An Exponential Time Parameterized Algorithm for Planar Disjoint Paths*

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

Pranabendu Misra
pmisra@mpi-inf.mpg.de
Max-Planck Institute for Informatics,
Saarland Informatics Campus
Saarbrücken, Germany

Michał Pilipczuk
michal.pilipczuk@mimuw.edu.pl
Institute of Informatics, University of
Warsaw
Warsaw, Poland

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

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

ABSTRACT

In the Disjoint Paths problem, the input is an undirected graph G on n vertices and a set of k vertex pairs, $\{s_i, t_i\}_{i=1}^k$, and the task is to find k pairwise vertex-disjoint paths such that the i 'th path connects s_i to t_i . In this paper, we give a parameterized algorithm with running time $2^{O(k^2)} n^{O(1)}$ for Planar Disjoint Paths, the variant of the problem where the input graph is required to be planar. Our algorithm is based on the unique linkage/treewidth reduction theorem for planar graphs by Adler et al. [JCTB 2017], the algebraic co-homology based technique developed by Schrijver [SICOMP 1994] for Disjoint Paths on directed planar graphs, and one of the key combinatorial insights developed by Cygan et al. [FOCS 2013] in their algorithm for Disjoint Paths on directed planar graphs. To the best of our knowledge our algorithm is the first parameterized algorithm to exploit that the treewidth of the input graph is small in a way completely different from the use of dynamic programming.

CCS CONCEPTS

• **Theory of computation** → **Fixed parameter tractability; Network flows.**

KEYWORDS

disjoint paths, planar graphs, parameterized complexity, network flow, homology

*This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416, no. 715744 and no. 677651), United States–Israel Binational Science Foundation grant no. 2018302, Israel Science Foundation (ISF) individual research grant (grant no. 1176/18) and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

[†] Additionally affiliated with University of Bergen, Norway and IRL 2000 ReLaX.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

STOC '20, June 22–26, 2020, Chicago, IL, USA

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6979-4/20/06...\$15.00

<https://doi.org/10.1145/3357713.3384250>

ACM Reference Format:

Daniel Lokshtanov, Pranabendu Misra, Michał Pilipczuk, Saket Saurabh, and Meirav Zehavi. 2020. An Exponential Time Parameterized Algorithm for Planar Disjoint Paths. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20)*, June 22–26, 2020, Chicago, IL, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3357713.3384250>

1 INTRODUCTION

In the Disjoint Paths problem, the input is an undirected graph G on n vertices and a set of k vertex pairs, $\{s_i, t_i\}_{i=1}^k$, and the task is to find k pairwise vertex-disjoint paths connecting s_i to t_i for each $i \in \{1, \dots, k\}$. The Disjoint Paths problem is a fundamental routing problem that finds applications in VLSI layout and virtual circuit routing, and has a central role in Robertson and Seymour's Graph Minors series. We refer to surveys such as [19, 39] for a detailed overview. The Disjoint Paths problem was shown to be NP-complete by Karp (who attributed it to Knuth) in a followup paper [22] to his initial list of 21 NP-complete problems [21]. It remains NP-complete even if G is restricted to be a grid [27, 30]. On directed graphs, the problem remains NP-hard even for $k = 2$ [18]. For undirected graphs, Perl and Shiloach [31] designed a polynomial time algorithm for the case where $k = 2$. Then, the seminal work of Robertson and Seymour [33] showed that the problem is polynomial time solvable for every fixed k . In fact, they showed that it is *fixed parameter tractable (FPT)* by designing an algorithm with running time $f(k)n^3$. The currently fastest parameterized algorithm for Disjoint Paths has running time $h(k)n^2$ [23]. However, all we know about h and f is that they are computable functions. That is, we still have no idea about what the running time dependence on k really is. Similarly, the problem appears difficult in the realm of approximation, where one considers the optimization variant of the problem where the aim is to find disjoint paths connecting as many of the $\{s_i, t_i\}$ pairs as possible. Despite substantial efforts, the currently best known approximation algorithm remains a simple greedy algorithm that achieves approximation ratio $O(\sqrt{n})$.

The Disjoint Paths problem has received particular attention when the input graph is restricted to be planar [2, 12, 15, 38]. Adler et al. [2] gave an algorithm for Disjoint Paths on planar graphs (Planar Disjoint Paths) with running time $2^{2^{O(k)}} n^2$, giving at least

a concrete form for the dependence of the running time on k for planar graphs. Schrijver [38] gave an algorithm for Disjoint Paths on *directed* planar graphs with running time $n^{O(k)}$, in contrast to the NP-hardness for $k = 2$ on general directed graphs. Almost 20 years later, Cygan et al. [12] improved over the algorithm of Schrijver and showed that Disjoint Paths on directed planar graphs is FPT by giving an algorithm with running time $2^{2^{O(k^2)}} n^{O(1)}$. The Planar Disjoint Paths problem is well-studied also from the perspective of approximation algorithms, with a recent burst of activity [7–11]. Highlights of this work include an approximation algorithm with factor $O(n^{9/19} \log^{O(1)} n)$ [8] and, under reasonable complexity-theoretic assumptions, hardness of approximating the problem within a factor of $2^{\Omega(\frac{1}{(\log \log n)^2})}$ [10].

In this paper, we consider the parameterized complexity of Planar Disjoint Paths. Prior to our work, the fastest known algorithm was the $2^{2^{O(k)}} n^2$ time algorithm of Adler et al. [2]. Double exponential dependence on k for a natural problem on planar graphs is something of an outlier—the majority of problems that are FPT on planar graphs enjoy running times of the form $2^{O(\sqrt{k} \text{ polylog } k)} n^{O(1)}$ (see, e.g., [13, 16, 17, 26, 32]). This, among other reasons (discussed below), led Adler [1] to pose as an open problem in GROW 2013¹ whether Planar Disjoint Paths admits an algorithm with running time $2^{k^{O(1)}} n^{O(1)}$. By integrating tools with origins in algebra and topology, we resolve this problem in the affirmative. In particular, we prove the following.

THEOREM 1.1. *The Planar Disjoint Paths problem is solvable in time $2^{O(k^2)} n^{O(1)}$.*

In addition to its value as a stand-alone result, our algorithm should be viewed as a piece of an on-going effort of many researchers to make the Graph Minor Theory of Robertson and Seymour algorithmically efficient. The graph minors project is abound with powerful algorithmic and structural results, such as the algorithm for Disjoint Paths [33], Minor Testing [33] (given two undirected graphs, G and H on n and k vertices, respectively, the goal is to check whether G contains H as a minor), the structural decomposition [34] and the Excluded Grid Theorem [36]. Unfortunately, all of these results suffer from such bad hidden constants and dependence on the parameter k that they have gotten their own term—“galactic algorithms” [28].

It is the hope of many researchers that, in time, algorithms and structural results from Graph Minors can be more algorithmically efficient, perhaps even practically applicable. Substantial progress has been made in this direction, examples include the simpler decomposition theorem of Kawarabayashi and Wollan [25], the faster algorithm for computing the structural decomposition of Grohe et al. [20], the improved unique linkage theorem of Kawarabayashi and Wollan [24], the linear excluded grid theorem on minor free classes of Demaine and Hajiaghayi [14], paving the way for the theory of Bidimensionality [13], and the polynomial grid minor theorem of Chekuri and Chuzhoy [6]. The algorithm for Disjoint

Paths is a cornerstone of the entire Graph Minor Theory, and a vital ingredient in the $g(k)n^3$ -time algorithm for Minor Testing. Therefore, *efficient* algorithms for Disjoint Paths and Minor Testing are necessary and crucial ingredients in an algorithmically efficient Graph Minors theory. This makes obtaining $2^{\text{poly}(k)} n^{O(1)}$ time algorithms for Disjoint Paths and Minor Testing a tantalizing and challenging goal.

Theorem 1.1 is a necessary basic step towards achieving this goal—a $2^{\text{poly}(k)} n^{O(1)}$ time algorithms for Disjoint Paths on general graphs also has to handle planar inputs, and it is easy to give a reduction from Planar Disjoint Paths to Minor Testing in such a way that a $2^{\text{poly}(k)} n^{O(1)}$ time algorithm for Minor Testing would imply a $2^{\text{poly}(k)} n^{O(1)}$ time algorithms for Planar Disjoint Paths. In addition to being a necessary step in the formal sense, there is strong evidence that an efficient algorithm for the planar case will be useful for the general case as well—indeed the algorithm for Disjoint Paths of Robertson and Seymour [33] relies on topology and essentially reduces the problem to surface-embedded graphs. Thus, an efficient algorithm for Planar Disjoint Paths represents a speed-up of the base case of the algorithm for Disjoint Paths of Robertson and Seymour. Coupled with the other recent advances [6, 13, 14, 20, 24, 25], this gives some hope that $2^{\text{poly}(k)} n^{O(1)}$ time algorithms for Disjoint Paths and Minor Testing may be within reach.

Known Techniques and Obstacles in Designing a $2^{\text{poly}(k)}$ Algorithm. All known algorithms for both Disjoint Paths and Planar Disjoint Paths have the same high level structure. In particular, given a graph G we distinguish between the cases of G having “small” or “large” treewidth. In case the treewidth is large, we distinguish between two further cases: either G contains a “large” clique minor or it does not. This results in the following case distinctions.

- (1) **Treewidth is small.** Let the treewidth of G be w . Then, we use the known dynamic programming algorithm with running time $2^{O(w \log w)} n^{O(1)}$ [37] to solve the problem. It is important to note that, assuming the Exponential Time Hypothesis (ETH), there is no algorithm for Disjoint Paths running in time $2^{o(w \log w)} n^{O(1)}$ [29], nor an algorithm for Planar Disjoint Paths running in time $2^{o(w)} n^{O(1)}$ [4].
- (2) **Treewidth is large and G has a large clique minor.** In this case, we use the good routing property of the clique to find an irrelevant vertex and delete it without changing the answer to the problem. Since this case will not arise for graphs embedded on a surface or for planar graphs, we do not discuss it in more detail.
- (3) **Treewidth is large and G has no large clique minor.** Using a fundamental structure theorem for minors called the “flat wall theorem”, we can conclude that G contains a large planar piece of the graph and a vertex v that is sufficiently insulated in the middle of it. Applying the unique linkage theorem [35] to this vertex, we conclude that it is irrelevant and remove it. For planar graphs, one can use the unique linkage theorem of Adler et al. [2]. In particular, we use the following result:

Any instance of Disjoint Paths consisting of a planar graph with treewidth at least $82k^{3/2}2^k$ and k terminal

¹The conference version of [2] appeared in 2011, before [1]. The document [1] erroneously states the open problem for Disjoint Paths instead of for Planar Disjoint Paths. That Planar Disjoint Paths is meant is evident from the statement that a $2^{2^{O(k)}} n^{O(1)}$ time algorithm is known.

pairs contains a vertex v such that every solution to Disjoint Paths can be replaced by an equivalent one whose paths avoid v .

This result says that if the treewidth of the input planar graph is (roughly) $\Omega(2^k)$, then we can find an irrelevant vertex and remove it. A natural question is whether we can guarantee an irrelevant vertex even if the treewidth is $\Omega(\text{poly}(k))$. Adler and Krause [3] exhibited a planar graph G with $k+1$ terminal pairs such that G contains a $(2^k + 1) \times (2^k + 1)$ grid as a subgraph, Disjoint Paths on this input has a unique solution, and the solution uses all vertices of G ; in particular, *no vertex of G is irrelevant*. This implies that the irrelevant vertex technique can only guarantee a treewidth of $\Omega(2^k)$, even if the input graph is planar.

Combining items (1) and (3), we conclude that the known methodology for Disjoint Paths can only guarantee an algorithm with running time $2^{2^{O(k)}} n^2$ for Planar Disjoint Paths. Thus, a $2^{\text{poly}(k)} n^{O(1)}$ -time algorithm for Planar Disjoint Paths appears to require entirely new ideas. As this obstacle was known to Adler et al. [1], it is likely to be the main motivation for Adler to pose the existence of a $2^{\text{poly}(k)} n^{O(1)}$ time algorithm for Planar Disjoint Paths as an open problem.

Our Methods. Our algorithm is based on a novel combination of two techniques that do not seem to give the desired outcome when used on their own. The first ingredient is the treewidth reduction theorem of Adler et al. [2] that proves that given an instance of Planar Disjoint Paths, the treewidth can be brought down to $2^{O(k)}$ (explained in item (3) above). This by itself is sufficient for an FPT algorithm (this is what Adler et al. [2] do), but as explained above, it seems hopeless that it will bring a $2^{\text{poly}(k)} n^{O(1)}$ -time algorithm.

We circumvent the obstacle by using an algorithm for a more difficult problem with a worse running time, namely, Schrijver's $n^{O(k)}$ -time algorithm for Disjoint Paths on directed planar graphs [38]. Schrijver's algorithm has two steps: a "guessing" step where one (essentially) guesses the homology class of the solution paths, and then a surprising homology-based algorithm that, given a homology class, finds a solution in that class (if one exists) in polynomial time. Our key insight is that for Planar Disjoint Paths, if the instance that we are considering has been reduced according to the procedure of Adler et al. [2], then we only need to iterate over $2^{O(k^2)}$ homology classes in order to find the homology class of a solution, if one exists. The proof of this key insight is highly non-trivial, and builds on a cornerstone ingredient of the recent FPT algorithm of Cygan et al. [12] for Disjoint Paths on directed planar graphs. To the best of our knowledge, this is the first algorithm that finds the exact solution to a problem that exploits that the treewidth of the input graph is small in a way that is different from doing dynamic programming. A technical overview of our methods will appear in the next section, and the detailed proofs are deferred to the full version of the paper.

2 ENUMERATING HOMOLOGY CLASSES

In this section we present an overview of our arguments to reduce the number of homology classes to $2^{O(k^2)}$.

Figure 1: Flow at a vertex.

Homology. Our starting point is Schrijver's view [38] of a collection of "non-crossing" (but possibly not vertex- or even edge-disjoint) sets of walks as flows. To work with flows (defined immediately), we deal with directed graphs. (In this context, undirected graphs are treated as directed graphs by replacing each edge by two parallel arcs of opposite directions.) Specifically, we denote an instance of Directed Planar Disjoint Paths as a tuple (D, S, T, g, k) where D is a directed plane graph, $S, T \subseteq V(D)$, $k = |S|$ and $g : S \rightarrow T$ is bijective. Then, a *solution* is a set $\mathcal{P}$ of pairwise vertex-disjoint directed paths in D containing, for each vertex $s \in S$, a path directed from s to $g(s)$.

In the language of flows, each arc of D is assigned a word with letters in $T \cup T^{-1}$ (that is, we treat the set of vertices T also as an alphabet), where $T^{-1} = \{t^{-1} : t \in T\}$. This collection of words is denoted by $(T \cup T^{-1})^*$ and let 1 denote the empty word. A word is *reduced* if, for all $t \in T$, the letters t and t^{-1} do not appear consecutively. Then, a *flow* is an assignment of reduced words to arcs that satisfies two constraints. First, when we concatenate the words assigned to the arcs incident to a vertex $v \notin S \cup T$ in clockwise order, where words assigned to ingoing arcs are reversed and their letters negated, the result (when reduced) is the empty word 1 (see Fig. 1). This is an algebraic interpretation of the standard flow-conservation constraint. Second, when we do the same operation with respect to a vertex $v \in S \cup T$, then when the vertex is in S , the result is $g(s)$ (rather than the empty word), and when it is in T , the result is t . There is a natural association of flows to solutions: for every $t \in T$, assign the letter t to all arcs used by the path from $g^{-1}(t)$ to t .

Roughly speaking, Schrijver proved that if a flow ϕ is given along with the instance (D, S, T, g, k) , then in *polynomial time* we can either find a solution or determine that there is no solution "similar to ϕ ". Specifically, two flows are *homologous* (which is the notion of similarity) if one can be obtained from the other by a set of "face operations" defined as follows.

DEFINITION 2.1. Let D be a directed plane graph with outer face f , and denote the set of faces of D by $\mathcal{F}$. Two flows ϕ and ψ are homologous if there exists a function $h : \mathcal{F} \rightarrow (T \cup T^{-1})^*$ such that (i) $h(f) = 1$, and (ii) for every arc $e \in A(D)$, $h(f_1)^{-1} \cdot \phi(e) \cdot h(f_2) = \psi(e)$ where f_1 and f_2 are the faces at the left-hand side and the right-hand side of e , respectively.

Then, a slight modification of Schrijver's theorem [38] readily gives the following corollary.

COROLLARY 2.1. There is a polynomial-time algorithm that, given an instance (D, S, T, g, k) of Directed Planar Disjoint Paths, a flow ϕ and a subset $X \subseteq A(D)$, either finds a solution of $(D - X, S, T, g, k)$

Figure 2: Two different ways of extracting a walk from a flow.

Figure 3: Face operations.

or decides that there is no solution of it such that the “flow associated with it” and ϕ are homologous in D .

Discrete Homotopy and Our Objective. While the language of flows and homology can be used to phrase our arguments, it also makes them substantially longer and somewhat obscure because it brings rise to multiple technicalities. For example, different sets of non-crossing walks may correspond to the same flow (see Fig. 2). Instead, we define a notion of *discrete homotopy*, inspired by (standard) homotopy. Specifically, we deal only with collections of non-crossing and *edge-disjoint* walks, called *weak linkages*. Then, two weak linkages are *discretely homotopic* if one can be obtained from the other by using “face operations” that push/stretch its walks across faces and keep them non-crossing and edge-disjoint (see Fig. 4). The discrete homotopy is an equivalence relation defined by face operations (see Fig 3). We note that the order in which face operations are applied is important in discrete homotopy (unlike homology)—we cannot stretch a walk across a face if no walk passes its boundary, but we can execute operations that will move a walk to that face, and then stretch it. We translate Corollary 2.1 to discrete homotopy (and undirected graphs) to derive the following result.

LEMMA 2.1. *There is a polynomial-time algorithm that, given an instance (G, S, T, g, k) of Planar Disjoint Paths, a weak linkage $\mathcal{W}$ in G and a subset $X \subseteq E(G)$, either finds a solution of $(G - X, S, T, g, k)$ or decides that no solution of it is discretely homotopic to $\mathcal{W}$ in G .*

In light of this result, our objective is reduced to the following.

Compute a collection of weak linkages such that if there exists a solution, then there also exists a solution (possibly a different one!) that is discretely homotopic to one of the weak linkages in our collection. To prove Theorem 1.1, the size of the collection should be upper bounded by $2^{O(k^2)}$.

Figure 4: Moving a walk of a weak linkage (in blue) onto the Steiner tree (the walk in purple) with “face operations” (e.g. a sub-path of the blue path is pushed giving the green sub-path).

Key Player: Backbone Steiner Tree. A key to the proof of our theorem is a very careful construction of a so-called *Backbone Steiner tree*. We use the term Steiner tree to refer to any tree in the *radial completion* of G (the graph obtained by placing a vertex on each face and making it adjacent to all vertices incident to the face) whose set of leaves is precisely $S \cup T$. In the first step, we consider an arbitrary Steiner tree as our Steiner tree R . Having R at hand, we have a more focused goal: we will zoom into weak linkages that are “pushed onto R ”, and we will only generate such weak linkages to construct our collection. Informally, a weak linkage is *pushed onto R* if all of the edges used by all of its walks are *parallel* to edges of R . We do not demand that the edges belong to R , because then the goal described immediately cannot be achieved—instead, we make $4n + 1$ parallel copies of each edge in the radial completion (the number $4n + 1$ arises from considerations in the “pushing process”), and then impose the above weaker demand. Now, our goal is to show that, if there exists a solution, then there also exists one that can be pushed onto R by applying face operations (in discrete homotopy) so that it becomes *identical* to one of the weak linkages in our collection (see Fig. 4).

At this point, one remark is in place. Our Steiner tree R is a sub-tree of the radial completion of G rather than G itself. Thus, if there exists a solution discretely homotopic to one of the weak linkages that we generate, it might not be a solution in G . We easily circumvent this issue by letting the set X in Lemma 2.1 contain all “fake” edges.

Partitioning a Weak Linkage Into Segments. For the sake of clarity, before we turn to present the next two steps taken to construct R , we begin with the (non-algorithmic) part of the proof where we analyze a (hypothetical) solution towards pushing it onto R . Our most basic notion in this analysis is that of a *segment*, defined as follows (see Fig. 5).

DEFINITION 2.2. *For a walk W in the radial completion of G that is edge-disjoint from R , a segment is a maximal sub-walk of W that does not “cross” R .*

Let $\text{Seg}(W)$ denote the set of segments of W . Clearly, $\text{Seg}(W)$ is a partition of W . Ideally, we would like to upper bound the number of segments of (all the paths of) a solution by $2^{O(k^2)}$. However, this will not be possible because, while R is easily seen to have only

Figure 5: Segments arising from the crossings of a walk with the Steiner tree.

Figure 6: Separators and flows for a long maximal degree-2 path P in R .

$O(k)$ vertices of degree 1 or at least 3, it can have “long” maximal degree-2 paths which can give rise to numerous segments (see Fig. 5). To be more concrete, we say that a maximal degree-2 path of R is *long* if it has more than 2^{ck} vertices (for some constant c), and it is *short* otherwise. Then, as the paths of a solution are vertex disjoint, the following observation is immediate.

OBSERVATION 2.1. *Let $\mathcal{P}$ be a solution. Then, its number of segments that have at least one endpoint on a short path, or a vertex of degree other than 2, of R , is upper bounded by $2^{O(k)}$.*

To deal with segments crossing only long paths, several new ideas are required. In what follows, we first explain how to handle segments going across different long paths, whose number can be bounded (unlike some of the other types of segments we will encounter).

Segments Between Different Long Paths. To deal with such segments, we modify R (in the second step of its construction). For each long path P with endpoints u and v , we will compute two minimum-size vertex sets, S_u and S_v , such that S_u separates (i.e., intersects all paths with one endpoint in each of the two specified subgraphs) the following subgraphs in the radial completion of G : (i) the sub-tree of R that contains u after the removal of a vertex u_1 of P that is “very close” to u , and (ii) the sub-tree of R that contains v after the removal of a vertex u_2 that is “close” to u . The condition satisfied by S_v is symmetric (i.e. u and v switch their roles; see Fig. 6). Here, “very close” refers to distance 2^{c_1k} and “close” refers to distance 2^{c_2k} on the path, for some constants $c_1 < c_2$. Let u' and v' be the vertices of P in the intersection with the separators S_u and S_v respectively. (The selection of u' not to be u itself is of use in the third modification of R .)

Figure 7: Detours in the Steiner tree.

To utilize these separators, we need their sizes to be upper bounded by $2^{O(k)}$. For our initial R , such small separators may not exist. However, the modification we present now will guarantee their existence. Specifically, we will ensure that R does not have any *detour*, which roughly means that each of its maximal degree-2 paths is a shortest path connecting the two sub-trees obtained once it is removed. More formally, we define a detour as follows (see Fig. 7).

DEFINITION 2.3. *A detour in R is a pair of vertices $u, v \in V_{\geq 3}(R) \cup V_{=1}(R)$ (i.e. the non-degree 2 vertices in R) that are endpoints of a maximal degree-2 path L of R , and a path P in the radial completion of G , such that (i) P is shorter than L , (ii) one endpoint of P belongs to the component of $R - V(L) \setminus \{u, v\}$ containing u , and (iii) one endpoint of P belongs to the component of $R - V(L) \setminus \{u, v\}$ containing v .*

By repeatedly “short-cutting” R , a process that terminates in a linear number of steps, we obtain a new Steiner tree R with no detour. We refer to this tree as the backbone Steiner tree.

LEMMA 2.2. *Let (G, S, T, g, k) be an instance of Planar Disjoint Paths. Then, a backbone Steiner tree R can be computed in time $2^{O(k)} n^{3/2} \log^3 n$.*

Now, if the separator S_u is large, then there is a large number of vertex-disjoint paths that connect the two sub-trees separated by S_u , and all of these paths are “long”, namely, of length at least $2^{c_2k} - 2^{c_1k}$. Based on a result by Bodlaender et al. [5] (whose application requires to work in the radial completion of G rather than G itself), we show that the existence of these paths implies that the treewidth of G is large. Thus, if the treewidth of G were small, all of our separators would have also been small. Fortunately, to guarantee this, we just need to invoke the following known result in a preprocessing step:

PROPOSITION 2.1 ([2]). *There is a $2^{O(k)} n^2$ -time algorithm that, given an instance (G, S, T, g, k) of Planar Disjoint Paths, outputs an equivalent instance (G', S, T, g, k) of Planar Disjoint Paths where G' is a subgraph of G whose treewidth is upper bounded by 2^{ck} for some constant c . Furthermore, any sequence of $S \cup T$ -free² concentric cycles in G' has length $2^{O(k)}$.*

Having separators of size $2^{O(k)}$, because segments going across different long paths must intersect these separators (or have an

²A cycle C is $S \cup T$ -free if no-vertex from $S \cup T$ lies in the strict interior of C .

Figure 8: A bad segment that contains all of $S \cup T$ in its cycle.

endpoint at distance $2^{O(k)}$ in R from some endpoint of a maximal degree-2 path), we immediately deduce the following.

OBSERVATION 2.2. *Let $\mathcal{P}$ be a solution. Then, its number of segments that have one endpoint on one long path, and a second endpoint on a different long path, is upper bounded by $2^{O(k)}$.*

Segments with Both Endpoints on the Same Long Path. We are now left with segments whose both endpoints belong to the same long path, which have two different kinds of behavior: they may or may not *spiral* around R , where spiraling means that the two endpoints of the segment belong to different “sides” of the path (see Fig. 5). By making sure that at least one vertex in $S \cup T$ is on the outer face of the radial completion of G , we ensure that the cycle formed by any non-spiraling segment together with the sub-path of R connecting its two endpoints does not enclose all of $S \cup T$; specifically, we avoid having to deal with segments as the one in Fig. 8.

While it is tempting to try to devise face operations that transform a spiraling segment into a non-spiraling one, this is not always possible. In particular, if the spiral “captures” a path P (of a solution), then when P and the spiral are pushed onto R , the spiral is not reduced to a simple path between its endpoints, but to a walk that “flanks” P . Due to such scenarios, dealing with spirals (whose number we are not able to upper bound) requires special attention. Before we turn to this task, let us consider the non-spiraling segments.

Non-Spiraling Segments. To achieve our main goal, we aim to push a (hypothetical) solution onto R so that the only few parallel copies of each edge will be used. Now, we argue that non-spiraling segments do not pose a real issue in this context. To see this, consider a less refined partition of a solution where some non-spiraling segments are “grouped” as follows (see Fig. 5).

DEFINITION 2.4. *A sub-walk of a walk W is a preliminary group of W if either (i) it has endpoints on two different maximal degree-2 paths of R or an endpoint in $V_{=1}(R) \cup V_{\geq 3}(R)$ or it is spiraling, or (ii) it is the union of an inclusion-wise maximal collection of segments not of type (i).*

The collection of preliminary groups of W is denoted by $\text{PreSegGro}(W)$. Clearly, it is a partition of W . For a weak linkage $\mathcal{W}$, $\text{PreSegGro}(\mathcal{W}) = \bigcup_{W \in \mathcal{W}} \text{PreSegGro}(W)$. Then,

OBSERVATION 2.3. *Let $\mathcal{W}$ be a weak linkage. The number of type-(ii) preliminary groups in $\text{PreSegGro}(\mathcal{W})$ is at most 1 plus the number of type-(i) preliminary groups in $\text{PreSegGro}(\mathcal{W})$.*

Roughly speaking, a type-(i) preliminary group is easily pushed onto R so that it becomes merely a simple path (see Fig. 5). Thus, by Observation 2.3, all type-(ii) preliminary groups of a solution in total do not give rise to the occupation of more than $x + 1$ copies of an edge, where x is the number of type-(i) preliminary groups.

Rollback Spirals and Winding Number. Unfortunately, the number of spirals can be huge. Nevertheless, we can pair-up *some* of them so that they will “cancel” each other when pushed onto R (see Fig. 9), thereby behaving like a type-(ii) preliminary group. Intuitively, we pair-up two spirals of a walk if one of them goes from the left-side to the right-side of the path, the other goes from the right-side to the left-side of the same path, and “in between” them on the walk, there are only type-(ii) preliminary groups and spirals that have already been paired-up. We refer to paired-up spirals as *rollback spirals*. (Not all spirals can be paired-up in this manner.) This gives rise to the following strengthening of Definition 2.4.

DEFINITION 2.5. *A sub-walk of a walk W is called a group of W if either (i) it is a non-spiral type-(i) preliminary group, or (ii) it is the union of an inclusion-wise maximal collection of segments not of type (i) (i.e., all endpoints of the segments in the group are internal vertices of the same maximal degree-2 path of R). The potential of a group is (roughly) 1 plus its number of non-rollback spirals.*

Now, rather than upper bounding the total number of spirals, we only need to upper bound the number of non-rollback spirals. To this end, we use the notion of *winding number*, informally defined as follows. Consider a solution $\mathcal{P}$, a path $Q \in \mathcal{P}$, and a long path P of R with separators S_u and S_v . As S_u and S_v are minimal separators in a triangulated graph (the radial completion is triangulated), they are cycles, and as at least one vertex in T belongs to the outer face, they form a ring (see Fig. 10). Each maximal sub-path of Q that lies inside this ring can either *visit* the ring, which means that both its endpoints belong to the same separator, or *cross* the ring, which means that its endpoints belong one to S_u and the other to S_v (see Fig. 10). Then, the (absolute value of the) *winding number* of a crossing sub-path is the number of times it “winds around” P inside the ring (see Fig. 10). At least intuitively, it should be clear that winding numbers and non-rollback spirals are related. In particular, each ring can only have $2^{O(k)}$ visitors and crossings sub-paths (because the size of each separator is $2^{O(k)}$), and we only have $O(k)$ rings to deal with. Thus, it is possible to show that if the winding number of every crossing sub-path is upper bounded by $2^{O(k)}$, then the total number of non-rollback spirals is upper bounded by $2^{O(k)}$ as well. The main tool we employ to bound the winding number of every crossing path is the following known result (rephrased to simplify the overview).

PROPOSITION 2.2 ([12]). *Let G be a graph embedded in a ring with a crossing path P . Let $\mathcal{P}$ and $\mathcal{Q}$ be two collections of vertex-disjoint crossings paths of the same size. (A path in $\mathcal{P}$ can intersect a path in $\mathcal{Q}$, but not another path in $\mathcal{P}$.) Then, G has a collection of crossing paths $\mathcal{P}'$ such that (i) for every path in $\mathcal{P}$, there is a path in $\mathcal{P}'$ with the same endpoints and vice versa, and (ii) the maximum difference*

Figure 9: Rollback spirals.

between (the absolute value of) the winding numbers with respect to P of any path in $\mathcal{P}'$ and any path in Q is at most 6.

To see the utility of Proposition 2.2, suppose momentarily that none of our rings has visitors. Then, if we could ensure that for each of our rings, there is a collection Q of vertex-disjoint paths of *maximum size* such that the winding number of each path in Q is a constant, Proposition 2.2 would have the following implication: if there is a solution, then we can modify it within each ring to obtain another solution such that each crossing sub-path of each of its paths will have a constant winding number (under the supposition that the rings are disjoint, which we will deal with later in the overview), see Fig. 11. Our situation is more complicated due to the existence of visitors—we need to ensure that the replacement $\mathcal{P}'$ does not intersect them. On a high-level, this situation is dealt with by first showing how to ensure that visitors do not “go too deep” into the ring on either side of it. Then, we consider an “inner ring” where visitors do not exist, on which we can apply Proposition 2.2. Afterwards, we are able to bound the winding number of each crossing path by $2^{O(k)}$ (but not by a constant) in the (normal) ring.

Modifying R within Rings. To ensure the existence of the aforementioned collection Q for each ring, we need to modify R . To this end, consider a long path P with separators S_u and S_v , and let P' be the sub-path of P inside the ring defined by the two separators. We compute a maximum-sized collection of vertex-disjoint paths $\text{Flow}(u, v)$ such that each of them has one endpoint in S_u and the other in S_v .³ Then, we prove a result that roughly states the following.

LEMMA 2.3. *There is a path P^* in the ring defined by S_u and S_v with the same endpoints as P' crossing each path in $\text{Flow}(u, v)$ at most once. Moreover, P^* is computable in linear time.*

Having P^* at hand, we replace P' by P^* . This is done for every maximal degree-2 path, and thus we complete the construction of R . However, at this point, it is not clear why after we perform these replacements, the separators considered earlier remain separators,

³This flow has an additional property: there is a tight collection of $C(u, v)$ of concentric cycles separating S_u and S_v such that paths in $\text{Flow}(u, v)$ do not “oscillate” too much between any two cycles in the collection. Such a maximum flow is said to be *minimal* with respect to $C(u, v)$.

Figure 10: A solution winding in a ring.

Figure 11: The “unwinding” of the solution.

or that we even still have a tree. Roughly speaking, a scenario as depicted in Fig. 12 can potentially happen. To show that this is not the case, it suffices to prove that there cannot exist a vertex that belongs to two different rings. Towards that, we apply another preprocessing operation: we ensure that the radial completion of G does not have 2^{c^k} (for some constant c) *concentric cycles* that contain no vertex in $S \cup T$ by using another result by Adler et al. [2]. Informally, a sequence of concentric cycles is a sequence of vertex-disjoint cycles where each one of them is contained inside the next one in the sequence. Having no such sequences, we prove the following.

Figure 12: The green and blue rings intersect, which can create cycles in R when replacing paths.

LEMMA 2.4. *Let R' be any Steiner tree. For every vertex v , there exists a vertex in $V(R')$ whose distance to v (in the radial completion of G) is 2^{c_k} for some constant c .*

To see why intuitively this lemma is correct, note that if v was “far” from R' in the radial completion of G , then in G itself v is surrounded by a large sequence of concentric cycles that contain no vertex in $S \cup T$. Having Lemma 2.4 at hand, we show that if a vertex belongs to a certain ring, then it is “close” to at least one vertex of the restriction of R to that ring. In turn, that means that if a vertex belongs to two rings, it can be used to exhibit a “short” path between one vertex in the restriction of R to one ring and another vertex in the restriction of R to the second ring. By choosing constants properly, this path is shown to exhibit a detour in R , and hence we reach a contradiction. (In this argument, we use the fact that for every vertex u , towards the computation of the separator, we considered a vertex u' of distance $2^{c_1 k}$ from u —this sub-path between u and u' is precisely that sub-path that we will shortcut.)

Pushing a Solution Onto R . So far, we have argued that if there is a solution, then there is also one such that the sum of the potential of all of the groups of all of its paths is at most $2^{O(k)}$. Additionally, we discussed the intuition why this, in turn, implies the following result.

LEMMA 2.5. *If there is a solution $\mathcal{P}$, then there is a weak linkage pushed onto R that is discretely homotopic to $\mathcal{P}$ and uses at most $2^{O(k)}$ copies of every edge.*

The formal proof of Lemma 2.5, is quite technical. On a high level, it consists of three phases. First, we push onto R all sequences of the solution—that is, maximal sub-paths that touch (but not necessarily cross) R only at their endpoints. Second, we eliminate some U-turns of the resulting weak linkage (see Fig. 13), as well as “move through” R segments with both endpoints being internal vertices of the same maximal degree-2 path of R and crossing it in opposing directions (called *swollen segments*). At this point, we are able to bound by $2^{O(k)}$ the number of segments of the pushed weak linkage. Third, we eliminate all of the remaining U-turns, and show that then, the number of copies of each edge used must be at most $2^{O(k)}$. We also modify the pushed weak linkage to be of a certain “canonical form”.

Figure 13: A walk going back and forth along a path of R , which gives rise to U-turns.

Generating a Collection of Pushed Weak Linkages. In light of Lemma 2.5 and Proposition 2.1, it only remains to generate a collection of $2^{O(k^2)}$ pushed weak linkages that includes all pushed weak linkages (of some canonical form) using at most $2^{O(k)}$ copies of each edge. (This part, along with the preprocessing and construction of R , are the algorithmic parts of our proof.)

This part of our proof is essentially a technical modification and adaptation of the work of Schrijver [38] (though we need to be more careful to obtain the bound $2^{O(k^2)}$). Thus, we only give a brief description of it in the overview. Essentially, we generate pairs of a *pairing* and a *template*: a pairing assigns, to each vertex v of R of degree 1 or at least 3, a set of pairs of edges incident to v to indicate that copies of these edges are to be visited consecutively (by at least one walk of the weak linkage under construction); a template further specifies, for each of the aforementioned pairs of edges, how many times copies of these edges are to be visited consecutively (but not which copies are paired-up). Clearly, there is a natural association of a pairing and a template to a pushed weak linkage. Further, we show that to generate all pairs of pairings and templates associated with the weak linkages we are interested in, we only need to consider pairings that in total have $O(k)$ pairs and templates that assign numbers bounded by $2^{O(k)}$ (because we deal with weak linkages using $2^{O(k)}$ copies of each edge):

LEMMA 2.6. *There is a collection $\mathcal{W}_{\text{ALL}}$ of $2^{O(k^2)}$ pairs of pairings and templates that, for any canonical pushed weak linkage $\mathcal{W}$ using only $2^{O(k)}$ copies of each edge, contains a pair (of a pairing and a template) “compatible” with $\mathcal{W}$. Further, such a collection $\mathcal{W}_{\text{ALL}}$ is computable in time $2^{O(k^2)}$.*

Using somewhat more involved arguments, we also prove the following.

LEMMA 2.7. *Any canonical pushed weak linkage is “compatible” with exactly one pair of a pairing and a template. Moreover, given a pair of a pairing and a template, if a canonical pushed weak linkage compatible with it exists, then it can be found in time polynomial in its size.*

These two lemmas complete the proof: we can indeed generate a collection of $2^{O(k^2)}$ pushed weak linkages containing all canonical pushed weak linkages using only $2^{O(k)}$ copies of any edge.

LEMMA 2.8. *Let (G, S, T, g, k) be an Yes-instance of Planar Disjoint Paths, and let R be a backbone Steiner tree. Then, there exists a collection of $2^{O(k^2)}$ weak linkages $\mathcal{W}_{\text{ALL}}$ such that there is a weak linkage $\mathcal{W} \in \mathcal{W}_{\text{ALL}}$ that is discretely homotopic to some solution of (G, S, T, g, k) . Further, collection can be enumerated in $2^{O(k^2)}n$ time.*

3 THE ALGORITHM

Having set up all required definitions and notions, we are ready to describe our algorithm that proves Theorem 1.1. Afterwards, we will analyze its running time and prove its correctness.

3.1 Execution of the Algorithm

We refer to this algorithm as PDPAlg. It takes as input an instance $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ of Planar Disjoint Paths, and its output is the decision whether this instance is a Yes-instance. The specification of the algorithm is as follows.

Step I: Preprocessing. First, PDPAlg invokes Proposition 2.1 to transform $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ into an equivalent instance (G, S, T, g, k) of Planar Disjoint Paths of bounded treewidth where $|V(G)| = O(|V(\tilde{G})|)$.

Step II: Computing a Backbone Steiner Tree. Second, PDPAlg invokes Lemma 2.2 with respect to (G, S, T, g, k) to compute a backbone Steiner tree, denoted by R .

Step III: Looping on $\mathcal{W}_{\text{ALL}}$. Now, PDPAlg invokes Lemma 2.8 to enumerate $\mathcal{W}_{\text{ALL}}$. For each weak linkage $\mathcal{W} \in \mathcal{W}_{\text{ALL}}$, the algorithm applies the algorithm of Lemma 2.1 to (G, S, T, g, k) and $\mathcal{W}$; if the algorithm finds a solution, then PDPAlg determines that $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ is Yes-instance and terminates.

Step IV: Reject. If the previous step fails, PDPAlg determines that $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ is a No-instance and terminates.

3.2 Running Time and Correctness

Let us first analyze the running time of PDPAlg.

LEMMA 3.1. *PDPAlg runs in time $2^{O(k^2)} n^{O(1)}$.*

PROOF. By Proposition 2.1, the computation of (G, S, T, g, k) in Step I is performed in time $2^{O(k)} n^2$. By Lemma 2.2, the computation of R in Step II is performed in time $2^{O(k)} n^{3/2} \log^3 n$. By Lemma 2.8, $|\mathcal{W}_{\text{ALL}}| = 2^{O(k^2)}$, and it can be enumerated in $2^{O(k^2)} n$ time. Finally, for each $\mathcal{W} \in \mathcal{W}_{\text{ALL}}$, Lemma 2.1 takes $n^{O(1)}$ time to test if there is a solution that is discretely homotopic to $\mathcal{W}$. Thus, PDPAlg runs in time $2^{O(k^2)} n^{O(1)}$. $\square$

The reverse direction of the correctness of PDPAlg is trivially true.

LEMMA 3.2. *Let $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ be an instance of Planar Disjoint Paths. If $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ is accepted by PDPAlg, then $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ is a Yes-instance.*

Now, we handle the forward direction of the correctness of PDPAlg.

LEMMA 3.3. *Let $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ be an instance of Planar Disjoint Paths. If $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ is a Yes-instance, then $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ is accepted by PDPAlg.*

PROOF. Suppose that $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$ is a Yes-instance of Planar Disjoint Paths. Then, by Lemma 2.1, (G, S, T, g, k) is a Yes-instance of Planar Disjoint Paths. Then by Lemma 2.8, there is a collection $\mathcal{W}_{\text{ALL}}$ of $2^{O(k^2)}$ weak linkages containing at least one weak linkage $\mathcal{W}^*$, that is discretely homotopic to some solution of (G, S, T, g, k) .

Then in Step III, Lemma 2.1 ensures that we obtain a solution to (G, S, T, g, k) in the iteration we consider $\mathcal{W}^*$. Hence PDPAlg accepts the instance $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$. $\square$

This proves Theorem 1.1. Lastly, we remark that the algorithm PDPAlg can be easily modified not only to reject or accept the instance $(\tilde{G}, \tilde{S}, \tilde{T}, \tilde{g}, k)$, but to return a solution in case of acceptance, within time $2^{O(k^2)} n^{O(1)}$.

REFERENCES

- [1] Isolde Adler. 2013. *6th Workshop on Graph Classes, Optimization, and Width Parameters 2013, List of Open Problems*.
- [2] Isolde Adler, Stavros G. Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. 2017. Irrelevant vertices for the planar Disjoint Paths Problem. *J. Comb. Theory, Ser. B* 122 (2017), 815–843. <https://doi.org/10.1016/j.jctb.2016.10.001>
- [3] Isolde Adler and Philipp Klaus Krause. 2010. A lower bound for the tree-width of planar graphs with vital linkages. *CoRR* abs/1011.2136 (2010). arXiv:1011.2136 <http://arxiv.org/abs/1011.2136>
- [4] Julien Baste and Ignasi Sau. 2015. The role of planarity in connectivity problems parameterized by treewidth. *Theor. Comput. Sci.* 570 (2015), 1–14. <https://doi.org/10.1016/j.tcs.2014.12.010>
- [5] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelco Penninx, Saket Saurabh, and Dimitrios M. Thilikos. 2016. (Meta) Kernelization. *J. ACM* 63, 5 (2016), 44:1–44:69. <https://doi.org/10.1145/2973749>
- [6] Chandra Chekuri and Julia Chuzhoy. 2016. Polynomial Bounds for the Grid-Minor Theorem. *J. ACM* 63, 5 (2016), 40:1–40:65. <https://doi.org/10.1145/2820609>
- [7] Julia Chuzhoy and David H. K. Kim. 2015. On Approximating Node-Disjoint Paths in Grids. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24–26, 2015, Princeton, NJ, USA (LIPIcs)*, Vol. 40. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 187–211.
- [8] Julia Chuzhoy, David H. K. Kim, and Shi Li. 2016. Improved approximation for node-disjoint paths in planar graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18–21, 2016*. ACM, 556–569.
- [9] Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. 2017. New hardness results for routing on disjoint paths. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19–23, 2017*. ACM, 86–99.
- [10] Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. 2018. Almost polynomial hardness of node-disjoint paths in grids. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25–29, 2018*. ACM, 1220–1233.
- [11] Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. 2018. Improved Approximation for Node-Disjoint Paths in Grids with Sources on the Boundary. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9–13, 2018, Prague, Czech Republic (LIPIcs)*, Vol. 107. 38:1–38:14.
- [12] Marek Cygan, Daniel Marx, Marcin Pilipczuk, and Michał Pilipczuk. 2013. The Planar Directed K-Vertex-Disjoint Paths Problem Is Fixed-Parameter Tractable. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26–29 October, 2013, Berkeley, CA, USA*. 197–206. <https://doi.org/10.1109/FOCS.2013.29>
- [13] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. 2005. Subexponential parameterized algorithms on bounded-genus graphs and H -minor-free graphs. *J. ACM* 52, 6 (2005), 866–893. <https://doi.org/10.1145/1101821.1101823>
- [14] Erik D. Demaine and Mohammad Taghi Hajiaghayi. 2008. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica* 28, 1 (2008), 19–36. <https://doi.org/10.1007/s00493-008-2140-4>
- [15] Guoli Ding, Alexander Schrijver, and Paul D. Seymour. 1992. Disjoint paths in a planar graph—a general theorem. *SIAM Journal on Discrete Mathematics* 5, 1 (1992), 112–116.
- [16] Fedor V. Fomin, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2016. Subexponential Parameterized Algorithms for Planar and Apex-Minor-Free Graphs via Low Treewidth Pattern Covering. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, New Brunswick, New Jersey, USA. IEEE Computer Society*, 515–524.
- [17] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. 2018. Excluded Grid Minors and Efficient Polynomial-Time Approximation Schemes. *J. ACM* 65, 2 (2018), 10:1–10:44.
- [18] Steven Fortune, John E. Hopcroft, and James Wyllie. 1980. The Directed Subgraph Homeomorphism Problem. *Theor. Comput. Sci.* 10 (1980), 111–121. [https://doi.org/10.1016/0304-3975\(80\)90009-2](https://doi.org/10.1016/0304-3975(80)90009-2)

- [19] András Frank. 1990. Packing paths, cuts, and circuits—a survey. *Paths, Flows and VLSI-Layout* (1990), 49–100.
- [20] Martin Grohe, Ken-ichi Kawarabayashi, and Bruce A. Reed. 2013. A Simple Algorithm for the Graph Minor Decomposition - Logic meets Structural Graph Theory. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*. SIAM, 414–431. <https://doi.org/10.1137/1.9781611973105.30>
- [21] Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*. 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9
- [22] Richard M Karp. 1975. On the computational complexity of combinatorial problems. *Networks* 5, 1 (1975), 45–68.
- [23] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. 2012. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B* 102, 2 (2012), 424–435. <https://doi.org/10.1016/j.jctb.2011.07.004>
- [24] Ken-ichi Kawarabayashi and Paul Wollan. 2010. A shorter proof of the graph minor algorithm: the unique linkage theorem. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. ACM, 687–694. <https://doi.org/10.1145/1806689.1806784>
- [25] Ken-ichi Kawarabayashi and Paul Wollan. 2011. A simpler algorithm and shorter proof for the graph minor decomposition. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*. ACM, 451–458. <https://doi.org/10.1145/1993636.1993697>
- [26] Philip N. Klein and Dániel Marx. 2014. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. SIAM, 1812–1830.
- [27] Mark R Kramer and J van Leeuwen. 1984. The complexity of wirerouting and finding minimum area layouts for arbitrary VLSI circuits. *Advances in computing research* 2 (1984), 129–146.
- [28] Richard J. Lipton and Kenneth W. Regan. 2013. *David Johnson: Galactic Algorithms*. Springer Berlin Heidelberg, Berlin, Heidelberg, 109–112. https://doi.org/10.1007/978-3-642-41422-0_20
- [29] Daniel Lokshantov, Dániel Marx, and Saket Saurabh. 2018. Slightly Super-exponential Parameterized Problems. *SIAM J. Comput.* 47, 3 (2018), 675–702. <https://doi.org/10.1137/16M1104834>
- [30] James F Lynch. 1975. The equivalence of theorem proving and the interconnection problem. *ACM SIGDA Newsletter* 5, 3 (1975), 31–36.
- [31] Yehoshua Perl and Yossi Shiloach. 1978. Finding Two Disjoint Paths Between Two Pairs of Vertices in a Graph. *J. ACM* 25, 1 (1978), 1–9. <https://doi.org/10.1145/322047.322048>
- [32] Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. 2018. Network Sparsification for Steiner Problems on Planar and Bounded-Genus Graphs. *ACM Trans. Algorithms* 14, 4 (2018), 53:1–53:73.
- [33] Neil Robertson and Paul D. Seymour. 1995. Graph Minors .XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B* 63, 1 (1995), 65–110. <https://doi.org/10.1006/jctb.1995.1006>
- [34] Neil Robertson and Paul D. Seymour. 2003. Graph Minors. XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B* 89, 1 (2003), 43–76. [https://doi.org/10.1016/S0095-8956\(03\)00042-X](https://doi.org/10.1016/S0095-8956(03)00042-X)
- [35] Neil Robertson and Paul D. Seymour. 2012. Graph Minors. XXII. Irrelevant vertices in linkage problems. *J. Comb. Theory, Ser. B* 102, 2 (2012), 530–563. <https://doi.org/10.1016/j.jctb.2007.12.007>
- [36] Neil Robertson, Paul D. Seymour, and Robin Thomas. 1994. Quickly Excluding a Planar Graph. *J. Comb. Theory, Ser. B* 62, 2 (1994), 323–348. <https://doi.org/10.1006/jctb.1994.1073>
- [37] Petra Scheffler. 1994. *A practical linear time algorithm for disjoint paths in graphs with bounded tree-width*. TU, Fachbereich 3.
- [38] Alexander Schrijver. 1994. Finding k Disjoint Paths in a Directed Planar Graph. *SIAM J. Comput.* 23, 4 (1994), 780–788. <https://doi.org/10.1137/S0097539792224061>
- [39] Alexander Schrijver. 2003. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer Science & Business Media.