vertex of p is deleted, except for v. From Lemma 5, it follows that v is an external node in G_i. Since S_i is a subgraph of G_i, it follows from Observation 1 that v is an external node of S_i too.

Now consider any node $u \in S_i$; we show $dist_{S_i}(u, B_i) < 3\gamma$. Let w be the closest external node in G to u. We know $dist_G(u, w) < (i+2)\gamma$. Any path in G from u to w must pass through B_i. Thus, $dist_G(u, w) = dist_G(u, B_i) + dist_G(B_i, w)$ (where $dist_G(u, B_i)$ denotes the shortest distance of u to any node in B_i, and symmetrically for $dist_G(B_i, w)$). Since every node in B_i is in one of $\{W_{i-1}, W_i, W_{i+1}\}$, and w is an external node, we have $dist_G(B_i, w) \geq (i-1)\gamma$. Thus, we get:

$$\begin{aligned} dist_G(u, B_i) + dist_G(B_i, w) &< (i+2)\gamma \\ dist_G(u, B_i) &< (i+2)\gamma - (i-1)\gamma = 3\gamma. \end{aligned}$$

Since the shortest path from u to B_i must lie in S_i, it follows that $dist_{S_i}(u, B_i) < 3\gamma$, and that the depth of S_i is less than 3γ.

In this way, we have reduced the problem of producing a cover for G into producing a cover for a zone S_i, whose depth is less than 3γ. The steps are presented in Algorithm Depth-Cover (Algorithm 4).

Algorithm 3: Planar-Cover(G, γ)

```
Input: Connected planar graph $G$; locality parameter $\gamma > 0$;
Output: A $\gamma$-cover for $G$;
Let $S_0, S_2, \ldots, S_\kappa$ be the different zones of $G$, where $\kappa = \lfloor depth(G)/\gamma \rfloor$;
$Z \leftarrow \emptyset$;
foreach $i$ from $0$ to $\kappa$ do
    foreach connected component $S$ of $S_i$ do
        $Z \leftarrow Z \cup$ Depth-Cover$(S, \gamma)$;
    end
end
return $Z$;
```

Theorem 4 *For any connected planar graph G and parameter $\gamma > 0$, Algorithm* Planar-Cover *returns in polynomial time a γ-cover Z^* with $rad(Z^*) < 16\gamma$ and $deg(Z^*) \leq 18$.*

Proof From Theorem 5, each call to Algorithm Depth-Cover(S, γ) results in a γ cover of S, that has radius no more than $4 \cdot \max\{2\gamma, \gamma + depth(S)\}$, and degree no more than 6. From Lemma 3, part (ii), it follows that for each vertex $v \in G$, the cover Z^* has a cluster that contains all of $N_\gamma(v, G)$. Hence, Z^* is a γ-cover for G.

Next, from Lemma 3, part (i), each vertex $v \in G$ participates in at most three instances of Algorithm Depth-Cover. Since the degree of each cover returned by Depth-Cover is no more than 6 (Theorem 5), it follows that the degree of Z^* is no more than 18.

The radius of Z is the maximum radius of a cover returned by Depth-Cover(S, γ). From Lemma 3, part (iii), $depth(S) < 3\gamma$. Using Theorem 5, we finally get $rad(Z^*) < 4 \cdot (4\gamma) = 16\gamma$.

Observation 1 *Let G' be a subgraph of a planar graph G. For any $v \in G'$, if v is external in G, then v is external in G' too.*

Lemma 4 *For a planar graph G, if v is an external node in G and u is a neighbor of v in G, then u is an external node in $G - \{v\}$.*

Proof Let the edges incident at v be $e_1, e_2, \ldots, e_k$. Assume without loss of generality that e_1 is an external edge, and that the edges $e_1, e_2, \ldots$ are in clockwise order centered at v. For $i = 1 \ldots k$, let v_i denote the vertex at the other end of e_i, and let $u = v_\ell$.

We simulate the removal of v from G by removing the edges $e_1, e_2, \ldots$ in order. Note that v_1 is an external node. When e_1 is removed, e_2 becomes an external edge and v_2 an external node. Proceeding thus, we get that after the removal of $e_1, e_2, \ldots, e_{\ell-1}$, v_ℓ is an external node. From Observation 1, it follows that v_ℓ remains an external node after all of $e_1, e_2, \ldots, e_k$ are deleted.

Lemma 5 *For a planar graph G, let v be an external node in G, and let $v, u_1, u_2, \ldots, u_k, u = u_{k+1}$ be a path from v to u in G. Then, u is an external node in $G - \{v, u_1, u_2, \ldots, u_k\}$.*

Proof For $i \geq 1$, let G_i denote the graph $G - v - \bigcup_{j=1}^{i-1} u_i$. By induction on i, we show that u_i is an external node of G_i. The base case, $i = 1$, follows from Lemma 4. For the inductive step, assume that u_j is an external node of G_j. It follows from Lemma 4 that u_{j+1} is an external node of G_{j+1}, completing the proof.

5.3 Basic Results for Planar Graphs.

We now prove some basic properties of planar graphs that will be useful further.

Lemma 6 *Let C be a connected planar graph with at least two vertices. It is possible to partition C into two subgraphs A and B such that the following conditions hold: (1)$V(A) \cap V(B) = \emptyset$, i.e. A and B have no common vertices, (2)$V(A) \cup V(B) = V(C)$ (3)A and B are connected, and (4)Both A and B have at least one vertex from f_C.*

Proof Consider any vertex $v \in f_C$. Consider the graph $C - v$. Since v is connected to at least one other vertex in f_C, there is at least one vertex, say $v' \in (C - v)$ that belongs to f_C. If $C - v$ is connected, then we have $A = \{v\}$, and $B = C - v$, and this satisfies conditions (1) to (4).

Suppose that $C - v$ is not connected. Let $C_1, C_2, \ldots, C_k$ denote the connected components in $G - v$. Suppose that C_1 is the component that contains v'. We set $B = C_1$, and A to be the graph $\{v\} \cup C_2 \cup C_3 \ldots \cup C_k$. We show that these sets A and B satisfy the conditions required in the lemma. Conditions (1) and (2) are obviously satisfied, since $A = C - B$. It is also clear that A and B have at least one vertex from f_C, and that B is connected. It remains to be shown that A is connected. Consider any two components C_i, C_j, for $i, j \neq 1$. For any two vertices $v_1, v_2 \in C_i$, there is always a path between them in A, since C_i is a connected component. For $v_1 \in C_i$ and $v_2 \in C_j$, there is a path between them through v. Thus, A is a also connected subgraph of C, proving the Lemma.

The *intersection* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is defined as $G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$. Let Y denote the edge-cut between A and B, $Y = \{e = (a,b) | (a \in A) \wedge (b \in B)\}$. Since C is connected, Y is non-empty. Let $f_C^A = f_C \cap A$, i.e. the portion of the walk f_C that consists of edges solely from A. Similarly, let $f_C^B = f_C \cap B$. The subgraph f_C^A is the union of some segments from the walk f_C, and similarly for f_C^B. The next lemma shows that f_C^A (f_C^B) is in fact a single connected component.

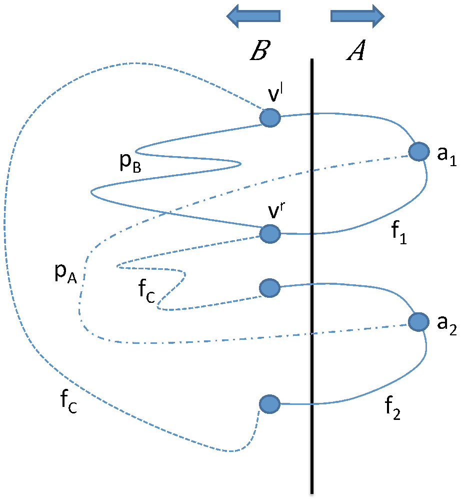

Fig. 3 Proof of Lemma 7

Lemma 7 *f_C^A and f_C^B are connected subgraphs of C.*

Proof We will prove that f_C^A is connected; a similar proof holds for f_C^B. We employ proof by contradiction. Suppose that f_C^A was disconnected, and there were two vertices $a_1, a_2 \in f_C^A$ such that there is no path between a_1 and a_2 in f_C^A. Let f_1 and f_2 denote the components of f_C^A that contain a_1 and a_2 respectively; we refer to f_1 and f_2 as "segments" since they are subsequences of the walk f_C. Let v^ℓ be the vertex that is adjacent to f_1 on f_C in the anti-clockwise direction, and let v^r be the vertex adjacent to f_1 on f_C in the clockwise direction. Clearly, both v^ℓ and v^r are in B (see Figure 3).

We show that v^ℓ and v^r must be distinct vertices. To prove this, suppose that $v^\ell = v^r$. Since v^ℓ occurs twice in the walk f_C, v^ℓ must be a cut-vertex in C separating a_1 and a_2. This implies that any path from a_1 to a_2 must pass through v^ℓ, which is not in A. Thus, there is no path between a_1 and a_2 which lies completely in A. This contradicts the fact that A is connected. Thus, v^ℓ and v^r must be distinct vertices.

Since $v^\ell, v^r \in B$, and B is connected, there exists a path p_B from v^ℓ to v^r that lies completely in B. This path must lie on or inside f_C, since f_C consists of all external edges of C. Similarly, there exists a path p_A from a_1 to a_2 in A, and this path also lies on or inside f_C. Note that the clockwise order of the vertices on f_C is a_1, v^r, a_2, v^ℓ, and we have two vertex disjoint and non-crossing paths, one (p_A) from a_1 to a_2, and the other (p_B) from v^ℓ to v^r. This is a contradiction, which completes the proof.

Let Y^{ext} denote the edges in $f_C \cap Y$, i.e. those edges in Y that are external in C.

Lemma 8 $1 \le |Y^{ext}| \le 2$.

Proof By definition of A and B, it follows that f_C has at least one vertex from each of A and B, say $a \in A$ and $b \in B$. Since f_C is connected, there is a path from a to b in f_C. This path should contain at least one edge in Y. Thus, there is at least one edge in $Y^{ext} = (f_C \cap Y)$, and hence $|Y^{ext}| \ge 1$.

From Lemma 7 it follows that f_C^A and f_C^B are both connected segments in f_C. The only edges in Y^{ext} are those edges connecting f_C^A to f_C^B, and there are no more than two such edges (only one if Y consists of a single cut-edge).

Let V_B be the nodes in B that are at the endpoint of edges in Y^{ext}. From Lemma 8, $1 \le |V_B| \le 2$. Let $p_B \in B$ be a shortest path connecting the nodes in V_B. If V_B has only one vertex, then p_B consists of just that vertex.

Lemma 9 *Let $q = v_1, v_2, \ldots, v_k$ be any path in B such that (1)p_B and q do not cross (they have no internal nodes in common), and (2)v_1 is the endpoint of some edge in Y. Then node v_k is not an external node of C.*

Proof Let V_A denote the nodes of A that are adjacent to Y^{ext}. From Lemma 8, $1 \le |V_A| \le 2$. Let p_A denote a shortest path between the nodes in V_A. The union of Y^{ext}, p_A, and p_B induce a cycle $\mathscr{C}$ in C (in case p_A and p_B are single points, $\mathscr{C}$ is a single edge). Let W denote the nodes of C that are contained within $\mathscr{C}$ (W is empty if $\mathscr{C}$ is a single edge). Finally, let $C(\mathscr{C})$ denote the subgraph of C that is induced by the union of the nodes in W and $\mathscr{C}$.

We first show that all the edges of Y are members of $C(\mathscr{C})$. Suppose for the sake of contradiction that there exists some edge $e = (u, v)$, where $e \in Y$, $u \in A$, $v \in B$, and $e \notin C(\mathscr{C})$. If $|Y^{ext}| = 1$, then it has to be $Y = Y^{ext} = \{e\}$, in other words, e is the only bridge edge between A and B. Consider now the case where $|Y^{ext}| = 2$. Suppose that $Y^{ext} = \{e_1, e_2\}$. Since A is connected, there is a path $\alpha \in A$ that connects edge e to a node in p_A; similarly,

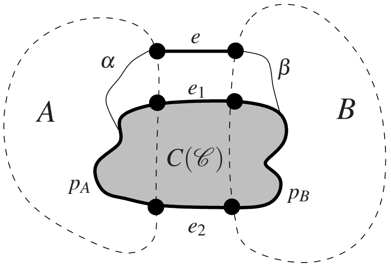

Fig. 4 This figure demonstrates the subgraphs and paths described in Lemma 9.

there is a path $\beta \in B$ that connects edge e to a node in p_B (see Figure 4). This implies that either e_1 or e_2 is not in the external face of C, a contradiction. Therefore, all the edges of Y are members of $C(\mathscr{C})$.

Since v_1 is adjacent to an edge in Y, we have that $v_1 \in C(\mathscr{C})$. Since q does not cross p_B, each node of q is a member of $C(\mathscr{C})$, that is, $q \in C(\mathscr{C})$. Let W_B denote the nodes of W that are members of B. The nodes of q are actually members of W_B, since none of the nodes of q are external in $C(\mathscr{C})$. Since the nodes of W_B are separated by the path p_B from the remaining nodes of B, in $B - p_B$, the nodes of W_B are in connected components consisting only of nodes of W_B. These connected components do not contain any external nodes of C, since W does not contain external nodes of C. Therefore, v_k will belong to such a connected component in $B - p_B$.

5.4 Algorithm Depth-Cover.

We now present Algorithm Depth-Cover, which constructs a γ-cover for a planar graph G. The radius of the cover is at most $4 \cdot \max\{2\gamma, \gamma + depth(G)\}$, and its degree is no more than 6. Note that the radius of the cover depends on the depth of the graph. The algorithm is formally described in Algorithm 4, and uses Algorithm 5 as a subroutine.

Similar to Algorithm Separator-Cover, the basic idea behind Algorithm Depth-Cover is to select appropriate shortest paths and cluster the nodes around them. This is achieved with the help of subroutine Subgraph-Clustering (Algorithm 5). An example execution is shown in Figure 5. Each time, a shortest path p is selected between external nodes. An invocation to Algorithm Shortest-Path-Cluster forms clusters I containing nodes around the shortest path. Then, the shortest path p and a neighborhood A around it are removed, which may breakup the graph into connected components. This process is repeated recursively for each connected component that contains external nodes of G, where a respective new shortest path is selected. The first invocation of Algorithm Subgraph-Clustering is with a trivial shortest path consisting of one external node v (in Figure 5.a node v_1), but as the algorithm progresses larger shortest paths are considered. The union of all the clusters from the shortest paths gives the resulting cover.

To control the degree of the cover, we choose to cluster each shortest path p with a locality parameter 2σ, and then remove the σ-neighborhood of the path, where the value of σ is specified below. The effect of this locality parameter choice is that each node participates in the clustering process of at most two paths. Since from Lemma 1 the resulting degree of a shortest path clustering is 3, the overall degree of a node is at most 6. Specifically, suppose that p is the shortest path to be clustered in a connected component H (which contains at least one external node of G). Consider a node v which is in the 2σ-neighborhood of p. Node v will be included in the set of clusters I produced from p. Let A be the set of nodes which are at distance at most σ from p. All nodes in A are removed after p is clustered. If $v \in A$, then v will be removed immediately after p is clustered, and hence v participates in only one path clustering. If $v \notin A$, then v belongs to some connected component B that results from the removal of A. If B does not contain external nodes of G then v is discarded, and thus, v has participated in only one path clustering (p's clustering) (see for example component B_2' in Figure 5.c). On the other hand, if B contains an external node of G then it will be recursively clustered (see for example component B_1 in Figure 5.b, and component B_2 in Figure 5.c). The shortest path p_B in B is chosen in such a way that v is in the σ-neighborhood of p_B. Thus, v will be removed after p_B is clustered. Hence, v participates in at most two shortest path clusterings, one from p and the other from p_B.

In order to guarantee that the resulting cover γ-satisfies every node in G we choose $\sigma = \gamma + \zeta$, where $\zeta = \max\{\gamma, depth(G)\}$. Consider as above a shortest path p in connected component H. When p is clustered to produce I then every node v in the ζ-neighborhood of A is γ-satisfied in H, since $\sigma = \gamma + \zeta$. One of the key properties (established in Lemma 10) of our algorithm is that if a node v is in the ζ-neighborhood of A then the γ-neighborhood of v is intact with respect to the original graph G. Therefore, I also γ-satisfies v in the graph G.

External nodes play an important role in the algorithm. First, external nodes are used in the formation of shortest paths. The initial shortest path in G consists of a single external node. Each time a shortest path p and its A surrounding set of nodes are removed, then in each resulting connected component B the new shortest path p_B is selected as follows. Suppose Y is an edge-cut between A and B (see Figure 5.a). Let Y^{ext} be the external edges of Y with respect to H. From Lemma 8, $1 \leq |Y^{ext}| \leq 2$. Let V_B be the set of nodes in B that are endpoints of edges in Y^{ext}; we have $1 \leq |V_B| \leq 2$. Path p_B is selected to be a shortest path in B between nodes in V_B (if V_B has only one vertex, then p_B consists of a single node). For example, in Figure 5.a $V_{B_1} = \{v_2, v_3\}$.

Second, external nodes are used to determine which connected components are to be processed recursively. As subgraphs of G are removed at each step of the algorithm, nodes which were originally external in G may be removed. A connected component which contains some external node of G can be safely recursively processed, since it is guaranteed (as shown in the analysis) that nodes already clustered in the previous step will be removed immediately in the next recursive step, enabling a node to participate in at most two path clusterings. Nodes in any connected component B that has no external nodes of G are γ-satisfied just before B is formed, and therefore, any such connected component B does not require further processing. The reasons why the nodes in B are already satisfied are: (i) every node in B is within distance at most $depth(G) \leq \zeta$ from some external node $u \in G$ which is removed; (ii) u is within distance $\sigma = \gamma + \zeta$ from a shortest path p (the one whose removed neighborhood A contains u); (iii) p is $2(\gamma + \zeta)$-satisfied; and (iv) the γ-neighborhood of every node in B is intact before A is removed (from the key property mentioned above for the nodes in the ζ-neighborhood of A).

Figure 5 depicts an example execution of Algorithm Depth-Cover with the first invocation (Figures 5.a and 5.b) and the second invocation (Figures 5.c and 5.d) of the subroutine Subgraph-Clustering. In the example we consider the special case $\zeta = \gamma = depth(G)$, and thus, $\sigma = 2\gamma$.

Algorithm Subgraph-Clustering(G, H, p, γ) is recursive, and parameters G and γ remain unchanged at each recursive invocation, while H and p change. Parameter H is a subgraph of G with at least one external node of G, and it is required to γ-satisfy all nodes in H.

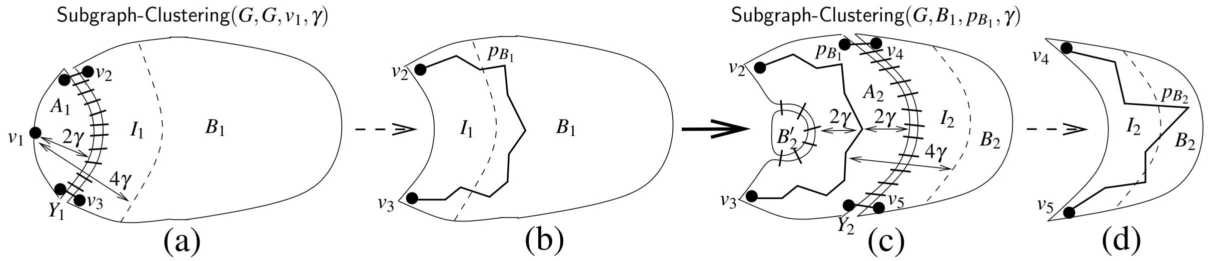

Fig. 5 Execution example of Algorithm Subgraph-Clustering.

Parameter p is a shortest path in H that will be used for clustering in the current invocation. Initially, $H = G$ and $p = v$, where v is an arbitrary external node of G.

```
Algorithm 4: Depth-Cover($G, \gamma$)
  Input: Connected planar graph $G$; locality parameter $\gamma > 0$;
  Output: $\gamma$-cover for $G$;
  Let $v$ be an external node of $G$;
  $Z \leftarrow$ Subgraph-Clustering($G, G, v, \gamma$);
  return $Z$;
```

```
Algorithm 5: Subgraph-Clustering($G, H, p, \gamma$)
  Input: Connected planar graph $G$; connected subgraph $H$ of $G$ (consisting of vertices that are still
         unsatisfied); shortest path $p \in H$ whose end nodes are external in $H$; locality parameter $\gamma > 0$;
  $\zeta \leftarrow \max\{\gamma, depth(G)\}$;   $\sigma \leftarrow \gamma + \zeta$;
  $I \leftarrow$ Shortest-Path-Cluster($H, p, 2\sigma$);
  $A \leftarrow N_\sigma(p, H)$;   $H' \leftarrow H - A$;
  $J \leftarrow \emptyset$;
  foreach connected component $B$ of $H'$ that contains at least one external node of $G$ do
      Let $Y$ be the edge-cut between $A$ and $B$ in subgraph $H$;
      Let $Y^{ext} \subseteq Y$ be the external edges of $Y$ in subgraph $H$;
      Let $V_B$ be the nodes of $B$ adjacent to the edges of $Y^{ext}$;
      Let $p_B$ be a shortest path in $B$ that connects the nodes in $V_B$;
      $J \leftarrow J \cup$ Subgraph-Clustering($G, B, p_B, \gamma$);
  end
  return $I \cup J$;
```

5.5 Analysis.

Our main result is Theorem 5, which bounds the radius and degree of the resulting covers from Algorithm Depth-Cover. It is convenient to represent the execution of Algorithm Depth-Cover as a tree T, where each vertex in T corresponds to some invocation of the subroutine Subgraph-Clustering. The root r of T corresponds to the first invocation with parameters (G, G, v, γ). Suppose, for example, that in the first invocation the removal of A creates two components H_1 and H_2 in G, for which the algorithm is invoked recursively with parameters (G, H_1, p_1, γ) and (G, H_2, p_2, γ). Then, these two invocations will correspond in T to the two children of the root. The leaf vertices correspond to subgraphs H_i that are not decomposed further, i.e. those subgraphs on which Subgraph-Clustering makes no recursive calls.

Let $\zeta = \max\{\gamma, depth(G)\}$, and $\sigma = \gamma + \zeta$, as in Algorithm Subgraph-Clustering. Suppose that vertex $w \in T$ corresponds to invocation (G, H, p, γ). We will denote by $H(w)$ the respective input graph H, and we will use a similar notation to denote the remaining parameters and variables used in this invocation; for example, $p(w)$ is the input shortest path while $A(w)$ is the respective σ-neighborhood of $p(w)$ in $H(w)$. As another example, using this notation, the resulting set of clusters is $Z = \bigcup_{w \in T} I(w)$.

We start with a key result which shows that a particular node retains the γ-neighborhood in G up to the point that it appears in the ζ-neighborhood of some A set.

Lemma 10 *For any node $v \in G$, there is a vertex $w \in T$ such that $N_\gamma(v, G) = N_\gamma(v, H(w))$ and $v \in N_\zeta(A(w), H(w))$.*

Proof By the construction of T, there is a path $s = w_1, w_2, \ldots, w_k$ in T such that: $k \geq 1$, $v \in H(w_i)$ for $1 \leq i \leq k$, $w_1 = r$ (the root of T), w_i is the parent of w_{i+1} for $1 \leq i \leq k-1$, and w_k does not have any child w' with $v \in H(w')$.

By the construction of T and s, $H(w_{i+1}) \subseteq H(w_i)$ for $1 \leq i \leq k-1$. Since $H(w_1) = H(r) = G$, $N_\gamma(v, G) = N_\gamma(v, H(w_1))$. Let k' be the largest integer i such that $N_\gamma(v, G) = N_\gamma(v, H(w_i))$, and let $s' = w_1, w_2, \ldots, w_{k'}$ be the subpath of s from the root till $w_{k'}$.

We examine two cases:

Case 1: $k' < k$
It holds that $v \in H(w_{k'})$, $v \in H(w_{k'+1})$, $N_\gamma(v, G) = N_\gamma(v, H(w_{k'}))$, and $N_\gamma(v, G) \neq N_\gamma(v, H(w_{k'+1}))$. According to Algorithm Subgraph-Clustering, v belongs to a connected component B of $H'(w_{k'})$, such that B contains an external node of G. Note that $B = H(w_{k'+1})$ and $H'(w_{k'}) = H(w_{k'}) - A(w_{k'})$. Clearly, $v \notin A(w_{k'})$, or else $k = k'$. Since the γ-neighborhood of v changes between $H(w_{k'})$ and $B = H(w_{k'+1})$, some node $u \in N_\gamma(v, H(w_{k'}))$ must be a member of $A(w_{k'})$ (note that only the nodes of $A(w_{k'})$ are removed from $H(w_{k'})$). Thus, $v \in N_\gamma(A(w_{k'}), H(w_{k'})) \subseteq N_\zeta(A(w_{k'}), H(w_{k'}))$. Therefore, $w_{k'}$ is the desired vertex of T.

Case 2: $k' = k$
In this case, it holds that $v \in H(w_k)$, no child w' of w_k has $v \in H(w')$, and $N_\gamma(v, G) = N_\gamma(v, H(w_k))$. According to Algorithm Subgraph-Clustering, there are two possible scenarios:

Case 2.1: $v \in A(w_k)$
This case trivially implies that $v \in N_\zeta(A(w_k), H(w_k))$. Thus, w_k is the desired vertex of T.

Case 2.2: $v \notin A(w_k)$
In this case, it holds that v belongs to a connected component X of $H'(w_k) = H(w_k) - A(w_k)$, such that X does not contain any external node of G. Since $depth(G) \leq \zeta$, there is a node $x \in G$ that is external in G and $x \in N_\zeta(v, G)$. Since X does not contain any external node of G, $x \notin N_\zeta(v, X)$. Therefore, $N_\zeta(v, X) \neq N_\zeta(v, G) = N_\zeta(v, H(w_k))$. Thus, the ζ-neighborhood of v changes between $H(w_k)$ and X. Hence, some node $u \in N_\zeta(v, H(w_k))$ is also a member of $A(w_k)$ (note that only the nodes of $A(w_k)$ are removed from $H(w_k)$), which implies $v \in N_\zeta(A(w_k), H(w_k))$. Therefore, w_k is the desired vertex of T.

Consequently, $w_{k'}$ is the desired vertex of T in all cases.

Lemma 11 *Z is a γ-cover for G.*

Proof From Lemma 10, for each node $v \in G$ there is a vertex $w \in T$ such that $N_\gamma(v,G) = N_\gamma(v,H(w))$ and $v \in N_\zeta(A(w),H(w))$. By Lemma 1, $p(w)$ is 2σ-satisfied by $I(w)$ in $H(w)$. Since $A(w) = N_\sigma(p(w),H(w))$, $A(w)$ is σ-satisfied by $I(w)$ in $H(w)$. Therefore, since $\sigma = \gamma + \zeta$, v is γ-satisfied by $I(w)$ in $H(w)$. Since $N_\gamma(v,G) = N_\gamma(v,H(w))$, $I(w)$ also γ-satisfies v in G. Since $Z = \bigcup_{w\in T} I(w)$, Z is a γ-cover for G.

Lemma 12 *$rad(Z) \leq 4 \cdot \max\{2\gamma, \gamma + depth(G)\}$.*

Proof We have that $Z = \bigcup_{w\in T} I(w)$, where each $I(w)$ is obtained by an invocation of Algorithm Shortest-Path-Cluster, with parameter $\beta = 2\sigma$. Therefore, by Lemma 1, for any $w \in T$, $rad(I(w)) \leq 2\beta = 4\sigma$. Since $\sigma = \gamma + \zeta = \gamma + \max\{\gamma, depth(G)\}$, we get $rad(Z) \leq 4 \cdot \max\{2\gamma, \gamma + depth(G)\}$.

Lemma 13 *$deg(Z) \leq 6$.*

Proof Consider an arbitrary node $v \in G$. We only need to show that $deg(v,Z) \leq 6$. Let $s = w_1, w_2, \ldots, w_k$ be the path in T as described in Lemma 10, i.e., s is such that: $v \in H(w_i)$ for $1 \leq i \leq k$, w_1 is the root of T, w_i is the parent of w_{i+1} for $1 \leq i \leq k-1$, and w_k does not have a child w' with $v \in H(w')$.

According to Algorithm Subgraph-Clustering, the only possible clusters that v can participate in are $I(w_1), I(w_2), \ldots, I(w_k)$. Let k^* denote the smallest index i in $\{1,\ldots,k\}$ such that $v \in I(w_i)$. We will show that $k^* \in \{k-1, k\}$. We examine two cases:

Case 1: $v \in A(w_{k^*})$
In this case, v will be removed with $A(w_{k^*})$, and therefore, v will not appear in any child of w_{k^*}. Consequently, $k^* = k$.

Case 2: $v \notin A(w_{k^*})$
In this case, let B denote the connected component of $H'(w_{k^*}) = H(w_{k^*}) - A(w_{k^*})$ that contains v. There are two subcases:

Case 2.1: *B does not contain an external node of G*
Since the algorithm does not recurse over any component of $H'(w_{k^*})$ that does not contain an external node of G, B is discarded at vertex w_{k^*}, and therefore, v will not appear in any child of w_{k^*}. Consequently, $k^* = k$.
Case 2.2: *B contains an external node of G*
In this case, we consider only the case $k^* < k$. If $k^* = k$, then we are already done. According to Algorithm Subgraph-Clustering applied at w_{k^*}, $B = H(w_{k^*+1})$. We will show that $v \in A(w_{k^*+1})$, which implies that $k^* + 1 = k$ (the reason is similar to the case where $v \in A(w_{k^*})$ above). Since $v \in I(w_{k^*})$, $v \in N_{2\sigma}(p(w_{k^*}),H(w_{k^*})) = N_\sigma(A(w_{k^*}),H(w_{k^*}))$. Thus, there is a node $u \in A(w_{k^*})$ such that $v \in N_\sigma(u,H(w_{k^*}))$. Let $g = u, x_1, x_2, \ldots, x_\ell, v$ be a shortest path between u and v in $H(w_{k^*})$. Clearly, $length(g) \leq \sigma$. Since Y is the edge-cut between $A(w_{k^*})$ and B, the path g must contain an edge of Y. Choose a node $x_y \in g$ such that $x_y \in B$, and x_y is adjacent to some edge of Y. Now, let $g' = x_y, x_{y+1}, \ldots, x_\ell, v$ be a subpath of g in B. Clearly, $length(g') \leq \sigma$ as well. Let p_B denote the shortest path chosen for component B as described in Algorithm Subgraph-Clustering.

Case 2.2.1: *p_B and g' cross (share a node)*
Then $v \in N_\sigma(p_B, B) = N_\sigma(p_B, H(w_{k^*+1}))$. Thus, $v \in A(w_{k^*+1})$, implying that v is removed at vertex w_{k^*+1}, and $k^* = k-1$.
Case 2.2.2: *p_B and g' do not cross*
By Lemma 9, in $B - p_B$, node v belongs to a connected component B' that has no external nodes of $H(w_{k^*})$. (Here, $H(w_{k^*})$ plays the role of C in the statement of Lemma 9, and the lemma can be applied because both $A(w_{k^*})$ and B hold external nodes of $H(w_{k^*})$, since $A(w_{k^*})$ contains a shortest path between external nodes of $H(w_{k^*})$ and B contains an external node of G which must be also an external node of $H(w_{k^*})$.) Since $H(w_{k^*})$ is a subgraph of G, Observation 1 implies that B' has no external nodes of G either. Thus, B' is discarded at the recursive invocation of the algorithm that corresponds to the vertex w_{k^*+1}. Consequently, $w_k = w_{k^*+1}$, which implies that $k^* = k-1$.

Consequently, $k^* \in \{k-1, k\}$. Thus, the only clusters that v could possibly belong to are $I(w_{k-1})$ and $I(w_k)$. Since for each $x \in T$, $I(x)$ is the result of an invocation of Algorithm Shortest-Path-Cluster, from Lemma 1, $deg(I(x)) \leq 3$. Therefore, $deg(v,Z) \leq deg(I(w_{k-1})) + deg(I(w_k)) \leq 6$.

It is easy to verify that Algorithm Depth-Cover computes the cover Z in polynomial time with respect to the size of G. Therefore, the main result in this section follows from Lemmas 11, 12, and 13.

Theorem 5 *For any connected planar graph G and $\gamma > 0$, Algorithm* Depth-Cover *returns in polynomial time a γ-cover Z with $rad(Z) \leq 4 \cdot \max\{2\gamma, \gamma + depth(G)\}$ and $deg(Z) \leq 6$.*

6 Conclusion.

We presented new algorithms for the construction of near-optimal covers for planar graphs, and efficient covers for other H-minor free graphs. Our results are based on the novel idea of finding appropriate shortest paths in the graph and clustering their neighborhood. After removing an area around the shortest paths, we solve the clustering problem recursively on the connected components of the residual graph. Our work has immediate implications on the efficiency of data structures used to solve fundamental distributed problems such as compact routing, distributed directories, synchronizers, and universal TSP. We get even better results for all these problems when the planar graphs have small depth, as in the outerplanar case.

Our planar graph construction can provide alternative bounds for degree and radius when we adjust the size of the bands in the zones. If we decrease the depth of a band, the radius of the cover will decrease, but the overlap between the zones will increase, resulting in a larger degree for the cover. An open problem is to find better tradeoffs in the radius and degree. Another open problem is to find better bounds for other significant special classes of graphs.

References

1. Ittai Abraham and Cyril Gavoille. Object location using path separators. *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, pages 188–197, 2006.
2. Ittai Abraham, Cyril Gavoille, Andrew Goldberg, and Dahlia Malkhi. Routing in networks with low doubling dimension. *Proc. International Conference on Distributed Computing Systems (ICDCS)*, 2006.

3. Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Compact routing for graphs excluding a fixed minor. *Proc. International Conference on Distributed Computing (DISC)*, pages 442–456, 2005.
4. Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Transactions on Algorithms*, 4(3):37:1–37:12, 2008.
5. Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, and Udi Wieder. Strong-diameter decompositions of minor free graphs. *Proc. ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 16–24, 2007.
6. Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Udi Wieder. Strong-Diameter Decompositions of Minor Free Graphs. *Theory of Computing Systems*, 47(4): 837–855, 2010
7. K. Alzoubi, X. Li, Y. Wang, P. Wan, and O. Frieder. Geometric Spanners for Wireless Ad Hoc Networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):408–421, 2003.
8. M. Arias, L. Cowen, K. Laing, R. Rajaraman, and O. Taka. Compact routing with name independence. *SIAM Journal on Discrete Mathematics*, 20(3):705–726, 2006.
9. Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw-Hill, 1st edition, 1998.
10. Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985.
11. Baruch Awerbuch, Shay Kutten, and David Peleg. On buffer-economical store-and-forward deadlock prevention. *IEEE Transactions on Communications*, 42(11):2934–2937, 1994.
12. Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 514–522, 1990.
13. Baruch Awerbuch and David Peleg. Sparse partitions (extended abstract). *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 503–513, 1990.
14. Baruch Awerbuch and David Peleg. Concurrent Online tracking of mobile users. *Proc. ACM SIGCOMM Symposium on Communication Architectures and Protocols*, pages 221–233 1991.
15. Baruch Awerbuch and David Peleg. Online tracking of mobile users. *Journal of the ACM*, 42(5):1021–1058, 1995.
16. Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.
17. Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Improved sparse covers for graphs excluding a fixed minor. Technical Report TR 06-16, Department of Computer Science, Rensselaer Polytechnic Institute, November 2006.
18. Greg N. Frederickson and Ravi Janardan. Efficient message routing in planar networks. *SIAM Journal on Computing*, 18(4):843–857, 1989.
19. Cyril Gavoille. Routing in distributed networks: overview and open problems. *SIGACT News*, 32(1):36–52, 2001.
20. Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Distributed Computing*, 16(2-3):111–120, 2003.
21. Anupam Gupta, Mohammad T. Hajiaghayi and Harald Räcke. Oblivious network design. *Proc. ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 970–979, 2006.
22. Mohammad T. Hajiaghayi, Robert Kleinberg, and Tom Leighton. Improved lower and upper bounds for universal TSP in planar metrics. *Proc. ACM-SIAM Symposium on Discrete algorithm (SODA)*, pages 649-658, 2006.
23. Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for TSP, Steiner tree, and set cover. *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 386-395, 2005.
24. Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 682–690, 1993.
25. Goran Konjevod, Andréa W. Richa, and Donglin Xia. Optimal-stretch name-independent compact routing in doubling metrics. *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, pages 198–207, 2006.
26. Goran Konjevod, Andréa W. Richa, and Donglin Xia. Optimal scale-free compact routing schemes in networks of low doubling dimension. *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 939–948, 2007.
27. Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., 1996.
28. David Peleg. Distance-dependent distributed directories. *Information and Computation*, 103(2):270–298, 1993.
29. David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, 2000.
30. David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, 1989.

31. Neil Robertson and Paul D. Seymour. Graph minors. V. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41:92–114, 1986.
32. Neil Robertson and Paul D. Seymour. Graph minors. XVI. excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, 2003.
33. Lior Shabtay and Adrian Segall. Low complexity network synchronization. *Proc. International Workshop on Distributed Algorithms (WDAG)*, pages 223–237, 1994.
34. Srivathsan Srinivasagopalan, Costas Busch, and S. Sitharama Iyengar. Oblivious buy-at-bulk in planar graphs. *Proc. International Conference on Algorithms and Computation (WALCOM)*, pages 33-44, 2011.
35. Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004.
36. Mikkel Thorup and Uri Zwick. Compact routing schemes. *Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2001.