SIAM J. DISCRETE MATH.
Vol. 33, No. 4, pp. 2326–2345




# FINDING DETOURS IS FIXED-PARAMETER TRACTABLE*

IVONA BEZÁKOVÁ†, RADU CURTICAPEAN‡, HOLGER DELL§, AND FEDOR V. FOMIN¶

**Abstract.** We consider the following natural "above-guarantee" parameterization of the classical LONGEST PATH problem: For given vertices $s$ and $t$ of a graph $G$ and integer $k$, the LONGEST DETOUR problem asks for an $(s,t)$-path in $G$ that is at least $k$ longer than a shortest $(s,t)$-path. Using insights into structural graph theory, we prove that LONGEST DETOUR is fixed-parameter tractable on undirected graphs and actually even admits a single-exponential algorithm, that is, one of running time $2^{O(k)} \cdot n^{O(1)}$. Up to the base of the exponential, this running time matches the best algorithms for finding a path of length at least $k$. Furthermore, we study the related EXACT DETOUR problem, which asks whether a graph $G$ contains an $(s,t)$-path that is exactly $k$ longer than a shortest $(s,t)$-path. For this problem, we obtain randomized algorithms with running times $2.746^k \cdot n^{O(1)}$ (for undirected graphs) and $4^k \cdot n^{O(1)}$ (for directed graphs) and a deterministic algorithm with running time $6.745^k \cdot n^{O(1)}$, showing that this problem is fixed-parameter tractable as well.

**Key words.** longest path, fixed-parameter tractable algorithm, above-guarantee parameterization, graph minors

**AMS subject classifications.** 05C83, 05C85, 68Q25, 68R05, 68R10, 68W40

**DOI.** 10.1137/17M1148566

## 1. Introduction.

The LONGEST PATH problem asks, given an undirected $n$-vertex graph $G$ and an integer $k$, to determine whether $G$ contains a path of length at least $k$, that is, a self-avoiding walk with at least $k$ edges. This problem is a natural generalization of the classical NP-complete HAMILTONIAN PATH problem, and the parameterized complexity community has paid a great deal of attention to it. For instance, Monien [31] and Bodlaender [4] showed that LONGEST PATH is fixed-parameter tractable with parameter $k$ and admits algorithms with running time $2^{O(k \log k)} n^{O(1)}$. This led Papadimitriou and Yannakakis [32] to conjecture that LONGEST PATH is solvable in polynomial time for $k = \log n$, and indeed, this conjecture was resolved in a seminal paper of Alon, Yuster, and Zwick [2], who introduced the method of color coding and derived from it the first algorithm with running time $2^{O(k)} n$. The LONGEST PATH problem occupies a central place in parameterized algorithmics, and several different approaches were developed in order to reduce the base of the exponent in the running time [21, 24, 9, 8, 25, 36, 17, 17, 3]. We refer the reader to two review articles in *Communications of ACM* [16, 26] as well as to the textbook [12, Chapter 10] for an extensive overview of parameterized algorithms for LONGEST PATH. Let us note that the fastest known randomized algorithm for LONGEST PATH is due to Björklund

*Received by the editors September 21, 2017; accepted for publication (in revised form) August 30, 2019; published electronically November 26, 2019.
https://doi.org/10.1137/17M1148566
**Funding:** The work of the first author was supported by NSF grant CCF-1319987. The work of the second author was supported by ERC grants PARAMTIGHT (280152) and SYSTEMATICGRAPH (725978), and VILLUM Foundation grant 16582 while working on this paper. The work of the fourth author was supported by NFR grant MULTIVAL.
†Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623 (ib@cs.rit.edu).
‡Basic Algorithms Research Copenhagen and IT University of Copenhagen, Copenhagen, Denmark (radu.curticapean@gmail.com).
§IT University of Copenhagen, Copenhagen, Denmark (hold@itu.dk).
¶Department of Informatics, University of Bergen, Bergen, 5020, Norway (fomin@ii.uib.no).

et al. [3] and runs in $1.657^k \cdot n^{O(1)}$ time, whereas the fastest known deterministic algorithm is due to Zehavi [37] and runs in $2.597^k \cdot n^{O(1)}$ time.

In the present paper, we study the Longest Path problem from the perspective of an "above-guarantee" parameterization that can attain small parameter values even for long paths: For a pair of vertices $s, t \in V(G)$, we use $d_G(s,t)$ to denote their distance, that is, the length of a shortest path from $s$ to $t$. In the Longest Detour problem, we then ask for an $(s,t)$-path of length at least $d_G(s,t)+k$, and we parameterize by this offset $k$ rather than the actual length of the path. In other words, the first $d_G(s,t)$ steps on a path sought by Longest Detour are "complimentary" and will not be counted towards the parameter value. This reflects the fact that shortest paths can be found in linear time and could be much better solutions for Longest Path than the paths of logarithmic length found by algorithms that parameterize by the path length.

We study two variants of the detour problem, one asking for a detour of length at least $k$, and another asking for a detour of length exactly $k$.

Longest Detour **Parameter:** $k$
**Input:** Graph $G$, vertices $s, t \in V(G)$, and integer $k$.
**Task:** Decide whether there is an $(s,t)$-path in $G$ of length at least $d_G(s,t)+k$.

Exact Detour **Parameter:** $k$
**Input:** Graph $G$, vertices $s, t \in V(G)$, and integer $k$.
**Task:** Decide whether there is an $(s,t)$-path in $G$ of length exactly $d_G(s,t)+k$.

Our parameterization above the length of a shortest path is a new example in the general paradigm of "above-guarantee" parameterizations, which was introduced by Mahajan and Raman [28, 29]. This paradigm was successfully applied to various problems, such as finding independent sets in certain planar graphs (where an independent set of size at least $\frac{n}{4}$ is guaranteed to exist by the Four Color Theorem) [14, 30], the maximum cut problem [11, 15], constraint satisfaction problems [1, 19], and the minimum vertex cover problem [18].

**Our results.** We establish in the following theorems that Longest Detour and Exact Detour are fixed-parameter tractable.

Theorem 1.1. *There is a deterministic algorithm that solves* Longest Detour *on undirected $n$-vertex graphs in time $2^{O(k)} \cdot n^{O(1)}$.*

To establish the correctness of this algorithm, we rely on nontrivial arguments in the theory of graph minors. While a running time of $2^{o(k)} \cdot n^{O(1)}$ is impossible for both Longest Detour and Exact Detour unless the exponential-time hypothesis of Impagliazzo and Paturi [22] fails, the following theorem establishes algorithms for Exact Detour with more precise running time guarantees.[1]

Theorem 1.2. *There is a bounded-error randomized algorithm that solves* Exact Detour *on undirected $n$-vertex graphs in time $2.746^k n^{O(1)}$ and on directed graphs in time $4^k n^{O(1)}$. For both undirected and directed graphs, there is a deterministic algorithm that runs in time $6.745^k n^{O(1)}$.*

[1] By a standard reduction, the exponential-time hypothesis rules out a $2^{o(n)} \cdot n^{O(1)}$ time algorithm for the Hamiltonian Path problem; see [12, Theorem 14.6]. An algorithm with running time $2^{o(k)} \cdot n^{O(1)}$ for Longest Detour and Exact Detour would clearly imply such an algorithm.

To prove Theorem 1.2, we construct a polynomial-time Turing reduction from Exact Detour to the standard parameterization of Longest Path, in which we ask on input $s, t$ and $k \in \mathbf{N}$ whether there is an $(s,t)$-path of length $k$. The reduction only makes queries to instances with parameter at most $2k+1$, and so pipelining it with the fastest known algorithms for Longest Path mentioned earlier yields the theorem.

We further observe that, pipelined with a self-reducibility argument, Theorems 1.1 and 1.2 can be used to explicitly construct the required detours. This incurs only a polynomial overhead in the running time.

**Techniques.** The main idea behind the algorithm for Longest Detour is a combinatorial theorem that shows the existence of specific large planar minors in graphs of large treewidth. Although the Excluded Grid Theorem [34] already shows that graphs of sufficiently large treewidth contain arbitrary fixed planar graphs, we circumvent the full machinery of this theorem and resort to more basic techniques. This allows us to show that *linear* treewidth suffices to guarantee our specific planar minors. More specifically, we show that there exists a constant $C \in \mathbf{N}$ such that every graph of treewidth at least $Ck$ contains a subgraph obtained from the complete graph $K_4$ by replacing every edge with a path of length at least $k$. We call such a subgraph a $(\geq k)$-*subdivided tetrahedron*. This result is shown using structural theorems of Leaf and Seymour [27] and Raymond and Thilikos [33].

With this combinatorial theorem at hand, we implement the following win/win approach: If the treewidth of the input graph is less than $Ck$, we use known algorithms [5, 17] to solve the problem in single-exponential time. Otherwise the treewidth of the input graph is at least $Ck$, and it thus contains a sufficiently subdivided tetrahedron as a subgraph. We show that any path traversing the two-connected component of this subgraph can be made at least $k$ steps longer by rerouting it to and through the subgraph. Indeed we prove that every pair of distinct vertices $u, v$ in a $(\geq k)$-subdivided tetrahedron has two $(u,v)$-paths that are fully contained in this subgraph and whose lengths differ by at least $k$. See also Figure 1.

The algorithm for Exact Detour is based on the following idea. We run breadth-first search (BFS) starting at vertex $s$. Then, for every $(s,t)$-path $P$ of length exactly $d_G(s,t)+k$, at most $k$ levels of the BFS-tree contain more than one vertex of $P$. Using this property, we are able to devise a simple algorithm for Exact Detour that makes queries to an oracle for Longest Path and reduces the instance to a directed acyclic graph (DAG), where the problem can be solved in polynomial time. The oracle queries have parameter $k'$ bounded by $2k+1$, so an algorithm for Longest Path with time $c^{k'}n^{O(1)}$ translates into an algorithm for Exact Detour with time $c^{2k}n^{O(1)}$.

The remaining part of the paper is organized as follows: Section 2 contains preliminaries used in the technical part of the paper. In section 3, we give an algorithm for Longest Detour, while section 4 is devoted to Exact Detour. We provide a search-to-decision reduction for Longest Detour and Exact Detour in section 5.

**2. Preliminaries.** Graphs $G$ may be undirected or directed. We denote by $uv$ an edge joining vertices $u, v \in V(G)$. A *path* is a self-avoiding walk in $G$; the *length* of the path is its number of edges. An $(s,t)$-path for $s, t \in V(G)$ is a path that starts at $s$ and ends at $t$. We allow paths to have length 0, in which case $s = t$ holds. For a vertex set $X \subseteq V(G)$, denote by $G[X]$ the subgraph induced by $X$.

A *tree decomposition* $\mathcal{T}$ of an undirected graph $G$ is a pair $(T, \{X_t\}_{t \in V(T)})$,

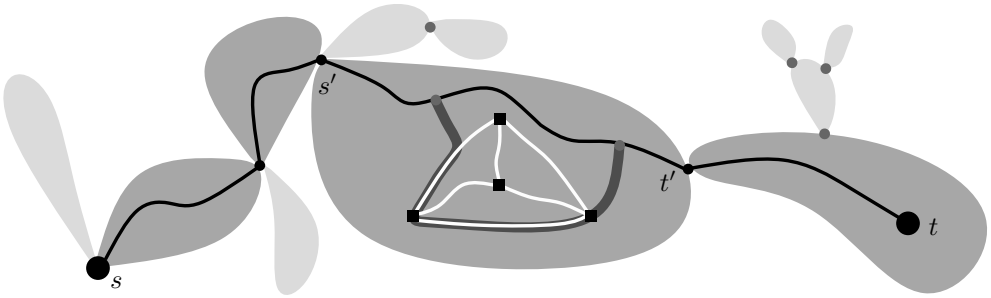



FIG. 1. *Overview of the high-treewidth situation in our algorithm for* LONGEST DETOUR. *Each* blob *is a block in the block-cut decomposition of $G$; the* dark gray blobs *together form the relevant part $G_{s,t}$ of $G$ (Definition* 3.1*). If a block of $G_{s,t}$ has high treewidth, it contains a large subdivided tetrahedron as a subgraph (Lemma* 3.10*), depicted in* white. *The shortest $(s,t)$-path (*black curve*) can be rerouted (*thick dark gray curve*) to and through the tetrahedron to become longer (Lemma* 3.9*).*

where $T$ is a tree in which every node $t$ is assigned a vertex subset $X_t \subseteq V(G)$, called a bag, such that the following three conditions hold:

(T1) Every vertex of $G$ is in at least one bag, that is, $V(G) = \bigcup_{t \in V(T)} X_t$.
(T2) For every $uv \in E(G)$, there is a node $t \in V(T)$ such that $X_t$ contains $u$ and $v$.
(T3) For every $u \in V(G)$, the set $T_u$ of all nodes of $T$ whose corresponding bags contain $u$, induces a connected subtree of $T$.

The *width* of the tree decomposition $\mathcal{T}$ is the integer $\max_{t \in V(T)} |X_t| - 1$, that is, the size of its largest bag minus 1. The *treewidth* of a graph $G$, denoted by $\mathrm{tw}(G)$, is the minimum width over all tree decompositions of $G$.

We will need two algorithmic results about treewidth: First, we require a single-exponential time algorithm for computing a constant approximation to the treewidth of a graph.

PROPOSITION 2.1 ([6]). *There is an algorithm that is given a graph $G$ and an integer $w$, runs in time $2^{O(w)} \cdot n$, and either outputs a tree decomposition of width at most $5w+4$ or correctly returns that $\mathrm{tw}(G) > w$ holds.*

Second, we need an algorithm for computing long paths in graphs of low treewidth.

PROPOSITION 2.2 ([5, 17]). *There is an algorithm that computes a longest path between two given vertices of a given $n$-vertex graph in time $2^{O(w)} \cdot n$ when a tree decomposition of width $w$ is given as additional input.*

Our main theorem uses topological graph minors, for which we introduce some notation here.

DEFINITION 2.3. *A* topological minor model *of $H$ in $G$ is a pair of functions $(f,p)$ with $f : V(H) \to V(G)$ and $p : E(H) \to 2^{E(G)}$ such that*

1. *$f$ is injective;*
2. *for every edge $uv \in E(H)$, the graph $G[p(uv)]$ is a path from $f(u)$ to $f(v)$ in $G$; and*
3. *for all edges $e, g \in E(H)$ with $e \neq g$, the paths $G[p(e)]$ and $G[p(g)]$ intersect only in endpoints or not at all.*

*The topological minor model $(f,p)$* induces *a subgraph $T$ of $G$, which consists of the union of all paths $G[p(uv)]$ over all $uv \in E(H)$. The vertices in $f(V(H))$ are the* branch vertices *of $T$, and $G[p(e)]$* realizes *the edge $e$ in $T$.*

**3. Win/win algorithm for LONGEST DETOUR.** Throughout this section, let $G$ be an undirected graph with $n$ vertices and $m$ edges, and let $s, t \in V(G)$ and $k \in \mathbb{N}$. We wish to decide in time $2^{O(k)} \cdot n^{O(1)}$ whether $G$ contains an $(s,t)$-path of length at least $d_G(s,t) + k$. We assume without loss of generality that $G$ is connected and that $s$ and $t$ are distinct (if $s = t$, then the only $(s,t)$-path is the path of length 0). Moreover, we can safely remove vertices $v$ that are not part of any $(s,t)$-path, as such vertices are irrelevant for the existence of a detour.

DEFINITION 3.1. *Let $G$ be a graph, and let $s, t \in V(G)$. The* $(s,t)$-relevant part of $G$ *is the graph induced by all vertices that are contained in at least one $(s,t)$-path. We denote it by $G_{s,t}$.*

The graph $G_{s,t}$ can be computed from $G$ in linear time. To this end, we use the *block-cut tree* of $G$, which is a tree where each vertex corresponds to a *block*, that is, a maximal biconnected component $B \subseteq V(G)$, or to a *cut vertex*, that is, a vertex whose removal disconnects the graph. A block $B$ and a cut vertex $v$ are adjacent in the block-cut tree if and only if $v \in B$. If $v$ is not a cut vertex, then there is a unique block $B_v$ that contains it.

LEMMA 3.2. *Let $s' = s$ if $s$ is a cut vertex, and $s' = B_s$ otherwise. Similarly let $t' = t$ if $t$ is a cut vertex, and $t' = B_t$ otherwise. Furthermore, let $P$ be the unique $(s', t')$-path in the block-cut tree of $G$. If $s \neq t$, then $G_{s,t}$ is the graph induced in $G$ by the union of all blocks visited by $P$.*

*Proof.* Let $v \in G_{s,t}$. Then there is an $(s,t)$-path that contains $v$; in particular, there are an $(s,v)$-path $p_1$ and a $(v,t)$-path $p_2$ such that $p_1$ and $p_2$ are internally vertex disjoint. If $v$ was not in one of the blocks visited by $P$, it would be hidden behind a cut vertex and $p_1$ and $p_2$ would have to intersect in the cut vertex; therefore, $v$ is contained in one of the blocks visited by $P$.

For the other direction, let $v$ be a vertex contained in a block $B$ visited by $P$. Suppose that $u$ is the cut vertex preceding $B$ in $P$ (or $u = s$ in case $B = B_s$) and $w$ is the cut vertex following $B$ in $P$ (or $w = t$ in case $B = B_t$). Then $u \neq w$ holds, and there are an $(s,u)$-path and a $(w,t)$-path that are vertex-disjoint. Since $B$ is biconnected, there are paths from $u$ to $v$ and from $v$ to $w$ that are internally vertex-disjoint. Combined, these path segments yield an $(s,t)$-path that visits $v$. ∎

We formulate an immediate implication of Lemma 3.2 that will be useful later.

COROLLARY 3.3. *The block-cut tree of $G_{s,t}$ is a $(B_s, B_t)$-path.*

Hopcroft and Tarjan [20] proved that the block-cut tree of a graph can be computed in linear time using depth-first search. Hence we obtain an algorithm for computing $G_{s,t}$ from $G$.

COROLLARY 3.4. *There is a linear-time algorithm that computes $G_{s,t}$ from $G$.*

**3.1. The algorithm.** By definition, the graph $G_{s,t}$ contains the same set of $(s,t)$-paths as $G$. Our algorithm for LONGEST DETOUR establishes a "win/win" situation as follows: We prove that if the treewidth of $G_{s,t}$ is "sufficiently large," then $(G, s, t, k)$ is a `YES` instance of LONGEST DETOUR (see Figure 1). Otherwise the treewidth is small, and we use a known treewidth-based dynamic programming algorithm for computing the longest $(s,t)$-path. Hence the algorithm builds upon the following subroutines:

1. The algorithm from Corollary 3.4, computing the relevant part $G_{s,t}$ of $G$ in time $O(n+m)$.

2. COMPUTE TREEWIDTH$(G, w)$ from Proposition 2.1, which is given $G$ and $w \in \mathbb{N}$ as input, and either constructs a tree decomposition $T$ of $G$ whose width is bounded by $5w + 4$ or outputs `LARGE`. If the algorithm outputs `LARGE`, then $\mathrm{tw}(G) > w$ holds. The running time is $2^{O(w)} \cdot n$.
3. LONGEST PATH$(G, T, s, t)$ from Proposition 2.2, which is given $G, s, t$ and additionally a tree decomposition $T$ of $G$, and outputs a longest $(s,t)$-path in $G$. The running time is $2^{O(w)} \cdot n^{O(1)}$, where $w$ denotes the width of $T$.

We now formalize what we mean by "sufficiently large" treewidth.

DEFINITION 3.5. *A function $f : \mathbb{N} \to \mathbb{N}$ is* detour-enforcing *if, for all $k \in \mathbb{N}$ and all graphs $G$ with vertices $s$ and $t$ such that $\mathrm{tw}(G_{s,t}) > f(k)$, the graph $G_{s,t}$ contains an $(s,t)$-path of length at least $d_G(s,t) + k$.*

THEOREM 3.6. *The function $k \mapsto 32k + 46$ is detour-enforcing.*

We defer the proof of this theorem to the following sections, and instead first state Algorithm D, which uses a detour-enforcing function $f$ to solve LONGEST DETOUR. Algorithm D turns out to be a fixed-parameter tractable algorithm already when any detour-enforcing function $f$ is known (as long as it is polynomial-time computable), and it becomes faster when detour-enforcing $f$ of slower growth are used.

ALGORITHM D (LONGEST DETOUR). *Let $f$ be a detour-enforcing function. Given $(G, s, t, k)$ as input, this algorithm decides whether the graph $G$ contains an $(s,t)$-path of length at least $d_G(s,t) + k$.*

D1 (Restrict to relevant part.) Compute $G_{s,t}$ using Corollary 3.4.
D2 (Compute shortest path.) Compute the distance $d$ between $s$ and $t$ in $G_{s,t}$.
D3 (Compute tree decomposition.) Call COMPUTE TREEWIDTH$(G_{s,t}, f(k))$.
  D3A (Small treewidth.) If the subroutine found a tree decomposition $T$, call the subroutine LONGEST PATH$(G_{s,t}, T, s, t)$. Output `YES` if there is an $(s,t)$-path of length at least $d + k$; otherwise output `NO`.
  D3B (Large treewidth.) If the subroutine returned `LARGE`, output `YES`.

We analyze the running time and correctness of Algorithm D.

LEMMA 3.7. *For any polynomial-time computable detour-enforcing function $f : \mathbb{N} \to \mathbb{N}$, Algorithm D solves* LONGEST DETOUR *in time $2^{O(f(k))} \cdot n^{O(1)}$.*

*Proof.* For the correctness, we first consider the case that COMPUTE TREEWIDTH outputs a tree decomposition $T$ of $G_{s,t}$ whose width is bounded by $5 \cdot f(k) + 4$. If so, step D3A invokes the algorithm for LONGEST PATH to compute a longest $(s,t)$-path in $G_{s,t}$ and outputs `YES` if and only if its length is at least $d(s,t) + k$. Since the $(s,t)$-paths in $G$ are precisely the $(s,t)$-paths in $G_{s,t}$, this output is correct. In the other case, COMPUTE TREEWIDTH outputs `LARGE`, which by Proposition 2.1 implies $\mathrm{tw}(G_{s,t}) > f(k)$. Since $f$ is detour-enforcing, the graph $G_{s,t}$ thus contains an $(s,t)$-path of length at least $d_G(s,t) + k$. Thus the output `YES` produced by the algorithm in step D3B is correct. We conclude that Algorithm D is correct.

For the running time, note that steps D1 and D2 run in polynomial time. Without loss of generality, $k \leq n$ holds, so computing $f(k)$ from $k$ takes time $n^{O(1)}$ by assumption on $f$. By Propositions 2.1 and 2.2, step D3 runs in time $2^{O(f(k))} n^{O(1)}$. We conclude that Algorithm D has the claimed running time. ∎

*Proof of Theorem* 1.1. We combine Theorem 3.6 and Lemma 3.7 to obtain an algorithm for LONGEST DETOUR that runs in single-exponential time as required. ∎

**3.2. Overview of the proof of Theorem 3.6.** In this proof, large subdivisions of the complete 4-vertex graph $K_4$, also called the *tetrahedron*, play an important

role. To prove Theorem 3.6, we first show that if a sufficiently large subdivided tetrahedron appears as a subgraph in $G_{s,t}$, then we can route some $(s,t)$-path through this subgraph and then exhibit a long detour within the subgraph. Then we prove that graphs of sufficiently large treewidth contain subdivided tetrahedra.

DEFINITION 3.8. *For $k \in \mathbb{N}$, a graph $F$ is a $(\geq k)$-*subdivided tetrahedron *if it can be obtained by replacing all edges of $K_4$ by internally vertex-disjoint paths of length at least $k$.*

Note that in the above definition, the path lengths do not need to agree for different edges. We show in section 3.3 that detours exist if the relevant part of the graph contains a subdivided tetrahedron as a subgraph.

LEMMA 3.9. *Let $G$ be a graph, let $s,t \in V(G)$, and let $k \in \mathbb{N}$. If $G_{s,t}$ contains a $(\geq k)$-subdivided tetrahedron as a subgraph, then $G$ contains an $(s,t)$-path of length at least $d_G(s,t)+k$.*

Since the graph obtained by subdividing each edge of $K_4$ exactly $k$ times is a planar graph on $O(k)$ vertices, the Excluded Grid Theorem of Robertson and Seymour [34] yields a function $f : \mathbb{N} \to \mathbb{N}$ such that every graph of treewidth at least $f(k)$ contains some $(\geq k)$-subdivided tetrahedron as a minor. Furthermore, since every subdivided tetrahedron has maximum degree 3, the tetrahedron occurs not only as a minor but also as a *subgraph* of $G$. Thus, Lemma 3.9 implies that $f$ is detour-enforcing, and a proof of this lemma immediately implies a weak version of Theorem 3.6.

By recent improvements on the Excluded Grid Theorem [7, 10], the function $f$ above is at most a polynomial. However, even equipped with this deep result we cannot obtain a single-exponential algorithm for LONGEST DETOUR using the approach of Lemma 3.7: It would require $f$ to be linear. In fact, excluding grids is too strong a requirement for us, since every function $f$ obtained as a corollary of the full Excluded Grid Theorem must be superlinear [35]. For this reason, we circumvent the Excluded Grid Theorem and prove the following lemma from more basic principles in section 3.4.

LEMMA 3.10. *Let $G$ be a graph, let $s,t \in V(G)$, and let $k \in \mathbb{N}$. If $\mathrm{tw}(G) \geq 32k+46$, then $G$ contains a subgraph that is isomorphic to a $(\geq k)$-subdivided tetrahedron.*

Together, Lemmas 3.10 and 3.9 imply Theorem 3.6.

*Proof of Theorem* 3.6. Let $G$ be a graph, let $s,t \in V(G)$, and let $k \in \mathbb{N}$ be such that $\mathrm{tw}(G_{s,t}) \geq 32k+46$. By Lemma 3.10, the graph $G_{s,t}$ contains a subdivided tetrahedron, and by Lemma 3.9 this implies that $G$ contains an $(s,t)$-path of length $d_G(s,t)+k$. This shows that $f$ is indeed detour-enforcing. ∎

**3.3. Proof of Lemma 3.9: Rerouting in subdivided tetrahedra.** Let $(G,s,t,k)$ be an instance for LONGEST DETOUR such that $s \neq t$ holds and $G_{s,t}$ contains a subgraph $M$ that is a $(\geq k)$-subdivided tetrahedron. We want to prove that $G_{s,t}$ contains an $(s,t)$-path of length at least $d_G(s,t)+k$. In fact, we construct the desired detour entirely in the subgraph $M$, for which reason we first need to route some $(s,t)$-path through $M$ via two "entry points," $u,v \in V(M)$; see Figure 1.

LEMMA 3.11. *There are two distinct vertices, $u,v \in V(M)$, and two vertex-disjoint paths, $P_s$ and $P_t$, in $G$ such that $P_s$ is an $(s,u)$-path, $P_t$ is a $(v,t)$-path, and they only intersect with $V(M)$ at $u$ and $v$, respectively.*

*Proof.* Since $M$ is biconnected, it is fully contained in a single block $B$ of $G_{s,t}$. By Corollary 3.3, the block-cut tree of $G_{s,t}$ is a path. Let $s'$ be the cut vertex preceding $B$ in this block-cut tree (or $s' = s$ if $B$ is the first block), and let $t'$ be the cut vertex following $B$ in the tree (or $t' = t$ if $B$ is the last block). Clearly $s', t' \in B$ and $s' \neq t'$ holds.

By the properties of the block-cut tree, there are an $(s, s')$-path $p_s$ and a $(t', t)$-path $p_t$, the two paths are vertex-disjoint, and they intersect $B$ only in $s'$ and $t'$, respectively. Since $B$ is biconnected, there are two vertex-disjoint paths from $\{s', t'\}$ to $V(M)$. Moreover, each path can be shortened if it intersects $V(M)$ more than once. Hence we have an $(s', u)$-path $p_1$ for some $u \in V(M)$ and a $(v, t')$-path $p_2$ for some $v \in V(M)$ with the property that $p_1$ and $p_2$ are disjoint and their internal vertices avoid $V(M)$.

We concatenate the paths $p_s$ and $p_1$ to obtain $P_s$ and the paths $p_2$ and $p_t$ to obtain $P_t$. ∎

Next we show that every subdivided tetrahedron contains a long detour between the two entry points; in fact, it contains a long detour between any two distinct vertices.

LEMMA 3.12. *For every $(\geq k)$-subdivided tetrahedron $M$ and every pair of distinct vertices $u, v \in V(M)$, there is a $(u, v)$-path of length at least $d_M(u, v) + k$ in $M$.*

*Proof.* Let us assume for contradiction that $M$ contains two distinct vertices $u, v$ between which there is no path of length at least $d_M(u, v) + k$. Write $\ell(P)$ for the length of a path. Then any pair $(P_1, P_2)$ of $(u, v)$-paths in the subdivided tetrahedron satisfies the inequality

$$\Big|\ell(P_1) - \ell(P_2)\Big| < k\,. \tag{3.1}$$

Indeed, if (3.1) were violated on any pair of $(u, v)$-paths $(P_1, P_2)$, then one of the paths, say $P_1$, would be at least $k$ steps longer than $P_2$. But since $\ell(P_2) \geq d_M(u, v)$ trivially holds, it would follow that $P_1$ has length at least $d_M(u, v) + k$, which contradicts our assumption.

In the following, let $b_1, \ldots, b_4$ be the four degree-3 vertices of $M$. We proceed by case analysis according to the relative positions of $u$, $v$, and the vertices $b_1, \ldots, b_4$. Up to symmetries of $K_4$, there are only three relevant cases to consider, which are shown in Figure 2. In each of the three cases, we choose two inequalities of type (3.1) in such a way that the inequalities together contradict the fact that the edges of the tetrahedron have been replaced by paths of length at least $k$.

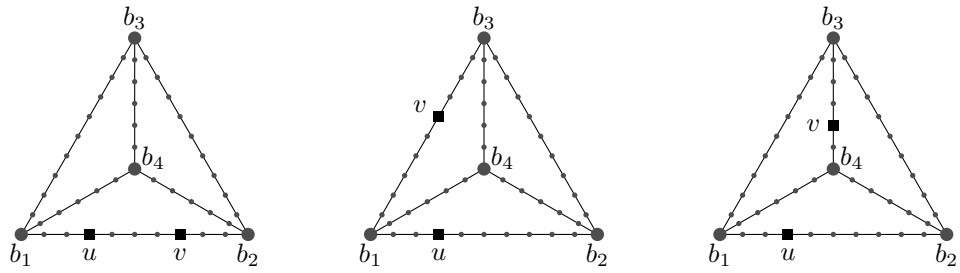
We call a path $P$ between vertices $x, y \in \{b_1, \ldots, b_4\} \cup \{u, v\}$ in $M$ *canonical* and abbreviate it by $xy$ if $P$ visits no other vertices from $\{b_1, \ldots, b_4\} \cup \{u, v\}$. Canonical paths $xy$ and $yz$ can be concatenated to a path $xyz$, which in turn can be decomposed unambiguously into $xy$ and $yz$.

*Case* (a): *Same edge.* If $u$ and $v$ lie on the same subdivided edge of the tetrahedron, we assume without loss of generality that they lie on the path from $b_1$ to $b_2$. We invoke (3.1) on the path pairs $(ub_1b_3b_4b_2v,\ ub_1b_4b_2v)$ and $(ub_1b_4b_3b_2v,\ ub_1b_3b_2v)$ to obtain the inequalities

$$\Big|\ell(ub_1b_3b_4b_2v) - \ell(ub_1b_4b_2v)\Big| < k \quad \text{and} \quad \Big|\ell(ub_1b_4b_3b_2v) - \ell(ub_1b_3b_2v)\Big| < k\,.$$

Summing these two inequalities, we obtain

$$\Big|\ell(ub_1b_3b_4b_2v) - \ell(ub_1b_4b_2v)\Big| + \Big|\ell(ub_1b_4b_3b_2v) - \ell(ub_1b_3b_2v)\Big| < 2k\,.$$



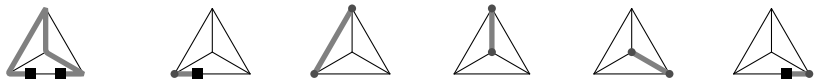

(a) Same edge. (b) Adjacent edges. (c) Nonadjacent edges.

FIG. 2. *Depicted are all three possible cases for the relative positions of vertices $u$ and $v$* (black squares) *in a $(\geq k)$-subdivided tetrahedron with degree-3 vertices $b_1, \ldots, b_4$* (large discs) *and at least $k = 5$ subdivision vertices* (small discs). *The vertices $u$ and $v$ lie on the same subdivided edge, on two adjacent subdivided edges, or on two nonadjacent subdivided edges of the subdivided tetrahedron.*

Applying the triangle inequality for sums of absolute values, this implies

$$\left|\ell(ub_1b_3b_4b_2v) - \ell(ub_1b_4b_2v) + \ell(ub_1b_4b_3b_2v) - \ell(ub_1b_3b_2v)\right| < 2k\,. \tag{3.2}$$

We expand all four paths in the sum into their canonical constituents; for example,



$$\ell(ub_1b_3b_4b_2v) = \ell(\ ub_1\ ) + \ell(\ b_1b_3\ ) + \ell(\ b_3b_4\ ) + \ell(\ b_4b_2\ ) + \ell(\ b_2v\ )\,.$$

After collecting canceling terms, inequality (3.2) drastically simplifies to

$$\left|2 \cdot \ell(\ b_3b_4\ )\right| < 2k\,.$$

This, however, contradicts the fact that $M$ is a $(\geq k)$-subdivided tetrahedron, in which the canonical path $b_3b_4$ has length $\ell(b_3b_4) \geq k$.

*Case* (b): *Adjacent edges.* If $u$ and $v$ lie on adjacent subdivided edges, say $u$ lies on $b_1b_2$ and $v$ on $b_1b_3$, then we invoke (3.1) on the path pairs $(ub_2b_3b_4b_1v,\ ub_2b_4b_1v)$ and $(ub_1b_4b_2b_3v,\ ub_1b_4b_3v)$ to obtain the inequalities

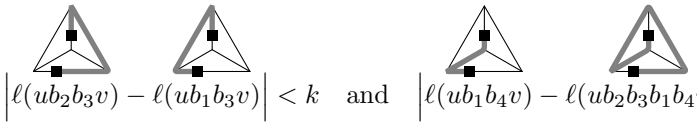
$$\left|\ell(ub_2b_3b_4b_1v) - \ell(ub_2b_4b_1v)\right| < k \quad \text{and} \quad \left|\ell(ub_1b_4b_2b_3v) - \ell(ub_1b_4b_3v)\right| < k\,.$$

By an argument analogous to that of the first case, we arrive at

$$\left|2 \cdot \ell(\ b_2b_3\ )\right| < 2k\,,$$

which is in contradiction with the construction of $M$.

*Case* (c): *Nonadjacent edges.* If $u$ and $v$ lie on nonadjacent subdivided edges, say $u$ lies on $b_1b_2$ and $v$ on $b_3b_4$, then we invoke (3.1) on the path pairs $(ub_2b_3v,\ ub_1b_3v)$ and $(ub_1b_4v,\ ub_2b_3b_1b_4v)$ to obtain the inequalities



$$\left|\ell(ub_2b_3v) - \ell(ub_1b_3v)\right| < k \quad \text{and} \quad \left|\ell(ub_1b_4v) - \ell(ub_2b_3b_1b_4v)\right| < k\,.$$

By an argument analogous to that of the previous cases, we arrive at

$$\left|2 \cdot \ell(\ b_1b_3\ )\right| < 2k\,,$$

which is in contradiction with the construction of $M$. ∎

*Proof of Lemma* 3.9. Let $d = d_G(s,t)$ be the length of a shortest $(s,t)$-path in $G$. Let $M$ be a $(\geq k)$-subdivided tetrahedron in $G_{s,t}$, and let $P_s$, $P_t$, $u$, and $v$ be the objects guaranteed by Lemma 3.11. Let $P_{uv}$ be a shortest $(u,v)$-path that only uses edges of $M$; its length is $d_M(u,v)$. Since the combined path $P_s, P_{uv}, P_t$ is an $(s,t)$-path, its length is at least $d$.

Finally, Lemma 3.12 guarantees that there is a $(u,v)$-path $Q_{uv}$ in $M$ whose length is at least $d_M(u,v)+k$. Therefore, the length of the $(s,t)$-path $P_s, Q_{uv}, P_t$ satisfies

$$\begin{aligned}\ell(P_s) + \ell(Q_{uv}) + \ell(P_t) &\geq \ell(P_s) + (d_M(u,v)+k) + \ell(P_t)\\ &= \ell(P_s) + \ell(P_{uv}) + \ell(P_t) + k \geq d + k\,.\end{aligned}$$

We constructed a path of length at least $d+k$ as required. ∎

**3.4. Proof of Lemma 3.10: Subdivided tetrahedra from high treewidth.** We prove Lemma 3.10 using a result of Leaf and Seymour [27], which shows that undirected graphs $G$ of treewidth $\Omega(k)$ contain every $k$-vertex forest $F$ as a minor. Their result additionally guarantees that the minor model of $F$ intersects a "highly connected region" of $G$ in a specific way. We adapt this result to the setting of *topological* minors by a standard argument. To prove Lemma 3.10, we choose a suitable forest $F$ and exploit the highly connected region of $G$ to extend the minor model of the forest $F$ to a large subdivided tetrahedron. A similar strategy was used by Raymond and Thilikos [33] to prove the existence of $k$-wheel minors in graphs of treewidth $\Omega(k)$.

To state the result by Leaf and Seymour, we require some additional notions from graph minor theory that will be used only in this section. First, their result is formulated in terms of minor models; we have only introduced *topological* minor models so far.

Definition 3.13. *Let $H$ and $G$ be undirected graphs. A* minor model *of $H$ in $G$ is a function $f : V(H) \to 2^{V(G)}$ such that*

1. *$G[f(v)]$ is connected for all $v \in V(H)$;*
2. *$f(u) \cap f(v) = \emptyset$ for all $u, v \in V(H)$ with $u \neq v$; and*
3. *for all $uv \in E(H)$, the graph $G$ has at least one edge between the vertex sets $f(u)$ and $f(v)$.*

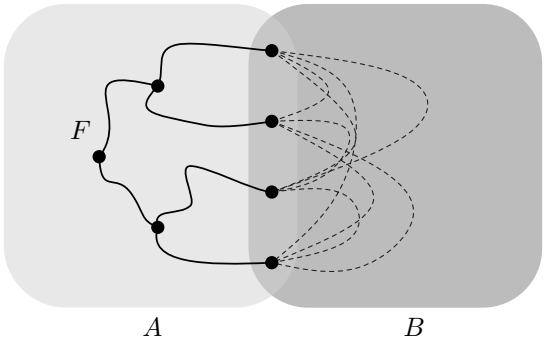



FIG. 3. *Illustration of the separation guaranteed by Lemma* 3.16. *The left side $G[A]$ contains a topological minor model of $F$ whose leaves are contained in $A \cap B$. The right side is highly connected in that $A \cap B$ is linked in $G[B]$. (In the separation $(A, B)$ actually constructed in the proof of Lemma* 3.16*, three further vertices of the minor model of $F$ would also be contained in $A \cap B$, which is not depicted here.)*

We also require the standard notions of a *linked* set and a *separation*, and the less standard notion of *left-containment*, which connects minor models and separations.

DEFINITION 3.14. *Let $G$ be a graph. We say that a set $S \subseteq V(G)$ is* linked in $G$ *if, for every $X, Y \subseteq S$ with $|X| = |Y|$, there are $|X|$ vertex-disjoint paths between $X$ and $Y$ that intersect $S$ exactly at its endpoints.*

*The pair $(A, B)$ with $A, B \subseteq V(G)$ is a* separation in $G$ *if the sets $A \setminus B$ and $B \setminus A$ are nonempty and no edge runs between them. The* order *of $(A, B)$ is the cardinality of $A \cap B$.*

*The separation $(A, B)$* left-contains *$H$ if $G[A]$ contains a minor model $f$ of $H$ that satisfies $|f(v) \cap A \cap B| = 1$ for every $v \in V(H)$.*

Leaf and Seymour [27, Proposition 4.3] proved that, for any forest $F$, every large-treewidth graph has a separation that left-contains $F$ and whose right side is linked.

LEMMA 3.15 ([27]). *Let $F$ be a nonempty forest, and let $G$ be a graph. If $\mathrm{tw}(G) \geq \frac{3}{2}|V(F)|-1$, then there exists a separation $(A, B)$ of order $|V(F)|$ in $G$ such that $(A, B)$ left-contains $F$ and $A \cap B$ is linked in $G[B]$.*

Since we only consider forests $F$ of maximum degree 3, we reformulate Lemma 3.15 in terms of topological minors, and we transform left-containment of $F$ to a weaker property that is guaranteed only at leaves of $F$. See Figure 3 for an illustration. The proof of our reformulation proceeds along the lines of the proof that $F$-minors are topological $F$-minors if $F$ has maximum degree 3 [13, Proposition 1.7.3].

LEMMA 3.16. *Let $F$ be a nonempty forest with maximum degree* 3*, and let $G$ be a graph. If $\mathrm{tw}(G) \geq \frac{3}{2}|V(F)| - 1$, then $G$ has a separation $(A, B)$ of order $|V(F)|$ such that*

(i) *there is a topological minor model $(f, p)$ of $F$ in $G[A]$;*
(ii) *for every leaf $v \in V(F)$, we have $f(v) \in A \cap B$;*
(iii) *$A \cap B$ is linked in $G[B]$.*

*Proof.* By Lemma 3.15, there is a separation $(A, B)$ of order $|V(F)|$ in $G$ such that $(A, B)$ left-contains $F$ and $A \cap B$ is linked in $G[B]$. Thus (iii) holds directly, and it remains to construct a topological minor model $(f, p)$ satisfying the other two conditions.

Since $(A, B)$ left-contains $F$, the graph $G[A]$ contains a minor model $f'$ of $F$ satisfying $|f'(v) \cap A \cap B| = 1$ for all $v \in V(F)$. From the minor model $f'$, we now construct a subgraph $X$ of $G$ that is the image of our topological minor model. Starting from the empty graph, we proceed as follows:

1. For each edge $e \in E(F)$, let $f'_e \in E(G)$ be an arbitrary edge between the vertex sets $f'(u)$ and $f'(v)$; such an edge exists by definition since $f$ is a minor model of $F$ in $G$. We add the edge $f'_e$ and its two endpoints to $X$.
2. For each vertex $v \in V(F)$, we define a set $X_v \subseteq f'(v) \subseteq V(G)$ as follows:
   (a) If $v$ is a leaf in $F$, we add the unique vertex in $f'(v) \cap A \cap B$ to $X_v$. Moreover, for the unique edge $e \in E(F)$ incident with $v$, we add the unique endpoint of $f'_e$ that is contained in $f'(v)$. (Now $X_v$ contains one or two vertices of $f'(v)$.)
   (b) If $v$ is not a leaf in $F$, we iterate over each edge $e \in E(F)$ incident with $v$ and add the unique endpoint of $f'_e$ to $X_v$ that is contained in $f'(v)$. (Now $X_v$ contains between one and $\deg(v)$ many vertices of $f'(v)$.)

   Now we add a connected subgraph $T_v$ of $G[f'(v)]$ to $X$ that contains $X_v$ and has the minimum number of edges.[2]

This concludes the definition of the subgraph $X$ in $G$. We now read off a topological minor model $(f, p)$ of $F$ from $X$. To this end, we define $f : V(F) \to V(G)$:

- For $v \in V(F)$ of degree 3, let $f(v)$ be the unique vertex in $f'(v)$ that has degree 3 in $X$. To see that this vertex exists and is unique, recall that $X[f'(v)]$ induces the tree $T_v$ that minimally connects the vertices in $X_v$. If $|X_v| = 3$, then $T_v$ has three leaves and a unique internal node of degree 3. If $|X_v| \in \{1, 2\}$, then one of the terminal nodes in $X_v$ has degree 3 in $X$.
- For $v \in V(F)$ of degree 2, let $f(v)$ be any vertex in $X[f'(v)]$.
- For $v \in V(F)$ of degree 1, let $f(v)$ be the unique vertex in $f'(v) \cap A \cap B$. Note that this guarantees claim (ii) of the lemma.

It is clear that $f$ is injective, and it can be checked easily that every edge $uv \in E(F)$ is realized by a path in $X$; these paths together give the function $p$ required for a topological minor model. Furthermore, the minimal choice of $T_v$ above ensures that such paths intersect only at endpoints. Thus claim (i) holds too. ∎

We now use Lemma 3.16 to prove the existence of large subdivided tetrahedra in graphs of high treewidth, thus proving Lemma 3.10.

*Proof of Lemma* 3.10. Let $k \in \mathbb{N}$, and let $G$ be a graph with $\mathrm{tw}(G) \geq 32k + 46 \geq \frac{3}{2}(21k+31) - 1$. We invoke Lemma 3.16 with a particular forest $F = T \dot\cup P$ on $21k+31$ vertices, where $T$ and $P$ are trees defined as follows (see also Figure 4):

- $T$ is obtained from the rooted full binary tree with 8 leaves by subdividing each edge exactly $k$ times. Let $L_T = \{x_1, \ldots, x_8\}$ be the set of leaves of $T$. The tree $T$ has $14k + 15$ vertices.
- $P$ is obtained from the path with 8 vertices by first subdividing each edge $k$ times, and then attaching degree-1 vertices $y_1, \ldots, y_8$ to the nonsubdivision vertices $z_1, \ldots, z_8$. We consider these vertices to be ordered along the natural order of the main path in $P$. Let $L_P = \{y_1, \ldots, y_8\}$ be the set of leaves of $P$. The tree $P$ has $7k + 16$ vertices.

Overall, $F = T \cup P$ has $21k + 31$ vertices and maximum degree 3, so Lemma 3.16 yields a separation $(A, B)$ in $G$ of order $|V(F)|$ such that $A \cap B$ is linked in $G[B]$, and there is a topological minor model $(f, p)$ of $F$ in $G[A]$ with $f(L_T \cup L_P) \subseteq A \cap B$.

[2] That is, $T_v$ is a minimum Steiner tree in the graph $G[f'(v)]$ for the terminal set $X_v$.

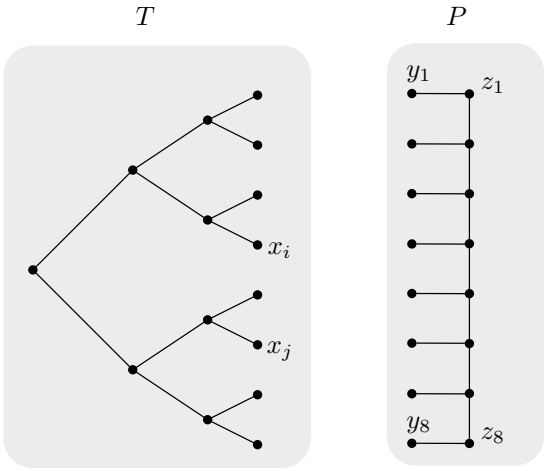



FIG. 4. *The forest $F = T \cup P$ that is found as a topological minor in $G[A]$, which will be extended to a model of the $(\geq k)$-subdivided tetrahedron by matching up its leaves using a linkage in $G[B]$. Each line between vertices in $T$ and between $z_1, \ldots, z_8$ represents a path of length $k+1$. The leaves of $T$ are labeled $x_1, \ldots, x_8$, and the ordering will be chosen in the proof of Lemma* 3.10.

We now show that the vertex-disjoint paths linking $A \cap B$ on the right can be used to complete the topological minor model of $F$ in $G[A]$ to one of a $(\geq k)$-subdivided tetrahedron in $G$.

For vertices $v \in V(F)$ and vertex sets $X \subseteq V(G)$, we abbreviate $v^G = f(v)$ and $X^G = f(X)$. For subgraphs $S$ of $F$, we write $S^G = f(S)$ for the image of $S$, which consists of $f(V(S))$ and the paths $p(uv)$ for $uv \in E(S)$.

Since $A \cap B$ is linked in $G[B]$, there are vertex-disjoint paths $Q_1, \ldots, Q_8$ between $L_T^G$ and $L_P^G$ in $G[B]$ that intersect $A \cap B$ only at their endpoints. Recall that the leaves $L_P = \{y_1, \ldots, y_8\}$ of $P$ and their neighbors $\{z_1, \ldots, z_8\}$ are ordered in the natural way along the path in $P$. This ordering is preserved by the topological minor model; that is, $y_1^G, \ldots, y_8^G$ follow along the main path in $P^G$ in this order. Without loss of generality, we label the paths $Q_1, \ldots, Q_8$ and the leaves $x_1, \ldots, x_8$ of $T$ in such a way that $Q_i$ for $i \in \{1, \ldots, 8\}$ is the path from $y_i^G$ to $x_i^G$.

Let $r$ denote the root of $T$ and write $\mathrm{root}(T^G) = r^G$. Write $T_1^G, T_2^G$ for the two subtrees of $T^G$ rooted at the two topmost branch vertices below $r^G$ in $T^G$. Let $\mathrm{lca}(x_1^G, x_8^G)$ denote the lowest common ancestor of $x_1^G$ and $x_8^G$ in $T^G$. In the following, we distinguish two cases according to the relative placement of $x_1^G$ and $x_8^G$ with respect to $T_1^G$ and $T_2^G$; see also Figure 5.

*Case* 1*:* $\mathrm{lca}(x_1^G, x_8^G) \neq \mathrm{root}(T^G)$ holds; that is, $x_1^G$ and $x_8^G$ are both contained in the same subtree. Assume without loss of generality that both are contained in the subtree $T_1^G$; the argument is otherwise analogous by exchanging the roles of $T_1^G$ and $T_2^G$. Let $i, j \in \{2, \ldots, 7\}$, with $i < j$, be arbitrary such that $x_i^G$ and $x_j^G$ are contained in $T_2^G$. Recall that a $(\geq k)$-subdivided tetrahedron has four branch vertices $b_1, \ldots, b_4$ of degree 3. To exhibit this tetrahedron as a topological minor in $G$, we map them to the following four vertices, respectively:

$$b_1^G := \mathrm{lca}(x_1^G, x_8^G), \quad b_2^G := \mathrm{lca}(x_i^G, x_j^G), \quad b_3^G := z_i^G, \quad b_4^G := z_j^G .$$

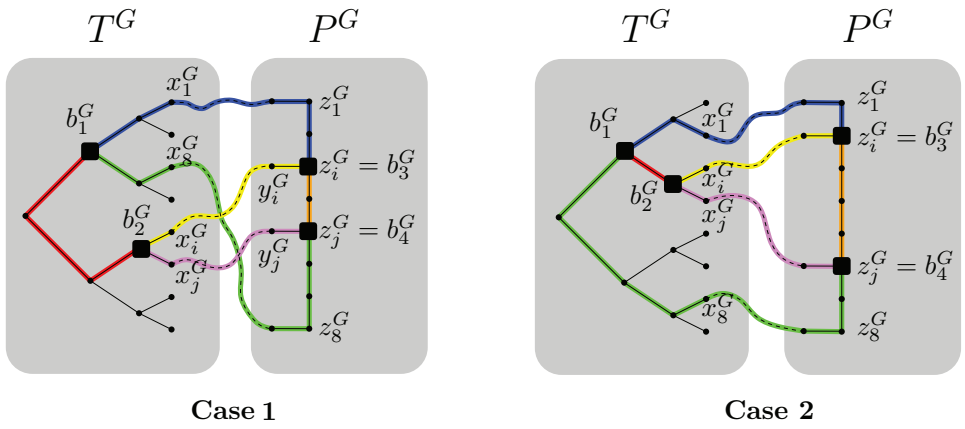



FIG. 5. *The two cases in the proof of Lemma* 3.10*, along with the configurations of* $T^G = f(T)$ *and* $P^G = f(P)$*, and the corresponding* $(\geq k)$*-subdivided tetrahedra.* Bullets *correspond to the images of nonsubdivision vertices of* $F$ *in* $G$*.* Lines *represent paths in* $G$ *whose length is guaranteed to be at least* $k+1$ *by subdivisions in* $F$*, with the exception of paths between* $y_r^G$ *and* $z_r^G$*, which have no length guarantee.* Dashed curves *correspond to the paths* $Q_1, \ldots, Q_8$ *between the leaves in* $f(F)$*.* Big squares *represent the degree-3 branch vertices* $b_1^G, b_2^G, b_3^G, b_4^G$ *of the subdivided tetrahedron.* Colored paths *depict the six paths of the subdivided tetrahedron. Some vertices of* $F$ *are hidden to avoid cluttering.*

These vertices are distinct, because $b_1^G$ is contained in $T_1^G$ and $b_2^G$ in $T_2^G$, and $b_3^G$ and $b_4^G$ are distinct vertices of $P^G$. We realize the paths of the subdivided tetrahedron in $G$ as follows (cf. Figure 5):

- $b_1b_2$ (*red*) is realized as the unique $(b_1^G, b_2^G)$-path in $T^G$.
- $b_3b_4$ (*orange*) is realized as the unique $(b_3^G, b_4^G)$-path in $P^G$.
- $b_1b_3$ (*blue*) is realized as the unique $(b_1^G, b_3^G)$-path in $T^G \cup P^G \cup Q_1$.
- $b_1b_4$ (*green*) is realized as the unique $(b_1^G, b_4^G)$-path in $T^G \cup P^G \cup Q_8$.
- $b_2b_3$ (*yellow*) is realized as the unique $(b_2^G, b_3^G)$-path in $T^G \cup P^G \cup Q_i$.
- $b_2b_4$ (*purple*) is realized as the unique $(b_2^G, b_4^G)$-path in $T^G \cup P^G \cup Q_j$.

For each of the six edges $b_ib_j$ with $i < j$, we have thus constructed a path in $G$. The paths are internally vertex-disjoint and have length at least $k$. Thus they form a topological minor model of a $(\geq k)$-subdivided tetrahedron in $G$ as required.

*Case 2:* $\mathrm{lca}(x_1^G, x_8^G) = \mathrm{root}(T^G)$ holds; that is, $x_1^G$ and $x_8^G$ are in different subtrees $T_1^G$ and $T_2^G$. Without loss of generality, let $x_1^G$ be contained in $T_1^G$. The tree $T_1$ has three other leaves apart from $x_1$; we ignore the direct sibling of $x_1$ and let $x_i$ and $x_j$ with $i < j$ be the remaining two leaves in $T_1$. To exhibit a subdivided tetrahedron as a topological minor in $G$, we define $b_1^G := \mathrm{lca}(x_1^G, x_i^G)$ and use the same definitions for $b_2^G, \ldots, b_4^G$ as in the previous case. We can now define the path connections between $b_1^G, \ldots, b_4^G$ exactly as before. Thus, a topological minor model of a $(\geq k)$-subdivided tetrahedron exists in $G$ in this case as well. ∎

**4. An algorithm for the EXACT DETOUR problem.** Recall that in the EXACT DETOUR problem, we wish to find an *exact* $k$*-detour*, that is, an $(s,t)$-path of length exactly $d(s,t)+k$. We devise an algorithm for EXACT DETOUR by reducing the problem to EXACT PATH. In EXACT PATH, we are given $(G, s, t, k)$ and wish to determine whether there is an $(s,t)$-path of length exactly $k$.

Before we state the algorithm, let us introduce some notation. For any $x \in V(G)$, we denote with $d(x)$ the distance $d_G(s,x)$ from $s$ to $x$ in $G$, and we let the $i$th level

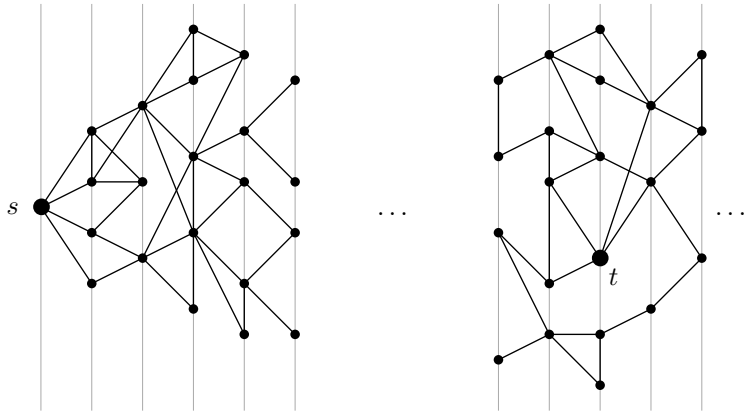



FIG. 6. *The* fine vertical lines *in this drawing of an example graph represent BFS levels, that is, vertices whose distance $d(v)$ from $s$ is equal.*

of $G$ be the set of all vertices $x$ with $d(x) = i$, as shown in Figure 6. We also abbreviate $d = d_G(s,t)$. For any $i, j$ with $0 \leq i < j \leq d$, we denote by $G_{[i,j]}$ the graph induced by the vertices $x$ with $i \leq d(x) \leq j$. These graphs can be computed in linear time using BFS starting at $s$. We abbreviate $G_i = G_{[i,i]}$. Furthermore, we write $G_{(a,b]}$ for the graph obtained from $G_{[a,b]}$ by deleting all edges in $G_a$, and we define $G_{[a,b)}$ and $G_{(a,b)}$ likewise.

The main observation leading to our algorithm is that every exact $k$-detour $P$ must traverse at least $d$ edges $uv$ with $d(v) = d(u) + 1$. As a consequence, $P$ cannot make detours in many levels, as formalized below.

DEFINITION 4.1. *Let $P$ be a path in $G$, and let $i \in \mathbf{N}$. We say that $i$ is a detour-level of $P$ if $P$ touches two or more vertices of $G_i$, or if $i > d$ holds and $P$ touches $G_i$.*

PROPOSITION 4.2. *If $P$ is an $(s,t)$-path in $G$ of length exactly $d + k$, then it has at most $k$ detour-levels. Moreover, at most $2k - 1$ edges of $P$ have both endpoints in detour-levels.*

*Proof.* Let $P$ be an $(s,t)$-path of length $d + k$. Clearly $P$ touches every level $G_i$ with $i \in \{0, \ldots, d\}$ at least once. Let $D_1$ be the number of indices $i \in \{0, \ldots, d\}$ such that $P$ touches $G_i$ more than once. Furthermore, let $D_2$ be the number of indices $i$ with $i > d$ such that $P$ touches $G_i$. Now $P$ has at least $d + D_1 + D_2 + 1$ vertices, because it must have $d + 1$ vertices to get from $s$ to $t$, and uses at least $D_1$ additional vertices in levels $G_i$ with $i \leq d$, and at least $D_2$ in levels with $i > d$. Since $P$ has exactly $d + k + 1$ vertices, we thus obtain $D_1 + D_2 \leq k$. The first claim follows, since $D_1 + D_2$ is the number of detour-levels of $P$.

For the second claim, note that for every $i \in \{0, \ldots, d-1\}$, the path $P$ has an edge $uv$ with $d(u) = i$ and $d(v) = i + 1$. Since there are at most $k$ detour-levels, at most $k - 1$ of these $d$ edges have both endpoints in detour-levels. Thus, at least $d - (k-1)$ of these edges $uv$ are incident to a non-detour-level. Since $P$ has exactly $d + k$ edges, at most $2k - 1$ of them have both endpoints in detour-levels. ∎

By Proposition 4.2, at most $2k - 1$ edges of $P$ touch detour-levels of $P$, and in particular, every exact $k$-detour $P$ can make only "short-range" detours of length at most $2k$. These short-range detours can occur far apart, but we can succinctly encode them in the following DAG with edge weights and multiple edges.

DEFINITION 4.3. *For a graph $G$ with vertices $s, t \in V(G)$, and $k \in \mathbb{N}$, the range-$k$ detour graph $G^{(k)}$ is a directed multigraph on the same vertex set as $G$. Starting with the empty set, we construct the edges of $G^{(k)}$ as follows:*

- *For each $u, v \in V(G)$ with $d(u) < d(v) < d$ and each $w \in \{1, \ldots, 2k+1\}$, we add an edge $uv$ of weight $w$ if there is a $(u, v)$-path of length exactly $w$ in the graph $G_{(d(u),d(v))}$.*
- *For each $u \in V(G)$ with $d - k \leq d(u) < d$ and each $w \in \{1, \ldots, 2k+1\}$, we add the edge $ut$ of weight $w$ if there is a $(u, t)$-path of length exactly $w$ in $G_{(d(u),d+k]}$.*

The construction of $G^{(k)}$ can be carried out in polynomial time if given access to an oracle for deciding the existence of $(s, t)$-paths with length at most $2k + 1$. Furthermore, the graph $G^{(k)}$ is acyclic by construction. Its relevance is given in the following lemma.

LEMMA 4.4. *The graph $G$ contains an $(s, t)$-path of length $d+k$ if and only if $G^{(k)}$ contains an $(s, t)$-path of total weight exactly $d + k$.*

*Proof.* Let $Q$ be an $(s, t)$-path in $G^{(k)}$ of total weight $d + k$, with edges $e_1, \ldots, e_\ell$ for some $\ell \in \mathbf{N}$. Each edge $e_i$ has an associated positive integer weight $w_i$, and $w_1 + \cdots + w_\ell = d + k$ holds. By construction of $G^{(k)}$, each edge $e_i$ corresponds to a path $Q_i$ in $G$ of length exactly $w_i$, and the internal points of $Q_i$ are in levels of $G$ that are strictly between the levels of $Q_i$'s endpoints. Since $Q$ is a path, the endpoint of $Q_i$ is the starting point of $Q_{i+1}$ for all $i \in \{1, \ldots, \ell - 1\}$. These facts imply the paths $Q_1, \ldots, Q_\ell$ are pairwise internally vertex-disjoint, and their concatenation forms an $(s, t)$-path in $G$ of length exactly $d + k$.

Conversely, if $P$ is an $(s, t)$-path in $G$ of total length $d + k$, then we claim that $G^{(k)}$ contains an $(s, t)$-path $Q$ of total weight $d + k$. Indeed, let $P$ be the path $v_0 e_1 v_1 \ldots v_{d+k-1} e_{d+k} v_{d+k}$ with $v_0 = s$ and $v_{d+k} = t$. We now partition the edges of $P$ into subpaths and show that each subpath causes a corresponding edge in $G^{(k)}$ to exist; together, these edges form a path of total weight $d + k$. So let $a_1, \ldots, a_\ell \in \{0, \ldots, d + k - 1\}$ be the unique increasing sequence of all indices less than $d+k$ such that $v_{a_j}$ is not in a detour-level of $P$. Since $v_0 = s$ is the only vertex in its level, we have $a_1 = 0$.

By Proposition 4.2, $P$ has at most $2k - 1$ edges with both endpoints in detour-levels. Thus the number of edges $w_j$ in the subpath $P_j$ of $P$ that starts at $v_{a_j}$ and ends at $v_{a_{j+1}}$ is at most $2k+1$. Let $d_j$ be the distance $d(s, v_{a_j})$ between $s$ and $v_{a_j}$, and analogously let $d_{j+1}$ be the distance $d(s, v_{a_{j+1}})$ between $s$ and $v_{a_{j+1}}$. By choice of $a_j$ and $a_{j+1}$, the path $P$ intersects levels $G_{d_j}$ and $G_{d_{j+1}}$ exactly once. Thus, $P_j$ is a path in graph $G_{(d_j, d_{j+1})}$. By construction, this implies that $G^{(k)}$ contains an edge $v_{a_j} v_{a_{j+1}}$ of weight exactly $w_j$.

Let $u = v_{a_\ell}$. Concatenating the sequence $P_1, \ldots, P_\ell$ gives an $(s, u)$-path in $G$. Moreover, the edges in $G^{(k)}$ corresponding to these subpaths form an $(s, u)$-path in $G^{(k)}$ whose total weight is equal to the length of $P_1, \ldots, P_\ell$. For the final segment from $u$ to $t$, we again invoke Proposition 4.2, which implies that $d(u) \geq d - k$ holds and that the subpath from $u$ to $t$ has length $w'$ at most $2k + 1$. By construction, this will add an edge $ut$ of weight $w'$ to $G^{(k)}$. Overall, we can conclude that $G^{(k)}$ has an $(s, t)$-path of total weight exactly $d + k$. ∎

Lemma 4.4 allows us to solve the exact $k$-detour problem in $G$ by determining whether $G^{(k)}$ contains an $(s, t)$-path of weight $d + k$. Since $G^{(k)}$ is a weighted DAG, and each weight is a small integer, this latter problem is polynomial-time solvable.

LEMMA 4.5. *There is a polynomial-time Turing reduction from* EXACT DETOUR *to* EXACT PATH. *On instances with parameter $k$, all queries have parameter at most $2k+1$.*

*Proof.* Given an instance $(G, s, t, k)$ for EXACT DETOUR, the reduction first annotates each vertex $v \in V(G)$ with its level $d(v)$, and computes all level sets $V(G_i)$. This process takes linear time $O(n+m)$ using BFS from $s$. Next, the algorithm computes the graph $G^{(k)}$, which can be done by querying the oracle for EXACT PATH for each $(G_{d(u),d(v)}, u, v, w)$ with $d(u) < d(v)$ and $w \in \{1, \ldots, 2k+1\}$; this takes polynomial time. Finally, we turn the weighted graph $G^{(k)}$ into an unweighted graph $G'$ by replacing each edge $uv$ of weight $w$ with a fresh $(u,v)$-path of length $w$. Clearly, the $(s,t)$-paths in $G^{(k)}$ of total weight $W$ correspond to $(s,t)$-paths in $G'$ of length $W$. Since $G'$ is an unweighted DAG, we can determine the existence of an $(s,t)$-path of length $W$ in polynomial time using BFS. ∎

Pipelining Lemma 4.5 with the fastest known $k$-path algorithms [3, 37, 26] gives single-exponential algorithms for EXACT DETOUR. Let us note, however, that slight generalizations of these algorithms are required, as they do not explicitly allow for specifying terminals $s$ and $t$. We describe these modifications below.

The randomized algorithm of Björklund et al. [3] is for a variant of EXACT PATH where the terminal vertices $s$ and $t$ are not given; that is, any path of length exactly $k$ yields a `YES` instance. Their algorithm applies to our problem as well, with the same running time. For readers familiar with the algorithm in Björklund et al. [3], we sketch an argument for this observation here. Recall that the idea is to reduce the problem to checking whether a certain polynomial is identically zero; this polynomial is defined by summing over all possible labeled walks of length $k$ (see [12, section 10.4.3]). We modify the polynomial by adding two leaf-edges, one incident to $s$ and one to $t$, and restricting our attention to $(k+2)$-walks that contain these two edges. The required information for such walks can still be computed efficiently as before. The crux of the proof is that walks that are not paths cancel out over a field of characteristic two; this argument works by a local reorientation of segments of the walk—an operation that does not change the vertices of the walk and must therefore keep $s$ and $t$ fixed. The graph $G$ contains a $k$-path if and only if the polynomial is not identically zero; this property remains true in our case. The rest of the argument goes through as before, so the algorithm of Björklund et al. applies to EXACT PATH with no significant loss in the running time. The randomized algorithm by Koutis and Williams [26] for directed graphs can be adapted to EXACT PATH in an analogous way.

The deterministic algorithm of Zehavi [37] also does not take terminal vertices as part of the input, but this algorithm actually applies even to the weighted version of the problem, to which the variant with specified terminals can easily be reduced. In the weighted $k$-path problem, we are given a graph $G$, weights $w_e$ on each edge, a number $k$, and a number $W$, and the question is whether there is a path of length exactly $k$ such that the sum of all edge weights along the path is at most $W$. We observe the following simple reduction from EXACT PATH (with terminal vertices $s$ and $t$) to the weighted $k$-path problem (without terminal vertices): Every edge gets assigned the same edge weight 2, except for the new leaf-edges at $s$ and $t$, which get edge weight 1. Now every path with exactly $k+2$ edges has weight at most $W = 2k+2$ if and only if the first and the last edges of the path are the leaf-edges we added. Due to this reduction, Zehavi's algorithm applies to EXACT PATH with no significant loss in the running time. We are ready to prove Theorem 1.2.

*Proof of Theorem* 1.2. Let $(G, s, t, k)$ be an instance of Exact Detour. We use Lemma 4.5 to solve this instance using an oracle for Exact Path which only needs to answer queries with parameter $k'$ at most $2k+1$. To answer these queries, we use the best known algorithm as a subroutine. Using the randomized algorithm by Björklund et al. [3], we obtain a running time of $1.657^{k'} n^{O(1)} \leq 2.746^k n^{O(1)}$. The reduction in Lemma 4.5 applies verbatim to directed graphs as well. In this case, the fastest known randomized algorithm is by Koutis and Williams [25, 36, 26] and runs in time $2^{k'} n^{O(1)}$, which yields a $4^k n^{O(k)}$ time algorithm for Exact Detour. Using the deterministic algorithm by Zehavi [37], we obtain a running time of $2.597^{k'} n^{O(1)} \leq 6.745^k n^{O(1)}$ for Exact Detour both in directed and undirected graphs. ∎

**5. Search-to-decision reduction.** Our graph-minor-based algorithm for the Longest Detour problem does not necessarily construct a detour, since the algorithm merely outputs "yes" on large-treewidth graphs. Similarly, our dynamic programming algorithm uses an algorithm for Exact Path as a subroutine, and these algorithms typically also do not output a path.

In this section, we present a search-to-decision reduction for Longest Detour and Exact Detour that uses a simple downward self-reducibility argument. In the interest of brevity, we focus on Longest Detour: Given a decision oracle for this problem, we show how to construct a detour with only polynomial overhead in the running time.

Algorithm B (*search-to-decision reduction*). *Given* $(G, s, t, k)$ *and access to an oracle for* Longest Detour*, this algorithm computes an* $(s, t)$*-path of length at least* $d_G(s, t) + k$.

B0 (Trivial case.) If $(G, s, t, k)$ is a `NO` instance of Longest Detour, halt and reject.

B1 (Add a new shortest path.) Let $d := d_G(s, t)$. Add $d$ new edges $p_1, \ldots, p_d$ and $d - 1$ new vertices, forming a new shortest $(s, t)$-path of length $d$.

B2 (Delete superfluous edges.) For each $e \in E(G) \setminus \{p_1, \ldots, p_d\}$ sequentially: If $(G - e, s, t, k)$ is a `YES` instance of Longest Detour, then set $G := G - e$.

B3 (Delete the added path.) Let $G := G - \{p_1, \ldots, p_d\}$.

B4 (Output detour.) Now $G$ is an $(s, t)$-path of length at least $d_G(s, t) + k$.

Lemma 5.1. *Given oracle access to* Longest Detour*, Algorithm B outputs a path of length at least* $d_G(s, t) + k$ *in polynomial time.*

*Proof.* It is clear that Algorithm B runs in polynomial time; we only need to show correctness. Let $G_0$ be the graph at the beginning of the algorithm, and let $G_1$ be the remaining graph after B2. If $(G_0, s, t, k)$ is a `YES` instance, then $(G_1, s, t, k)$ is also a `YES` instance. Moreover, deleting any edge from $E(G) \setminus \{p_1, \ldots, p_d\}$ would turn it into a `NO` instance. Since $(G_1, s, t, k)$ is a `YES` instance, it contains an $(s, t)$-path on edges $q_1, \ldots, q_\ell$ for $\ell \geq d_G(s, t) + k$. Since the `YES` instance is minimal (no edge could be deleted without turning it into a `NO` instance), we have $E(G_1) = \{p_1, \ldots, p_d\} \cup \{q_1, \ldots, q_\ell\}$.

Finally, since the path $p_1, \ldots, p_d$ was added to $G$ in an edge-disjoint way, it is edge-disjoint from every other $(s, t)$-path in $G$. Therefore, by removing $\{p_1, \ldots, p_d\}$ from $G_1$, we get the path $q_1, \ldots, q_\ell$, of length at least $d_G(s, t) + k$. ∎

**6. Conclusion.** We have shown that finding detours of length at least $k$ is fixed-parameter tractable in undirected graphs, and likewise for finding detours of length exactly $k$ in undirected and directed graphs. For undirected planar graphs, our algorithm for Exact Detour can be sped up to run in time $2^{O(\sqrt{k})} n^{O(1)}$ by using

an improved algorithm for Exact Path. Likewise, there is an $2^{O(\sqrt{k})}n^{O(1)}$ time algorithm for Longest Detour: Planar graphs admit detour-enforcing functions of order $O(\sqrt{k})$, since the $k$-subdivided tetrahedron is a minor of a square grid with side-lengths $\Omega(\sqrt{k})$, which in turn is a minor of every planar graph of treewidth $\Omega(\sqrt{k})$ [35].

The main problem left open by our work is to settle the complexity of Longest Detour in directed graphs. So far, our attempts to mimic the algorithm for undirected graphs did not succeed. Kawarabayashi and Kreutzer [23] proved that every directed graph of sufficiently large directed treewidth contains a large directed grid as a "butterfly minor," a particular minor notion in directed graphs. It is tempting to use their result in order to obtain a win/win algorithm for Longest Detour on directed graphs, but there are several obstacles on this path. In fact, we do not know if the problem is in the class XP, that is, if there is an algorithm that solves directed Longest Detour in time $n^{f(k)}$ for some function $f$. Can one even find an $(s,t)$-path of length at least $d_G(s,t)+1$ in polynomial time, that is, any path that is not a shortest path?

**Acknowledgments.** We thank Daniel Lokshtanov, Meirav Zehavi, Petr Golovach, Saket Saurabh, Stephan Kreutzer, and Tobias Mömke for fruitful discussions.

## REFERENCES

[1] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, *Solving MAX-r-SAT above a tight lower bound*, Algorithmica, 61 (2011), pp. 638–655, https://doi.org/10.1007/s00453-010-9428-7.

[2] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. ACM, 42 (1995), pp. 844–856, https://doi.org/10.1145/210332.210337.

[3] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, *Narrow sieves for parameterized paths and packings*, J. Comput. System Sci., 87 (2017), pp. 119–139, https://doi.org/10.1016/j.jcss.2017.03.003.

[4] H. L. Bodlaender, *On linear time minor tests with depth-first search*, J. Algorithms, 14 (1993), pp. 1–23, https://doi.org/10.1006/jagm.1993.1001.

[5] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof, *Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth*, Inform. and Comput., 243 (2015), pp. 86–111, https://doi.org/10.1016/j.ic.2014.12.008.

[6] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk, *A $c^k n$ 5-approximation algorithm for treewidth*, SIAM J. Comput., 45 (2016), pp. 317–378, https://doi.org/10.1137/130947374.

[7] C. Chekuri and J. Chuzhoy, *Polynomial bounds for the grid-minor theorem*, J. ACM, 63 (2016), 40, https://doi.org/10.1145/2820609.

[8] J. Chen, J. Kneis, S. Lu, D. Mölle, S. Richter, P. Rossmanith, S.-H. Sze, and F. Zhang, *Randomized divide-and-conquer: Improved path, matching, and packing algorithms*, SIAM J. Comput., 38 (2009), pp. 2526–2547, https://doi.org/10.1137/080716475.

[9] J. Chen, S. Lu, S.-H. Sze, and F. Zhang, *Improved algorithms for path, matching, and packing problems*, in Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2007, pp. 298–307.

[10] J. Chuzhoy and Z. Tan, *Towards tight(er) bounds for the excluded grid theorem*, in Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2019, pp. 1445–1464, https://doi.org/10.1137/1.9781611975482.88.

[11] R. Crowston, M. Jones, G. Muciaccia, G. Philip, A. Rai, and S. Saurabh, *Polynomial kernels for lambda-extendible properties parameterized above the Poljak-Turzik bound*, in IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), LIPIcs 24, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, pp. 43–54, https://doi.org/10.4230/LIPIcs.FSTTCS.2013.43.

[12] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, 2015, https://doi.org/10.1007/978-3-319-21275-3.

[13] R. Diestel, *Graph Theory*, 5th ed., Grad. Texts in Math. 173, Springer-Verlag, 2017.

[14] Z. Dvořák and M. Mnich, *Large independent sets in triangle-free planar graphs*, SIAM J.

Discrete Math., 31 (2017), pp. 1355–1373, https://doi.org/10.1137/16M1061862.
[15] M. Etscheid and M. Mnich, *Linear kernels and linear-time algorithms for finding large cuts*, Algorithmica, 80 (2018), pp. 2574–2615, https://doi.org/10.1007/s00453-017-0388-z.
[16] F. V. Fomin and P. Kaski, *Exact exponential algorithms*, Comm. ACM, 56 (2013), pp. 80–88, https://doi.org/10.1145/2428556.2428575.
[17] F. V. Fomin, D. Lokshtanov, F. Panolan, and S. Saurabh, *Efficient computation of representative families with applications in parameterized and exact algorithms*, J. ACM, 63 (2016), 29, https://doi.org/10.1145/2886094.
[18] G. Gutin, E. J. Kim, M. Lampis, and V. Mitsou, *Vertex cover problem parameterized above and below tight bounds*, Theory Comput. Syst., 48 (2011), pp. 402–410, https://doi.org/10.1007/s00224-010-9262-y.
[19] G. Gutin, L. van Iersel, M. Mnich, and A. Yeo, *Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables*, J. Comput. System Sci., 78 (2012), pp. 151–163, https://doi.org/10.1016/j.jcss.2011.01.004.
[20] J. Hopcroft and R. Tarjan, *Algorithm* 447: *Efficient algorithms for graph manipulation*, Comm. ACM, 16 (1973), pp. 372–378, https://doi.org/10.1145/362248.362272.
[21] F. Hüffner, S. Wernicke, and T. Zichner, *Algorithm engineering for color-coding with applications to signaling pathway detection*, Algorithmica, 52 (2008), pp. 114–132, https://doi.org/10.1007/s00453-007-9008-7.
[22] R. Impagliazzo and R. Paturi, *On the complexity of k-SAT*, J. Comput. System Sci., 62 (2001), pp. 367–375, https://doi.org/10.1006/jcss.2000.1727.
[23] K. Kawarabayashi and S. Kreutzer, *The directed grid theorem*, in Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC), ACM, 2015, pp. 655–664, https://doi.org/10.1145/2746539.2746586.
[24] J. Kneis, D. Mölle, S. Richter, and P. Rossmanith, *Divide-and-color*, in Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG), Lecture Notes in Comput. Sci. 4271, Springer, 2006, pp. 58–67, https://doi.org/10.1007/11917496_6.
[25] I. Koutis, *Faster algebraic algorithms for path and packing problems*, in Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP), Lecture Notes in Comput. Sci. 5125, Springer, 2008, pp. 575–586, https://doi.org/10.1007/978-3-540-70575-8_47.
[26] I. Koutis and R. Williams, *Algebraic fingerprints for faster algorithms*, Comm. ACM, 59 (2016), pp. 98–105, https://doi.org/10.1145/2742544.
[27] A. Leaf and P. D. Seymour, *Tree-width and planar minors*, J. Combin. Theory Ser. B, 111 (2015), pp. 38–53, https://doi.org/10.1016/j.jctb.2014.09.003.
[28] M. Mahajan and V. Raman, *Parameterizing above guaranteed values: MaxSat and MaxCut*, J. Algorithms, 31 (1999), pp. 335–354, https://doi.org/10.1006/jagm.1998.0996.
[29] M. Mahajan, V. Raman, and S. Sikdar, *Parameterizing above or below guaranteed values*, J. Comput. System Sci., 75 (2009), pp. 137–153, https://doi.org/10.1016/j.jcss.2008.08.004.
[30] M. Mnich, *Large independent sets in subquartic planar graphs*, in Proceedings of the 10th International Workshop on Algorithms and Computation (WALCOM), Lecture Notes in Comput. Sci. 9627, Springer, 2016, pp. 209–221, https://doi.org/10.1007/978-3-319-30139-6_17.
[31] B. Monien, *How to find long paths efficiently*, in Analysis and Design of Algorithms for Combinatorial Problems, North-Holland Math. Stud. 109, North-Holland, 1985, pp. 239–254, https://doi.org/10.1016/S0304-0208(08)73110-4.
[32] C. H. Papadimitriou and M. Yannakakis, *On limited nondeterminism and the complexity of the V-C dimension*, J. Comput. System Sci., 53 (1996), pp. 161–170, https://doi.org/10.1006/jcss.1996.0058.
[33] J. Raymond and D. M. Thilikos, *Low polynomial exclusion of planar graph patterns*, J. Graph Theory, 84 (2017), pp. 26–44, https://doi.org/10.1002/jgt.22009.
[34] N. Robertson and P. D. Seymour, *Graph minors. V. Excluding a planar graph*, J. Combin. Theory Ser. B, 41 (1986), pp. 92–114, https://doi.org/10.1016/0095-8956(86)90030-4.
[35] N. Robertson, P. D. Seymour, and R. Thomas, *Quickly excluding a planar graph*, J. Combin. Theory Ser. B, 62 (1994), pp. 323–348, https://doi.org/10.1006/jctb.1994.1073.
[36] R. Williams, *Finding paths of length k in $O^*(2^k)$ time*, Inform. Process. Lett., 109 (2009), pp. 315–318, https://doi.org/10.1016/j.ipl.2008.11.004.
[37] M. Zehavi, *Mixing color coding-related techniques*, in Proceedings of the 23rd Annual European Symposium on Algorithms (ESA), Lecture Notes in Comput. Sci. 9294, Springer, 2015, pp. 1037–1049, https://doi.org/10.1007/978-3-662-48350-3_86.