

A Polynomial Kernel for Face Cover on Non-Embedded Planar Graphs

Thekla Hamm 

TU Eindhoven, The Netherlands

Sukanya Pandey 

RWTH Aachen University, Germany

Krisztina Szilágyi 

Czech Technical University in Prague, Czech Republic

Abstract

Given a planar graph, a subset of its vertices called terminals, and $k \in \mathbb{N}$, the FACE COVER NUMBER problem asks whether the terminals lie on the boundaries of at most k faces of some embedding of the input graph. When a plane graph is given in the input, the problem is known to have a polynomial kernel [16]. In this paper, we present the first polynomial kernel for FACE COVER NUMBER when the input is a planar graph (without a fixed embedding). Our approach overcomes the challenge of not having a predefined set of face boundaries by building a kernel bottom-up on an SPR-tree while preserving the essential properties of the face cover along the way.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Parameterized complexity and exact algorithms; Mathematics of computing $\rightarrow$ Graph algorithms

Keywords and phrases Kernelization, Planar Graphs, SPQR-tree

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.50

Related Version *Full Version*: <https://arxiv.org/abs/2601.04169> [17]

Funding *Thekla Hamm*: Supported by the Austrian Science Fund (J4651-N) during part of this work. *Krisztina Szilágyi*: Supported under the project Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22_008/0004590) and CTU Global Postdoc Fellowship Program.

1 Introduction

Problems in network design have been extensively studied on the class of planar graphs. Many of these problems take as input a graph and a subset of its vertices distinguished as terminals, and the goal is to find an optimal subgraph connecting or separating the terminals that satisfies certain constraints. Even when the input is restricted to the class of planar graphs, a large number of these problems remain intractable if the number of terminals is superconstant [14, 29, 28, 26, 8, 15, 23]. This motivates the quest for further restrictions on the input that may make the problems tractable. Besides bounding the actual number of terminals, one can target weaker restrictions, which impose conditions on the placement of the terminals in the input planar graph. One such restriction could be to require that the terminals lie on the boundary of a single face of the planar input graph. Some well-known NP-hard problems like STEINER TREE, MULTIWAY CUT, and SHORTEST DISJOINT PATHS can be solved in polynomial time [11, 2, 6, 4] in this case. It is therefore natural to wonder: *What if the terminals lie on the boundaries of $k > 1$ faces?*

The *terminal face cover number* of a planar graph is the minimum number of faces, over all embeddings, that jointly cover all the terminals. Parameterization by the terminal face cover number was first considered by Erickson et al. in their seminal paper [11] to design an XP algorithm for STEINER TREE. Recently, it has received a lot of attention, be it for flow and



© Thekla Hamm, Sukanya Pandey, and Krisztina Szilágyi;
licensed under Creative Commons License CC-BY 4.0

43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).

Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thăng

Article No. 50; pp. 50:1–50:18



Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



cut problems [25, 24, 6, 27], shortest path problems [7, 13], minimum non-crossing walks [10], and minimum Steiner tree [20]. In this paper, we focus on the problem of computing the terminal face cover number. Formally, the problem is defined as follows:

FACE COVER NUMBER

Instance: Planar graph G , $T \subseteq V(G)$, integer k

Question: Does there exist a set of faces of size at most k in some embedding of G that covers all the vertices of T ?

Bienstock and Monma [3] initiated the study of FACE COVER NUMBER and showed that it can be solved in FPT time parameterized by k . Their algorithm runs in $c^k \cdot n$ time, for some constant $c > 1$. They also studied a related problem where the input consists of a plane graph, that is, a planar graph with a fixed embedding. We call this problem EMBEDDED FACE COVER NUMBER. For this problem, they considered parameterization by the number of terminals t , and gave the first subexponential FPT algorithm with a running time of $2^{\mathcal{O}(\sqrt{t})}$.

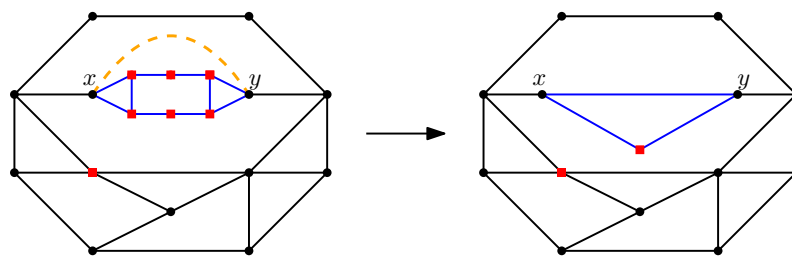
A special case of EMBEDDED FACE COVER NUMBER with $T = V(G)$, parameterized by the face cover number, was studied by Kloks et al. [21]. They showed that the problem can be solved in time $\mathcal{O}^*(c^{\sqrt{k}})$. Subsequently, their result was improved by Fernau and Juedes [12], whose algorithm runs in time $\mathcal{O}^*(2^{24.551\sqrt{k}})$ and further by Koutsonas and Thilikos [22] who gave an algorithm with a running time of $\mathcal{O}^*(2^{10.1\sqrt{k}})$. This problem can be easily reduced to RED-BLUE DOMINATING SET, a variant of DOMINATING SET on 2-colored graphs. All the aforementioned algorithms *indirectly* find the minimum face cover of the input plane graph by solving RED-BLUE DOMINATING SET. A direct FPT algorithm for the problem was given by Abu-Khzam et al. [1]. However, with a running time of $\mathcal{O}^*(4.6056^k)$, their algorithm is not subexponential in the parameter.

Given the fixed-parameter tractability of both the above problems, a natural line of inquiry is whether there exists a polynomial kernel for the problem. For EMBEDDED FACE COVER NUMBER, Garnero et al. [16] showed that a linear kernel exists by reduction to RED-BLUE DOMINATING SET. Before their work, a quadratic kernel was known to exist due to the result of Kloks et al. [21]. To date, it was not known whether FACE COVER NUMBER has a polynomial kernel. We answer this question in the affirmative by presenting a cubic kernel for the problem parameterized by the face cover number.

► **Theorem 1.** *Given an instance (G, T, k) of FACE COVER NUMBER, we can produce a kernel (G', T', k') in polynomial time such that $|G'| + k' \in \mathcal{O}(k^3)$.*

Overview of the algorithm. Our approach is to perform dynamic programming over an SPR-tree of the given planar graph. SPR-trees are classic data structures that have often been useful for algorithm design on planar graphs. At a high level, an SPR-tree can be viewed as a tree-structured decomposition of a planar graph into its 3-connected subgraphs. Every node of this tree-structure is associated with a graph called the *skeleton* of that node. The graph induced at any node is obtained by replacing each of the so-called *virtual edges* (edges of the skeleton that are not actual edges of the planar graph) by the graphs induced at the children of that node. In this sense, endpoints of virtual edges can be viewed as an “interface” between the graph induced at a node and the graphs induced at its children. Each node of the SPR-tree has a type – S, P or R – depending on the structure of its skeleton.

With this in mind, we aim to construct a kernel for FACE COVER NUMBER in a leaves-to-root fashion along an SPR-tree of the input graph. For this, a natural attempt at the dynamic programming step is to try to replace parts of the graph that correspond to an



■ **Figure 1** (left) A skeleton in black and orange with the orange dashed edge xy being a virtual edge to a child node. The graph that replaces this virtual edge is blue. Terminals are red squares. The blue graph has face cover number one (all its terminals lie on its rectangularly drawn face). Hence, a kernel for the blue graph could in principle be a triangle on x , y and a terminal (this also has face cover number one). This would even still contain the interface (x, y) between the blue and black graph. (right) However, if we replace the blue graph with its possible triangular kernel, the face cover of the entire graph drops from two to one.

SPR-tree rooted at a child of that SPR-tree by an inductively assumed kernel. However, this naïve approach comes with some technical obstacles: how do we replace a subgraph with a kernel in which we are not ensured that the interface to the subgraph remains? On a more fundamental level, this simple idea is doomed to fail because while the face cover number of the kernel may be the same, it might interact differently with the rest of the graph than the subgraph we replace it with (see Figure 1). Consequently, we need to strengthen what we inductively assume of kernels constructed so far and then also demonstrate in each dynamic programming step that these assumptions are satisfied after carrying out the reduction steps. Crucially, our stronger assumptions allow us to preserve the way in which the face cover of the graph induced at any child node interacts with the graph induced at the parent node during kernelization. We call a kernel that satisfies our stronger assumptions a *nice kernel*, and our target is to dynamically compute such a kernel.

While many parts of our dynamic programming procedure require us to devise reduction rules that are specific to the skeleton and hence the type of node we are considering (S, P or R), some subgraphs that are induced at the children of a node have sufficiently *simple* face covers. Hence, we can replace them by constant-sized subgraphs and obtain a nice kernel in which we only need to handle subgraphs induced at children that have significantly more complex interactions with the skeleton of the parent.

We refer to the simple subgraphs as *terminal-free*, *unproblematic* or *semi-problematic* components depending on whether they contain terminals, and if so, whether they can be covered optimally using only faces that interact with the rest of the graph. We describe polynomial-time algorithms to recognize each of them. We call the remaining complex subgraphs induced by children *problematic*. At each node type, the overall strategy is to bound the number of virtual edges that we need to consider by a polynomial in the prospective face cover number k . Once we achieve that, we can then reduce the skeleton in a way that we can bound the number of its non-virtual edges in k . Both of these are the steps that are the most specific to the node types. Finally, we inductively replace the problematic virtual components by their kernels and any possible remaining non-problematic ones by their constant-size replacements – this last step does not depend on the particular node type.

Concerning the node-type specific arguments, simple reduction rules like edge deletions and contractions can be used at S-nodes and P-nodes. However, such simple reduction rules are difficult to control at R-nodes, which, of all the node types admit the structurally most complex skeletons, namely 3-connected planar graphs. Deleting or contracting edges from

a 3-connected planar graph does not, in general, preserve 3-connectivity. This is of great relevance for face cover numbers because, before any reduction, the skeleton has a unique embedding, a property that we cannot easily preserve during reduction. For this reason, during the kernelization for R-nodes, we switch to the embedded setting, and formulate reduction rules that take the initial fixed embedding into account. To transition back to the setting in which we consider a non-embedded graph, we re-establish 3-connectivity by what we call “*rigidizing*” the embedded nice kernel. While doing so, we take care not to change the sets of terminals of the nice kernel that occur together on a face, and hence obtain a non-embedded nice kernel at R-nodes.

Organization. In Section 2, we give necessary definitions and notation. In Section 3, we give formal definitions of *terminal-free*, *unproblematic* or *semi-problematic* and *problematic* components, and algorithms to distinguish them. Then, in Section 4 we define the notion of a *nice kernel* and describe the replacements of the components from Section 3, which are not node-type specific. In Section 5, we turn to the actual formulation of the dynamic program. We conclude in Section 6. The omitted proofs are marked with † and can be found in the full version of the paper [17].

2 Preliminaries

Graph terminology. We follow standard graph terminology [9]. A *graph* is a pair $G = (V, E)$, such that $E \subseteq \binom{V}{2}$. The elements of V are called the vertices of the graph and the elements of E are called edges. The number of vertices of a graph is its order denoted by n . A *subgraph* $G' = (V', E')$ of G , written as $G' \subseteq G$, is a graph such that $V' \subseteq V$ and $E' \subseteq E$. G' is an *induced subgraph* of G , if for all $x, y \in V'$, if $xy \in E$ then $xy \in E'$.

A *path* is a non-empty graph $P = (V', E')$ of the form $V' = \{x_0, x_1, \dots, x_k\}$ and $E' = \{x_0x_1, \dots, x_{k-1}x_k\}$, where all x_i are distinct. The vertices x_0 and x_k are called the *endpoints* of P . An *induced path* is a path in which all the vertices except the endpoints have degree two. The number of edges in a path is its *length*.

A graph is *connected* if there is a path between any two vertices in a graph, and *disconnected* otherwise. A graph is *biconnected* if there does not exist any vertex whose removal from the graph results in a disconnected graph. Similarly, a graph is called *3-connected* if there does not exist a subset of vertices of size two whose removal disconnects the graph.

SPR-trees. An *SPR-tree* is a rooted tree in which each node t is associated with a multigraph called the *skeleton* of the node. The nodes, and their skeletons, have one of three types:

1. **S node:** These are nodes whose skeleton forms a cycle of three or more vertices.
2. **P node:** These are nodes whose skeleton is a graph with exactly two vertices and at least three edges between them. Such a graph is called a dipole.
3. **R node:** These are nodes whose skeleton forms a 3-connected graph that is not a cycle or a dipole.

Each edge between two nodes t and t' of the SPR-tree is associated to one edge with an ordering of its endpoints in the skeleton of t and one edge with an ordering of its endpoints in the skeleton of t' . Overall, the edge of the skeleton of each node is allowed to be associated to at most one edge of the SPR-tree in this way. These skeleton edges that are associated to SPR-tree edges are called *virtual* edges. The edges of the skeleton of any node that are not virtual are called *real*. For the subtree rooted at a node t of the SPR-tree the graph *induced* by it is inductively defined as follows.

- if t is a leaf, the graph induced by the subtree rooted at t is the skeleton of t .
- if t has children $t_1, \dots, t_c$, the graph induced by the subtree rooted at t is the graph arising from the skeleton of t by doing the following for each $i \in [c]$: Insert the graph induced by the subtree rooted at t_i ; identify the first and second endpoint of the virtual edge in the skeleton of t_i that is associated to tt_i with the first and second endpoint respectively of the virtual edge in the skeleton of t that is associated to tt_i ; and lastly, delete the virtual edges corresponding to tt_i . Intuitively speaking, the graph induced by the subtrees rooted at t_i are “glued-in” in place of the respective virtual edges of the skeleton of t .

We say that an SPR-tree is an SPR-tree of the graph that it induces.

► **Lemma 2** (Hopcroft and Tarjan [18]). *An SPR-tree of a biconnected planar graph can be constructed in linear time.*

For a node t of an arbitrary fixed SPR-tree of G , we denote the subgraph induced by the subtree rooted at t by G_t . We call the subgraph induced by a subtree rooted at a child of t a *virtual component* of G_t . Let c_1, c_2 be the endpoints of the virtual edges associated to tt' where t' is a child of t . We call c_1 and c_2 the *corners* of the virtual component of G_t associated with t' . We call the edge c_1c_2 the *corner edge* of $G_{t'}$. We refer to the virtual component of G_t associated with t' plus its corner edge as the *enhancement* of that virtual component. In an embedding of an enhancement, we call the two faces containing the corner edge *external faces* and the remaining faces *internal faces*. If G_t itself has no corners, i.e. t is the root of the SPR-tree, then G_t is equal to its enhancement (we pick an arbitrary edge of G_t and regard it as the corner edge and its endpoints as corners).

Kernelization. A kernelization for a parameterized problem $\Pi \in \Sigma^* \times \mathbb{N}$ is an algorithm that takes as input an instance (I, k) of Π , runs in time $\text{poly}(|I| + k)$ and outputs an instance (I', k') of Π such that: (1) (I, k) is a YES-instance if and only if (I', k') is a YES-instance (2) $|I'| \leq f(k)$, for some computable function f , and (3) $k' \leq g(k)$, for some computable function g . The output (I', k') of a kernelization is referred to as a *kernel*. We say that (I', k') is a *polynomial kernel* if $|I'| + k' \leq \text{poly}(k)$.

Embedded graphs. A *drawing* of a connected planar graph $G = (V, E)$ is defined as a mapping of the vertices in V to points in $\mathbb{R}^2$ and edges to simple curves between the points corresponding to its endpoints, such that: (1) distinct edges have distinct pair of endpoints, and (2) the interior of an edge does not contain any vertex or any point of another edge. Given a planar drawing, the (clockwise) cyclic order of edges incident on every vertex is *fixed*. The set of circular orderings of edges around each vertex is called a *rotation system*. A (*combinatorial*) *embedding* is an equivalence class of planar drawings defined by their rotation systems. We frequently identify vertices and edges with their drawings and embeddings with an arbitrary drawing from its represented equivalence class.

The restrictions of embeddings and drawings to a subgraph of a graph are called *subembeddings* and *subdrawings* respectively. Let G be a plane graph and t a node in its SPR-decomposition. *Flipping* G_t is defined by reversing the rotation system restricted to G_t (i.e. for the corner vertices, we only reverse edges in G_t).

For every drawing G of a planar graph, the inclusion-maximal regions of $\mathbb{R}^2 \setminus G$ are called *faces*. The unique unbounded face is the *outer face*, all other faces are *inner faces*. Let f be a face. We say that a face f *covers* a vertex v if the drawing of v lies on the boundary of f . For the two external faces of the enhancement G_t , we define their corresponding faces in G as

the faces whose boundaries restricted to G_t are equal. Given a face cover F of G , we define the induced face cover F_t of the enhancement of G_t as the set of internal faces of G_t that are in F together with the external faces corresponding to a face in F . The *embedded face cover number* of an embedded graph G with terminals $T \subseteq V(G)$ is the minimum number of faces needed to cover T . The *face cover number* of a graph G with terminals $T \subseteq V(G)$ ($\text{fcn}(G) = \text{fcn}(G, T)$) is the minimum number of faces over all embeddings of G that cover the terminals of G .

One-connected case

By standard arguments in face cover literature which we repeat below for self-containedness, we can reduce to considering a biconnected input graph. Also, note that we can reduce to connected graphs as follows. If G is disconnected, we kernelize every connected component separately, and then we check if there are more than k components that need at least 2 faces for the face cover (using the existing FPT algorithm from [3] for $k = 1$). If there are, we have a NO-instance.

The following definition will be useful to show that we can reduce to the case of a biconnected input graph. A *cut vertex* in G is a vertex v such that $G - v$ has two or more connected components. A *block* of graph G is a maximal biconnected induced subgraph. A block is *trivial* if it contains exactly one edge and *non-trivial* otherwise. A *block-cut tree* of a graph G has a vertex for each block and cut vertex in G . We denote the block-cut tree by $\mathcal{B}$. There exists an edge in $\mathcal{B}$ for a pair of nodes corresponding to a block and cut vertex in G if and only if the cut vertex belongs to the block.

► **Lemma 3** (Hopcroft and Tarjan [19]). *A block-cut tree of a graph can be computed in linear time.*

We now give the reduction to the biconnected case.

► **Lemma 4.** *If we can compute $\text{fcn}(G, T)$ for every 2-connected planar graph G and every $T \subseteq V(G)$, then we can compute $\text{fcn}(G, T)$ for every planar graph G and $T \subseteq V(G)$.*

Proof. If G is not 2-connected, consider the block-cut tree $\mathcal{B}$ of G . We will compute the FACE COVER NUMBER inductively, starting from the leaves. Let v be a cutvertex with children $B_1, \dots, B_t$ and let G^v be the subgraph of G that corresponds to the subtree of $\mathcal{B}$ rooted at v . We distinguish two cases:

- If v is a terminal: we set $\text{fcn}(G^v, T \cap G^v) = \sum \text{fcn}(B_i, T \cap B_i) - (t - 1)$, since we know that in each B_i , the vertex v will be covered, and we can ensure that the face used to cover it is the outer face of B_i .
- If v is not a terminal: for each $i \in [t]$, we compute $k_i = \text{fcn}(B_i, T \cap B_i)$ and $k'_i = \text{fcn}(B_i, (T \cap B_i) \cup \{v\})$. If there is exists $i > 1$ such that $k_i = k'_i$, we can ensure that the outer face of B_i covers v . Thus adding v to T does not increase the face cover number. We add v to T and use the above argument to compute $\text{fcn}(G^v, T \cap G^v)$. Otherwise, we set $\text{fcn}(G^v, T \cap G^v) = \sum \text{fcn}(B_i, T \cap B_i)$. ◀

3 Types of Virtual Components

To facilitate constructing the kernel, we will divide virtual components into 4 groups. The terminal-free, unproblematic and semi-problematic will be replaced by constant-sized gadgets, while the problematic ones will eventually be replaced by inductively constructed kernels.

► **Definition 5** (Types of virtual components). *We distinguish different types of virtual components.*

1. *Virtual components that have no terminals except possibly the corners – we call such virtual components terminal-free.*
2. *Virtual components that have a face cover consisting of exactly one external face – we call such virtual components unproblematic.*
3. *Virtual components that are not unproblematic, have a face cover consisting of both external faces but also have face cover number one – we call such virtual components semi-problematic.*
4. *All other virtual components which we call problematic.*

For unproblematic virtual components, we say that an embedding witnesses their unproblematicness if in that embedding one of the external faces covers all terminals. For semi-problematic components, we say that an embedding witnesses their semi-problematicness if the face cover number of that embedding is one, but both the external faces can jointly cover all the terminals.

We first argue that the types of virtual components can be recognized in polynomial time and then present kernelization steps for such virtual components in Section 4. Obviously, terminal-free virtual components can be recognized in linear time and to recognize problematic components, one merely needs to recognize unproblematic and semi-problematic components. For this, we can give dynamic programs along the SPR-subtree.

► **Lemma 6** (†). *Deciding whether a virtual component is unproblematic is in P.*

► **Lemma 7** (†). *Deciding whether a virtual component is semi-problematic is in P.*

4 Nice Kernels and Basic Virtual Component Replacement

In this section, we describe a kernelization procedure that inductively assumes that we have special kernels, which we refer to as *small nice kernels* of the enhancement of each virtual component of the graph induced at the node of the SPR-tree under consideration.

► **Definition 8.** *Let (G, T) be an instance of FACE COVER NUMBER. Given a node t of the SPR-tree of G consider its enhancement G_t with corner vertices c_1, c_2 . We define $\text{fcn}_0(G_t, T)$, $\text{fcn}_1(G_t, T)$, $\text{fcn}_2(G_t, T)$ as the sizes of the smallest face cover of the enhancement of G_t such that they contain exactly 0, 1, 2 external faces respectively. Let K be a graph with terminal set $T' \subseteq V(K)$. We say that K is a nice kernel of G_t if the following conditions are fulfilled:*

K1 K contains vertices c_1, c_2 and the edge c_1c_2 .

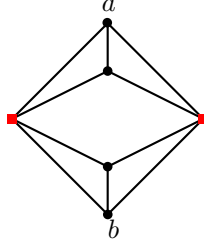
K2 For $i \in \{0, 1, 2\}$, we have that $\text{fcn}_i(G_t, T \cap V(G_t)) = \text{fcn}_i(K, T') \leq k$ or that $\text{fcn}_i(G_t, T \cap V(G_t)) > k$ and $\text{fcn}_i(K, T') > k$.

K3 For every $C \subseteq \{c_1, c_2\}$, we have that $\text{fcn}_0(G_t, (T \cap V(G_t)) \setminus C) = \text{fcn}_0(K, T' \setminus C) \leq k$ or that $\text{fcn}_0(G_t, (T \cap V(G_t)) \setminus C) > k$ and $\text{fcn}_0(K, T' \setminus C) > k$,

A nice kernel is called small if

K4 There is some constant c (that does not depend on K) such that $c \cdot |V(K)|^{\frac{1}{3}}$ is the minimum number of internal faces used in any minimum face cover of G_t .

The condition (K1) ensures that we can construct the kernel inductively, i.e. that in G_t , we can replace a virtual component corresponding to its child by a nice kernel for it. Intuitively, we want G_t and K to behave the same, and in particular interact with the rest of the graph in the same way. The conditions (K2) and (K3) say that if the face cover sizes of G_t and K



■ **Figure 2** The gadget for semi-problematic virtual components. Red squares are terminals. Note that if a and b are fixed to be on the outer face, then the face boundaries are unique.

differ, both of the face covers are too big to lead to a global solution. One important thing to note is that the corner vertices c_1, c_2 might be covered by from the “outside” of G_t . Thus we need to compare face covers of G_t and K covering all terminals except for the corners. In (K3), we only compare the values of fcn_0 as the values of fcn_1 and fcn_2 are equal by (K2) (because the corners are covered by the corresponding external faces).

Ultimately, we will design a leaves-to-root dynamic program along the SPR-tree to obtain small nice kernels for each G_t with t being a node of an SPR-tree of G . We can verify that a small nice kernel at the root of the SPR-tree is in fact a polynomial kernel.

► **Lemma 9.** *Let r be the root of the considered SPR-tree for G . A small nice kernel K of G_r is a polynomial-sized kernel for G as an instance of FACE COVER NUMBER.*

Proof. By the fact that for r , G_r has no corners and $G = G_r$ coincides with its enhancement, (K1)–(K3) are obsolete or equivalent to expressing instance equivalence of G and K as instances of FACE COVER NUMBER.

Moreover, by (K4) there is a constant c such that $|V(K)| \leq \frac{1}{c}(k-2)^3$ where k is the face cover number of G (and hence also of K). This shows that K is a cubic kernel for G as an instance of FACE COVER NUMBER. ◀

The next reduction rules replace “simple” virtual components by constant-sized graphs.

► **Reduction Rule 1.** *Replace every virtual component without terminals by an edge between its corners.*

► **Reduction Rule 2.** *Replace every unproblematic component by a P_3 whose endpoints are identified with the corners and the degree-two vertex is a terminal.*

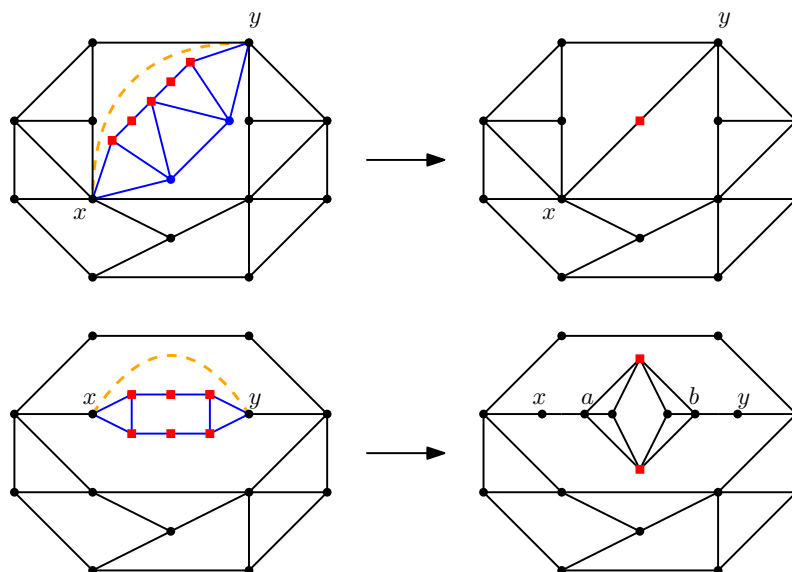
We call the above graph a *terminal-subdivided edge*.

► **Reduction Rule 3.** *Replace every semi-problematic component by the gadget depicted in Figure 2 by connecting one corner to the vertex a of the gadget by an edge and the other corner to the vertex b .*

As defined above, let t be a node of the SPR-tree of G and let G_t be its enhancement.

► **Lemma 10** (†). *Let K and T' be the graph that arises from G_t by applying the described replacements (Reduction Rule 1, Reduction Rule 2 and Reduction Rule 3) and the set of terminals in it respectively. Then K is a nice kernel of G_t .*

Next, we show that inductively replacing nice kernels for problematic virtual components yields a nice kernel.



■ **Figure 3** Exemplary replacements made in Reduction Rule 2 (top) and Reduction Rule 3 (bottom). The virtual edges are shown as orange dashed curves, their corresponding virtual components are shown in blue in the left figures and their corresponding replacements are shown on the right.

► **Lemma 11.** *Let C be a problematic component of G_t and $\tilde{C}$ be a nice kernel for C . Let $\tilde{G}_t$ be the graph obtained from G_t by replacing C by $\tilde{C}$. Then $\tilde{G}_t$ is a nice kernel for G_t .*

Proof. Firstly, note that the replacement does not affect the skeleton of t , so the corner vertices and corner edge of G_t are preserved, i.e. (K1) is satisfied. Let c_1, c_2 be the corner vertices of C and let C' be the enhancement of C .

Let us now show (K2). Let $i \in \{0, 1, 2\}$. Fix an embedding $\mathcal{F}$ of G_t witnessing $\text{fcn}_i(G_t, T)$ and let F be the corresponding face cover. We aim to construct a face cover $\tilde{F}$ of $\tilde{G}_t$ from F . Informally, we partition F into two sets, F_C and F_U , where F_U is the set of faces unaffected by the replacement of C by $\tilde{C}$, and F_C contains the remaining faces of F . Using the fact that $\tilde{C}$ is a nice kernel for C , we construct a face cover $\tilde{F}_{C'}$ of $\tilde{C}$. The face cover $\tilde{F}$ will contain the faces in F_U and the faces corresponding to $\tilde{F}_{C'}$.

Formally, we define $F_U = \{f \in F : f \cap V(C) \subseteq \{c_1, c_2\}\}$. In other words, F_U contains faces in F that are disjoint from C except possibly for its corner vertices. Let $F_C = F \setminus F_U$. The faces in F_C correspond to faces of C' : those that are completely contained in C correspond to internal faces of C (with the same set of vertices) and those that contain vertices outside of C correspond to the external faces of C' . Let $F_{C'}$ be the set of faces of C' corresponding to F_C . Note that $F_{C'}$ is a face cover of C' covering all terminals in $T \cap V(C)$ except possibly for c_1, c_2 (as these vertices may be covered by faces in F_U). Let $j \in \{0, 1, 2\}$ be the number of faces in $F_{C'}$ corresponding to external faces of C' . We distinguish two cases depending on the value of j :

If $j \in \{1, 2\}$. in this case, all terminals of C' are covered by a face in $F_{C'}$. By an exchange argument, it is easy to see that $F_{C'}$ is optimal, i.e. it witnesses $\text{fcn}_j(C, T \cap V(C))$, so we have $\text{fcn}_j(C, T \cap V(C)) = |F_{C'}| = |F_C|$. Fix an embedding of $\tilde{C}$ witnessing $\text{fcn}_j(\tilde{C}, T' \cap V(\tilde{C}))$ and let $\tilde{F}_{C'}$ be the corresponding face cover. We set $T_1 = T$, $T'_1 = T'$.

If $j = 0$. in this case, c_1 and c_2 might not be covered by a face in $F_{C'}$. Let T_1, T'_1 be sets of terminals obtained by removing the corner vertices not covered by a face in F_C from T and T' respectively. Similarly, we have that $\text{fcn}_0(C, T'_1 \cap V(C)) = |F_C|$. Fix an embedding of $\tilde{C}$ witnessing $\text{fcn}_0(\tilde{C}, T'_1 \cap V(\tilde{C}))$ and let $\tilde{F}_{C'}$ be the corresponding face cover.

We embed $\tilde{G}_t$ by using the above embedding of $\tilde{C}$ and leaving the remaining part of $\tilde{G}_t$ the same as in $\mathcal{F}$. If $j = 1$, we embed $\tilde{C}$ inside $\tilde{G}_t$ such that the external face of $\tilde{C}$ that belongs to $F_{C'}$ corresponds to the face in F_C containing c_1, c_2 . If $j \in \{0, 2\}$ we can embed $\tilde{C}$ with an arbitrary choice of external faces (i.e. the way it is “flipped” inside $\tilde{G}_t$ is irrelevant). Now we have obtained an embedding of $\tilde{G}_t$.

We define $\tilde{F}_C$ as the set of faces of $\tilde{G}_t$ corresponding to $\tilde{F}_{C'}$. Note that the faces in F_U are faces of $\tilde{G}_t$ (since they do not contain vertices from $V(C) \setminus \{c_1, c_2\}$) and that $F_U \cap \tilde{F}_C = \emptyset$.

We claim that $\tilde{F} = F_U \cup \tilde{F}_C$ is a face cover of $\tilde{G}_t$. Clearly, all terminals except c_1, c_2 are covered: those in $V(\tilde{C})$ are covered by a face in $\tilde{F}_C$, and the others are covered by a face in F_U . The vertices c_1 and c_2 (if they are terminals) are covered by a face in $\tilde{F}_C$ if they were covered by a face in F_C , and otherwise they are covered by a face in F_U . Thus $\tilde{F}$ is a face cover of $\tilde{G}_t$. It remains to show the size bounds.

If $|F| \leq k$, then we have $\text{fcn}_j(C, T_1 \cap V(C)) = |F_C| \leq k$, so $|\tilde{F}_C| = \text{fcn}_j(\tilde{C}, T'_1 \cap V(\tilde{C})) = \text{fcn}_j(C, T_1 \cap V(C))$ by (K2) and (K3). Thus $|\tilde{F}_C| = |F_C|$, so $|\tilde{F}| = |F|$. If $|F| > k$, we distinguish two cases. If $|F_C| > k$, then $|\tilde{F}_C| > k$, so $|\tilde{F}| > k$. If $|F_C| \leq k$, then $|\tilde{F}_C| = |F_C|$, so $|\tilde{F}| = |F| > k$.

To show (K3), we remove one or both corner vertices from T and T' and proceed the same as the case $j = 0$. ◀

5 Node-Type Specific Kernelization Steps

In the following subsections, we describe the node-specific reduction rules that will allow us to obtain small nice kernels at each node type. By Lemma 9, this implies a polynomial kernel for the entire graph. For each of these sections, consider a node t of the SPR-tree of G .

5.1 R-node

It is well-known that any 3-connected planar graph, in particular the skeleton of an R-node, has a unique embedding up to homeomorphism [5]. Our strategy is to fix this embedding of the skeleton of an R-node, kernelize it based on this fixed embedding, possibly losing 3-connectivity, and then rigidize the resulting kernel to regain control over the faces that contain terminals.

We first make the following general observation about 3-connected embedded graphs.

► **Proposition 12** (†). *In any 3-connected planar graph with at least four vertices, no three vertices share more than one face.*

Proposition 12 can easily be adapted to when some edges are subdivided by terminals, in particular as is the case in the replacement for unproblematic virtual components in Section 4. For ease of notation, we refer to graphs that may arise from 3-connected planar graphs by replacing some edges by terminal-subdivided edges as *almost 3-connected planar* and to the graph that arises from the skeleton of G in this way as *extended skeleton*.

► **Proposition 13** (†). *In any embedding of any almost 3-connected planar graph with at least seven vertices, no four vertices share more than one face.*

Let us now actually focus on G_t where t is an R-node of the SPR-tree of G . For problematic virtual components, we make the following simple observation.

► **Observation 14** (†). *If G_t has more than k problematic virtual components, then it has no face cover of size at most k .*

We can also bound the number of semi-problematic virtual edges that we need to consider.

► **Proposition 15** (†). *If there is a face of the skeleton of G_t with more than k semi-problematic virtual edges on its boundary, then there is no face cover of size at most k for this embedding.*

► **Corollary 16** (†). *If there are more than k^2 semi-problematic virtual edges in the skeleton of G_t , then there is no face cover of size at most k .*

Finally, we reduce the number of unproblematic virtual edges and in fact, the total number of terminals in the skeleton. For this, we first make a general statement about almost 3-connected planar graphs.

► **Proposition 17** (†). *In the embedding of an almost 3-connected planar graph, a face with more than $3k$ terminals on its boundary has to be in any face cover of size at most k .*

Together with the safeness of the basic replacement step for unproblematic virtual components (cf. the proof of Lemma 10), this immediately can be used to justify the following reduction rule which we refer to as *terminal-heavy face reduction*.

► **Lemma 18** († Safeness of terminal-heavy face reduction). *Replacing all unproblematic virtual components by terminal-subdivided edges and then for each face of the extended skeleton with more than $3k + 1$ terminals, turning all but an arbitrary set of $3k + 1$ of these terminals into non-terminals yields a nice kernel.*

► **Proposition 19** (†). *If after exhaustive application of terminal-heavy face reduction for G_t , there are more than $3k^2 + k$ terminals, then there is no face cover of size at most k .*

Assume from now on that terminal-heavy face reduction was exhaustively applied to G_t ; in particular, we will no longer explicitly mention it as a precondition in the following theorem statements. We have now bounded the number of terminals and edges in the extended skeleton, which we will want to replace by inductively assumed kernels. It remains to reduce the parts of the skeleton that contain neither. During this modification of the extended skeleton, for the first time, we might lose the uniqueness of its embedding. We will fix its embedding and take it into account in the following modifications. For this, we explicitly speak about nice kernels of EMBEDDED FACE COVER NUMBER, rather than nice kernels for FACE COVER NUMBER i.e. fcn gets replaced by its fixed-embedding analogue in (K1)–(K3).

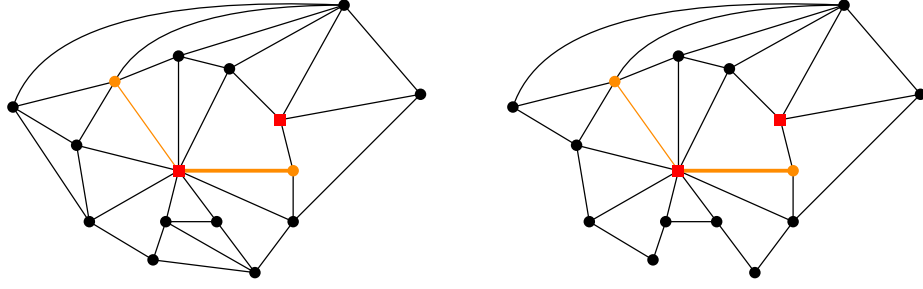
We can bound the number of faces containing at least two of the following objects: terminals, corners or endpoints of virtual edges. We refer to any such vertex as *interesting*.

► **Proposition 20** (†). *The number of faces with at least two interesting vertices in the skeleton of G_t is bounded above by $\mathcal{O}(k^2)$.*

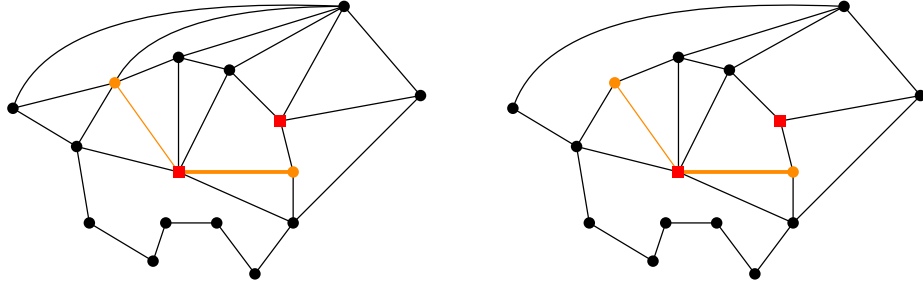
► **Reduction Rule 4** (Boring edge removal). *Delete edges of the extended skeleton that are only on boundaries of faces that do not contain interesting vertices (see Figure 4).*

Note that these edges in the reduction rule above are uniquely determined because the embedding of the extended skeleton is fixed.

Next, we consider each face f of the skeleton of G that has some arbitrary fixed terminal, corner or endpoint of a virtual edge x on its boundary.



■ **Figure 4** Example of the result of boring edge removal (right) applied to a graph (left) in which terminals are red squares, and virtual edges and their endpoints are orange (the thick one being the corner edge).



■ **Figure 5** Example of the result of private face merging around the left terminal (left) and the result of exhaustive private face merging (right) applied to the graph from Figure 4.

► **Reduction Rule 5** (Private face merging). *If there exists a face f in the extended skeleton that has at most one fixed interesting vertex, say x , on its boundary, and is incident on another face f' containing x but no other interesting vertex, delete the shared boundary of f and f' connected to x (see Figure 5).*

► **Proposition 21** († Safeness of private face merging). *Let $\tilde{G}$ be the graph obtained from G_t by exhaustively applying Reduction Rule 4 and Reduction Rule 5. For an arbitrary embedding of G_t , $\tilde{G}$ embedded according to the restriction of the embedding of G_t to $\tilde{G}$ is a nice kernel of G_t for EMBEDDED FACE COVER NUMBER.*

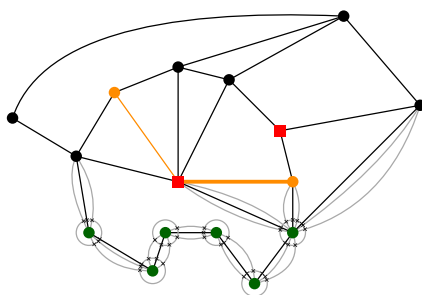
Boring edge removal and private face merging allow us to reduce the problem to a setting in which the number of faces of the skeleton is bounded.

► **Lemma 22** (†). *Let $\tilde{S}$ be the graph obtained from the skeleton of G_t by exhaustively applying Reduction Rule 4 and Reduction Rule 5, embedded according to the restriction of the unique embedding of the skeleton of G_t to $\tilde{S}$. $\tilde{S}$ has at most $\mathcal{O}(k^3)$ faces.*

So far, we have shown an upper bound only on the number of faces. To obtain a small number of vertices rather than faces, we apply the following reduction rule.

► **Reduction Rule 6** (Boring edge contraction). *Replace each induced path of length at least two in the skeleton, whose interior is free from interesting vertices, by a single edge. Further, we delete vertices of degree one that are not interesting vertices.*

Let $\tilde{G}$ be the graph obtained after applying Reduction Rule 4 and Reduction Rule 5 to G_t embedded in an arbitrary way. Then let $\tilde{G}'$ be the embedded graph obtained after exhaustively applying Reduction Rule 6. The sets of terminals occurring together on any face of $\tilde{G}$ do not change in $\tilde{G}'$. Hence, this is safe with respect to EMBEDDED FACE COVER NUMBER by the same argument as Proposition 21.



■ **Figure 6** $G^{\otimes U}$ with G from Figure 5 (right) and U the set of green vertices. Edges that are not subdivisions of original edges are colored gray, subdivision vertices are indicated as crosses. Notice that only one vertex of U would remain after boring edge contraction.

► **Proposition 23.** *Let $\tilde{G}$ be the graph obtained from G_t by exhaustively applying boring edge removal, private face merging and then boring edge contraction. For an arbitrary embedding of G_t , let $\tilde{G}$ be embedded according to the restriction of the embedding of G_t to $\tilde{G}$ in which edges that replace paths trace these paths at an ε -distance. Then, $\tilde{G}$ is a nice kernel of G_t for EMBEDDED FACE COVER NUMBER.*

Finally, we are at a point where we can start bounding the number of vertices.

► **Lemma 24** (†). *Let $\tilde{S}$ be the graph obtained from the skeleton of G_t by exhaustively applying boring edge removal, private face merging and then boring edge contraction. $|V(\tilde{S})| \in \mathcal{O}(k^3)$.*

We will now re-establish uniqueness of the embedding of our kernelized skeleton by *rigidizing* (i.e. adding structures to the embedded graph that render it 3-connected).

► **Definition 25.** *Let H be an embedded planar graph and $U \subseteq V(H)$. The rigidization of H around U is the embedded graph $H^{\otimes U}$ arising from H in the following way. For each edge $uv \in E(H)$ where $\{u, v\} \cap U \neq \emptyset$ introduce two new copies of uv , drawn at an ε -distance from uv . Subdivide each of uv and its copies by $|\{u, v\} \cap U|$ many vertices, which we refer to as subdivision vertices at ε -distance from $\{u, v\} \cap U$. Introduce the unique planar cycle on the subdivision vertices at ε -distance of each vertex in U .*

See Figure 6 for an example of the above definition. Rigidization does not significantly increase the size of a graph:

► **Observation 26** (†). *Let H be an embedded planar graph and $U \subseteq V(H)$. $|V(H^{\otimes U})| \in \mathcal{O}(|V(H)|)$.*

For us, the intended purpose of rigidization is to fix an embedding, which is formally supported by the following lemma.

► **Lemma 27** (†). *Let H be an arbitrary embedded connected planar graph, and U be the set of vertices that are contained in a separator of size at most two of H . $H^{\otimes U}$ is 3-connected.*

It is straightforward to verify that the graph $\tilde{S}$ obtained from the skeleton of G_t by exhaustively applying boring edge removal, private face merging and then boring edge contraction is connected. Hence $\tilde{S}^{\otimes U}$ where U is the set of all cutvertices and vertices in 2-separators of $\tilde{S}$ is planar and 3-connected by Lemma 27 and hence has a unique embedding which allows us to return from considering EMBEDDED FACE COVER NUMBER to FACE COVER NUMBER on abstract graphs. That safeness is maintained in this step crucially relies on the fact that we never rigidize around interesting vertices.

► **Lemma 28** (†). *Let $\tilde{S}$ be obtained from the skeleton of G_t by exhaustively applying boring edge removal, private face merging and then boring edge contraction. No vertex separator of size at most two in $\tilde{S}$ contains an interesting vertex.*

The following theorem essentially shows instance equivalence of what will be – up to the replacement of virtual components – the small nice kernel for G_t .

► **Theorem 29** (†). *Let $\tilde{G}$ and $\tilde{S}$ be the graphs obtained from G_t and its skeleton respectively by exhaustively applying boring edge removal, private face merging and then boring edge contraction. $\tilde{G}^{\otimes U}$ where U is the set of all cut-vertices and vertices that are contained in a separator of size two in $\tilde{S}$ is a nice kernel of G_t for FACE COVER NUMBER.*

Proof Sketch. (K1) is trivially satisfied by construction.

For arguing (K2) and (K3), fix an embedding of G_t in which its minimum face cover number using a specified number of external faces and covering a specified set of corners (we refer to this as *generalized face cover number*) can be realized. By Proposition 23, $\tilde{G}$ is a nice kernel for G_t as an instance for EMBEDDED FACE COVER NUMBER. By Lemma 27, $\tilde{S}^{\otimes U}$ has a unique embedding which necessarily coincides with the unique one of the skeleton of G_t on its restriction to $\tilde{S}$ where copies of edges are embedded at an ε -distance from the corresponding original edge and subdivision vertices are drawn in cycles at ε -distance around the corresponding vertices in U . Further, by construction, replacing the virtual edges in this embedding by the embeddings of the corresponding virtual components of G_t results in an embedding of $\tilde{G}^{\otimes U}$ in which the faces are the same as in the embedding of $\tilde{G}$ apart from some vertices in U being replaced by subdivision vertices in some face cycles and there being new face cycles consisting only of vertices in U and subdivision vertices. By Lemma 28, this means that the sets of terminals, corners and virtual edge endpoints from the skeleton of G_t that occur together on face boundaries are the same in the fixed embedding of $\tilde{G}$ and the described embedding of $\tilde{G}^{\otimes U}$. This means that the abstract graph $\tilde{G}^{\otimes U}$ has at most the same face cover number as the embedded graph $\tilde{G}$, i.e. the generalized face cover number of G_t as an abstract graph.

Conversely, we can also argue that the $\tilde{G}^{\otimes U}$ has at least the same generalized face cover as G_t . Similarly as the other direction of inequality, we can start from an embedding of $\tilde{G}^{\otimes U}$ and transform it to an embedding of G_t which realizes the same face cover number even when requiring the inclusion of a specific number of external faces. ◀

As a final step, we replace all virtual components by their corresponding constant-size gadgets if they are semiproblematic and their inductively assumed kernels if they are problematic. By Lemma 10 this results in a nice kernel. We can also show that this final nice kernel which we call K is also small.

► **Theorem 30** (†). *K as obtained by the process described in this subsection is small.*

5.2 S-node

In an S-node t , we first take care of the real edges in the skeleton and terminal-free virtual components as follows.

► **Reduction Rule 7.** *If there are at least two terminals in the skeleton of the S-node that are not corner vertices, turn all but one into non-terminals. If there is a path of length at least two consisting of real edges, replace it by a single real edge.*

► **Reduction Rule 8.** *Contract each terminal-free virtual component to a vertex.*

Our next task is to reduce the number of unproblematic components. If we consider only fcn_1 and fcn_2 , it would suffice to keep only one unproblematic component (since all the other ones will be covered by the outer faces that are in the face cover). However, in case of fcn_0 , we do not use any of the outer faces in the face cover, so we have to use internal faces of the unproblematic components. In this case, having at least $k + 1$ unproblematic components guarantees that fcn_0 will be larger than k .

If there is an unproblematic component without internal faces (e.g. a terminal subdivided edge), we say that $\text{fcn}_0 = \infty$, and we can show that the same holds after applying the reduction rules (i.e. after applying reduction rules, there is no face cover without external faces).

► **Reduction Rule 9.** *If there are at least $k + 1$ unproblematic virtual components, take all but $k + 1$ of them and contract all their edges. For each remaining unproblematic component, if it has a face cover consisting of one internal face, replace it by a triangle two of whose vertices are corner vertices of the component, and the third vertex is a terminal. Otherwise, replace it by a terminal subdivided edge.*

For the same reason, we need to keep $k + 1$ semi-problematic components:

► **Reduction Rule 10.** *If there are at least $k + 1$ semi-problematic virtual components, take all but $k + 1$ of them and contract all their edges. Replace the remaining semi-problematic components by using the gadget in Figure 2.*

Finally, we replace each unreplaced problematic virtual component by its inductively assumed kernel for its enhancement restricted to the virtual component.

► **Lemma 31** (†). *Consider an S -node t and let $\tilde{G}_t$ be the graph obtained by applying the reduction rules exhaustively. Then $\tilde{G}_t$ is a nice kernel for G_t .*

► **Lemma 32** (†). *After applying Reduction Rule 7, 8, 9, 10 exhaustively on G_t , we obtain a small nice kernel.*

5.3 P-node

Since we assume that G has no parallel edges, no leaf node of the SPR-tree of G is a P-node. While the graph G has no parallel edges, the application of certain reduction rules may create parallel edges. We deal with those edges as follows.

Let t be a parallel node and let G_t be the enhancement of the graph induced on the sub-SPR-tree rooted at t . In the following discussion, after applying each of the reduction rules exhaustively the graph obtained from G_t will be denoted by G'_t .

► **Reduction Rule 11.** *Delete all but one real edge of the skeleton of G_t from G_t .*

► **Proposition 33** (†). *Reduction Rule 11 creates a nice kernel.*

Trivially, there can be at most k problematic virtual components in G_t . We will next argue that we can bound the number of terminal-free, unproblematic and problematic virtual components linearly in k .

► **Reduction Rule 12.** *Delete all but one terminal-free virtual component from G_t .*

► **Proposition 34** (†). *Applying Reduction Rule 12 creates a nice kernel.*

We claim that after we apply Reduction rules 11 and 12 exhaustively, the number of virtual child components of a P-node is bounded linearly in k . We say that two virtual child components *share an internal face* if some internal face of the P-node skeleton is formed by merging external faces contained in the respective face covers of the child virtual components.

► **Proposition 35** (†). *If there are more than $4k + 2$ virtual components in G'_t , then there does not exist a face cover of size at most k in G .*

By the above proposition, the number of virtual edges in the skeleton of a P-node is $\mathcal{O}(k)$. Replacing the corresponding virtual components by small graphs in G'_t helps to bound its size. We now apply Reduction Rule 1 through Reduction Rule 3 on the graph G'_t exhaustively. Additionally, we replace every problematic virtual component by its small nice kernel.

► **Proposition 36** (†). *After applying Reduction Rules 11, 12 and the Reduction rules 1 to 3 exhaustively on G_t , the resulting graph is a small nice kernel.*

6 Conclusion

This paper presents the first polynomial kernel for the FACE COVER NUMBER problem, with a cubic dependence on the parameter. For a restricted version of the problem – where the embedding is provided as part of the input and the terminal set includes all vertices of the graph – a linear kernel was previously known. A natural direction for future research is to close the gap between the kernel sizes for the embedded and non-embedded variants of the problem.

Open Question 1. *Does there exist a linear kernel for FACE COVER NUMBER?*

Addressing this question likely necessitates a deeper understanding of the structural properties of 3-connected planar graphs.

Following the fixed-parameter tractable (FPT) algorithm developed by Bienstock and Monma [3], there have been no subsequent improvements for the non-embedded case. In particular, no subexponential-time algorithm is currently known for FACE COVER, prompting the following question:

Open Question 2. *Is there an algorithm that solves FACE COVER in $2^{\mathcal{O}(\sqrt{k})}$ time, where k is the face cover number?*

It is important to note that while bidimensionality theory applies to the variant with $T = V(G)$, the general form of the problem does not fall within this framework. Addressing Open Question 2 will therefore require the development of novel algorithmic ideas beyond existing techniques.

References

- 1 Faisal N. Abu-Khzam, Henning Fernau, and Michael A. Langston. A Bounded Search Tree Algorithm for Parameterized Face Cover. *Journal of Discrete Algorithms*, 6(4):541–552, 2008. doi:10.1016/J.JDA.2008.07.004.
- 2 Marshall Bern. Faster Exact Algorithms for Steiner Trees in Planar Networks. *Networks*, 20:109–120, 2006.
- 3 Daniel Bienstock and Clyde L. Monma. On the Complexity of Covering Vertices by Faces in a Planar Graph. *SIAM Journal of Computing*, 17(1):53–76, 1988. doi:10.1137/0217004.

- 4 Glencora Borradaile, Amir Nayyeri, and Farzad Zafarani. Towards Single Face Shortest Vertex-Disjoint Paths in Undirected Planar Graphs. In *Proc. ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 227–238. Springer, 2015. doi:10.1007/978-3-662-48350-3_20.
- 5 Gunnar Brinkmann. A simple and elementary proof of whitney’s unique embedding theorem. *Ars Math. Contemp.*, 20(1):195–197, 2021. doi:10.26493/1855-3974.2334.331.
- 6 Danny Chen and Xiadong Wu. Efficient Algorithms for k -Terminal Cuts on Planar Graphs. *Algorithmica*, 38:299–316, February 2004. doi:10.1007/S00453-003-1061-2.
- 7 Danny Z. Chen and Jinhui Xu. Shortest Path Queries in Planar Graphs. In *Proc. of STOC 2000*, pages 469–478. ACM, 2000. doi:10.1145/335305.335359.
- 8 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994. doi:10.1137/S0097539792225297.
- 9 Reinhard Diestel. *Graph Theory*. Springer Publishing Company, Incorporated, 5th edition, 2017.
- 10 Jeff Erickson and Amir Nayyeri. Shortest Non-Crossing Walks in the Plane. In *Proc. of SODA 2011*, pages 297–208. SIAM, 2011. doi:10.1137/1.9781611973082.25.
- 11 Ranel E. Erickson, Clyde L. Monma, and Arthur F. Veinott. Send-and-Split Method for Minimum-Concave-Cost Network Flows. *Mathematics of Operations Research*, 12(4):634–664, 1987. doi:10.1287/MOOR.12.4.634.
- 12 Henning Fernau and David W. Juedes. A geometric approach to parameterized algorithms for domination problems on planar graphs. In *Proc. MFCS 2004*, volume 3153 of *Lecture Notes in Computer Science*, pages 488–499. Springer, 2004. doi:10.1007/978-3-540-28629-5_37.
- 13 Greg N. Frederickson. Planar Graph Decomposition and All Pairs Shortest Paths. *Journal of ACM*, 38(1):162–204, 1991. doi:10.1145/102782.102788.
- 14 M. R. Garey and David S. Johnson. The Rectilinear Steiner Tree Problem is NP-Complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.
- 15 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual Approximation Algorithms for Integral Flow and Multicut in Trees. *Algorithmica*, 18:3–20, 1997. doi:10.1007/BF02523685.
- 16 Valentin Garnero, Ignasi Sau, and Dimitrios M. Thilikos. A Linear Kernel for Planar Red-Blue Dominating Set. *Discrete Applied Math*, 217:536–547, 2017. doi:10.1016/J.DAM.2016.09.045.
- 17 Thekla Hamm, Sukanya Pandey, and Krisztina Szilágyi. A polynomial kernel for face cover on non-embedded planar graphs, 2026. arXiv:2601.04169.
- 18 J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973. doi:10.1137/0202012.
- 19 John E. Hopcroft and Robert Endre Tarjan. Efficient Algorithms for Graph Manipulation [H] (algorithm 447). *Commun. ACM*, 16(6):372–378, 1973. doi:10.1145/362248.362272.
- 20 Sándor Kisfaludi-Bak, Jesper Nederlof, and Erik Jan van Leeuwen. Nearly ETH-tight algorithms for planar steiner tree with terminals on few faces. *ACM Trans. Algorithms*, 16(3):28:1–28:30, 2020. doi:10.1145/3371389.
- 21 Ton Kloks, Chuan-Min Lee, and Jiping Liu. New Algorithms for k -Face cover, k -Feedback Vertex set, and k -Disjoint cycles on plane and planar graphs. In Ludek Kucera, editor, *Proc. WG 2002*, volume 2573 of *Lecture Notes in Computer Science*, pages 282–295. Springer, 2002. doi:10.1007/3-540-36379-3_25.
- 22 Athanassios Koutsonas and Dimitrios M. Thilikos. Planar feedback vertex set and face cover: Combinatorial bounds and subexponential algorithms. *Algorithmica*, 60(4):987–1003, 2011. doi:10.1007/S00453-010-9390-4.
- 23 Mark R. Kramer and Jan van Leeuwen. The Complexity of Wire-Routing and Finding Minimum Area Layouts for Arbitrary VLSI Circuits. *Advances in Computing Research*, 2:129–146, 1984.

- 24 Robert Krauthgamer, James R. Lee, and Havana Rika. Flow-Cut Gaps and Face Covers in Planar Graphs. In *Proc. of SODA 2019*, pages 525–534. SIAM, 2019. doi:10.1137/1.9781611975482.33.
- 25 Robert Krauthgamer and Havana Inbal Rika. Refined Vertex Sparsifiers of Planar Graphs. *SIAM Journal on Discrete Mathematics*, 34(1):101–129, 2020. doi:10.1137/17M1151225.
- 26 James F. Lynch. The equivalence of theorem proving and the interconnection problem. *SIGDA Newsletter*, 5(3):31–36, 1975.
- 27 Sukanya Pandey and Erik Jan van Leeuwen. Planar Multiway Cut with Terminals on Few Faces. In *Proc. SODA 2022*, pages 2032–2063. SIAM, 2022. doi:10.1137/1.9781611977073.81.
- 28 Werner Schwärzler. On the complexity of the planar edge-disjoint paths problem with terminals on the outer boundary. *Combinatorica*, 29(1):121–126, 2009. doi:10.1007/S00493-009-2407-4.
- 29 Jens Vygen. NP-Completeness of Some Edge-Disjoint Paths Problems. *Discrete Applied Mathematics*, 61(1):83–90, 1995. doi:10.1016/0166-218X(93)E0177-Z.