

Improved Sparse Covers for Graphs Excluding a Fixed Minor ^{*}

Costas Busch[†]

Ryan LaFortune[‡]

Srikanta Tirthapura[§]

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 no more than $24\gamma - 8$ and degree (maximum cluster overlap) no more than 18. For every n node graph that excludes a fixed minor, 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 of $O(\gamma)$, it was required to have degree polynomial in n . Since sparse covers have many applications in distributed computing, including compact routing, distributed directories and synchronizers, our improved cover construction results in improved algorithms for all these problems, for the class of minor-free graphs.

1 Introduction

A *cover* Z of a graph G is a set of connected components 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 in Z that contains the entire γ -neighborhood of v . Two locality metrics characterize the cover: the *radius*, denoted $rad(Z)$, which is the maximum radius of any of its clusters,¹ and 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 the construction of distance-dependent distributed directories [21, 22, 12], compact routing schemes [9, 22, 23, 26, 4, 3], network synchronizers [10, 7, 20, 22], and transformers for certain classes of distributed algorithms [9]. 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 [11] present an algorithm for constructing a sparse cover on 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 $O(\log n)$. This is the best known result for general graphs. There is an inherent trade-off between

^{*}This work is supported by NSF grants CNS-0520102 and CNS-0520009.

[†]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA, buschc@cs.rpi.edu

[‡]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA, laforr@cs.rpi.edu

[§]Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA, snt@iastate.edu

¹The radius of a cluster $C \in Z$ is the minimum *eccentricity* of any vertex in C , where the eccentricity of a vertex $v \in C$ is the maximum distance from v to any node in C .

the radius and the degree of a cover. It is known ([22, Theorem 16.2.4]) that for every $k \geq 3$, there exist graphs and values of γ (i.e. $\gamma = 1$) 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 trade-off 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 or $K_{3,3}$, *series-parallel graphs*, which exclude K_4 , and *trees*, which exclude K_3 .

1.1 Contributions

We give improved bounds for planar graphs and other graphs excluding fixed minors:

1. For any planar graph G , we present an algorithm for computing a sparse cover Z with $rad(Z) \leq 24\gamma - 8$ and $deg(Z) \leq 18$. This cover is optimal (modulo constant factors) with respect to both the degree and the radius. To our knowledge, this is the first optimal construction for planar graphs.
2. For any graph G that excludes a fixed minor 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 the excluded minor H .

The algorithms run in polynomial time with respect to G . For the class of minor free graphs, our construction improves upon the previous work of Awerbuch and Peleg [11] by providing a smaller radius. For planar graphs, our construction simultaneously improves both the degree and the radius.

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 constant c) of every path $p \in P$ using shortest-path clustering, and we then remove P together with its $c'\gamma$ -neighborhood from G , for some $c' < c$. This gives residual connected components G'_1, \dots, G'_r which contain the remaining unclustered nodes as a subset. We apply the same procedure recursively to each G'_i by identifying appropriate shortest paths in them. The algorithm terminates when there are no remaining nodes.

For 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 . 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 [11] 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 trade-off is reached between the radius and the degree. 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

Name-Independent Compact Routing. Consider a distributed system where nodes have arbitrary identifiers. A *routing scheme* is a method which 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 $1, \dots, n$. 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* of a routing scheme is the number of bits (per node) used to store the routing table. A routing scheme is *compact* if its stretch as well as memory overhead are “small”.

There is a trade-off 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 trade-offs between the stretch and memory overhead of routing, including [23, 26, 4, 3, 6].

Sparse covers can be used to provide efficient name-independent routing schemes (for example, see [7]). A hierarchy of *regional* routing schemes is created based on a hierarchy of covers $Z_1, Z_2, \dots, Z_\delta$, 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. 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 [11], 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 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 fixed minor-free graphs, Abraham, Gavoille and Malkhi [3] present name-independent compact routing schemes with $O(1)$ stretch and $\tilde{O}(1)$ maximum space per node. However, their paper does not provide the explicit power of $\log n$ inside the \tilde{O} , hence, we cannot directly compare our results with those of [3]. It is also noted in [3] that it is an open problem to construct efficient sparse covers for planar graphs with $O(\gamma)$ radius and $O(1)$ degree.

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 [25] gives a *labeled* routing scheme for planar graphs with stretch $(1 + \epsilon)$ and memory overhead of $O((1/\epsilon) \log^2 n)$ maximum bits per node. Name-independent routing scheme is clearly less restrictive to the user than labeled routing, and hence a harder problem.

Directories for Mobile Objects. A directory is a basic service in a distributed system which, given an object 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 trade-off between the cost of implementing the *find* and *move* operations. 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 [13]).

Awerbuch and Peleg[13, 22] 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 $Stretch_{move} = O(\log^2 n)$ ([13, Corollary 5.4.8])²

Our construction of sparse covers yields improved directories for mobile objects for planar and 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, 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 the previously known directories.

Synchronizers. Many distributed algorithms are designed assuming a synchronous model where the processors execute and communicate in time synchronized rounds [20, 7]. 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 which enable the execution of synchronized algorithms in asynchronous systems [8, 22, 20, 7]. A synchronizer uses logical rounds to simulate the time rounds of the synchronous algorithm.

One of the most efficient synchronizers is called ZETA [24]. 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 better time to simulate a round. For planar graphs, our covers give a synchronizer with $O(1)$ time and average messages per node. For fixed minor-free graphs, the time with our covers is $O(1)$ and uses $O(\log n)$ messages per node on average.

An initial exposition of our work appears as a technical report [15].

1.3 Related Work

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

For planar graphs, our algorithm yields a much better tradeoff than [5] since we give a radius of no more than $24\gamma - 6$, and a degree of no more than 18, while their cover (by using $r = 3$, since a planar graph must exclude $K_{3,3}$) gives a diameter of 64γ (which translates to a radius of at least 32γ) and the degree of the cover is 840 (this can be derived from the proof of Theorem 1.2 in page 6 of [5]).

Klein, Plotkin and Rao [19], obtain sparse covers for minor-free graphs with degree $O(1)$ but with a *weak diameter* $O(\gamma)$ where the $O(\gamma)$ length 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, 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 minor-free graphs can have large doubling dimensions, this does not yield efficient sparse covers for these graphs.

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

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. We then give in Section 4 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.

2 Definitions and Preliminaries

Graph Basics. All the graphs we consider in this paper are weighted. Consider a graph $G = (V, E, \omega)$, where V is the set of nodes, E is the set of edges, and ω is a weight function $E \rightarrow R^+$ that assigns a weight $\omega(e) > 0$, to every edge $e \in E$. For simplicity, we will also write $G = (V, E)$. 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 $q = v_1, v_2, \dots, v_k$ where nodes may be repeated. The length of q is defined as: $length(q) = \sum_{i=0}^{k-1} \omega(v_i, v_{i+1})$. 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, \dots, 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 j -neighborhood of a node v in G is $N_j(v, G) = \{w \in V | dist_G(v, w) \leq j\}$. For $V' \subseteq V$, the j -neighborhood of V' in G is $N_j(V', G) = \{N_j(v, G) | v \in V'\}$. 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, $rad(G) = \infty$.

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 use C to refer to $G(C)$. Let $Z = \{C_1, C_2, \dots, C_k\}$ 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 the nodes in X is taken with respect to G).

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) $S = P_1 \cup P_2 \cup \dots \cup P_\ell$, where for each $1 \leq i \leq \ell$ subgraph P_i is the union of k_i shortest paths in $G - \bigcup_{1 \leq j < i} P_j$, (ii) $\sum_i k_i \leq k$, 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 mesh is 1-separable by taking S to be the middle row path. Trees are also 1-separable by taking S to be the *center node* whose branching subtrees have at most $n/2$ nodes. Thorup [25] shows how to compute in polynomial time a 3-path separator for planar graphs.

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 [25] for the class of minor-free graphs:

Theorem 2.1 (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.*

We note that in Theorem 2.1, the parameter k is exponential in the size of the minor. Some interesting classes of H -minor-free graphs are: trees, which exclude K_3 , outerplanar graphs, which exclude K_4 or $K_{2,3}$, series-parallel graphs, which exclude K_4 , and planar graphs, which exclude K_5 or $K_{3,3}$.

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 1 (Shortest-Path-Cluster) contains the details of the construction of R . Lemma 3.1 establishes the correctness of Algorithm 1.

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 β -satisfies p ;

Suppose $p = v_1, v_2, \dots, v_\ell$;

// partition p into subpaths p_1, p_2, \dots, p_s of length at most β

$i \leftarrow 1; j \leftarrow 1$;

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, \dots, 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 ;

Lemma 3.1 *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 p_i , there is a path of length at most 2β from v_i to u . Thus, $\text{rad}(R) \leq 2\beta$.

For property *iii*, suppose for the sake of contradiction that $\deg(R) \geq 4$. 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, $\text{dist}_G(v_j, v_k) > \beta$, since otherwise, v_k would have been included in p_j . Similarly, $\text{dist}_G(v_k, v_l) > \beta$. Since $\text{dist}_G(v_i, v_l) > \text{dist}_G(v_j, v_k) + \text{dist}_G(v_k, v_l)$, it follows that $\text{dist}_G(v_i, v_l) > 2\beta$, a contradiction. Thus, $\deg(R) \leq 3$. ■

4 Sparse Cover for k -Path Separable Graphs

We now present Algorithm 2 (Separator-Cover), which returns a cover with a small radius and degree for any graph that has a k -path separator. Theorem 4.1 establishes the correctness and the properties of Algorithm Separator-Cover, and uses Lemma 4.1, which gives some useful properties about clusters. The proofs have been moved to the appendix due to space constraints, and proof sketches are provided here.

Algorithm 2: Separator-Cover(G, γ)

Input: Connected graph G that is k -path separable; locality parameter $\gamma > 0$;

Output: γ -cover for G ;

// base case

if G consists of a single vertex v **then**

$Z \leftarrow \{v\}$;

return Z ;

end

// main case

Let $S = P_1 \cup P_2 \cup \dots \cup P_l$ be a k -path separator of G ;

for $i = 1$ to l **do**

foreach $p \in P_i$ **do**

$A_i \leftarrow \text{Shortest-Path-Cluster}(G - \bigcup_{1 \leq j < i} P_j, p, 2\gamma)$;

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, \dots, G'_r denote the connected components of G' ;

$B \leftarrow \bigcup_{1 \leq i \leq r} \text{Separator-Cover}(G'_i, \gamma)$;

$Z \leftarrow A \cup B$;

return Z ;

Lemma 4.1 *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 .*

Sketch of proof: The set of clusters C γ -satisfies not only W but also $N_\gamma(W, G)$ in G . For any vertex $v \notin N_\gamma(W, G)$, since $N_\gamma(v, G) \cap W = \emptyset$, if a set of clusters γ -satisfies v in $G - W$, it also γ -satisfies v in G . Hence $C \cup D$ is a γ -cover for G . ■

Theorem 4.1 *For any connected k -path separable graph G with n nodes, and locality parameter $\gamma > 0$, Algorithm `Separator-Cover`(G, γ) returns set Z which has the following properties: (i) Z is a γ -cover for G ; (ii) $\text{rad}(Z) \leq 4\gamma$; (iii) $\text{deg}(Z) \leq 3k(\lg n + 1)$.*

Sketch of proof: For property *i*, we note that the algorithm repeatedly constructs a 2γ cover of a set of shortest paths P in graph G and recursively clusters $G - P$. From Lemma 4.1, a γ -cover of $(G - P)$, when combined with a 2γ -cover of P in G yields a γ -cover of G . The formal proof uses induction on number of vertices in G .

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 3.1, the radius of each cluster is at most $2\beta = 4\gamma$. Thus, $\text{rad}(Z) \leq 4\gamma$.

For property *iii*, we visualize the recursive invocations of the algorithm as a tree T , where each node of T is associated with an input graph on an invocation of the recursive algorithm. Due to the balancing property of a path separator, the depth of the tree is no more than $\lg n$. Each vertex $v \in G$ appears in the graphs corresponding to all nodes in T that lie on a single path starting from the root; the number of tree nodes on this path is no more than $(\lg n + 1)$. The number of clusters that v appears in is bounded by $(\lg n + 1)$ times $3k$, the factor of $3k$ arising due to the fact that calling `Shortest-Path-Cluster` on each of the k shortest paths in the separator for any node in T can lead to v appearing in no more than 3 clusters (due to Lemma 3.1). ■

Upon combining Theorem 4.1 with Theorem 2.1, we get the following.

Theorem 4.2 *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) $\text{rad}(Z) \leq 4\gamma$; (iii) $\text{deg}(Z) \leq 3k(\lg n + 1)$; where $k = k(H)$ is a parameter that depends on the size of the excluded minor H .*

5 Sparse Cover for Planar Graphs

Since every planar graph is 3-path separable [25], Theorem 4.1 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 $O(\gamma)$ and degree $O(1)$, both of which are optimal up to constant factors.

Consider a connected planar graph $G = (V, E)$. If G is disconnected, then it can be handled by clustering each connected component separately. Consider also an embedding of G in the Euclidean plane where no two edges cross each other. In the following discussion, we use G to refer to the planar embedding of the graph. The edges of G divide the Euclidean plane into closed geometric regions called *faces*. The *external face* is a special face that surrounds the whole graph; the other faces are *internal*. A node may belong to multiple faces, while an edge to at most two faces. A node (edge) that belongs to the external face will be called external. For any node $v \in G$ we denote by $\text{depth}(v, G)$ the shortest distance between v and an external node of G . We also define $\text{depth}(G) = \max_{v \in V} \text{depth}(v, G)$; note that $\text{depth}(G) \geq 0$. The vertices of G are divided into *layers* as follows. Layer L_0 consists of all external nodes, and layer L_i consists of all nodes whose depth is i . Note also that any subgraph of G is planar.

At a high level, our algorithm for a sparse cover of a planar graph G breaks up a graph into many overlapping planar subgraphs called *zones* such that (1) the depth of each zone is not much greater than γ ,

and (2) clustering each zone separately is sufficient to cluster the whole graph. This way, we can focus on clustering only planar graphs whose depth is not much more than γ . Thus, our algorithm is divided into two parts: (i) Algorithm **Depth-Cover**, which clusters graph G such that $\text{depth}(G) \leq \gamma$ and (ii) Algorithm **Planar-Cover**, which clusters arbitrary planar graphs using **Depth-Cover** as a subroutine. We now proceed with the description of Algorithms **Depth-Cover** and **Planar-Cover** in Sections 5.1 and 5.2 respectively.

5.1 Algorithm Depth-Cover

We now present Algorithm **Depth-Cover** which constructs a γ -cover for planar graph G for the case $\gamma \geq \max(\text{depth}(G), 1)$. The resulting cover has radius no more than 8γ and degree no more than 6. We describe the intuition here, and the algorithm is formally described in Algorithm **Depth-Cover** which uses subroutine **Subgraph-Clustering** to do most of the work. The proofs are in the appendix.

Depth-Cover allows us to focus on satisfying only the external nodes in G . Since $\text{depth}(G) \leq \gamma$, if a set of clusters S 2γ -satisfies every external node in the graph, then S is a γ -cover for G . The reason is that every internal node u is within a distance of γ from some external node v , and the cluster that contains the 2γ -neighborhood of v will also contain the γ -neighborhood of u , and will γ -satisfy u . *We now focus on constructing a set of clusters that 2γ -satisfies each external node of G .*

The algorithm begins by selecting an external node of G , which is also trivially a shortest path p in G . Through shortest-path clustering, it constructs a set of clusters I which 4γ -satisfies p in G , and deletes A , the 2γ -neighborhood of p in G . Let the resulting connected components in $G - A$ be $\mathcal{B} = B_1, B_2, \dots$. By Lemma 4.1, the union of 2γ covers of B_i s with I results in a 2γ -cover of G . Further, since we are only interested in 2γ -satisfying every external node of G , we need not further consider any component in \mathcal{B} that does not contain an external node of G . Thus, the algorithm proceeds by recursively clustering every component in \mathcal{B} that contains at least one external node of G .

Let $B \in \mathcal{B}$ be a component with at least one external node of G . Shortest path p_B is selected as follows. Suppose Y is an edge-cut between A and B (see Figure 1.a), and Y' are the external edges of Y with respect to G . It can be shown that $1 \leq |Y'| \leq 2$ (the proof appears in the appendix). Let V_B be the set of nodes in B that are endpoints of edges in Y' ; 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 1.a $V_{B_1} = \{v_2, v_3\}$.

As we show in the analysis, for every node $v \in I$, $v \notin A$, either: (i) v appears in the 2γ -neighborhood of p_B for one of the connected components $B = B_i$, or (ii) v is in a connected component B' that does not contain any external nodes of G (for example, see component B'_2 in Figure 1.c). In either case, node v will be removed in next recursive call, which deletes the 2γ -neighborhood of p_B . Thus, v participates in at most two shortest-path clusterings (of p and p_B) and it is satisfied by at least one of these two clusterings. Since each instance of shortest-path clustering contributed at most 3 to the degree of v , the total degree of v in the cover is bounded by 6.

It is useful to compare the algorithm for clustering a planar graph with shortest path clustering using path separators, as in Section 4. When separators are used, the graph is decomposed into small pieces upon the removal of the separator (which is a set of shortest paths), and the depth of this recursion is bounded by $\lg n$. However, a vertex of the graph maybe be involved in clusters due to $\lg n$ such separators. In the case of planar graph, the resulting components B_i are not necessarily much smaller than G , but the shortest paths are chosen so that the resulting clusters have little overlap.

Figure 1 depicts an example execution of Algorithm **Depth-Cover** with the first invocation (Figures 1.a and 1.b) and second invocation (Figures 1.c and 1.d) of the algorithm.

In algorithm **Subgraph-Clustering**(G, H, p, γ), parameters G and γ remain unchanged at each recursive invocation, while H and p change. Parameter H is the subgraph of G with at least one external node of

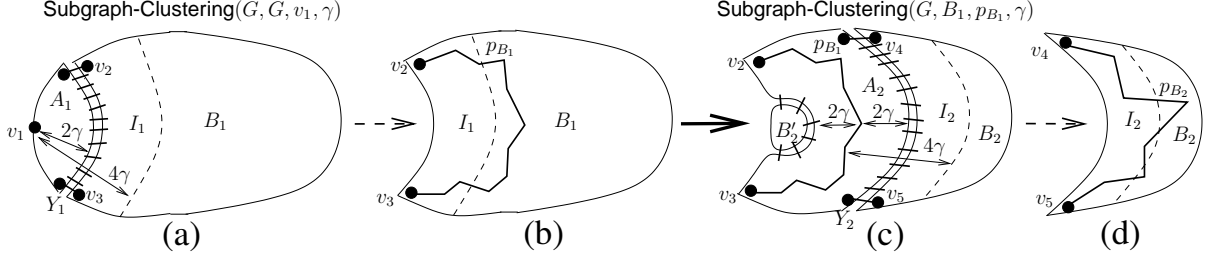


Figure 1: Execution example of Algorithm Subgraph-Clustering

G , and it is required to 2γ -satisfy all nodes in H that are external nodes of G . Parameter p is a shortest path in H which will be used in clustering in the current invocation. Initially, $H = G$ and $p = v_1$, where v_1 is an arbitrary external node of G .

Algorithm 3: Depth-Cover(G, γ)

Input: Connected planar graph G ; parameter $\gamma \geq \max(\text{depth}(G), 1)$;

Output: A γ -cover for G ;

Let v be an external node of G ;

$Z \leftarrow \text{Subgraph-Clustering}(G, G, v, \gamma)$;

return Z ;

Lemma 5.1 For any connected planar graph G and $\gamma \geq \max(\text{depth}(G), 1)$, Algorithm Depth-Cover returns in polynomial time a γ -cover Z with $\text{rad}(Z) \leq 8\gamma$ and $\text{deg}(Z) \leq 6$.

5.2 General Planar Cover

We now describe the main Algorithm Planar-Cover which given planar graph G , constructs a γ -cover with radius $O(\gamma)$ and degree $O(1)$, for any $\gamma \geq 1$. In the algorithm we do the following. If $\gamma \geq \text{depth}(G)$, then we invoke Algorithm Depth-Cover(G, γ). However, if $\gamma < \text{depth}(G)$, we first divide G into zones of layers, and then cluster each zone with Algorithm Depth-Cover. The union of the zone clusters gives the resulting cover for G . (The detailed algorithm appears in the appendix as Algorithm 5.)

We now describe how to construct the zones. Suppose that $\gamma < \text{depth}(G)$. Let $\kappa = \lceil (\text{depth}(G)+1)/\gamma \rceil$. We first divide the graph into bands of γ layers, $W_j = \bigcup_{(j-1)\gamma \leq i \leq j\gamma-1} L_i$, for $1 \leq j < \kappa$, where the last band has at most γ layers $W_\kappa = \bigcup_{(\kappa-1)\gamma \leq i \leq \text{depth}(G)} L_i$. The main goal is to γ -satisfy the nodes in each band W_i . However, in G the γ -neighborhoods of the nodes in W_i may appear in the adjacent bands W_{i-1} and W_{i+1} . For this reason we form the 3γ -zone S_i , consisting of bands W_{i-1} , W_i , and W_{i+1} (in particular, $S_i = G(W_{i-1} \cup W_i \cup W_{i+1})$, where $W_0 = W_{\kappa+1} = \emptyset$). S_i contains the whole γ -neighborhood of W_i .

In this way, we have reduced the problem of satisfying band W_i to the problem of producing a cover for zone S_i , which can be solved with Algorithm Depth-Cover. Since zone S_i is a planar graph consisting of (at most) 3γ layers, $\text{depth}(S_i) \leq 3\gamma - 1$. We invoke Algorithm Depth-Cover($S_i, 3\gamma - 1$) with locality parameter $3\gamma - 1$, since in Algorithm Depth-Cover the locality parameter has to be at least as much as the depth of the input graph. The resulting cover for G is the union of all the covers for the zones.

Using Lemma 5.1, and the observation that every node participates in at most three zones, we obtain the main result for planar graphs.

Algorithm 4: Subgraph-Clustering(G, H, p, γ)

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 ; parameter $\gamma \geq \max(\text{depth}(G), 1)$;

$I \leftarrow \text{Shortest-Path-Cluster}(H, p, 4\gamma)$;

$A \leftarrow N_{2\gamma}(p, H)$; $H' \leftarrow H - A$;

$J \leftarrow \emptyset$;

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 $Y' \subseteq Y$ be the external edges of Y in subgraph H ;

Let V_B be the nodes of B adjacent to the edges of Y' ;

Let p_B be a shortest path in B which connects all the nodes in V_B ;

$J \leftarrow J \cup \text{Subgraph-Clustering}(G, B, p_B, \gamma)$;

end

return $I \cup J$;

Theorem 5.1 For any connected planar graph G and parameter $\gamma \geq 1$, Algorithm Planar-Cover returns in polynomial time a γ -cover Z with $\text{rad}(Z) \leq 24\gamma - 8$ and $\text{deg}(Z) \leq 18$.

References

- [1] Ittai Abraham and Cyril Gavoille. Object location using path separators. In *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. In *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. In *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. In *SPAA*, pages 20–24, 2004.
- [5] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, and Udi Wieder. Strongly-bounded sparse decompositions of minor free graphs. Technical Report MSR-TR-2006-192, Microsoft Research, December 2006.
- [6] M. Arias, L. Cowen, K. Laing, R. Rajaraman, and O. Taka. Compact routing with name independence. In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, pages 184–192, 2003.
- [7] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw-Hill, 1st edition, 1998.
- [8] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4), 1985.
- [9] Baruch Awerbuch, Shay Kutten, and David Peleg. On buffer-economical store-and-forward deadlock prevention. In *INFOCOM*, pages 410–414, 1991.

- [10] Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 514–522, 1990.
- [11] Baruch Awerbuch and David Peleg. Sparse partitions (extended abstract). In *IEEE Symposium on Foundations of Computer Science*, pages 503–513, 1990.
- [12] Baruch Awerbuch and David Peleg. Online tracking of mobile users. In *Proc. ACM SIGCOMM Symposium on Communication Architectures and Protocols*, 1991.
- [13] Baruch Awerbuch and David Peleg. Online tracking of mobile users. *Journal of the ACM*, 42(5):1021–1058, 1995.
- [14] Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.
- [15] Costas Busch, Ryan LaFortune, and Srikanta Tirathapura. Improved sparse covers for graphs excluding a fixed minor. Technical Report TR 06-16, Department of Computer Science, Rensselaer Polytechnic Institute, November 2006.
- [16] Greg N. Frederickson and Ravi Janardan. Efficient message routing in planar networks. *SIAM Journal on Computing*, 18(4):843–857, 1989.
- [17] Cyril Gavoille. Routing in distributed networks: overview and open problems. *SIGACT News*, 32(1):36–52, 2001.
- [18] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Distributed Computing*, 16(2-3):111–120, 2003.
- [19] Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proc. 25th annual ACM Symposium on Theory of computing (STOC)*, pages 682–690, 1993.
- [20] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., 1996.
- [21] David Peleg. Distance-dependent distributed directories. *Information and Computation*, 103(2), 1993.
- [22] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
- [23] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3), 1989.
- [24] Lior Shabtay and Adrian Segall. Low complexity network synchronization. In *WDAG '94: Proceedings of the 8th International Workshop on Distributed Algorithms*, pages 223–237, London, UK, 1994. Springer-Verlag.
- [25] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004.
- [26] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2001.

A Proofs of Section 4

Proof of Lemma 4.1 Let C be a set of clusters that 2γ -satisfies a set of nodes W in 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 . ■

Proof of Theorem 4.1 For any connected k -path separable graph G with n nodes, and locality parameter $\gamma > 0$, Algorithm **Separator-Cover**(G, γ) returns set Z which has 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 *i*, 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, the algorithm is clearly correct. For the inductive case, suppose that for every k -separable graph with less than n vertices, the algorithm returns a γ -cover for the graph. Let G be a k -separable graph with n vertices.

The last part of the algorithm recursively calls **Separator-Cover** on every connected component in G' . Since the number of vertices of G' is less than n , the number of vertices in each G'_i is less than n . By the inductive assumption, for each $i = 1, 2, \dots, r$, **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 $i = 1, 2, \dots, l + 1$, 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 $i = \nu$, i.e. $A_\nu \cup A_{\nu+1} \cup \dots \cup A_l \cup B$ is a γ -cover for G_ν . Consider $G_{\nu-1} = G_\nu \cup P_{\nu-1}$. From the correctness of Algorithm **Shortest-Path-Cluster** (Lemma 3.1), we have that $A_{\nu-1}$ 2γ -satisfies $P_{\nu-1}$ in $G_{\nu-1}$. Since $A_\nu \cup A_{\nu+1} \cup \dots \cup A_l \cup B$ is a γ -cover for $G_\nu - P_{\nu-1}$, using Lemma 4.1 we have $A_{\nu-1} \cup A_\nu \cup \dots \cup A_l \cup B$ is a γ -cover for $G_{\nu-1}$, 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 *ii*, we note that each cluster is obtained from an invocation of Algorithm **Shortest-Path-Cluster** with input argument $\beta = 2\gamma$. From Lemma 3.1, the radius of each cluster is at most $2\beta = 4\gamma$. Thus, $rad(Z) \leq 4\gamma$.

For property *iii*, we visualize the recursive invocations of the algorithm as a tree T , where each node is associated with an input graph on an invocation of the recursive algorithm. For each node $v \in T$, let $G(v)$ denote the associated input graph and $N(v)$ denote the number of vertices in $G(v)$. Let r denote the root, thus $G(r) = G$. Clearly, for each vertex $v \in T$, $G(v)$ is a connected subgraph in G , and the leaves represent components that require no further recursive calls. The depth of any node in T is defined as the distance from the root. The depth of the tree is defined as the maximum depth of any node. For any node $v \in T$, by the property of the path separator, we have for each child v' of v , $N(v') \leq N(v)/2$. Since $N(r) = n$, any node at a depth of i has no more than $n/2^i$ vertices. Since every leaf has at least 1 vertex, the depth of the tree is no more than $\lg n$.

Consider any node $u \in G$. Suppose u belongs to $G(v)$ for some node v in T . At v , clusters are formed by calling **Shortest-Path-Cluster** no more than k times. From Lemma 3.1, u appears in no more than 3 clusters returned by each call of **Shortest-Path-Cluster**. Thus, due to all clusters formed at any node v , u appears in no more than $3k$ clusters. Further, if v_1, v_2, \dots, v_x are the children of v , it is clear that $G(v_1), G(v_2), \dots, G(v_x)$ are all disjoint from each other. Thus, u can belong to at most one component among $G(v_1), G(v_2), \dots, G(v_x)$. Since the depth of T is no more than $\lg n$, node u can belong to $G(v)$ for no more than $\lg n + 1$ nodes $v \in T$. Thus, u can belong to at most $3k(\lg n + 1)$ clusters in total, implying that $\deg(Z) \leq 3k(\lg n + 1)$. ■

B Proofs of Section 5

B.1 Basic Results for Planar Graphs

Here we prove some basic properties of planar graphs which will be used in the correctness and performance analysis of our algorithms for planar graphs.

For any planar graph G , it holds that the subgraph consisting of the edges of the face is a connected. This observation also holds for the subgraph induced by the edges of the external face. The *intersection* of any two graphs G_1 and G_2 is denoted $G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$. The following lemma can be easily verified as a property of all planar graphs.

Lemma B.1 *Let G' be a subgraph of a planar graph G . If $v \in G \cap G'$, and v is external in G , then v is external in G' too.*

Consider now a connected planar graph C that consists of two connected subgraphs A and B which are node-disjoint, and a set of edges Y which is an edge-cut between A and B (the removal of Y partitions C into A and B). Further, each of A and B contains at least one node external to C . Let Y' denote the edges of Y that are external in C .

Lemma B.2 *For any two nodes $u, v \in A \cap C$ which are external in C , there exists a walk $w = u, x_1, x_2, \dots, x_k, v$, with $k \geq 0$, such that $x_i \in A$, and each edge of the walk is external in C .*

Proof: Suppose for the sake of contradiction that there exists two nodes $u, v \in A \cap C$ that are external in C , such that there does not exist a walk $w = u, x_1, x_2, \dots, x_k, v$, with $k \geq 0$, such that $x_i \in A$, and each edge of the walk is external in C . Let f_A be the external face of A , and f_C be the external face of C . Let S be the set of connected components (we will refer to them as *segments*) in $f_A \cap f_C$ (all the nodes and edges that are external in both A and C). Let $s_u \in S$ be the segment that contains u . Similarly, let $s_v \in S$ be the segment that contains v .

We know that in C , there exists a walk of external edges that connects s_u to s_v . Thus, in A , external edges have been removed (edges from Y'). All removed edges span from A to B . Let $l_u, r_u, l_v, r_v \in B$ and $e_{l_u}, e_{r_u}, e_{l_v}, e_{r_v} \in Y'$ be removed edges (see Figure 2) (note it is possible that $r_u = r_v$ and it is possible that $l_u = l_v$). Since B is connected, there exists a walk from l_u to l_v residing entirely in B . This walk cannot go through A since $V(A) \cap V(B) = \emptyset$, so it can go directly from l_u to l_v , or all the way around A (see Figure 2). If it goes all the way around A , it must enclose e_{r_u} and e_{r_v} , since this walk cannot include the end nodes of s_u or s_v because they are in A . Hence, e_{r_u} and e_{r_v} are not in the external face of C , and could not have been in Y' , a contradiction. Therefore, it must go directly from l_u to l_v .

Similarly, since B is connected, there exists a walk from r_u to r_v residing entirely in B . By symmetry, this walk goes directly from r_u to r_v as well, without going all the way around A . Once again, we know B is connected, so there must exist a walk from l_u to r_u residing entirely in B (see Figure 2). If the walk goes

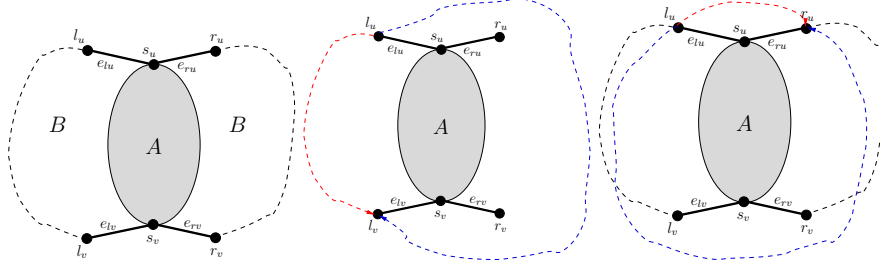


Figure 2: For Lemma B.2: the figure on the left shows a configuration of removed edges that are external in C and span from A to B (note, if the lemma were not true, B would be disconnected), the figure in the middle demonstrates the walk options from l_u to l_v , and the figure on the right demonstrates the walk options from l_u to r_u .

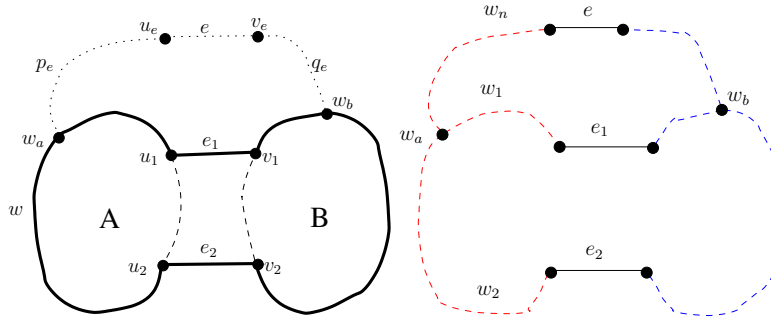


Figure 3: For Case 2 of Lemma B.3: the figure on the left demonstrates a possible setup, and the figure on the right demonstrates one of the two possible path configurations.

directly from l_u to r_u , it must enclose the external segment s_u , a contradiction. So the walk must go all the way around A , and therefore encloses the external segment s_v , a contradiction.

Therefore, for any two nodes $u, v \in A \cap C$ that are external in C , there exists a walk $w = u, x_1, x_2, \dots, x_k, v$, with $k \geq 0$, such that $x_i \in A$, and each edge of the walk is external in C . ■

Lemma B.3 $1 \leq |Y'| \leq 2$.

Proof: First, we show that $|Y'| \geq 1$. Let f_C be the external face of C . Let $u \in A$ and $v \in B$ be two external nodes in C . Clearly, $u, v \in f_C$. Since f_C is connected, there is a path p connecting u and v . Since Y is an edge-cut for A and B , p contains an edge in Y . Thus, one of the edges of Y is external in C , which implies that $|Y'| \geq 1$.

We now show that $|Y'| \leq 2$. Suppose for the sake of contradiction that $|Y'| > 2$. Choose two edges $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$, where $e_1, e_2 \in Y'$, $u_1, u_2 \in A$, and $v_1, v_2 \in B$. Let p be the walk from u_1 to u_2 consisting only of edges external in A and in C . Similarly, let q be the walk from v_1 to v_2 consisting only of edges external in B and in C . We know these walks exist from Lemma B.2. Construct a closed walk w using the edges in $p \cup q \cup e_1 \cup e_2$.

There are two cases to examine:

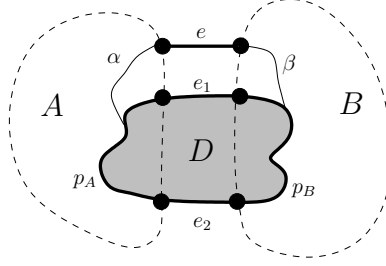


Figure 4: This figure demonstrates the subgraphs and paths described in Lemma B.4.

Case 1: w is an external face of C .

There exists an external edge $e \in Y'$ such that $e \neq e_1$ and $e \neq e_2$. w does not contain e since $e \notin A$ and $e \notin B$. Therefore, e is not in the external face of C , a contradiction.

Case 2: w is not an external face of C .

That is, there exists an external edge $e = (u_e, v_e)$, such that $e \in Y'$, $u_e \in A$, $v_e \in B$, and e is not contained within w . Since $u_e \in A$, there must exist a walk of external edges p_e from u_e to some node w_a belonging to w within A , such that $E(p_e) \cap E(w) = \emptyset$, $V(p_e) \cap V(w) = \{w_a\}$, and p_e is the shortest of such walks. Similarly, since $v_e \in B$, there must exist a walk of external edges q_e from v_e to some node w_b belonging to w within B , such that $E(q_e) \cap E(w) = \emptyset$, $V(q_e) \cap V(w) = \{w_b\}$, and q_e is the shortest of such walks (see Figure 3). These walks exist from Lemma B.2. Within w , there exists two walks consisting entirely of external edges from w_a to w_b , one goes through the edge e_1 , and the other through the edge e_2 (from Lemma B.2). Take the shortest of such walks and call them w_1 and w_2 respectively. It is clear that $w = w_1 \cup w_2$. Let w_n be the walk from w_a to u_e (p_e), the edge e , and the walk from v_e to w_b (q_e). We now have three walks w_1 , w_2 , and w_n , that connect w_a to w_b . The subpaths belonging to A may have common nodes and edges, and the subpaths belonging to B may have common nodes and edges. However, each walk has a unique external edge (w_1 has e_1 , w_2 has e_2 , and w_n has e). In any possible configuration, one of these external edges (either e_1 or e_2) is completely enclosed by the other two walks (see Figure 3), and is therefore not in the external face of C , a contradiction.

Since in both cases we obtained a contradiction, $|Y'| \leq 2$. ■

Let V_B be the nodes adjacent to the edges in Y' , which are in the graph B . From Lemma B.3, $1 \leq |V_B| \leq 2$. Let $p_B \in B$ be a shortest path connecting the nodes in V_B . Let $q = v_1, v_2, \dots, v_k$ be any path in B with the following properties: p_B and q do not intersect (they have no nodes in common), and v_1 is adjacent to an edge in Y .

Lemma B.4 *Node v_k belongs to a connected component of $B - p_B$ that does not contain any external nodes of C .*

Proof: Let V_A denote the nodes of A adjacent to Y' . From Lemma B.3, $1 \leq |V_A| \leq 2$. Let p_A denote a shortest path between the nodes in V_A . The union of the edges of Y' , p_A , and p_B induce a connected subgraph \hat{C} of C . Let W denote the set of nodes of C that are contained inside the internal faces (if they exist) of \hat{C} . Finally, let D denote the subgraph of C that is induced by the union of the nodes in W and \hat{C} .

Now, we show that all the edges of Y are members of D . 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 $e \notin D$. Consider first the case where $|Y'| = 1$. Let $e = (u, v) \in Y'$, with $u \in A$ and $v \in B$. We have that $p_B = v$. Thus, q intersects p_B , a contradiction. Consider now the case where $|Y'| = 2$. Suppose that $Y' = \{e_1, e_2\}$. Since A is connected, there is a path $\alpha \in A$ that connects edge e to a node in p_A ; similarly, 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 D .

Since v_1 is adjacent to an edge in Y , we have that $v_1 \in D$. Since q does not intersect p_B , each node of q is a member of D , that is, $q \in D$. 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 D . 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 $B - p_B$. ■

B.2 Proofs of Section 5.1

We first argue about the correctness of the for-loop in Algorithm `Subgraph-Clustering`. In particular, we want to show that path p_B exists. Consider an iteration with a connected component B of H' . Since B contains an external node of G , and B is a subgraph of both H and G , from Lemma B.1, B contains an external node of H too. Let C be the graph consisting of A , B , and the edges of Y . By the construction of the path p , we know that graph A also contains external nodes of H (the end nodes of p are external in H). Therefore, Lemma B.3 can be applied to C , which bounds the number of external edges in Y to be either one or two, giving $1 \leq |V_B| \leq 2$. Therefore, path p_B exists and it can be constructed in a way that its (at most two) end points are external nodes of H .

We continue with Lemma 5.1, which is a consequence of Lemmas B.6, B.7, and B.8 that we prove below. In the analysis, it is convenient to represent the execution of Algorithm `Depth-Cover` as a tree T , where each node in T corresponds to some invocation of the algorithm. The root r of T corresponds to the first invocation with parameters (G, G, v_1, 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. Suppose that node $w \in T$ 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 2γ -neighborhood of $p(w)$ in $H(w)$. As another example, using this notation, the resulting set of clusters is $Z = \bigcup_{w \in T} I(w)$.

Lemma B.5 *For any node $v \in G$, there is a node $w \in T$ such that $N_\gamma(v, G) = N_\gamma(v, H(w))$ and $v \in N_\gamma(A(w), H(w))$.*

Proof: By the construction of T , there is a path $s = w_1, w_2, \dots, w_k$, such that: $s \in T$, $k \geq 1$, $v \in H(w_i)$ 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 $s' = w_1, w_2, \dots, w_{k'}$, where $1 \leq k' \leq k$, be the longest subpath of s with the property that $N_\gamma(v, G) = N_\gamma(v, H(w_i))$ for $1 \leq i \leq k'$. 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, $v \in N_\gamma(A(w_{k'}), H(w_{k'}))$. Therefore, $w_{k'}$ is the desired node 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 $v \in N_\gamma(A(w_k), H(w_k))$. Thus, w_k is the desired node 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 $\text{depth}(G) \leq \gamma$, there is a node $x \in G$ that is external in G and $x \in N_\gamma(v, G)$. Since X does not contain any external node of G , $x \notin N_\gamma(v, X)$. Therefore, $N_\gamma(v, X) \neq N_\gamma(v, G) = N_\gamma(v, H(w_k))$. Thus, the γ -neighborhood of v changes between $H(w_k)$ and X . Hence, some node $u \in N_\gamma(v, H(w_k))$ 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 $v \in N_\gamma(A(w_k), H(w_k))$. Therefore, w_k is the desired node of T .

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

Lemma B.6 Z is a γ -cover for G .

Proof: From Lemma B.5, for each node $v \in G$ there is a node $w \in T$ such that $N_\gamma(v, G) = N_\gamma(v, H(w))$ and $v \in N_\gamma(A(w), H(w))$. By Lemma 3.1, $p(w)$ is 4γ -satisfied by $I(w)$ in $H(w)$. Since $A(w) = N_{2\gamma}(p(w), H(w))$, $A(w)$ is 2γ -satisfied by $I(w)$ in $H(w)$, which implies that 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 B.7 $\text{rad}(Z) \leq 8\gamma$.

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 $\beta = 4\gamma$. Therefore, by Lemma 3.1, for any $w \in T$, $\text{rad}(I(w)) \leq 2\beta = 8\gamma$, which implies that $\text{rad}(Z) \leq 8\gamma$. ■

Lemma B.8 $\text{deg}(Z) \leq 6$.

Proof: Consider an arbitrary node $v \in G$. We only need to show that $\text{deg}(v, Z) \leq 6$. Let $s = w_1, w_2, \dots, w_k$ be the path in T as described in Lemma B.5. According to Algorithm **Subgraph-Clustering**, the only possible clusters that v can participate in are $I(w_1), I(w_2), \dots, I(w_k)$. Let i denote the smallest index such that $v \in I(w_i)$. We will show that $i \in \{k-1, k\}$. We examine two cases:

Case 1: $v \in A(w_i)$

In this case, v will be removed with $A(w_i)$, and therefore, v will not appear in any child of w_i . Consequently, $w_i = w_k$, hence, $i = k$.

Case 2: $v \notin A(w_i)$

In this case, v is a member of a connected component B of $H'(w_i) = H(w_i) - A(w_i)$. There are two subcases:

Case 2.1: B does not contain any external node of G

In this case, B is discarded, and therefore, v will not appear in any child of w_i . Consequently, $w_i = w_k$, hence, $i = k$.

Case 2.2: B contains an external node of G

If $w_i = w_k$, the situation is similar as above, with $i = k$. So suppose that $i < k$. According to Algorithm **Subgraph-Clustering**, $B = H(w_{i+1})$. We will show that $v \in A(w_{i+1})$, which implies that $w_{i+1} = w_k$ (the reason is similar to the case where $v \in A(w_i)$ above). Since $v \in I(w_i)$, $v \in N_{4\gamma}(p, H(w_i)) = N_{2\gamma}(A(w_i), H(w_i))$. Thus, there is a node $u \in A(w_i)$ such that $v \in N_{2\gamma}(u, H(w_i))$. Let $g = u, x_1, x_2, \dots, x_k, v$ be a shortest path between u and v in $H(w_i)$. Clearly, $\text{length}(g) \leq 2\gamma$. Since $u \in A(w_i)$ and v is a member of a connected component B of $H'(w_i) = H(w_i) - A(w_i)$ with an external node of G , that path g must contain an edge of Y (or else $H(w_i)$ is disconnected). Choose the node x_y such that $x_y \in g$, $x_y \in B$, and x_y is adjacent to some edge of Y . Now, let $g' = x_y, x_{y+1}, \dots, x_k, v$ be a subpath of g in B . Clearly, $\text{length}(g') \leq 2\gamma$ as well.

Case 2.2.1: p_B and g' intersect

Then $v \in N_{2\gamma}(p_B, B) = N_{2\gamma}(p_B, H(w_{i+1}))$. Thus, $v \in A(w_{i+1})$. Therefore, $w_{i+1} = w_k$, which implies that $i = k - 1$.

Case 2.2.2: p_B and g' do not intersect

Thus, by Lemma B.4, in $B - p_B$, node v belongs to a connected component B' that has no external nodes of C . Since C is a subgraph of G , Lemma B.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 node w_{i+1} . Consequently, $w_k = w_{i+1}$, which implies that $i = k - 1$.

Consequently, $i \in \{k - 1, k\}$. 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 3.1, $\text{deg}(I(x)) \leq 3$. Therefore,

$$\text{deg}(v, Z) \leq \text{deg}(I(w_{k-1})) + \text{deg}(I(w_k)) \leq 6.$$

■

It is easy to verify that algorithm **Depth-Cover** computes cover Z in polynomial time with respect to the size of G . Therefore, the main result in this section, Lemma 5.1, follows from Lemmas B.6, B.7, and B.8.

B.3 Proofs of Section 5.2

The main algorithm for planar covers which handles an arbitrary locality parameter γ is Algorithm 5 (**Planar-Cover**). We give the analysis of the algorithm.

Lemma B.9 For $\gamma < \text{depth}(G)$ and $1 \leq i \leq \kappa$ it holds that: (i) $\text{depth}(S_i) \leq 3\gamma - 1$; (ii) $N_\gamma(W_i, G) = N_\gamma(W_i, S_i)$.

$depth(u, G)$. However, by the construction of W_k and W_i ,

$$\begin{aligned} depth(v, G) &\geq depth(u, G) + \gamma(i - k - 1) + 1 \\ &\geq depth(u, G) + \gamma + 1, \end{aligned}$$

a contradiction. The case where $k \geq i + 2$ is similar.

Therefore, $u \in S_i$, which implies that $N_\gamma(v, G) \subseteq S_i$. For each $u \in N_\gamma(v, G)$, there is a path p with length at most γ between u and v using only nodes in $N_\gamma(v, G)$. Since $N_\gamma(v, G) \subseteq S_i$, by the construction of S_i , the same path p also exists in S_i . Therefore, $N_\gamma(v, G) = N_\gamma(v, S_i)$. Consequently, $N_\gamma(W_i, G) = N_\gamma(W_i, S_i)$, as required. ■

Proof of Theorem 5.1 For any connected planar graph G and parameter $\gamma \geq 1$, Algorithm Planar-Cover returns in polynomial time a γ -cover Z with $rad(Z) \leq 24\gamma - 8$ and $deg(Z) \leq 18$.

Proof: If $\gamma \geq depth(G)$, Corollary 5.1 implies that Z is a γ -cover for G with $rad(Z) \leq 8\gamma$ and $deg(Z) \leq 6$. Thus, we only need to consider the case where $\gamma < depth(G)$.

Let Z_i denote the union of the clusters of the connected components of S_i returned by Algorithm Depth-Cover. From Lemma B.9, $depth(S_i) \leq 3\gamma - 1$. Since Algorithm Depth-Cover is invoked with parameter $3\gamma - 1$, from Corollary 5.1, we have that Z_i is a γ -cover of graph S_i , with $rad(Z_i) \leq 24\gamma - 8$, and $deg(Z_i) \leq 6$. From Lemma B.9, $N_\gamma(W_i, G) = N_\gamma(W_i, S_i)$. Thus, if a node $v \in W_i$ is γ -satisfied by a cluster C in S_i , then v is also γ -satisfied by C in G . Therefore, band W_i is γ -satisfied by Z_i in G . Since $Z = \bigcup_{i \in [\kappa]} Z_i$, and $V(G) = \bigcup_{i \in [\kappa]} W_i$, we have that Z is a γ -cover for the graph G .

Clearly, $rad(Z) \leq 24\gamma - 6$. For the degree, we observe that every node participates in at most three zones. A node in band W_i can participate in S_i, S_{i-1}, S_{i+1} and no other zones. Consequently for each vertex v , the degree $deg(v, Z)$ is bounded by $3 \times 6 = 18$. ■