Decomposition of Map Graphs with Applications

Fedor V. Fomin
University of Bergen, Norway
fomin@ii.uib.no

Daniel Lokshtanov
University of California, Santa Barbara, USA
daniello@ucsb.edu

Fahad Panolan
University of Bergen, Norway
fahad.panolan@ii.uib.no

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University of the Negev, Beer-Sheva, Israel
meiravze@bgu.ac.il

Abstract

Bidimensionality is the most common technique to design subexponential-time parameterized algorithms on special classes of graphs, particularly planar graphs. The core engine behind it is a combinatorial lemma of Robertson, Seymour and Thomas that states that every planar graph either has a $\sqrt{k} \times \sqrt{k}$-grid as a minor, or its treewidth is $\mathcal{O}(\sqrt{k})$. However, bidimensionality theory cannot be extended directly to several well-known classes of geometric graphs like unit disk or map graphs. This is mainly due to the presence of large cliques in these classes of graphs. Nevertheless, a relaxation of this lemma has been proven useful for unit disk graphs. Inspired by this, we prove a new decomposition lemma for map graphs, the intersection graphs of finitely many simply-connected and interior-disjoint regions of the Euclidean plane. Informally, our lemma states the following. For any map graph G, there exists a collection $(U_1, \ldots, U_t)$ of cliques of G with the following property: *G either contains a $\sqrt{k} \times \sqrt{k}$-grid as a minor, or it admits a tree decomposition where every bag is the union of $\mathcal{O}(\sqrt{k})$ cliques in the above collection.*

The new lemma appears to be a handy tool in the design of subexponential parameterized algorithms on map graphs. We demonstrate its usability by designing algorithms on map graphs with running time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ for Connected Planar $\mathcal{F}$-Deletion (that encompasses problems such as Feedback Vertex Set and Vertex Cover). Obtaining subexponential algorithms for Longest Cycle/Path and Cycle Packing is more challenging. We have to construct tree decompositions with more powerful properties and to prove sublinear bounds on the number of ways an optimum solution could "cross" bags in these decompositions.

For Longest Cycle/Path, these are the first subexponential-time parameterized algorithm on map graphs. For Feedback Vertex Set and Cycle Packing, we improve upon known $2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$-time algorithms on map graphs.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Computational geometry

Keywords and phrases Longest Cycle, Cycle Packing, Feedback Vertex Set, Map Graphs, FPT

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.60

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `http://arxiv.org/abs/1903.01291`.

Acknowledgements This work is supported by the European Research Council (ERC) via grant LOPPRE, reference 819416, the Norwegian Research Council via project MULTIVAL, and Israel Science Foundation individual research grant no. 1176/18.

1 Introduction

In this paper, we develop new proof techniques to design parameterized subexponential-time algorithms for problems on map graphs, particularly problems that involve hitting or connectivity constraints. The class of map graphs was introduced by Chen, Grigni, and Papadimitriou [7, 8] as a modification of the class of planar graphs. Roughly speaking, map graphs are graphs whose vertices represent countries in a map, where two countries are considered adjacent if and only if their boundaries have at least one point in common; this common point can be a single common point rather than necessarily an edge as standard planarity requires. Formally, a *map* $\mathcal{M}$ is a pair $(\mathscr{E}, \omega)$ defined as follows : $\mathscr{E}$ is a plane graph (i.e., a planar graph with an embedding) where each connected component of $\mathscr{E}$ is biconnected, and ω is a function that maps each face f of $\mathscr{E}$ to 0 or 1. A face f of $\mathscr{E}$ is called *nation* if $\omega(f) = 1$ and *lake* otherwise. The graph associated with $\mathcal{M}$ is the simple graph G where $V(G)$ consists of the nations of $\mathcal{M}$, and $E(G)$ contains $\{f_1, f_2\}$ for every pair of faces f_1 and f_2 that are adjacent (that is, share at least one vertex). Accordingly, a graph G is called a map graph if there exists a map $\mathcal{M}$ such that G is the graph associated with $\mathcal{M}$.

Every planar graph is a map graph [7, 8], but the converse does not hold true. Moreover, map graphs can have cliques of any size and thus they can be "highly non-planar". These two properties of map graphs can be contrasted with those of H-minor free graphs and unit disk graphs: the class of H-minor free graphs generalizes the class of planar graphs, but can only have cliques of constant size (where the constant depends on H), while the class of unit disk graphs does not generalize the class of planar graphs, but can have cliques of any size. At least in this sense, map graphs offer the best of both worlds. Nevertheless, this comes at the cost of substantial difficulties in the design of efficient algorithms on them.

Arguably, the two most natural and central algorithmic questions concerning map graphs are as follows. First, we would like to efficiently recognize map graphs, that is, determine whether a given graph is a map graph. In 1998, Thorup [29] announced the existence of a polynomial-time algorithm for map graph recognition. Although this algorithm is complicated and its running time is about $\mathcal{O}(n^{120})$, where n is the number of vertices of the input graph, no improvement has yet been found; the existence of a simpler or faster algorithm for map graph recognition has so far remained an important open question in the area (see, e.g., [9]).

The second algorithmic question – or rather family of algorithmic questions – concerns the design of efficient algorithms for various optimization problems on map graphs. Most well-known problems that are NP-complete on general graphs remain NP-complete when restricted to planar (and hence on map) graphs. Nevertheless, a large number of these problems can be solved faster or "better" when restricted to planar graphs. For example, nowadays we know of many problems that are APX-hard on general graphs, but which admit polynomial time approximation schemes (PTASes) or even efficient PTASes (EPTASes) on planar graphs (see, e.g., [4, 14, 15, 22]). Similarly, many parameterized problems that on general graphs cannot be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ unless the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi and Zane [24] fails, admit parameterized subexponential-time algorithms on planar graphs (see, e.g., [1, 2, 14, 27]). It is compelling to ask whether the algorithmic results and techniques for planar graphs can be extended to map graphs.

For approximation algorithms, Chen [6] and Demaine et al. [12] developed PTASes for the Maximum Independent Set and Minimum r-Dominating Set problems on map graphs. Moreover, Fomin et al. [21, 22] developed an EPTAS for Treewidth-η Modulator for any fixed constant $\eta \geq 0$, which encompasses Feedback Vertex Set (FVS) and Vertex Cover (VC). For parameterized subexponential-time algorithms on map graphs, the situation is less explored. While on planar graphs there are general algorithmic methods – in particular, the powerful theory of bidimensionality [15, 13] – to design parameterized subexponential-time algorithms, we are not aware of any general algorithmic method that can be easily adapted to map graphs. Demaine et al. [12] gave a parameterized algorithm for Dominating Set, and more generally for (k, r)-Center, with running time $2^{\mathcal{O}(r \log r\sqrt{k})} n^{\mathcal{O}(1)}$ on map graphs. Moreover, Fomin et al. [21, 22] gave $2^{\mathcal{O}(k^{0.75} \log k)} n^{\mathcal{O}(1)}$-time parameterized algorithms for FVS and Cycle Packing on map graphs. Additionally, Fomin et al. [21, 22] noted that the same approach yields $2^{\mathcal{O}(k^{0.75} \log k)} n^{\mathcal{O}(1)}$-time parameterized algorithms for Vertex Cover and Connected Vertex Cover (CVC) on map graphs. However, the existence of a parameterized subexponential-time algorithm for Longest Path/Cycle on map graphs was left open. (In these problems we are asked whether an n-vertex graph contains a path/cycle of length at least k.) Furthermore, time complexities of $2^{\mathcal{O}(k^{0.75} \log k)} n^{\mathcal{O}(1)}$, although having subexponential dependency on k, remain far from time complexities of $2^{\mathcal{O}(\sqrt{k} \log k)} n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$ that commonly arise for planar graphs [27]. We remark that time complexities of $2^{\mathcal{O}(\sqrt{k} \log k)} n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$ are particularly important since they are often known to be essentially optimal under the aforementioned ETH [27].

In the field of Parameterized Complexity, Longest Path/Cycle , FVS and Cycle Packing serve as testbeds for development of fundamental algorithmic techniques such as color-coding [3], methods based on polynomial identity testing [25, 26, 30, 5], cut-and-count [11], and methods based on matroids [19]. By combining the bidimensionality theory of Demaine et al. [13] with efficient algorithms on graphs of bounded treewidth [17, 10], Longest Path/Cycle, Cycle Packing and FVS are solvable in time $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$ on planar graphs. Furthermore, the parameterized subexponential-time "tractability" of these problems can be extended to graphs excluding some fixed graph as a minor [15].

Our results. We design parameterized subexponential-time algorithms with running time $2^{\mathcal{O}(\sqrt{k} \log k)} \cdot n^{\mathcal{O}(1)}$ for a number of natural and well-studied problems on map graphs.

Let $\mathcal{F}$ be a family of connected graphs that contains at least one planar graph. Then Connected Planar $\mathcal{F}$-Deletion (or just $\mathcal{F}$-Deletion) is defined as follows. The input is a graph G and a non-negative integer k, and our objective is to test whether there exists a set S of at most k vertices such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor. $\mathcal{F}$-Deletion is a general problem and several problems such as VC, FVS, Treewidth-η Vertex Deletion, Pathwidth-η Vertex Deletion, Treedepth-η Vertex Deletion, Diamond Hitting Set and Outerplanar Vertex Deletion are its special cases. We give the first parameterized subexponential algorithm for this problem on map graphs, which runs in time $2^{\mathcal{O}(\sqrt{k} \log k)} \cdot n^{\mathcal{O}(1)}$. Our approach for $\mathcal{F}$-Deletion also directly extends to yield $2^{\mathcal{O}(\sqrt{k} \log k)} \cdot n^{\mathcal{O}(1)}$-time parameterized algorithms for CVC and Connected Feedback Vertex Set (CFVS) on map graphs. (In this versions we are asked if there is a *connected* vertex cover or a feedback vertex set of size at most k.)

With additional ideas, we derive the first subexponential-time parameterized algorithm on map graphs for Longest Path/Cycle. Our technique also allows to improve the running time for Cycle Packing (does a map graph contains at least k vertex-disjoint cycles) from $2^{\mathcal{O}(k^{0.75} \log k)} \cdot n^{\mathcal{O}(1)}$ to $2^{\mathcal{O}(\sqrt{k} \log k)} \cdot n^{\mathcal{O}(1)}$. Our results are summarized in Table 1.

Our methods. The starting point of our study is the technique of bidimensionality [15, 13]. The core engine behind this technique is a combinatorial lemma of Robertson, Seymour and Thomas [28] that states that every planar graph either has a $\sqrt{k} \times \sqrt{k}$-grid as a minor, or its treewidth is $\mathcal{O}(\sqrt{k})$. Unfortunately, a clique on $k - 1$ vertices has no $\sqrt{k} \times \sqrt{k}$-grid as a minor

Table 1 Parameterized complexity of problems on map graphs. For $\mathcal{F}$-DELETION, LONGEST CYCLE, and LONGEST PATH no faster (than on general graphs) algorithms were known.

	Our results	Previous work
(CONNECTED) VERTEX COVER	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$ [22]
(CONNECTED) FEEDBACK VERTEX SET	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$ [22]
$\mathcal{F}$-DELETION	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ [18]
LONGEST CYCLE/PATH	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ [10]
CYCLE PACKING	$2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$	$2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$ [22]

and its treewidth is $k-2$. Because classes of geometric graphs such as unit disk graphs and map graphs can have arbitrarily large cliques, the combinatorial lemma is inapplicable to them. Nevertheless, a relaxation of this lemma has been proven useful for unit disk graphs. Specifically, every unit disk graph G has a natural partition $(U_1,\ldots,U_t)$ of $V(G)$ such that each part induces a clique with "nice" properties – in particular, it has neighbors only in a *constant number* (to be precise, this constant is at most 24) of other parts; it was shown that G either has a $\sqrt{k}\times\sqrt{k}$-grid as a minor, or it has a tree decomposition where every bag is the union of $\mathcal{O}(\sqrt{k})$ of these cliques [20]. In particular, given a parameterized problem where any two cliques have constant-sized "interaction" in a solution, it is implied that any bag has $\mathcal{O}(\sqrt{k})$-sized "interaction" with all other bags in a solution. For any map graph G, there also exists a natural collection of subsets of $V(G)$ that induce cliques with "nice" properties. However, not only are these cliques not vertex disjoint, but each of these cliques can have neighbors in *arbitrarily many* other cliques.

In this paper, we first prove that every map graph either has a $\sqrt{k}\times\sqrt{k}$-grid as a minor, or it has a tree decomposition where every bag is the union of $\mathcal{O}(\sqrt{k})$ of the cliques in the above collection. For $\mathcal{F}$-DELETION, CVC, and CFVS, this combinatorial lemma alone already suffices to design $2^{\mathcal{O}(\sqrt{k}\log k)}\cdot n^{\mathcal{O}(1)}$-time algorithms on map graphs. Indeed, we can choose a fixed constant $c>0$ so that in case we have a $c\sqrt{k}\times c\sqrt{k}$-grid as a minor, there does not exist a solution, and otherwise we can solve the problem by using dynamic programming over the given tree decomposition. Specifically, since every bag is the union of $\mathcal{O}(\sqrt{k})$ cliques, and the size of each clique is upper bounded by $\mathcal{O}(k)$ (once we know that no $c\sqrt{k}\times c\sqrt{k}$-grid exists), only $\mathcal{O}(\sqrt{k})$ vertices in the bag are not to be taken into a solution – there are only $2^{\mathcal{O}(\sqrt{k}\log k)}$ choices to select these vertices, and once they are selected, the information stored about the remaining vertices is the same as in normal dynamic programming over a tree decomposition of $\mathcal{O}(\sqrt{k})$ width.

This approach already substantially improves upon the previously best known algorithms for FVS, VC and CVC of Fomin et al. [21, 22]. However, $2^{\mathcal{O}(\sqrt{k}\log k)}\cdot n^{\mathcal{O}(1)}$-time algorithms for LONGEST PATH/CYCLE and CYCLE PACKING on map graphs require more efforts. The main reason why we cannot apply the same arguments as for unit disk graphs is the following. Recall that for unit disk graphs, given a parameterized problem where any two cliques have constant-sized "interaction" in a *solution* (in our case, this means a path/cycle on at least k vertices, or a cycle packing of k cycles), it is implied that any bag has $\mathcal{O}(\sqrt{k})$-sized "interaction" with all other bags in a solution. Here, interaction between two cliques refers to the number of edges in a solution "passing" between these two cliques; similarly, interaction between a bag B and a collection of other bags refers to the number of edges in a solution that have one endpoint in B and the other endpoint in some bag in the collection. In this context, dealing with map graphs is substantially more difficult than dealing with unit disk graphs. In map graphs vertices in a clique can have neighbors in arbitrarily many other cliques in the collection rather than only in a constant number as in unit disk graphs. This is why it is difficult to obtain an $\mathcal{O}(\sqrt{k})$-sized "interaction" as for unit disk graphs.

Hence, we are forced to take a different approach for map graphs by bounding "the interaction within a clique across all the bags of a decomposition". Towards this, we first need to strengthen our tree decomposition. To explain the new properties required, we note that every clique in the aforementioned collection of cliques, say $\mathcal{K}$, is either a single vertex or the neighborhood of some "special vertex" in an exterior bipartite graph (see Section 2). Further, every vertex of G occurs as a singleton in $\mathcal{K}$. We construct our decomposition in a way such that every bag is not necessarily a union of $\mathcal{O}(\sqrt{k})$ cliques in $\mathcal{K}$, but a union of carefully chosen subcliques of $\mathcal{O}(\sqrt{k})$ cliques in $\mathcal{K}$ (with one subclique for each of these $\mathcal{O}(\sqrt{k})$ cliques); subcliques of the same clique chosen in different bags may be different. We then prove properties that roughly state that, if we look at the collection of bags that include some vertex v of G, then this collection induces a subtree and a path as follows: ($\clubsuit$) *the subtree consists of the bags that correspond to the singleton clique v, and the path goes "upwards" (in the tree decomposition) from the root of this subtree.* We thereby implicitly derive that in every bag B, every subclique of size larger than 1 can only have as neighbors vertices that are *(i)* in the bag B itself or in one of its descendants, or *(ii)* in cliques that have a subclique in the bag B. In particular, this means that if we prove that there exists a solution such that for any clique K in $\mathcal{K}$, the number of edges in $E(K)$ that "cross any bag B" (i.e., the edges in $E(K)$ with one endpoint in B and the other in the collection of all bags that are not descendants of B) is a constant, then we obtain a bound of $\mathcal{O}(\sqrt{k})$ on the interaction between any bag B and the collection of all bags that are not descendants of B. We prove the mentioned statement using property ($\clubsuit$). The proof that such a property simultaneously holds for all cliques and all bags is the most challenging part of the proof.

In Section 3 we give our special tree decomposition of map graphs and in Section 4 we explain its application in the algorithm for Longest Cycle on map graphs. For the proofs of results marked with $\star$ and all other results we refer to the full version of the paper.

2 Preliminaries

For any $t \in \mathbb{N}$, we use $[t]$ and $[t]_0$ as shorthands for $\{1, 2, \ldots, t\}$ and $\{0, 1, \ldots, t\}$, respectively. For a set U, we use 2^U to denote the power set of U. For a sequence $\sigma = x_1x_2 \ldots x_n$ and any $1 \leq i \leq j \leq n$, the sequence $\sigma' = x_i \ldots x_j$ is called a *segment* of σ. For a sequence $\sigma = x_1x_2 \ldots x_n$ and a subset $Z \subseteq \{x_1, \ldots, x_n\}$, the restriction of σ on Z, denoted by $\sigma|_Z$, is the sequence obtained from σ by deleting the elements of $\{x_1, \ldots, x_n\} \setminus Z$.

Graphs. We use standard notation and terminology from the book of Diestel [16] for graph-related terms. Given a graph G, let $V(G)$ and $E(G)$ denote its vertex-set and edge-set, respectively. For a set $\mathcal{Q}$ of graphs we slightly abuse terminology and let $V(\mathcal{Q})$ and $E(\mathcal{Q})$ denote the union of the sets of vertices and edges of the graphs in $\mathcal{Q}$, respectively. For a vertex subset $X \subseteq V(G)$ in a graph G, $E(X)$ denotes the set $\{\{u, v\} \in E(G) \colon u, v \in X\}$. For a graph G and a degree-2 vertex $v \in V(G)$, by *contracting* v, we mean deleting v from G and adding an edge between the two neighbors of v in G.

A binary tree is a rooted tree where each node has at most two children. In a labelled binary tree, for each node with two children one of the children is labelled as "left child" and the other child is labelled as "right child". A *postorder transversal* of a labelled binary tree T is the sequence σ of $V(T)$ where for each node $t \in V(T)$, t appears after all its descendants, and if t has two children, then the nodes in the subtree rooted at the left child appear before the nodes in the subtree rooted at the right child. For a binary tree T, we say that a sequence σ of $V(T)$ is a postorder transversal if there is a labelling of T such that σ is its postorder transversal.

▶ **Definition 2.1** (Treewidth). *A* tree decomposition *of a graph G is a pair $\mathcal{T} = (T_{\mathcal{T}}, \beta_{\mathcal{T}})$, where T is a rooted tree and $\beta_{\mathcal{T}}$ is a function from $V(T_{\mathcal{T}})$ to $2^{V(G)}$, that satisfies the following three conditions. (We use the term* nodes *to refer to the vertices of $T_{\mathcal{T}}$.)*
(a) $\bigcup_{x \in V(T_{\mathcal{T}})} \beta_{\mathcal{T}}(x) = V(G)$.
(b) *For every edge $\{u, v\} \in E(G)$, there exists $x \in V(T_{\mathcal{T}})$ such that $\{u, v\} \subseteq \beta_{\mathcal{T}}(x)$.*
(c) *For every vertex $v \in V(G)$, the set of nodes $\{t \in V(T_{\mathcal{T}}) : v \in \beta_{\mathcal{T}}(t)\}$ induces a (connected) subtree of $T_{\mathcal{T}}$.*
The width *of $\mathcal{T}$ is $\max_{x \in V(T_{\mathcal{T}})} |\beta_{\mathcal{T}}(x)| - 1$. Each set $\beta_{\mathcal{T}}(x)$ is called a* bag*. Moreover, $\gamma_{\mathcal{T}}(x)$ denotes the union of the bags of x and its descendants. The* treewidth *of G is the minimum width among all possible tree decompositions of G, and it is denoted by* $\mathsf{tw}(G)$.

▶ **Definition 2.2.** *A tree decomposition $\mathcal{T} = (T_{\mathcal{T}}, \beta_{\mathcal{T}})$ of a graph G is* nice *if for the root r of $T_{\mathcal{T}}$, it holds that $\beta_{\mathcal{T}}(r) = \emptyset$, and each node $v \in V(T_{\mathcal{T}})$ is of one of the following types.*
- ***Leaf**: v is a leaf in $T_{\mathcal{T}}$ and $\beta_{\mathcal{T}}(v) = \emptyset$. This bag is labelled with **leaf**.*
- ***Forget vertex**: v has exactly one child u, and there exists a vertex $w \in \beta_{\mathcal{T}}(u)$ such that $\beta_{\mathcal{T}}(v) = \beta_{\mathcal{T}}(u) \setminus \{w\}$. This bag is labelled with **forget** (w).*
- ***Introduce vertex**: v has exactly one child u, and there exists a vertex $w \in \beta_{\mathcal{T}}(v)$ such that $\beta_{\mathcal{T}}(v) \setminus \{w\} = \beta_{\mathcal{T}}(u)$. This bag is labelled with **introduce**(w).*
- ***Join**: v has exactly two children, u and w, and $\beta_{\mathcal{T}}(v) = \beta_{\mathcal{T}}(u) = \beta_{\mathcal{T}}(w)$. This bag is labelled with **join**.*

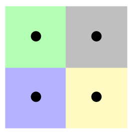
We will use the following folklore observation and proposition in the later sections.

▶ **Observation 2.3.** *Let $\mathcal{T}$ be a nice tree decomposition of a graph G. For any $v \in V(G)$, there is exactly one node $t \in V(T_{\mathcal{T}})$ such that t is labelled with **forget**(v).*

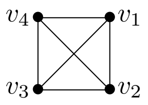
▶ **Proposition 2.4** (Theorem 7.23 in [10], [23, 28]). *There exists an $\mathcal{O}(n^2)$ time algorithm that given an n-vertex planar graph G and $t \in \mathbb{N}$, either outputs a (nice) tree decomposition of G of width less than $5t$, or constructs a $t \times t$ grid minor in G.*

Map graphs. Map graphs are the intersection graphs of finitely many connected and interior-disjoint regions of the Euclidean plane. Map graphs can be represented as the *half-squares of planar bipartite graphs*. For a bipartite graph B with bipartition $V(B) = W \uplus U$, the half-square of B is the graph G with vertex set W and edge set is defined as follows: two vertices in W are adjacent in G if they are at distance 2 in B. It is known that the half-square of a planar bipartite graph is a map graph [7, 8]. Moreover, for any map graph G, there exists a planar bipartite graph B such that G is a half-square of B [7, 8]; we refer to such B as a planar bipartite graph *corresponding* to the map graph G (see Figure 1).

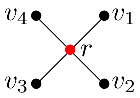
Throughout this paper, we assume that any input map graph G is given with a corresponding planar bipartite graph B. This assumption is made without loss of generality in the sense that if G is given with an embedding instead to witness that it is a map graph, then B is easily computable in linear time [7, 8]. We remark that we consider map graphs as simple graphs, that is, there are no multiple edges between two vertices u and v, even if there are two or more internally vertex-disjoint paths of length 2 between u and v in B. For a map graph G with a corresponding planar bipartite graph B having bipartition $V(B) = W \uplus U$, we refer to the vertices in $W = V(G)$ simply as *vertices* and the vertices in U as *special vertices*.



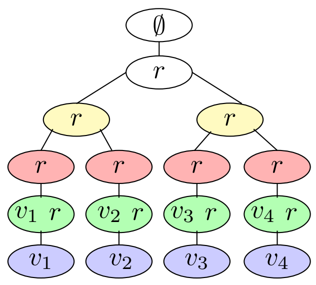
(a) A map $\mathcal{M}$.



(b) A map graph G associated with $\mathcal{M}$.



(c) A corresponding planar bipartite graph B of the map graph G. Here, red colored vertices are special vertices.



(d) A tree decomposition $\mathcal{D} = (T_{\mathcal{D}}, \beta_{\mathcal{D}})$ of B of width < 2. The nodes colored blue, green, red, and yellow are labelled with **introduce**(v_i), **introduce**(r), **forget**(v_i) and **join**, resp.

Figure 1 Example of a map graph G, and a corresponding planar bipartite graph B. Figure 1d represents a tree decomposition of B (obtained after deleting the leaves of a nice tree decomposition).

Moreover, we denote the special vertices by $S(G)$. Notice that for any $s \in S(G)$, $N_B(s)$ forms a clique in G; we refer to these cliques as *special cliques* of G. We remark that the collection $\mathcal{K}$ of cliques mentioned in Section 1 refers to $\{N_B(s) \colon s \in S(G)\} \cup \{\{v\} \colon v \in V(G)\}$.

3 Few Cliques Tree Decomposition of Map Graphs

In this section, we define a special tree decomposition for map graphs. This decomposition will be derived from a tree decomposition of the bipartite planar graph corresponding to the given map graph. Once we have defined our new decomposition, we will gather a few of its structural properties that will be useful in designing fast subexponential time algorithms.

▶ **Definition 3.1.** *Let G be a map graph with a corresponding planar bipartite graph B. Let $\mathcal{D} = (T_{\mathcal{D}}, \beta_{\mathcal{D}})$ be a tree decomposition of B of width less than ℓ. A pair $\mathcal{D}' = (T_{\mathcal{D}'}, \beta_{\mathcal{D}'})$ is called the ℓ-*few cliques tree decomposition derived from $\mathcal{D}$, or simply an $(\ell, \mathcal{D})$-FewCliTD, if it is constructed as follows (see Figure 2).*

1. *The tree $T_{\mathcal{D}'}$ is equal to $T_{\mathcal{D}}$. Whenever $\mathcal{D}'$ and $\mathcal{D}$ are clear from context, we denote both $T_{\mathcal{D}'}$ and $T_{\mathcal{D}}$ by T.*
2. *For each node $t \in V(T)$, $\beta_{\mathcal{D}'}(t) = (\beta_{\mathcal{D}}(t) \cap V(G)) \cup (\bigcup_{s \in \beta_{\mathcal{D}}(t) \cap S(G)} N_B(s) \cap \gamma_{\mathcal{D}}(t))$. That is, for each node $t \in V(T)$, we derive $\beta_{\mathcal{D}'}(t)$ from $\beta_{\mathcal{D}}(t)$ by replacing every special vertex $s \in \beta_{\mathcal{D}}(t) \cap S(G)$ by $N_B(s) \cap \gamma_{\mathcal{D}}(t)$.*

In words, the second item states that for every vertex $v \in V(G)$ and node $t \in V(T)$, we have that $v \in \beta_{\mathcal{D}'}(t)$ if and only if either (i) $v \in \beta_{\mathcal{D}}(t) \cap V(G)$ or (ii) $v \in N_B(s)$ for some $s \in S(G) \cap \beta_{\mathcal{D}}(t)$ and $v \in \beta_{\mathcal{D}}(t')$ for some node t' in the subtree of T rooted at t.

We can prove that the $(\ell, \mathcal{D})$-FewCliTD $(T, \beta_{\mathcal{D}'})$ in Definition 3.1 is a tree decomposition of G (see the full version of the paper for a proof). We remark that if we replace the term $N_B(s) \cap \gamma_{\mathcal{D}}(t)$ by the term $N_B(s)$ in the second item of Definition 3.1, then we still derive a tree decomposition, but then some of the properties proved later do not hold true.

To simplify statements ahead, from now on, we have the following notation.

Throughout the section, we fix a map graph G, a corresponding planar bipartite graph B of G, an integer $\ell \in \mathbb{N}$, a nice tree decomposition $\mathcal{D}$ of B of width less than ℓ and an *ℓ-few cliques tree decomposition* $\mathcal{D}'$ of G derived from $\mathcal{D}$ using Definition 3.1

Recall that $T = T_{\mathcal{D}} = T_{\mathcal{D}'}$ and that for each node $t \in V(T)$, $\beta_{\mathcal{D}'}(t)$ was obtained from $\beta_{\mathcal{D}}(t)$ by replacing every special vertex $s \in S(G)$ with $N_B(s) \cap \gamma_{\mathcal{D}}(s)$.

▶ **Definition 3.2.** *For a node $t \in V(T)$, we use Original(t) to denote the set $\beta_{\mathcal{D}}(t) \cap \beta_{\mathcal{D}'}(t)$, Fake$(t)$ to denote the set $\beta_{\mathcal{D}'}(t) \setminus \beta_{\mathcal{D}}(t)$, and Cliques$(t)$ to denote the set $\{N_B(s) \colon s \in S(G) \cap \beta_{\mathcal{D}}(t)\}$ of special cliques of G.*

Informally, for a node $t \in V(T)$, Original(t) denotes the set of vertices of $V(G)$ present in the bag $\beta_{\mathcal{D}}(t)$, Fake(t) denotes the set of "new" vertices added to $\beta_{\mathcal{D}'}(t)$ while replacing special vertices in $\beta_{\mathcal{D}}(t)$, and Cliques(t) is the set of special cliques in G that consist of one for each special vertex $s \in \beta_{\mathcal{D}}(t)$. For example, let t be the node in Figure 1d that is labelled with **forget**(v_1) by $\mathcal{D}$. Then, Original$(t) = \emptyset$, Fake$(t) = \{v_1\}$ and Cliques$(t) = \{\{v_1, \dots, v_4\}\}$.

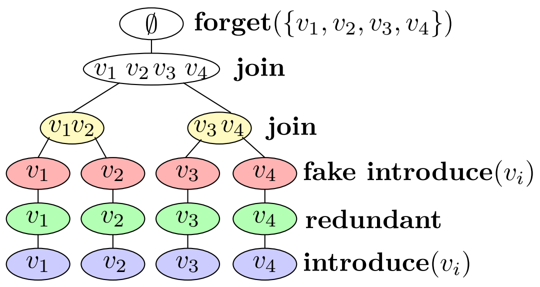
In the remainder of this section we prove properties related to $\mathcal{D}$ and $\mathcal{D}'$, which we use later in the paper. Towards the formulation of the first property, consider the tree decomposition $\mathcal{D}'$ in Figure 2 and the set of its nodes whose bags contain the vertex v_1 as a "fake" vertex. This set of nodes forms a path with one end-vertex being the unique node t_{v_1} of T labelled with **forget**(v_1) by $\mathcal{D}$ and the other end-vertex being an ancestor of t_{v_1}. In fact, the set of nodes $Q = \{t \in V(T) \colon v_1 \in \mathsf{Fake}(t) \text{ and } r \in \beta_{\mathcal{D}}(t)\}$ forms the unique path in T from t_{v_1} to t_r where t_r is the unique child of the node labelled with **forget**(r) by $\mathcal{D}$. This observation is abstracted and formalized in the following lemma.

▶ **Lemma 3.3.** *Let $v \in V(G)$ and $s \in S(G)$ such that $v \in N_B(s)$ and $Q = \{t \in V(T) \colon v \in$ Fake(t) and $s \in \beta_{\mathcal{D}}(t)\} \neq \emptyset$. Let x be the node in T labelled with **forget**(v) by $\mathcal{D}$, and y be the unique child of the node labelled with **forget**(s) by $\mathcal{D}$. Then, y is an ancestor of x, and Q induces a path in T which is the unique path between x and y in T.*

Proof. First, we prove that Q induces a (connected) subtree of T. Suppose not. Then, there exist two connected components C_1 and C_2 of $T[Q]$ such that there exists a path P in T from a vertex in C_1 to a vertex in C_2 whose internal vertices all belong to $V(T) \setminus Q$. By Property (c) of the tree decomposition $\mathcal{D}$, we have that $s \in \beta_{\mathcal{D}}(t)$ for any $t \in V(P)$. Moreover, there is an internal vertex w of P such that w is an ancestor of one of the end-vertices of P. This implies that $v \in \gamma_{\mathcal{D}}(w)$, because v belong to the bags of the endpoints of P (by the definition of Q and Fake). As we have also shown that $s \in \beta_{\mathcal{D}}(t)$ for all $t \in V(P)$, this implies that $w \in Q$, which is a contradiction. Hence, we have proved that $T[Q]$ is connected.

Next, we prove that $T[Q]$ is a path such that one of its endpoints is a descendant of the other. Towards this, it is enough to prove that (i) for any distinct $t, t' \in Q$, either t is a descendant of t' or t' is a descendant of t. For the sake of contradiction, assume that there exist $t, t' \in Q$ such that neither t is a descendent of t' nor t' is a descendent of t. By the definition of Q and because $t, t' \in Q$, we have that $v \in \gamma_{\mathcal{D}}(t)$ and $v \in \gamma_{\mathcal{D}}(t')$. Thus by Property (c) of the tree decomposition $\mathcal{D}$, we have that $v \in \beta_{\mathcal{D}}(t)$ and $v \in \beta_{\mathcal{D}}(t')$. Because $v \in \mathsf{Fake}(t)$ and $v \in \mathsf{Fake}(t')$, this is a contradiction to the definition of Fake.

It remains to prove that y is an ancestor of x and that x and y are endpoints of $T[Q]$. First, we prove that x is an end-vertex of the path $T[Q]$. Let x' be the only child of x. To prove x is an end-vertex of the path $T[Q]$, it is enough to show that $x \in Q$ and $x' \notin Q$. Since x is labelled with **forget**(v) by $\mathcal{D}$, we have that $v \notin \beta_{\mathcal{D}}(x)$, $v \in \beta_{\mathcal{D}}(x')$, and $v \in \gamma_{\mathcal{D}}(x)$. This implies that $v \in \mathsf{Original}(x')$ and hence $x' \notin Q$. Now, we prove that $x \in Q$. For this purpose, let $R = \{t \in V(T) \colon s \in \beta_{\mathcal{D}}(t)\}$. Clearly, $Q \subseteq R$. By Property (c) of the tree decomposition $\mathcal{D}$, we have that $T[R]$ is connected. We have already proved that $T[Q]$ is a path and since $Q \subseteq R$, $T[Q]$ is a path in $T[R]$. Since x is labelled with **forget**(v) by $\mathcal{D}$, for any node x'' in the subtree rooted at x and $x'' \neq x$, either $v \in \beta_{\mathcal{D}}(x'')$ or $v \notin \gamma_{\mathcal{D}}(x'')$ (this fact follows from Property (c) of $\mathcal{D}$). This implies that Q contains no node in the subtree of T rooted at x and not equal to x. Moreover, observe that there exists a node $x^\star$ in the subtree of T rooted



$\emptyset$ **forget**$(\{v_1,v_2,v_3,v_4\})$
$v_1\ v_2\ v_3\ v_4$ **join**
v_1v_2
$v_3\ v_4$ **join**
v_1 v_2 v_3 v_4 **fake introduce**(v_i)
v_1 v_2 v_3 v_4 **redundant**
v_1 v_2 v_3 v_4 **introduce**(v_i)

Figure 2 An $(\ell,\mathcal{D})$-*FewCliTD* $\mathcal{D}'$ derived from the nice tree decomposition $\mathcal{D}$ in Figure 1d using Definition 3.1. The labels of the nodes in $\mathcal{D}'$ are mentioned on the right.

at x such that $\{s,v\} \subseteq \beta_{\mathcal{D}}(x^\star)$ and hence $x^\star \in R$. Now, since Q is non-empty and $T[Q]$ is connected, we have that $s \in \beta_{\mathcal{D}}(x)$. Since $v \notin \beta_{\mathcal{D}}(x)$, $v \in \gamma_{\mathcal{D}}(x)$ and $s \in \beta_{\mathcal{D}}(x)$, we conclude that $x \in Q$. Thus, we have proved that x is an end-vertex of the the path $T[Q]$.

Next we prove that y is the other end-vertex of the path $T[Q]$ and y is an ancestor of x. Since y is the only child of the node y' labelled with **forget**(s), we have that $s \in \beta_{\mathcal{D}}(y)$ and $s \notin \beta_{\mathcal{D}}(y')$. This implies that $y' \notin Q$. Thus to prove that y is an end-vertex of the path $T[Q]$, it is enough to prove that $y \in Q$. Since $s \in \beta_{\mathcal{D}}(x)$, $s \in \beta_{\mathcal{D}}(y)$, $s \notin \beta_{\mathcal{D}}(y')$ and y' is the parent of y, by Property (c) of $\mathcal{D}$, we have that y is an ancestor of x. This also implies that $v \in \gamma_{\mathcal{D}}(y)$ and $v \notin \beta_{\mathcal{D}}(y)$. Hence, $y \in Q$. This completes the proof of the lemma. ◀

In the next lemma we show that for any special vertex $s \in S(G)$ and any node t in T labelled with **introduce**(s) by $\mathcal{D}$, it holds that t and its child carry the "same information".

▶ **Lemma 3.4** ($\star$). *Let $s \in S(G)$ and t be a node in T labelled with **introduce**(s) by $\mathcal{D}$. Let t' be the only child of t. Then, Original(t) = Original(t') and Fake(t) = Fake(t').*

Next, we see a property of nodes $t \in V(T)$ labelled with **join**.

▶ **Lemma 3.5** ($\star$). *Let t be a node in T labelled with **join** by D, and t_1 and t_2 are its children. Then, Original(t) = Original(t_1) = Original(t_2), Cliques(t) = Cliques(t_1) = Cliques(t_2), Fake$(t_1) \cap$ Fake$(t_2) = \emptyset$, and Fake(t) = Fake$(t_1) \cup$ Fake(t_2).*

Now, we define a notion of *nice ℓ-few cliques tree decomposition* of G as the tree decomposition of G derived from a nice tree decomposition $\mathcal{D}$ of B of width $< \ell$ (see Definition 3.1) with additional labeling of nodes. In what follows, we describe this additional labeling of nodes. Towards this, observe that because of Lemma 3.4, for any special vertex $s \in S(G)$ and any node $t \in V(T)$ labelled with **introduce**(s) by $\mathcal{D}$, the bags $\beta_{\mathcal{D}'}(t)$ and $\beta_{\mathcal{D}'}(t')$ carry the "same information" where t' is the only child of t. Informally, one may choose to handle these nodes by contracting them. However, to avoid redundant proofs ahead, instead of getting rid of such nodes, we label them with **redundant** in $\mathcal{D}'$. Next, we explain how to label other nodes of T in the decomposition $\mathcal{D}'$ (see Figure 2). To this end, let $t \in V(T)$.

- If t is labelled with **leaf** by $\mathcal{D}$, then we label t with **leaf**. Here, $\beta_{\mathcal{D}'}(t) = \emptyset$.
- If t is labelled with **introduce**(v) by $\mathcal{D}$ for some $v \in V(G)$, then we label t with **introduce**(v). In this case, t has only one child t' in T and $\beta_{\mathcal{D}'}(t) \setminus \{v\} = \beta_{\mathcal{D}'}(t')$.
- If t is labelled with **forget**(v) by $\mathcal{D}$ for some $v \in V(G)$ and $v \in$ Fake(t), then we label t with **fake introduce**(v). In this case, t has only one child t' and $\beta_{\mathcal{D}'}(t) = \beta_{\mathcal{D}'}(t')$, but Original$(t)$ = Original$(t') \setminus \{v\}$ and Fake(t) = Fake$(t') \cup \{v\}$.

- If t is labelled with **forget**(v) by $\mathcal{D}$ for some $v \in V(G)$ and $v \notin \mathsf{Fake}(t)$, then we label t with **forget**(v). In this case, t has only one child t', $\beta_{\mathcal{D}'}(t) = \beta_{\mathcal{D}'}(t') \setminus \{v\}$, $\mathsf{Original}(t) = \mathsf{Original}(t') \setminus \{v\}$ and $\mathsf{Fake}(t) = \mathsf{Fake}(t')$.
- Suppose t is labelled with **forget**(s) by $\mathcal{D}$ for some $s \in S(G)$. Then, t has only one child t'. Here, we label t with **forget**$(\beta_{\mathcal{D}'}(t') \setminus \beta_{\mathcal{D}'}(t))$. In this case, $\mathsf{Fake}(t) \subseteq \mathsf{Fake}(t')$ and $\mathsf{Original}(t) = \mathsf{Original}(t')$.
- If t is labelled with **join** by $\mathcal{D}$, then we label t with **join**. Let t_1 and t_2 be the children of t. Then, $\mathsf{Original}(t) = \mathsf{Original}(t_1) = \mathsf{Original}(t_2)$, $\mathsf{Cliques}(t) = \mathsf{Cliques}(t_1) = \mathsf{Cliques}(t_2)$, $\mathsf{Fake}(t_1) \cap \mathsf{Fake}(t_2) = \emptyset$, and $\mathsf{Fake}(t) = \mathsf{Fake}(t_1) \cup \mathsf{Fake}(t_2)$. (See Lemma 3.5).
- If t is labelled with **introduce**(s) for some $s \in S(G)$, then we label t with **redundant**.

This completes the definition of the nice ℓ-few cliques tree decomposition of G derived from $\mathcal{D}$, to which we simply call an $(\ell, \mathcal{D})$-NFewCliTD. Notice that for each node t in T, $|\mathsf{Original}(t)| + |\mathsf{Cliques}(t)| \leq \ell$. That is, for any node $t \in V(T)$, there exist $i, j \in \mathbb{N}$ such that $i + j \leq \ell$, the cardinality of $\mathsf{Original}(t)$ is at most i, and the vertices in $\beta_{\mathcal{D}'}(t) \setminus \mathsf{Original}(t)$ were obtained from at most j special cliques. From now on, we assume that $\mathcal{D}'$ is an $(\ell, \mathcal{D})$-NFewCliTD of a map graph G.

Since the number of nodes with label **forget**(v) in the tree decomposition $\mathcal{D}$ is exactly one for any $v \in V(B)$ (see Observation 2.3), *at most one* node in T is labelled with **fake introduce**(v) in $\mathcal{D}'$. This is formally stated in the following observation.

▶ **Observation 3.6.** *Let $t \in V(T)$ and $v \in \mathsf{Fake}(t)$. Then,*
(i) *there is a unique node $t' \in V(T)$ such that t' is labelled with **fake introduce**(v) in $\mathcal{D}'$,*
(ii) *t is an ancestor of t' or $t = t'$, and*
(iii) *for any node t'' in the unique path between t and t', we have that $v \in \mathsf{Fake}(t'')$.*

The correctness of Observation 3.6 follows from Observation 2.3 and Lemma 3.3. The discussion above, along with Proposition 2.4, implies the following lemma.

▶ **Lemma 3.7** ($\star$)**.** *Given a map graph G, a corresponding planar bipartite graph B, and an integer $\ell \in \mathbb{N}$, in time $\mathcal{O}(n^2)$, one can either correctly conclude that B contains an $\ell \times \ell$ grid as a minor, or compute a nice tree decomposition $\mathcal{D}$ of B of width less than 5ℓ and a $(5\ell, \mathcal{D})$-NFewCliTD of G.*

Lastly, we prove an important property of $\mathcal{D}'$. In particular, the edges considered in the following lemma are precisely those that connect the vertices "already seen" (when we use dynamic programming (DP)) with vertices to "see in the future".

▶ **Lemma 3.8.** *For any node $t \in V(T)$, the edges with one endpoint in $\gamma_{\mathcal{D}'}(t)$ and other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ are of two kinds: (i) edges incident with vertices in $\mathsf{Original}(t)$, and (ii) edges belonging to some special clique in $\mathsf{Cliques}(t)$ (these edges are incident to vertices in $\mathsf{Fake}(t)$).*

Proof. Fix $t \in V(T)$. Since $\mathcal{D}'$ is a tree decomposition of G, for any edge $e \in E(G)$ with one endpoint in $\gamma_{\mathcal{D}'}(t)$ and other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$, the endpoint of e in $\gamma_{\mathcal{D}'}(t)$ should belong to $\beta_{\mathcal{D}'}(t)$. Let u be the endpoint of e that belongs to $\beta_{\mathcal{D}'}(t)$, and v be the other endpoint of e. Notice that the set $\beta_{\mathcal{D}'}(t)$ is partitioned into $\mathsf{Original}(t)$ and $\mathsf{Fake}(t)$, so u belongs to either $\mathsf{Original}(t)$ or $\mathsf{Fake}(t)$, and in the former case we are done. We now assume that $u \in \mathsf{Fake}(t)$.

Since $\{u, v\} = e \in E(G)$, there is a special vertex $s \in S(G)$ such that $\{u, s\}, \{v, s\} \in E(B)$. If $s \in \beta_{\mathcal{D}}(t)$, then the edge $\{u, v\}$ belongs to the special clique $K = N_B(s)$ in G and $K \in \mathsf{Cliques}(t)$. We claim that indeed $s \in \beta_{\mathcal{D}}(t)$. Towards this, notice that $u \in \mathsf{Fake}(t)$. This implies that $u \notin \beta_{\mathcal{D}}(t)$, but u is present in a bag $\beta_{\mathcal{D}}(t')$ of some descendant t' of t. Moreover, since $\{u, s\} \in E(B)$, we further know that $\{u, s\} \subseteq \beta_{\mathcal{D}}(t_1)$ for some descendent t_1 of t. Since

$\{v, s\} \in E(B)$ and $v \notin \gamma_{\mathcal{D}'}(t)$, there is a bag $\beta_{\mathcal{D}}(t_2)$ such that $\{v, s\} \subseteq \beta_{\mathcal{D}}(t_2)$ and t_2 is not a descendent of t. Thus, since $s \in \beta_{\mathcal{D}}(t_1) \cap \beta_{\mathcal{D}}(t_2)$, by Property (c) of the tree decomposition $\mathcal{D}$, we have that $s \in \beta_{\mathcal{D}}(t)$. This completes the proof of the lemma. ◀

4 Longest Cycle

In this section we prove the following theorem.

▶ **Theorem 4.1.** *Longest Cycle on map graphs can be solved in $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ time.*

Notice that if there is a special vertex $s \in S(G)$ such that $|N_B(s)| \geq k$, then G has a cycle of length at least k, because $N_B(s)$ forms a clique in G. Moreover, observe that if there is a "large enough" grid in B, then we can answer Yes. These observations along with Lemma 3.7 lead to the following lemma.

▶ **Lemma 4.2** ($\star$). *There is an algorithm that given an instance (G, B, k) of Longest Cycle, runs in time $\mathcal{O}(n^2)$, and either correctly concludes that G has a cycle of length at least k, or outputs a nice tree decomposition $\mathcal{D}$ of B of width $< 5\sqrt{2k}$ and a $(5\sqrt{2k}, \mathcal{D})$-NFewCliTD $\mathcal{D}'$ of G such that for each node $t \in V(T)$, $|\beta_{\mathcal{D}'}(t)| \leq 5\sqrt{2} \cdot k^{1.5}$.*

Because of Lemma 4.2, to prove Theorem 4.1, we may assume that the input contains a $(5\sqrt{2k}, \mathcal{D})$-NFewCliTD $\mathcal{D}'$ of G (derived from a nice tree decomposition $\mathcal{D}$ of B) such that for each node $t \in V(T)$, $|\beta_{\mathcal{D}'}(t)| \leq 5\sqrt{2} \cdot k^{1.5}$. That is, from now on, we fix our input to be an instance (G, B, k) of Longest Cycle, a nice tree decomposition $\mathcal{D}$ of B and a $(5\sqrt{2k}, \mathcal{D})$-NFewCliTD $\mathcal{D}'$ of G such that for each node $t \in V(T)$, $|\beta_{\mathcal{D}'}(t)| \leq 5\sqrt{2} \cdot k^{1.5}$. Towards the proof of Theorem 4.1, the main ingredient is to prove the following claim: if G has a cycle of length ℓ, then there is a cycle C of length ℓ, with the following property.

For each node $t \in V(T)$, the number of edges of $E(C)$ with one endpoint in $\beta_{\mathcal{D}'}(t)$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is upper bounded by $\mathcal{O}(\sqrt{k})$.

The above mentioned property is encapsulated in the following sublinear crossing lemma.

▶ **Lemma 4.3** (Sublinear Crossing Lemma ($\star$)). *Let C be a cycle in G. Then there is a cycle C' of the same length as C such that for any node $t \in V(T)$, the number of edges in $E(C')$ with one endpoint in $\beta_{\mathcal{D}'}(t)$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is at most $20\sqrt{2k}$.*

Lemma 4.3 lies at the heart of the proof of Theorem 4.1 and is one of the main technical contributions of the paper. Assuming Lemma 4.3, the proof of Theorem 4.1 is by designing a DP algorithm on a $(5\sqrt{2k}, \mathcal{D})$-NFewCliTD of G. This part is similar to the algorithm for Longest Cycle in [20] on a so called *special path decomposition*. The main ingredient of Lemma 4.3 is the following lemma (Lemma 4.4). The proof of Lemma 4.3 is by an inductive argument assuming Lemma 4.4. The rest of the section is devoted to the proof of Lemma 4.4.

▶ **Lemma 4.4.** *Let C be a cycle in G and K be a special clique in G. Then, there is a cycle C' of the same length as C such that $E(C') \setminus E(K) = E(C) \setminus E(K)$ and for any node $t \in V(T)$, the number of edges of $E(C') \cap E(K)$ with one endpoint in $\mathsf{Fake}(t) \cap K$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is at most 4.*

Before formally proving Lemma 4.4, we give a high level overview of the proof and an auxiliary lemma which we use in the proof of Lemma 4.4. The proof idea is to change the edges of $E(K) \cap E(C)$ in C (because in Lemma 4.4 our objective is to bound the "crossing edges" from a subset of $E(K)$ for each node $t \in V(T)$) to obtain a new cycle C' of the same length as C that satisfies the following property: (i) for any node $t \in V(T)$, the number of edges of $E(C') \cap E(K)$ with one endpoint in $\mathsf{Fake}(t) \cap K$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is at most 4. For the ease of presentation, assume that $K \subseteq V(C)$. Now, consider the graph $\mathcal{P}$ obtained from the cycle C after deleting edges in $E(K)$. Without loss of generality assume that $E(C) \cap E(K) \neq \emptyset$. Otherwise, Lemma 4.4 is true where $C' = C$. We consider $\mathcal{P}$ as a collection of vertex-disjoint paths where the end-vertices of the paths are in K. Some paths in $\mathcal{P}$ may be of length 0. Let Z be the set of end-vertices of the paths in $\mathcal{P}$. Clearly, $Z \subseteq K$. We will "complete" the collection of paths $\mathcal{P}$ to a cycle by adding edges from $E(Z)$ satisfying Statement (i). Any cycle C' with $V(C') = V(\mathcal{P})$ has the same length as C. So, all the work that is required for us is to complete the collection of paths $\mathcal{P}$ to a cycle by adding edges from $E(Z)$ satisfying Statement (i). Towards that, let $\widehat{\sigma} = v_1, \ldots, v_{k'}$ be an arbitrary sequence of vertices in Z. We show (in Claim 4.7) that (ii) there is a subset of edges $F \subseteq E(Z)$ such that $E(\mathcal{P}) \cup F$ forms a cycle C' with vertex set $V(\mathcal{P})$ and for any $j \in [k']$, the number of edges in F with one endpoint in $\{v_1, \ldots, v_j\}$ and the other in $\{v_{j+1}, \ldots, v_{k'}\}$ is at most 2. This implies that for any $1 \leq i \leq j \leq k'$, the number of edges in F with one endpoint in $\{v_i, \ldots, v_j\}$ and the other in $Z \setminus \{v_i, \ldots, v_j\}$ is at most 4. In the light of Statement (ii), our aim will be to prove that (iii) for any node $t \in V(T)$, there exist $1 \leq i \leq j \leq k'$ such that $\mathsf{Fake}(t) \cap Z \subseteq \{v_i, \ldots, v_j\}$ and no vertex in $Z \setminus \gamma_{\mathcal{D}'}(t)$ belongs to $\{v_i, \ldots, v_j\}$. Then, Statement (i) will follow (because edges of C' incident with vertices in $K \setminus Z$ are from $E(G) \setminus E(K)$ and will not be counted in Statement (i)). In fact, we will prove that there is a sequence σ on Z (derived from a postorder transversal of T) such that Statement (iii) is true (see Claim 4.8). The proof of Statement (ii) is encapsulated in the following lemma.

▶ **Lemma 4.5** ($\star$). *Let $\ell \geq 3$ be an integer. Let $u_1, \ldots, u_\ell$ be a sequence of vertices in a graph H where $X = \{u_1, \ldots, u_\ell\}$ is a clique in H. Let $\mathcal{Q}$ be a family of vertex-disjoint paths in H (which possibly contains paths of length 0) such that each $v \in X$ is an end-vertex of a path in $\mathcal{Q}$ and $E(\mathcal{Q}) \cap E(X) = \emptyset$. Then, there is a set $F \subseteq E(X)$ with the following conditions.*

(a) *$E(\mathcal{Q}) \cup F$ forms a cycle containing all the vertices of $V(\mathcal{Q})$,*

(b) *For any $j \in [\ell]$, the number of edges in F with one endpoint in $\{u_1, \ldots, u_j\}$ and the other in $\{u_{j+1}, \ldots, u_\ell\}$ is at most 2.*

Next, we move to a formal proof of Lemma 4.4.

Proof of Lemma 4.4. Without loss of generality, assume that $K \subseteq V(C)$. Otherwise, we can consider the statement of the lemma for cycle C in the graph $G' = G - (K \setminus V(C))$ and special clique $K \cap V(C)$ of G'. We also assume that $E(C) \cap E(K) \neq \emptyset$, else the correctness is trivial because we can take C' as C.

Let π' be a postorder transversal of the nodes in the rooted binary tree T, and let π be the restriction of π' where we only keep the nodes that are labeled with **fake introduce**(v) for some $v \in K$. Denote $\pi = t_1, \ldots, t_{k''}$ such that each t_i, $i \in [k'']$, is labelled with **fake introduce**(x_i) where $x_i \in K$. Notice that $\bigcup_{t \in V(T)} \mathsf{Fake}(t) \cap K = \{x_1, \ldots, x_{k''}\}$ (by Observation 3.6). Let σ_1 be the sequence $x_1, \ldots, x_{k''}$ and $U = \{x_1, \ldots, x_{k''}\}$. Let σ_2 be a fixed arbitrary sequence of $K \setminus U$, i.e., all the vertices of K that are never "fakely introduced". Let σ be the sequence which is a concatenation of σ_1 and σ_2.

Let $\mathcal{P} = (V(C), E(C) \setminus E(K))$. That is, $\mathcal{P}$ is the graph obtained by deleting edges of $E(K)$ from the cycle C. Notice that each connected component of $\mathcal{P}$ is a path (may be of length 0) with end-vertices in K. Let Z be the set of end-vertices of the paths in $\mathcal{P}$. Notice that for any vertex $u \in K \setminus Z$, both edges of C incident with u are from $E(C) \setminus E(K)$ (see the left part of Figure 3). That is, $E(C) \setminus E(K) = E(C) \setminus E(Z) = E(\mathcal{P})$. Since we seek a cycle C' in which $E(C') \setminus E(K) = E(C) \setminus E(K)$, no edge of C' incident with u for any vertex $u \in K \setminus Z$, is in $E(K)$. That is, all the edges of $E(C') \cap E(K)$ will belong to $E(Z)$.

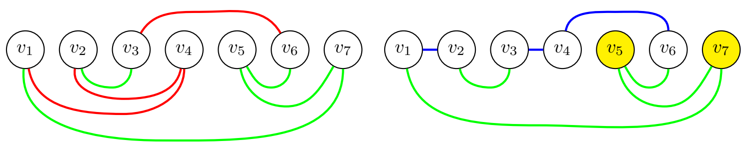


Figure 3 Left part illustrates a cycle C interacting with a special clique $K = \{v_1, \dots, v_7\}$. The red curves represent edges in $E(C) \cap E(K)$ and green curves represent paths in C with endpoints in K and (at least one) internal vertices in $V(G) \setminus K$. Thus, $\mathcal{P}$ is the collection of paths that is a union of the set of two "green" paths ($(v_2 - v_3)$ and $(v_1 - v_7 - v_5 - v_6)$) and $\{[v_4]\}$. Here $Z = \{v_1, v_2, v_3, v_4, v_6\}$. Any edge of $E(C)$ incident with v_5 and v_7 (i.e., vertices in $K \setminus Z$) are from $E(C) \setminus E(K)$. The right part illustrates the proof of Claim 4.7. The edges of $E(C') \setminus E(C)$ mentioned in the proof of Claim 4.7 are colored blue.

▶ **Observation 4.6.** *Let C' be a cycle in G such that $E(C') \setminus E(K) = E(C) \setminus E(K)$ and $t \in V(T)$. The number of edges of $E(C') \cap E(K)$ with one endpoint in $\mathsf{Fake}(t) \cap K$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is equal to the number of edges of $E(C') \cap E(Z)$ with one endpoint in $\mathsf{Fake}(t) \cap Z$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$.*

Let $Z = \{v_1, \dots, v_{k'}\}$ and $\sigma' = \sigma|_Z = v_1, \dots, v_{k'}$. The main ingredients of the proof are the following two claims.

▷ Claim 4.7. There is a cycle C' of the same length as C such that (i) $E(C') \setminus E(Z) = E(C) \setminus E(Z)$, and (ii) for any $j \in [k']$, the number of edges of $E(C') \cap E(Z)$ with one endpoint in $\{v_1, \dots, v_j\}$ and the other in $\{v_{j+1}, \dots, v_{k'}\}$ is at most 2.

The proof of Claim 4.7 follows from Lemma 4.5.

▷ Claim 4.8. For any node $t \in V(T)$, there is a segment σ'' of σ' such that each vertex in $\mathsf{Fake}(t) \cap Z$ appears in σ'', and each vertex in $Z \setminus \gamma_{\mathcal{D}'}(t)$ does not appear in σ''.

Proof. Fix a node $t \in V(T)$. Recall that $\sigma = \sigma_1\sigma_2$ and $\sigma' = \sigma|_Z$. Here, the set of vertices present in σ_1 is $U = (\bigcup_{t \in V(T)} \mathsf{Fake}(t) \cap K) \supseteq (\bigcup_{t \in V(T)} \mathsf{Fake}(t) \cap Z)$ (because $Z \subseteq K$), and no vertex in σ_2 is from U. This implies that all the vertices of $\mathsf{Fake}(t) \cap Z$ are in the sequence σ_1. That is, the sequence σ'' we seek is also a sequence of $\sigma_1|_Z$ and this is the reason we defined σ to be $\sigma_1\sigma_2$. Thus, to prove the claim it is enough to prove that there is a segment σ_1' of $\sigma_1|_Z$ such that each vertex in $\mathsf{Fake}(t) \cap Z$ appears in σ_1' and each vertex in $Z \setminus \gamma_{\mathcal{D}'}(t)$ does not appear in σ_1'.

Recall that $\sigma_1 = x_1 \dots x_{k''}$ is obtained from the sequence $\pi = t_1, \dots, t_{k''}$. In turn, recall that π is the restriction of the postorder transversal π' of T, where for each $i \in [k'']$, t_i is labelled with **fake introduce**(x_i) for $x_i \in K$. Let W_t be the nodes of the subtree of T rooted at t, and $V_t = \{v \in K \colon \text{there is } t' \in W_t \text{ such that } t' \text{ is labelled with } \textbf{fake introduce}(v)\}$.

The vertices in W_t appear consecutively in π. Thus, we can let π_t be the minimal segment of π that contains all the nodes in V_t. Let $i, j \in [k'']$ be such that $\pi_t = t_i, \dots, t_j$. Now, we define σ_t be the segment $x_i, \dots, x_j$ of σ_1. Now we prove the claim. By conditions (i) and (ii) in Observation 3.6, $\mathsf{Fake}(t) \cap Z \subseteq V_t$. Clearly, no vertex in $Z \setminus \gamma_{\mathcal{D}'}(t)$ is in V_t. This implies that each vertex in $\mathsf{Fake}(t) \cap Z$ appears in σ_t and no vertex from $Z \setminus \gamma_{\mathcal{D}'}(t)$ appears in σ_t. In turn, this implies that $\sigma_t|_Z$ is the required segment σ'' of $\sigma' = \sigma|_Z$. ◁

Now, having the above two claims, we are ready to prove the lemma. By Claim 4.7, we have that there is a cycle C' such that (i) $E(C') \setminus E(Z) = E(C) \setminus E(Z)$, and (ii) for any $j \in [k']$, the number of edges of $E(C) \cap E(Z)$ with one endpoint in $\{v_1, \dots, v_j\}$ and other in $\{v_{j+1}, \dots, v_{k'}\}$ is at most 2. By Claim 4.8, we know that for any $t \in V(T)$, there is a segment

σ'' of σ' such that all vertices in $\mathsf{Fake}(t) \cap Z$ appear in a segment σ'' and no vertex from $Z \setminus \gamma_{\mathcal{D}'}(t)$ appears in σ''. That is, there exist $i, j \in [k']$ such that $\mathsf{Fake}(t) \cap Z \subseteq \{v_i, \ldots, v_j\}$ and $(Z \setminus \gamma_{\mathcal{D}'}(t)) \cap \{v_i, \ldots, v_j\} = \emptyset$. Therefore, by *(ii)*, the number of edges of $E(C') \cap E(Z)$ with one endpoint in $\mathsf{Fake}(t) \cap Z$ and the other in $V(G) \setminus \gamma_{\mathcal{D}'}(t)$ is at most 4. Then, by Observation 4.6, the proof of the lemma is complete. ◀

References

1 J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002.

2 Jochen Alber, Henning Fernau, and Rolf Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. *J. Algorithms*, 52(1):26–56, 2004.

3 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. Assoc. Comput. Mach.*, 42(4):844–856, 1995.

4 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41(1):153–180, 1994.

5 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.

6 Zhi-Zhong Chen. Approximation algorithms for independent sets in map graphs. *Journal of Algorithms*, 41:20–40, 2001.

7 Zhi-Zhong Chen, Enory Grigni, and Christos H. Papadimitriou. Planar map graphs. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC '98)*, pages 514–523. ACM, 1998.

8 Zhi-Zhong Chen, Enory Grigni, and Christos H. Papadimitriou. Map graphs. *J. Assoc. Comput. Mach.*, 49(2):127–138, 2002.

9 Zhi-Zhong Chen, Michelangelo Grigni, and Christos H. Papadimitriou. Recognizing Hole-Free 4-Map Graphs in Cubic Time. *Algorithmica*, 45(2):227–262, 2006. doi:10.1007/s00453-005-1184-8.

10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

11 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 150–159. IEEE, 2011.

12 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. *ACM Trans. Algorithms*, 1(1):33–47, 2005. doi:10.1145/1077464.1077468.

13 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential Parameterized Algorithms on Graphs of Bounded Genus and H-minor-free Graphs. *J. ACM*, 52(6):866–893, 2005.

14 Erik D. Demaine and MohammadTaghi Hajiaghayi. Bidimensionality: new connections between FPT algorithms and PTASs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, pages 590–601, New York, 2005. ACM-SIAM.

15 Erik D. Demaine and MohammadTaghi Hajiaghayi. The Bidimensionality Theory and Its Algorithmic Applications. *Comput. J.*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

16 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

17 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Decompositions. *Algorithmica*, 58(3):790–810, 2010. doi:10.1007/s00453-009-9296-1.

18 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 470–479. IEEE, 2012.

19 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM*, 63(4):29, 2016. `doi:10.1145/2886094`.

20 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, Hitting and Packing Cycles in Subexponential Time on Unit Disk Graphs. *Discrete & Computational Geometry*, January 2019. `doi:10.1007/s00454-018-00054-x`.

21 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Bidimensionality and geometric graphs. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1563–1575. SIAM, 2012.

22 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Excluded grid minors and efficient polynomial-time approximation schemes. *J. ACM*, 65(2):Art. 10, 44, 2018. `doi:10.1145/3154833`.

23 Qian-Ping Gu and Hisao Tamaki. Improved Bounds on the Planar Branchwidth with Respect to the Largest Grid Minor Size. *Algorithmica*, 64(3):416–453, November 2012. `doi:10.1007/s00453-012-9627-5`.

24 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

25 Ioannis Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586, 2008.

26 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. `doi:10.1145/2742544`.

27 Dániel Marx. The Square Root Phenomenon in Planar Graphs. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 7966 of *Lecture Notes in Computer Science*, page 28. Springer, 2013.

28 N. Robertson, P. Seymour, and R. Thomas. Quickly Excluding a Planar Graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994. `doi:10.1006/jctb.1994.1073`.

29 Mikkel Thorup. Map Graphs In Polynomial Time. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS 1998)*, pages 396–405. IEEE Computer Society, 1998.

30 Ryan Williams. Finding Paths of Length k in $O^*(2^k)$ Time. *Inf. Process. Lett.*, 109(6):315–318, 2009.