(Almost-)Optimal FPT Algorithm and Kernel for T-Cycle on Planar Graphs

Harmender Gahlawat[*][†]
harmendergahlawat@gmail.com

Abhishek Rathod[‡]
arathod@post.bgu.ac.il

Meirav Zehavi[‡]
meiravze@bgu.ac.il

September 14, 2025

Abstract

Research of cycles through specific vertices is a central topic in graph theory. In this context, we focus on a well-studied computational problem, T-Cycle: given an undirected n-vertex graph G and a set of k vertices $T \subseteq V(G)$ termed *terminals*, the objective is to determine whether G contains a simple cycle C through all the terminals. Our contribution is twofold: (i) We provide a $2^{O(\sqrt{k} \log k)} \cdot n$-time fixed-parameter deterministic algorithm for T-Cycle on planar graphs; (ii) We provide a $k^{O(1)} \cdot n$-time deterministic kernelization algorithm for T-Cycle on planar graphs where the produced instance is of size $k \log^{O(1)} k$.

Both of our algorithms are optimal in terms of both k and n up to (poly)logarithmic factors in k under the ETH. In fact, our algorithms are the first subexponential-time fixed-parameter algorithm for T-Cycle on planar graphs, as well as the first polynomial kernel for T-Cycle on planar graphs. This substantially improves upon/expands the known literature on the parameterized complexity of the problem.

1 Introduction

Combinatorial properties of (simple) cycles passing through a specific set of vertices (along with, possibly, other vertices) in a graph—and, in particular, combinatorial properties of the graphs containing them—have been intensively explored for over six decades. As observed by Björklund et al. [7], the public interest in this topic can even be dated back to 1898, when Lewis Carroll challenged the readers of Vanity Fair with a riddle linked to this problem. However, the flurry of scientific results on this topic has, roughly, begun with Dirac's result [16] from 1960, stating that given k vertices in a k-connected (undirected) graph G, G has a cycle through all of them. For a few illustrative examples of works in Graph Theory that followed up on this result, we refer to [10,18,23–26,40,41,45]. Indeed, as stated by Kawarabayashi [27]: "Since 1960's, cycles through a vertex set or an edge set are one of central topics in all of graph theory."

Computationally, given an undirected n-vertex graph G and a set of k vertices $T \subseteq V(G)$ referred to as *terminals* (for any $k \in \{1, 2, \ldots, n\}$), the objective of the T-Cycle problem is to determine whether G contains a (simple) cycle C that passes through all (but not necessarily only) the terminals. Henceforth, such a cycle is referred to as a T-*loop*. Due to the immediate relation between the T-Cycle and Disjoint Paths problems (formalized later in this introduction), the T-Cycle problem has been long known (since the seminal work of Robertson and Seymour [39]) to be *fixed-parameter tractable*, that is, solvable in time $f(k) \cdot n^c$ for some computable function f of k and some fixed constant c. However, here, the best known f is galactic, satisfying $f(k) \geq 2^{2^{2^{2^{2^k}}}}$, and $c = 2$ [28]. Still, already in 1980, LaPaugh and Rivest [33] provided a linear time algorithm for T-Cycle when $k = 3$, which involved no large hidden constants.

---

[*]LIMOS, Université Clermont Auvergne, Clermont-Ferrand, France
[†]Laboratoire G-SCOP, Grenoble-INP, France
[‡]Ben-Gurion University of the Negev, Beersheba, Israel

The first breakthrough for the general case—having k as part of the input—was established in 2008 by Kawarabayashi [27], who proved that:[1]

- The T-CYCLE problem is solvable in time $2^{2^{O(k^{10})}} \cdot n^2$. We remark that using the developments in [12] on the Grid-Minor Theorem that were first published shortly after the work of Kawarabayashi, it can be easily seen that his algorithm can be made to run in time $2^{O(k^c)} \cdot n^2$ for some large constant c.

- Restricted to planar graphs, the T- CYCLE problem is solvable in time $2^{O(k \log^4 k)} \cdot n^2$.

We note that all algorithms mentioned so far are deterministic. In 2010, by making novel use of the idea behind the breakthrough technique of algebraic fingerprints [5, 6], Björklund et al. [7] developed a randomized algorithm for T-CYCLE that runs in time $O(2^k \cdot n)$, which, prior to this manuscript, has remained the fastest known algorithm for this problem even if restricted to planar graphs. We note that however, prior to this manuscript, both of Kawarabayashi's algorithms have remained the best-known deterministic ones for the problem even if restricted to planar graphs.

With the above context in mind, the first of the two main contributions of our manuscript can be summarized in the following theorem. Specifically, we present the first subexponential-time fixed-parameter algorithm for the T-CYCLE problem on planar graphs.

Theorem 1. *Restricted to planar graphs, the T-CYCLE problem is solvable in time $2^{O(\sqrt{k} \log k)} \cdot n$.*

Notably, under the Exponential Time Hypothesis (ETH), the time complexity in Theorem 1 is tight up to the logarithmic factor in the exponent, and, on general graphs, no sub-exponential-time (fixed-parameter or not) algorithm exists for T-CYCLE. This can be seen from the fact that HAMILTONICITY is the special case of T-CYCLE when T is the entire vertex set of the graph, and HAMILTONICITY is not solvable in times $2^{o(\sqrt{n})}$ and $2^{o(n)}$ on planar and general graphs, respectively, under the ETH [14].

The existence of a subexponential-time fixed-parameter algorithm for the T-CYCLE problem on planar graphs might seem surprising when considered in the context of two other well-known "terminals-based" problems on planar graphs: Specifically, under the ETH, the STEINER TREE problem on planar graphs is not solvable in time $2^{o(k)} \cdot n^{O(1)}$ [36], and the DISJOINT PATHS problem was only recently shown to be solvable in time $2^{k^{O(1)}} \cdot n$ [13], where, in both cases, k denotes the number of terminals.

Following-up on the work of Björklund et al. [7], Wahlström [43] studied the compressability of T-CYCLE. To discuss Wahlström's contribution, let us first present the definitions of compression and kernelization. Formally, we say that a (decision) parameterized problem Π admits a *compression* into a (decision, not necessarily parameterized) problem Π' if there exists a polynomial-time algorithm (called a *compression algorithm*) that, given an instance (I, k) of Π, produces an equivalent instance J of Π' such that $|J| \leq f(k)$ for some computable function f of k. When f is polynomial, then the compression is said to be *polynomial-sized* (or, for short, *polynomial*), and when $\Pi' = \Pi$, then the compression is termed *kernelization*. Kernelization is, perhaps, the most well-studied research subarea of Parameterized Complexity after that of fixed-parameter tractability [20]. In fact, kernelization was termed "the lost continent of polynomial time" [19] by one of the two fathers of the field of parameterized complexity. Indeed, kernelization is, essentially, the only mathematical framework to rigorously reason about the effectiveness of preprocessing procedures, which are ubiquitous in computer science in general. Here, the focus is on polynomial-sized kernels (and compressions), since the existence of a (not necessarily polynomial) kernel for a problem is equivalent to that problem being fixed-parameter tractable [11], while, on the other hand, there exist numerous problems known to be fixed-parameter tractable but which do not admit a polynomial kernel unless the polynomial hierarchy collapses [20] (with the first examples having been discovered already in 2008 [8]).

Wahlström [43] proved that T-CYCLE admits a compression of size $O(k^3)$. The target problem is a somewhat artificial algebraic problem, which, in particular, is not known to be in NP, and hence this compression does not imply the existence of a polynomial-sized kernelization. Over the years, Wahlström's result on the compressability of T-CYCLE has become an extremely intriguing finding in the field of Parameterized Complexity: Since its discovery and until this day, T-CYCLE remains the *only* natural problem known to admit a polynomial-sized compression but not known to admit

---

[1]Kawarabayashi [27] states that when k is a fixed constant, the dependency on n in the runtimes of his algorithms can be made linear, but no details of a proof to support this statement are given.

a polynomial-sized kernelization! In fact, the resolution of the kernelization complexity of T-Cycle (on general graphs) is one of the biggest problems in the field. We remark that the ideas behind this compression also yield an alternative $O(2^k \cdot n)$-time algorithm for T-Cycle (in [43]), which is, similarly to Björklund et al. [7]'s algorithm, also randomized and based on algebraic arguments.

With the above context in mind, the second of the two main contributions of our manuscript can be summarized in the following theorem.

Theorem 2. *Restricted to planar graphs, the T-Cycle problem admits a $k^{O(1)} \cdot n$-time kernelization algorithm of size $k \cdot \log^{O(1)} k$.*

Again, the bounds in the theorem are almost tight in terms of both time complexity and, more importantly, the size of the reduced instance. First, we cannot expect a time complexity of $o(n)$ as we need to, at least, read the input. (We note that the $k^{O(1)}$ factor is essentially negligible because the time complexity of an exact algorithm to solve this problem afterwards will, anyway, have at least a $2^{\Omega(\sqrt{k})}$ factor under the ETH as argued earlier). Second, we cannot expect the reduced instance to be of size $o(k)$ (or even of size $k-1$), else we can repeatedly reapply the kernelization algorithm until it will yield a constant-sized instance that is solvable in constant time, thereby solving the (NP-hard) problem in polynomial time.

As a side note, we remark that our proof might shed light on the kernelization complexity of T-Cycle on general graphs as well: We believe that the scheme that we employ—on a high-level, the usage of a polynomial-time treewidth reduction to a polynomial function of k coupled with a polynomial kernel for the combined parameter k plus treewidth, and on a lower level, the specific arguments used in our proofs to implement both procedures—*might* be liftable to general graphs.

Combining Theorems 1 and 2, we conclude the following corollary, which is, up to a logarithmic factor in the exponent, essentially the best one can hope for.

Corollary 3. *Restricted to planar graphs, the T-Cycle problem is solvable in time $2^{O(\sqrt{k} \log k)} + k^{O(1)} \cdot n$.*

1.1 Related Problems

We first note that when the T-Cycle problem is extended to directed graphs, it becomes NP-hard already when $k = 2$ [21]. Various other extensions/generalizations of T-Cycle were studied in the literature, including from the perspective of parameterized complexity. For example, Kawarabayashi et al. [29] developed an algorithm for detecting a T-loop whose length has a given parity; for fixed k, the running time is polynomial in n, but the dependency on k is unspecified. For another example, Kobayashi and Kawarabayashi [30] developed an algorithm for detecting an induced T-loop in a planar graph in time $2^{O(k^{3/2} \log k)} n^2$. The survey of additional results of this nature is beyond the scope of this paper.

Highly relevant to the T-Cycle problem is the Disjoint Paths problem. Here, given a graph G and a set $T = \{(s_i, t_i) : i \in \{1, 2, \ldots, k\}\}$ of pairs of vertices (termed terminals) in G, the objective is to determine whether G contains k vertex-disjoint[2] paths $P_1, P_2, \ldots, P_k$ so that the endpoints of P_i are s_i and t_i. The T-Cycle problem can be reduced to the Disjoint Paths problem with an overhead of $k! = k^{O(k)}$: Given an instance (G, T) of T-Cycle, for every ordering $v_0, v_1, \ldots, v_{k-1}$ of T, create an instance (G, T') of Disjoint Paths where $T' = \{(v_i, v_{(i+1) \mod k}) : i \in \{0, 1, \ldots, k-1\}\}$; then, (G, T) is a yes-instance of T-Cycle if and only if at least one of the constructed instances of Disjoint Paths is. Being foundational to the entire graph minor theory [34], the Disjoint Paths problem is of great importance to both algorithm design and graph theory, and has been intensively studied for several decades. Most relevant to this manuscript is to mention that, currently, the fastest algorithms for Disjoint Paths run in $f(k) \cdot n^2$ time for $f(k) > 2^{2^{2^{2^{2^k}}}}$ on general graphs [28], and in $2^{k^{O(1)}} \cdot n$ time on planar graphs [13]. It is worth mentioning that Korhonen, Pilipczuk, and Stamoulis [32] recently provided an FPT algorithm for Disjoint Paths with almost-linear running time dependence on the number of vertices and edges, i.e., with running time $f(k) \cdot (m+n)^{1+o(1)}$. Also, the Disjoint Paths problem does not admit a polynomial kernel w.r.t. k even on planar graphs [44], but it admits a polynomial kernel w.r.t. k plus the treewidth of the input graph on planar graphs [44] (on general graphs, it is unknown whether Disjoint Paths admits a polynomial kernel w.r.t. k plus treewidth).

---

[2] With the exception that a vertex can occur in multiple paths if it is an endpoint in all of them.

Also quite relevant to the T-CYCLE problem is the k-CYCLE (or k-PATH) problem, which is also, perhaps, the second or third most well-studied problem in Parameterized Complexity (see, e.g., the dozens of papers cited in [35]). Here, given a graph G and a nonnegative integer k, the objective is to determine whether G contains a simple cycle (or path) on k vertices. The techniques used to solve k-CYCLE can be trivially lifted to solve T-CYCLE when parameterized by the *sought size of a solution* ℓ (given as an extra parameter in the input), which can be arbitrarily larger than T (e.g., in the extreme case where the entire graph is a single cycle, $\ell = n$ while $k = 1$). In particular, this means that, on general graphs, T-CYCLE is solvable in time $O(2^k 1.66^{\ell-k} \cdot (n+m))$ by a randomized algorithm [6] (or time $O(2.56^k 1.66^{\ell-k} \cdot (n+m))$ by a deterministic algorithm [42]), and on planar graphs, T-CYCLE is solvable in time $2^{O(\sqrt{\ell})} \cdot (n+m)$ by a deterministic algorithm (e.g., via [17]).

1.2 Techniques

We present a high-level overview of our proofs in the next section. Here, we only briefly highlight (in a very informal manner) the novelties in our proofs, and put them in the context of the known literature. Essentially, the following theorem reveals some of the core combinatorial arguments that underlie our algorithmic results. Thus, we believe that this result may be of independent interest.

Theorem 4. *There exists a $k^{O(1)} \cdot n$-time algorithm that, given an instance (G, T) of T-CYCLE on planar graphs, outputs an equivalent instance (G', T) of T-CYCLE on planar graphs along with a set of vertices U such that:*

1. *G' is a subgraph of G whose treewidth is bounded by $O(\sqrt{k} \log k)$, and*

2. *$|U| \in O(k \log k)$ and the treewidth of $G' - U$ is bounded by $O(\log k)$.*

Indeed, from here, Theorem 1 directly follows by using a $2^{O(tw)} \cdot n$-time algorithm for T-CYCLE parameterized by the treewidth tw of the planar graph G' as a black box (e.g., via [17]), while Theorem 2 requires additional non-trivial work, described later in this section.

Our proof of Theorem 4 consists of three main ingredients. Towards the first ingredient, we consider a sequence of concentric cycles, and classify the "segments" of an (unknown) solution T-loop that go inside this sequence into types, similarly to how it is done in [4] for the DISJOINT PATHS problem. Here, a cheap solution is one that uses as few edges as possible that do not belong to the sequence of concentric cycles. Then, we prove the following result, which is the first ingredient.

Lemma 5 (Informal Statement of Lemma 19). *For a "cheap" solution T-loop and a sequence of concentric cycles that does not contain any terminal, there can be at most constantly many segments of the same type.*

We note that in [4], it is proved that the number of segments of the same type for a DISJOINT PATHS solution is $2^{O(k)}$. Our proof of Lemma 5 is based on an elegant (and novel) re-routing argument (see Section 2), which has no relation to that in [4]. This argument is, in particular, perhaps the cornerstone of the proof of Theorem 4.

With Lemma 5 at hand, we then turn to prove our second ingredient.

Lemma 6 (Informal Statement of Theorem 29). *For a "cheap" solution T-loop and a sequence of concentric cycles that does not contain any terminal, no segment goes more than $O(\log k)$ cycles deep into the sequence.*

We note that in [4], it is proved that no segment (for a DISJOINT PATHS solution) goes more than $2^{O(k)}$ cycles deep into the sequence. For the proof of our lemma, we make use of one intermediate result from [4] about DISJOINT PATHS solutions, which states that there can be $O(k)$ segments of different types. However, besides that, the proof of our lemma is different, based on a simple analysis of the "segment-tree" induced by the segments (see Section 2). We note that rerouting arguments have been used for other terminal-based routing problems [30]. But, in our knowledge, our rerouting arguments are first to achieve a sublinear bound on the number of concentric cycles used by a cheap solution. Further, we believe that another strength of our result is that the arguments used to prove Lemma 6 are rather simple, and therefore easy to reuse in future works.

In particular, Lemma 6 implies that within a $c \log k$-sized (for some constant c) sequence of concentric cycles that does not contain any terminal, every vertex that lies inside the innermost cycles is irrelevant, that is, can be deleted without turning a yes-instance into a no-instance. In

particular, this implies that within a $c \log k \times c \log k$-grid minor that does not contain any terminal, the "middle-most" vertex is irrelevant. Here, it is important to mention that the main component in the proof of Kawarabayashi [27] is a lemma that (essentially) states that within a $ck \times ck$-grid minor that does not contain any terminal, the "middle-most" vertex is irrelevant. Our proof (that is, the arguments briefly described so far) is completely different, and, in particular, yields an exponential improvement (from ck to $c \log k$).

From here, it is easy to provide a $k^{O(1)} \cdot n^2$-time algorithm that given an instance (G, T) of T-CYCLE on planar graphs, outputs an equivalent instance (G', T) of T-CYCLE on planar graphs such that G' is a subgraph of G whose treewidth is bounded by $O(\sqrt{k} \log k)$. Indeed, this can be done by, as long as possible, computing a $c \log k \times c \log k$ grid minor that does not contain any terminal, and removing its "middle-most" vertex, where each iteration requires time $k^{O(1)} \cdot n$, and where $O(n)$ iterations are performed in total.

To reduce the time complexity to be linear in n, we make use of Reed's [37] approach of dividing the plane and working on each "piece" simultaneously, together with an argument by Cho et al. [13] that further simplifies the pieces such that none of the piece contains a sequence of "too many" concentric cycles. Here, the proof is mostly based on a similar approach described in the aforementioned two papers with minor adaptations. However, we need to add one novel and critical step. Briefly, by applying the known approach (with the minor adaptations to our case) we directly obtain pieces such that: *(i)* the total number of vertices on their boundaries is $O(k \log k)$; *(ii)* no piece contains a sequence of more than $c \log k$ many concentric cycles. This already yields a set of vertices U such that $|U| \in O(k \log k)$ and the treewidth of $G' - U$ (where G' is the current graph) is bounded by $O(\log k)$, by taking the union of boundaries (that is what we need for the second item in Theorem 4). However, from this, the aforementioned two papers (i.e., [37] and [13]) directly get their tree decomposition of G' by (implicitly) creating a tree decomposition for each piece, gluing them together, and putting the vertices of U in all bags. For these two previous papers, this was sufficient—they get pieces of treewidth $2^{O(k)}$ and U of size $2^{O(k)}$, and thus by doing this, they do not lose anything. However, for us, this yields a tree decomposition of width $O(k \log k)$, while we need width $O(\sqrt{k} \log k)$. We overcome this by adding a new step, where we attempt to remove (possibly many) boundary vertices, and after performing this step, we are able to directly argue that the graph G' itself (rather than just each one of its pieces individually) does not have a sequence of more than $c \log k$ many concentric cycles that do not contain any terminal (see Section 2). Thus, we get the bound $O(\sqrt{k} \log k)$.

Lastly, having Theorem 4 at hand, let us address the additional work we need to perform in order to prove Theorem 2. Here, we use two powerful theorems/machineries:

- Machinery I [9] (see also [20]): Suppose we are given a set of vertices U with $|U| \in O(k \log k)$ so that the treewidth of $G' - U$ is bounded by $O(\log k)$. Then, one can construct in $k^{O(1)} \cdot n$-time a so-called protrusion decomposition of G' with "very good parameters". That is, roughly speaking, the vertex set of G' can be partitioned into a "universal" part $U^\star \supseteq U$ of size $O(k \log^2 k)$ and $O(k \log^2 k)$ additional "protrusion" parts such that each protrusion part induces a graph of treewidth $O(\log k)$, has $O(\log k)$ many vertices that have neighbors in $U^\star$, while having no neighbors in the other protrusions.

- Machinery II [44]: A generalization of the DISJOINT PATHS problem on planar graphs, where, roughly speaking, we need to preserve the yes/no answer for every possible pairing of the terminals vertices, admits a kernel of size $O((k' \cdot w')^{12})$ where k' is the number of terminals and w' is the treewidth of the graph, and the output graph is a minor of the input graph.

Thus, we can compute the protrusion decomposition using the set obtained from Theorem 1 (which, in particular, contains all terminals), and then kernelize each of the protrusions individually using Machinery II, where the terminal set is defined as those vertices having neighbors outside the protrusion. However, this will not yield a time complexity of $k^{O(1)} \cdot n$ since Machinery II only specifies a polynomial (rather than linear) dependency on n, and, furthermore, this will yield a huge polylogarithmic dependency on k in the size of the reduced instance.

To overcome this, we employ two novel twists in this approach. First, we create a "nested" protrusion decomposition, or, alternatively, one can think about this as "bootstrapping" the entire proof for each protrusion. In more details, for each protrusion P, we yet again apply all the arguments in our proofs so far to now find a set of vertices U_P with $|U_P| \in O(\log k \log \log k)$ so that the treewidth of $P - U_P$ is bounded by $O(\log \log k)$. Then, for each protrusion, we yet again compute a protrusion decomposition, but now with each "subprotrusion" having only $O(\log \log k)$ many vertices

with neighbors outside the subprotrusion, and with treewidth $O(\log \log k)$. Then, the second twist is that instead of directly kernelizing the subprotrusion, we think of Machinery II as an existential rather than an algorithmic result—we know that there *exists* a $\log^{O(1)} \log k$-sized graph that can replace our subprotrusion. Then, we simply guess this replacement! That is, we generate all possible $\log^{O(1)} \log k$-sized graphs. For each generated graph P', we test whether:

1. P' is equivalent to the subprotrusion with the same terminal set. That is, we must preserve the yes/no answers to the instances corresponding to all possible terminal pairings.

2. P' is a minor of P.

This test can be done using a standard dynamic programming-based algorithm over tree decomposition, since both the treewidth of the subprotrusion and the size of P' are bounded by $\log^{O(1)} \log k$. Specifically, we have $2^{\log^{O(1)} \log k} < k^{O(1)}$ many guesses for the replacement, and the "validity" of each can be checked in time $k^{O(1)} \cdot n_P$ where n_P is the number of vertices in P. We believe that this concept of "nested" protrusion decomposition can find applications in improving kernelization algorithms, in terms of both kernel size and running time, for other terminal-based problems.

Additional discussion and related open problems can be found in Section 8.

2 Overview of Our Proofs

In this section, we provide an overview of our proofs. We will only informally define notions required to explain the overview. These notions are formally defined in later sections. Furthermore, to ease the presentation, we borrow some figures here from later sections, which will appear again in their respective sections.

2.1 FPT Algorithm

Our first contribution is an FPT algorithm for T-Cycle in planar graphs that has a running time subexponential in k (which is, $2^{O(\sqrt{k} \log k)}$) and linear in n. Since T-Cycle can be solved in $2^{O(\text{tw})} n$ time (e.g., using [17]), a natural way to attack our problem is via the technique of *treewidth reduction* by removal of some *irrelevant vertices*. In fact, this technique was used by Kawarabayashi [27] to reduce the treewidth to be polynomial in k. To get the algorithm that we desire, we need to

1. reduce the treewidth to $O(\sqrt{k} \log k)$ by removal of some irrelevant vertices, and

2. to ensure that this entire removal procedure is performed in $k^{O(1)} \cdot n$ time.

Irrelevant vertices: We begin with establishing that if there is a vertex v that is "sufficiently separated" from each vertex in T, then v is irrelevant. A set of cycles $\mathcal{C} = (C_0, \ldots, C_r)$ is *concentric* if the cycles in $\mathcal{C}$ are vertex disjoint and cycle C_{i-1} is contained in the interior of C_i (for $i \in [r]$). See Figure 1a for an illustration. If there exists a sequence of $\ell + 1$ concentric cycles $C_0, \ldots, C_\ell$ such that v in the interior of C_0 and T is in the exterior of C_ℓ, then we say that v is *ℓ-isolated*. Let D_i denote the disk corresponding to the interior of C_i.

The first component of our algorithm is a proof that establishes that there exists some $g(k) \in O(\log k)$, which we explicitly compute, such that each $g(k)$-isolated vertex is irrelevant, and hence, can be removed from G safely. To this end, we use a notion called *CL-configuration*, which is a pair $\mathcal{Q} = (\mathcal{C}, L)$, where $\mathcal{C} = (C_0, \ldots, C_r)$ is a set of concentric cycles such that $D_r \cap T = \emptyset$, and L is a T-loop. See Figure 1b for an illustration. The connected components of $D_i \cap L$ are said to be C_i-*segments*. Intuitively, we say that a C_j-segment S_a is in the *zone* of a C_j-segment S_b if S_a lies in the region encompassed by S_b and the cycle C_j. This can be used to define a partial order $\prec$ on the C_j-segments of the set of CL-configuration $\mathcal{Q}$ as follows: we say that $S_a \prec S_b$ if and only if S_a lies in the zone of S_b. We need the following notion of segment types (see Figure 2a for an illustration.).

Definition 7 (**Segment types**). *Let $\mathcal{Q} = (\mathcal{C}, L)$ be a depth r CL-configuration of G. Moreover, let S_1 and S_2 be two C_j-segments of $\mathcal{Q}$ such that S_i, for $i \in [2]$, has endpoints u_i and v_i. We say that S_1 and S_2 have the same C_j-type if:*

1. *there exist paths P and P' on C_j connecting an endpoint of S_1 with an endpoint of S_2 such that these paths do not pass through the other endpoints of S_1 and S_2,*

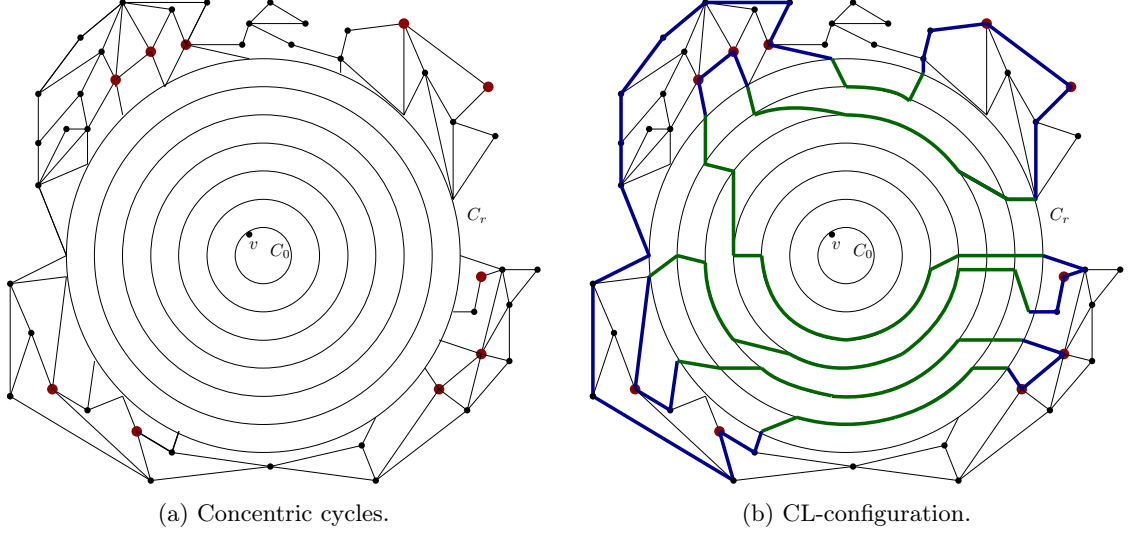(a) Concentric cycles.　　　　　　　　　(b) CL-configuration.

Figure 1: (a) A set $\mathcal{C}$ of concentric cycles and v is r-isolated. (b) A CL-configuration $(\mathcal{C}, L)$. The segments of L are represented in green and the edges of L outside of D_r are shown in blue. Hence, the green edges along with the blue edges combine to give the loop L. In both subfigures, the terminal vertices are highlighted in red.

2. *no segment of $\mathcal{Q}$ has both endpoints on P or on P', and*

3. *the closed-interior of the cycle $P \cup S_1 \cup P' \cup S_2$ does not contain the disk D_0.*

A C_j-type of segment is an equivalence class of segments of $\mathcal{Q}$. See Figure 2b for an illustration of segment types.
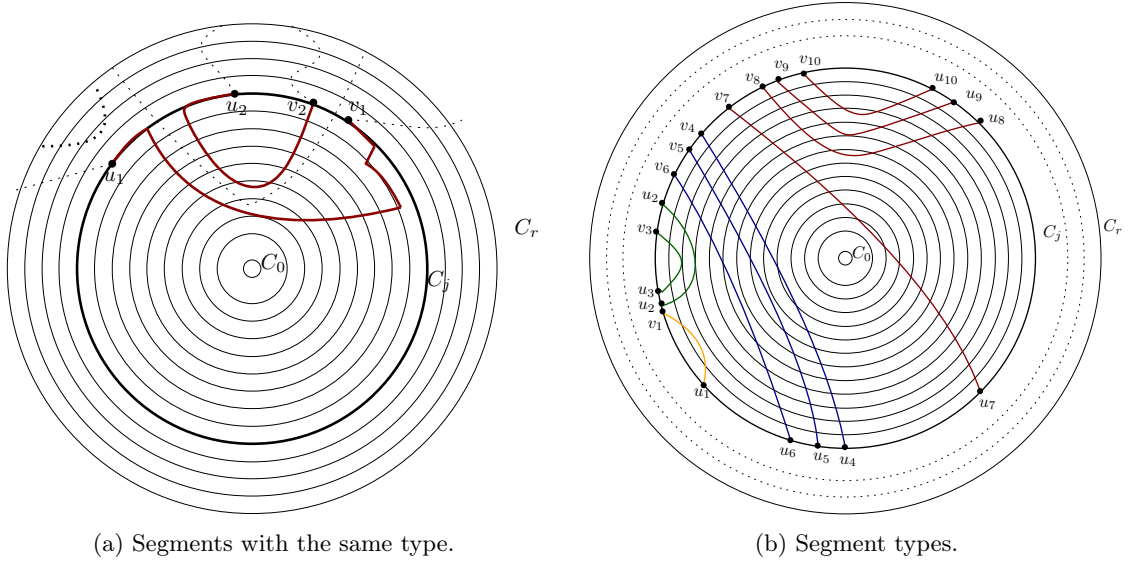

(a) Segments with the same type.　　　　　(b) Segment types.

Figure 2: (a) S_1 and S_2 are C_j-segments with endpoints u_1, v_1 and u_2, v_2, respectively. Moreover, P and P' are the (u_1, u_2)-path and (v_1, v_2)-path along C_j, respectively. Here, S_1 and S_2 have the same C_j-type. (b) Here, segments $S_1, \ldots, S_{10}$ are C_j-segments such that segment S_i has endpoints u_i and v_i. Segments S_4 and S_7 are not of the same C_j-type because of condition (3) in Definition 15, and S_6 and S_2 are not of the same C_j-type because of condition (2) in Definition 15. Further, when S_6 and S_2 are restricted to D_{j-2}, they have the same C_{j-2}-type. Finally, $S_{10} \prec S_9 \prec S_8 \prec S_7$, and similarly, $S_6 \prec S_5 \prec S_4$.

Finally, for any T-loop, say L, we define a *cost* function that corresponds to the number of edges in L that are not contained in any cycle of $\mathcal{C}$. The main idea behind this is that a T-loop has to
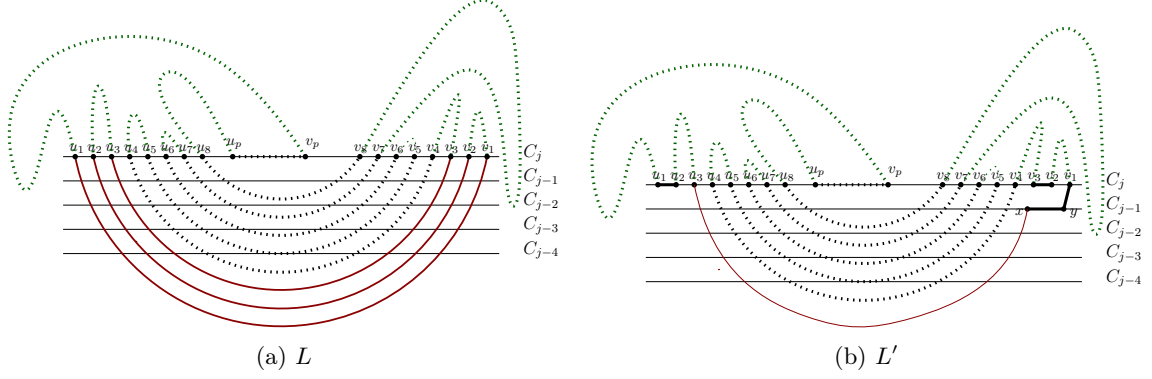
7

Figure 3: (a) Here, S_i is the C_j segment with endpoints u_i and v_i, segments S_1, S_2, S_3 are illustrated in red, and segments $S_1, \ldots, S_8$ have the same C_j-type. (b) In the rerouting, we remove the segment S_2 completely (reducing the cost) and use a part of the segment S_1 (not increasing the cost), and additionally use some paths along concentric cycles C_j and C_{j-1} that have cost 0.

"pay" every time it goes to a "deeper" cycle of $\mathcal{C}$. Then, a T-loop L is *cheap* if there is no other loop with cost strictly lesser than that of L. We would then like to show that any cheap T-loop will not go "very deep" in $\mathcal{C}$, deeming the deeper cycles of $\mathcal{C}$ irrelevant.

Now, consider a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$ of depth r. We are ready to explain our first main lemma, which says that if L is a cheap T-loop, then there cannot be more than 3 segments with the same C_j-type, for $j \in \{0, \ldots, r\}$. To prove this, we show that if we have more than 3 segments with the same C_j-type, then we can reroute L in such a manner that we get a new T-loop L' which is cheaper than L. See Figure 3 for an illustration of the intuition. This gives us Lemma 5 from Section 1.

The next component of the proof is to establish, using our above result, that a cheap T-loop will not go beyond $O(\log k)$ levels deep, and we compute a $g(k) \in O(\log k)$ such that every vertex that is $g(k)$-isolated, i.e., contained in $D_{r-g(k)-1}$ is irrelevant. To this end, we use a novel idea based on segment forests. In the discussion that follows, $\mathcal{Q}$ is a CL-confiugation of depth r.

For every $j \in [r]$, we use the hierarchy induced by $\prec$ to associate a forest structure called a *segment forest*. To begin with, if a C_j-segment S_a is in the zone of a C_j segment S_b, then we say that S_b is an *ancestor* of S_a and (S_a is a *descendant* of S_b). If S_b is an ancestor of S_a such that there is no C_j-segment S_c that satisfies $S_a \prec S_c \prec S_b$, then S_b is a parent of S_a (and S_a is a child of S_b). Modeling all parent child relationships with edges gives rise to a forest that we call the *j-th segment forest* of $\mathcal{Q}$. See Figure 4 for an example. A C_j-segment S_p is an *m-th generation* ancestor of S_q if S_p is an ancestor of S_q and the length of the path from S_p to S_q in the j-th segment forest is m. The *height* of a segment forest is the length of the longest path from a leaf segment to a segment that has no ancestors. The *width m j-subtree* of a C_j-segment S is the tree induced by all C_j-segments that are descendants of S at a distance of at most m in the j-th segment forest.

For a segment S, its eccentricity e is defined as $e = \min_i V(C_i \cap S) \neq \emptyset$. Let m be any positive integer. We will make in Lemma 27 the observation that in the width m e-subtree T rooted at the k-th generation ancestor of S, either all ancestors of S in T have the same type or there is a segment in T that is not an ancestor of S. This observation coupled with Lemma 5 will tell us that the width three e-subtree rooted at the 3-rd generation ancestor of a segment S of eccentricity e has a segment that is not an ancestor of S. The fact that the height of the r-segment forest of $\mathcal{Q}$ is at most logarithmic follows then from an inductive argument. In [4, Lemma 5], it is shown that a cheap solution for DISJOINT PATHS has $O(k)$ different types of segments. Using this result, it is easy to check that the same holds true for T-CYCLE. Leveraging this fact, we deduce that the height of the r-segment forest of $\mathcal{Q}$ is $O(\log k)$. This gives us Lemma 6 from Section 1.

Fast Removal of Irrelevant Vertices. Since we have established that all $g(k)$-isolated vertices are irrelevant, our next task is to remove $g(k)$-isolated vertices in $k^{O(1)} \cdot n$ time. Note that there is a rather straightforward way to remove such vertices by, as long as possible, computing a $g(k) \times g(k)$ grid minor that does not contain any terminal, and removing its "middle-most" vertex. Here, each iteration requires $k^{O(1)} \cdot n$ time, and hence, in total, this algorithm requires $k^{O(1)} \cdot n^2$ time.

To remove the irrelevant vertices in $k^{O(1)} \cdot n$ time, we use Reed's [37] approach of "cutting" the
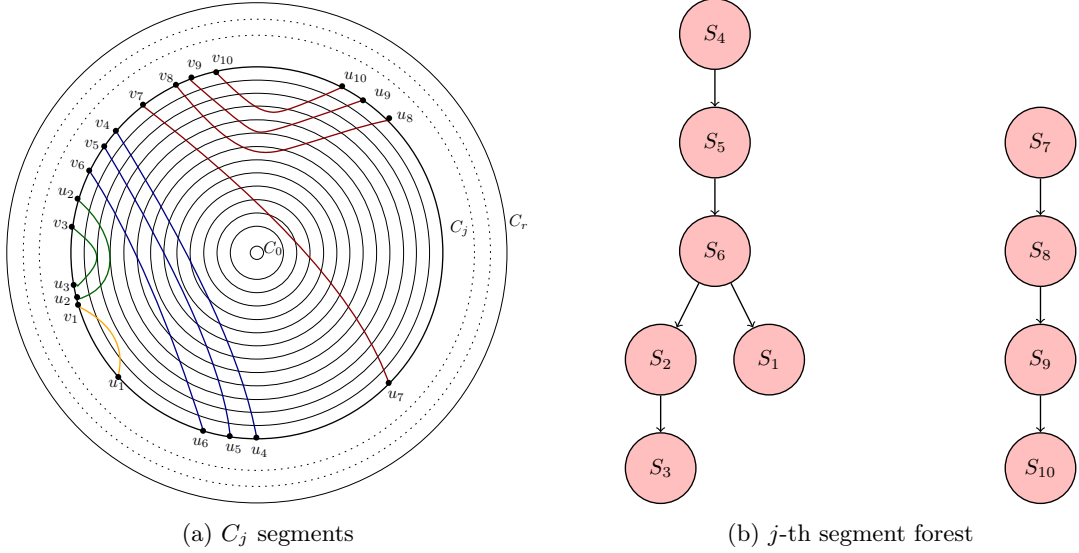
(a) C_j segments

(b) j-th segment forest

Figure 4: Subfigure (a) depicts C_j segments of a CL configuration $\mathcal{Q}$ of depth r. Here, segments $S_1, \ldots, S_{10}$ are C_j-segments such that segment S_i has endpoints u_i and v_i. All C_j-segments of the same type are depicted in a common color. The colors red, blue green and yellow are used to distinguish segment types. Subfigure (b) shows the j-th segment forest of $\mathcal{Q}$.

plane into more "manageable" planes, and then working on these pieces simultaneously. To this end, we consider the "punctured" planes, where, intuitively, a *c-punctured* plane means a punctured plane with c connected components in its boundary. Note that it is desirable that all terminals lie on the boundary of a punctured plane. This can be achieved, to begin with, by considering the graph to be embedded on a k-punctured plane, where each terminal is a trivial hole. Then, the idea of Reed is to recursively cut the plane along some curves of "bounded" size into "nice" subgraphs, where a subgraph H is *nice* if every vertex of H that is $g(k)$-isolated from the boundary of H is also $g(k)$-isolated in H. More specifically, we use the following result of Reed.

Proposition 8 (Lemma 2 in [37]). *Let H be a nice subgraph of G embedded on a c-punctured plane $\boxdot$ such that $c > 2$. Then, we can compute both a non-crossing proper closed curve J contained in $\boxdot$ and a (possibly empty) set X of vertices that are $g(k)$-isolated from the boundary of $\boxdot$ in $O(|V(H)|)$ time such that*

1. *the number of vertices of $H - X$ intersected by J is at most $6g(k) + 6$, and*

2. *$\boxdot \setminus J$ contains at most 3 components each of which has less than c holes.*

A careful analysis shows that by a recursive application of this *cut reduction* procedure, we finally get at most $4k + 8$ components, each of which is embedded on either a 2-punctured plane or a 1-punctured plane and the total number of vertices on the boundaries of these punctured planes is $O(kg(k))$, i.e., $O(k \log k)$. Since we apply this cut reduction $O(k)$ times, the cutting-plane procedure takes $O(k \cdot n)$ time in total.

The novel component of this part of our algorithm is to decide for each vertex v on the boundary of these punctured planes if v is $g(k)$-isolated in G, and remove if it is. We can perform this check in $O(n)$ time for one vertex, and since there are $O(k \log k)$ vertices altogether on the boundaries of punctured planes, we can remove all $g(k)$-isolated vertices from the boundaries of punctured planes in $O(k \log k \cdot n)$ time. After this step, no vertex on the boundary of a punctured plane is $g(k)$-isolated. Finally, we use the irrelevant vertex removal algorithm of Cho et al. [13] which, given a c-punctured plane $\boxdot$ where $c \leq 2$, removes all vertices from $\boxdot$ that are $4g(k)$-isolated from the boundary of $\boxdot$. Their algorithm in total requires $O(n)$ time for each c-punctured plane ($c \in [2]$), and since there are $O(k)$ such planes, this step requires $O(k \cdot n)$ time in total.

At this point, we prove that no vertex in G is $5g(k)$-isolated. In particular, we show that if a vertex v in $\boxdot$ is $5g(k)$-isolated, then there exists a vertex on the boundary of $\boxdot$ that is $g(k)$-isolated. An illustration of the proof is provided in the Figure 5.

Finally, using a result of Adler et al. [3] (Proposition 11), it follows that if there is no vertex in a planar graph G that is $5g(k)$-isolated from a set of vertices T of size k, then the treewidth of G is
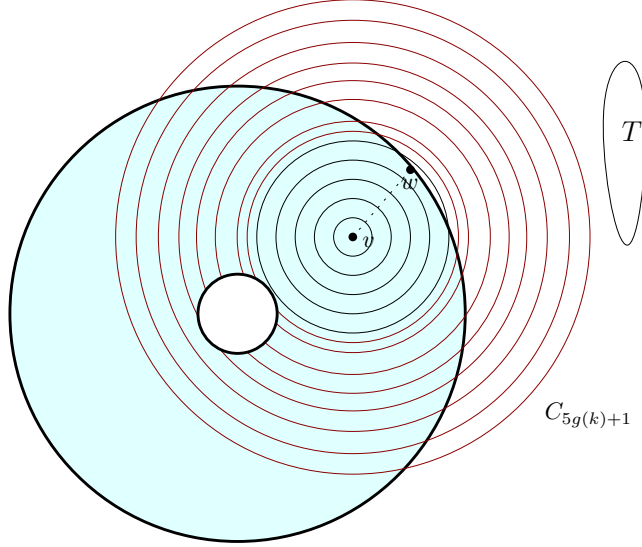
Figure 5: A 2-punctured plane $\boxdot$ is depicted (in blue) by its two holes highlighted in bold. v is a vertex in $\boxdot$ which is $5g(k) + 1$-isolated in G, and hence there is a sequence of $C_0, \ldots, C_{5g(k)+1}$ concentric cycles separating v and T. Since v is not $4g(k)$-isolated from boundary of $\boxdot$, there is some boundary vertex w of $\boxdot$ such that $w \in D_{4g(k)}$. In this case, a sequence of concentric cycles $C_{4g(k)+1}, \ldots, C_{5g(k)+1}$ separate w from T, implying that w is $g(k)$-isolated.

$O(\sqrt{k}g(k))$, i.e., $O(\sqrt{k}\log k)$. This gives us Theorem 4 from Section 1. Indeed, the treewidth of the reduced graph, say G', obtained after removing irrelevant vertices from G, is $O(\sqrt{k}\log k)$. Moreover, let U be the set of all boundary vertices of all $O(k)$ many punctured planes ($|U| \in O(k\log k)$), then observe that in $G' - U$, there is no sequence of $4g(k) + 1$ concentric cycles, and hence the treewidth of $G' - U$ is upper bounded by $4g(k)$, i.e., $O(\log k)$. Now, a dynamic-programming based algorithm can be used to solve our problem in $2^{O(\sqrt{k})\log k} \cdot n$ time. We refer to the procedure described in the above steps as Reed decomposition algorithm.

2.2 Kernelization Algorithm

The main tool used in our kernelization algorithm is the so-called *protrusion decomposition*. Please refer to [20, Chapter 15] for a detailed discussion on this topic. Below, we provide a brief overview starting with a few definitions.

Let G be a graph. Given a vertex set $U \subseteq V(G)$, its *boundary* ∂U, is the set of vertices in U that have at least one neighbor outside of U. A vertex set $W \subseteq V(G)$ is called a *treewidth-η-modulator* if $\mathrm{tw}(G - W) \leq \eta$. A vertex set $Z \subseteq V(G)$ is a *q-protrusion* if $\mathrm{tw}(Z) \leq q$ and $|\partial Z| \leq q$. Finally, given a vertex set $S \subset V(G)$, we use $S^+ = N_G[S]$ to denote the set of vertices that are in the closed neighborhood of S in G.

A partition $X_0, X_1, \ldots, X_\ell$ of vertex set $V(G)$ of a graph G is called an (α, β, γ)-*protrusion decomposition* of G if α, β and γ are integers such that:

- $|X_0| \leq \alpha$,

- $\ell \leq \beta$,

- X_i for every $i \in [\ell]$ is a γ-protrusion of G,

- for $i \in [\ell]$, $N_G(X_i) \subseteq X_0$ (here, $N_G(X_i)$ denotes the open neighborhood of X_i in G).

For $i \in [\ell]$, the sets X_i are called *protrusions*. For every $i \in [\ell]$, we set $B_i = X_i^+ \setminus X_i$.

A central result about protrusions is that if a planar graph G has a treewidth-η modulator S, then G has a $(O(|\eta| \cdot |S|), O(|\eta| \cdot |S|), O(|\eta|))$-protrusion decomposition with $X_0 \supseteq S$ [20, Lemma 15.14]. Moreover, such a protrusion can be computed efficiently using Algorithm 1. The key motivation behind our use of protrusion decompositions for kernelization is Theorem 4, which readily gives us a graph $\tilde{G}$ and a vertex set $U \subset V(\tilde{G})$ of size $O(k\log k)$ that is a treewidth-$O(\log k)$-modulator of $\tilde{G}$, where (G, T) and $(\tilde{G}, T)$ are equivalent as T-CYCLE instances.

10

Apart from protrusions, the other main tool we use is a blackbox from [44]. In [44], the authors define the notion of B-linkage equivalence. Specifically, two graphs G_1, G_2 with a common vertex set B are said to be B-linkage equivalent w.r.t. DISJOINT PATHS if for every set of pairs of terminals $M \in B^2$, the disjoint path instances (G_1, M) and (G_2, M) are equivalent. The key result we use from [44] is that a planar graph G can be replaced by a smaller planar graph H that is B-linkage equivalent w.r.t. DISJOINT PATHS, where the size of H is polynomial in the treewidth of G and $|B|$. This leads us to our first kernelization algorithm outlined in Figure 6 that serves as a forerunner to the linear time algorithm summarized in Figure 7.
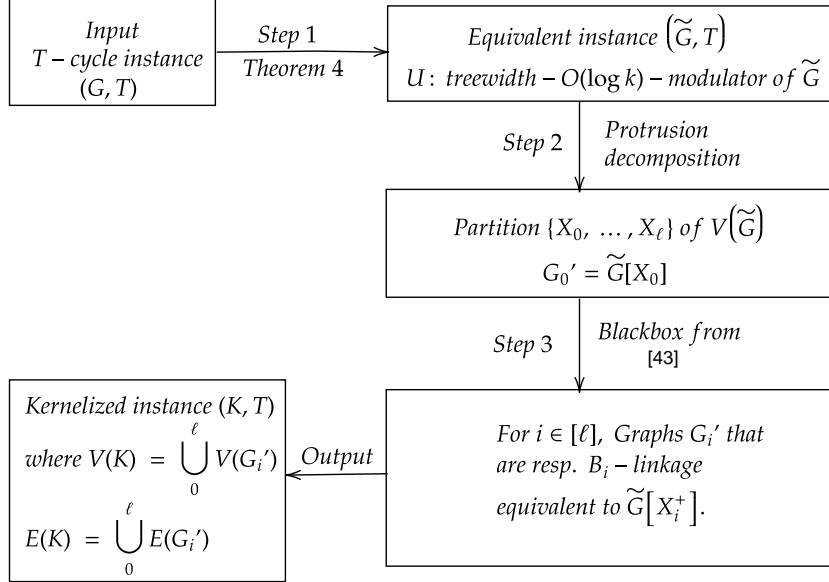

Figure 6: Here, we depict the Steps in Kernelization Algorithm-I. The arrows are labelled with the Step# and the tool used, and every box contains the output data associated to that box.

One of the key insights behind the linear time kernelization algorithm is to treat the black box result from [44] as an existential result rather than an algorithmic one. Thus, instead of using the algorithm from [44] for replacing graphs with smaller B-linkage equivalent counterparts, one could guess the replacement graph instead. One obstacle in doing this is that $|B_i|$ where $B_i = X_i^+ \setminus X_i$ and the treewidth of $\tilde{G}[X_i^+]$ for $i \in [\ell]$ are both $O(\log k)$ making the guesses expensive. A potential remedy, which is our other key insight, is to apply a nested protrusion decomposition. However, a roadblock to this remedy is that we do not have a small treewidth modulator for graphs $\tilde{G}[X_i^+]$ right away. This is fixed by adapting the Reed decomposition algorithm for T-CYCLE to B_i-linkage equivalence w.r.t. T-CYCLE. We stress here that the notion of linkage equivalence w.r.t. DISJOINT PATHS differs from the notion of linkage equivalence w.r.t. T-CYCLE. Below, we give an intuitive definition for linkage equivalence w.r.t. T-CYCLE.

Given a T-CYCLE instance (G, T), let G_1 and G_1' be subgraphs of G with a common set of vertices $B \subset V(G_1) \cap V(G_1')$. Then, G_1 is B-linkage equivalent w.r.t. T-CYCLE to G_1' if

- a T-loop in G can be written as a union of collections of paths $\mathcal{P}_1 \cup \mathcal{P}_2$ with endpoints of paths in $\mathcal{P}_1 \cup \mathcal{P}_2$ lying in B such that paths in $\mathcal{P}_1$ use edges that belong to G_1 and paths in $\mathcal{P}_2$ use edges outside of G_1 if and only if in the graph H obtained by replacing G_1 with G_1' in G there is a T-loop that can be written as a union of collections of paths $\mathcal{P}_1' \cup \mathcal{P}_2$ where paths in $\mathcal{P}_1'$ use edges that belong to G_1'.

A vertex $v \in G_1$ is B-linkage irrelevant if $G_1 - v$ is B-linkage equivalent w.r.t. T-CYCLE to G_1. It is easy to check that vertices that are B-linkage irrelevant in G_1 are also irrelevant for the T-CYCLE problem in G.

Returning to our discussion, an adaptation of the Reed decomposition algorithm allows us to delete B_i-linkage irrelevant vertices from $\tilde{G}[X_i^+]$ for every $i \in [\ell]$. This has two important outcomes. First, for every $i \in [\ell]$, we get smaller graphs G_i' that are B_i-linkage equivalent to $\tilde{G}[X_i^+]$. Second, for every $i \in [\ell]$, the Reed decomposition algorithm also computes sets $B_i' \supseteq B_i$ such that the sets B_i'
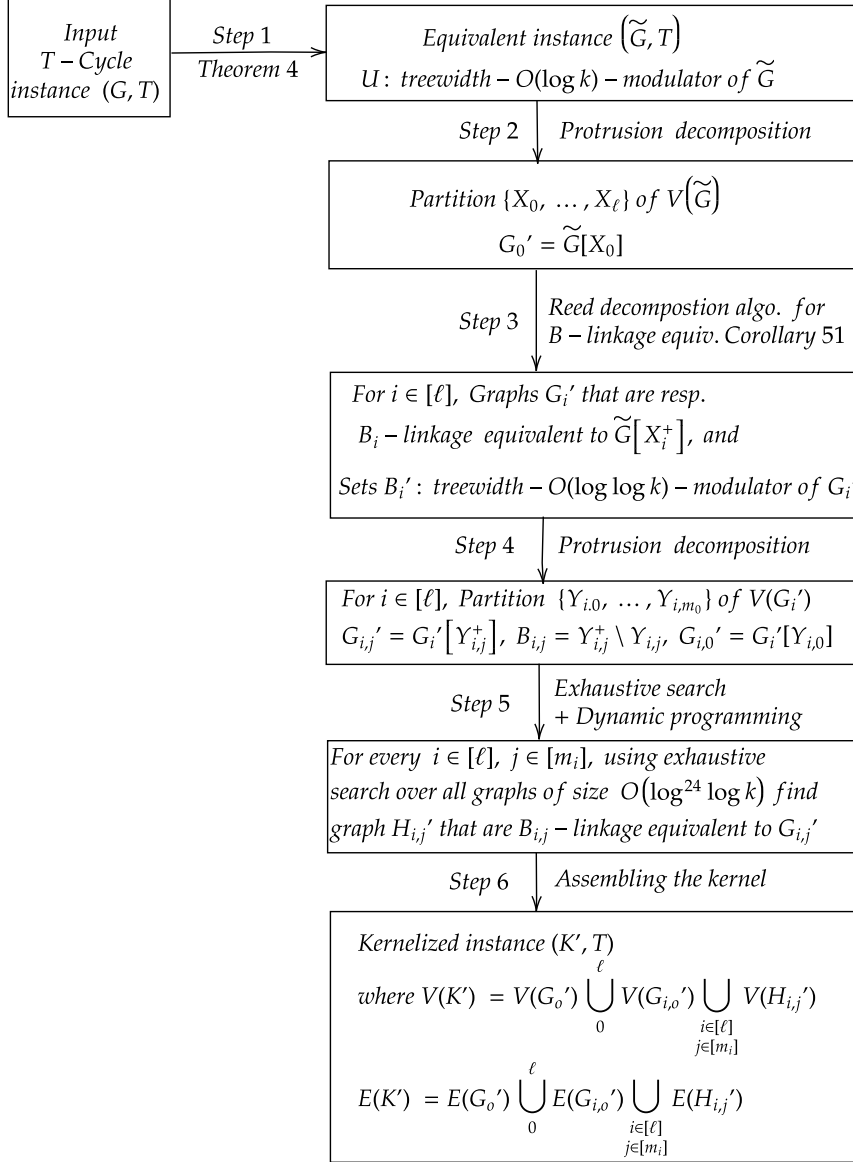
Figure 7: Here, we depict the Steps in Kernelization Algorithm-II. The arrows are labelled with the Step# and the tool used, and every box contains the output data associated to that box.

are treewidth-$O(\log \log k)$-modulators of G'_i. In fact, the sets B'_i are boundary vertices in the Reed decomposition algorithm.

With that at hand, we can apply protrusion decomposition to graphs G'_i giving us "subprotrusions". Specifically, computing a protrusion decomposition of G'_i for every $i \in [\ell]$, gives a partition $\{Y_{i,0}, \ldots, Y_{i,m_i}\}$ of $V(G'_i)$. For $i \in [\ell]$, $j \in [m_i]$, let $G'_{ij} = G'_i[Y^+_{i,j}]$, where $Y^+_{i,j} = N_{G'_i}[Y_{i,j}]$, and $B_{i,j} = Y^+_{i,j} \setminus Y_{i,j}$. Also, let $G'_{i,0} = G'_i[Y_{i,0}]$ for $i \in [\ell]$.

To now apply [44] as an existential result, we need to guess graphs of size $\log^{O(1)} \log k$. Checking if a guessed graph is $B_{i,j}$ linkage equivalent w.r.t. DISJOINT PATHS to a subprotrusion $G'_{i,j}$ and replacing it while preserving planarity involves the use of two dynamic programming subroutines: MINOR CONTAINMENT and DISJOINT PATHS, the details of which can be found in Section 7. We do these replacements for all $i \in [\ell]$, $j \in [m_i]$. The replacement graphs for $G'_{i,j}$ are denoted by $H'_{i,j}$.

The kernel for the T-CYCLE problem is given by assembling the graph K' where

$$V(K') = V(G'_0) \bigcup_{i \in [\ell]} V(G'_{i,0}) \bigcup_{\substack{i \in [\ell] \\ j \in [m_i]}} V(H'_{i,j}) \text{ and } E(K') = E(G'_0) \bigcup_{i \in [\ell]} E(G'_{i,0}) \bigcup_{\substack{i \in [\ell] \\ j \in [m_i]}} E(H'_{i,j}).$$

Figure 7 outlines the key steps in the linear time kernelization algorithm. We show in Section 7 that $|V(K')| \leq k \log^4 k$, and K' can be computed in $k^{O(1)} \cdot n$ time.

3 Preliminaries

For integers $i \leq j$, let $[i,j] = \{i, i+1, \ldots, j-1, j\}$. Moreover, for $\ell \in \mathbb{N}$, let $[\ell] = [1, \ell]$. Given a set A, the Cartesian product $A \times A$ is denoted by A^2.

3.1 Graph Theory

For a graph G, let $V(G)$ and $E(G)$ denote the vertex set and edge set of G, respectively. When G is clear from the context, we denote $|V(G)|$ by n. For a vertex set $X \subseteq V(G)$, we use $N_G(X)$ to denote the open neighborhood of X in G and $N_G[X]$ to denote the closed neighborhood of X in G. For a graph G and a vertex set $U \subseteq V(G)$, we use $G[U]$ to denote the graph induced by U. Moreover, let $G - U = G[V(G) \setminus U]$, and for $x \in V(G)$, let $G - x = G - \{x\}$.

For two distinct vertices $u, v \in V(G)$, a (u, v)-*path* is a path with endpoints u and v. Given $S, X, Y \subseteq V(G)$, we say that X *separates* X and Y if $G - S$ has no path with an endpoint in X and an endpoint in Y. A *cycle* is a connected subgraph of G where each vertex has degree 2. A cycle C *separates* X and Y if $V(C)$ separates X and Y. Given a subset $T \subseteq V(G)$, a T-*loop* is a simple cycle C such that $V(T) \subseteq V(C)$. We are interested to study the following problem.

T-CYCLE
Input: A graph G, and a set $T \subseteq V(G)$ of vertices.
Question: Does G have a cycle C such that $T \subseteq V(C)$?

A vertex v in G is *irrelevant* if $G - v$ has a T-loop whenever G has a T-loop. When we say that we delete a path P from a graph G, we delete all internal vertices of P from G.

A graph H is a *minor* of G if we can obtain H from G by using the following three operations: vertex deletion, edge deletion, and edge contractions. If H is a minor of G, we also say that G has a H minor. A $p \times q$-grid is the graph obtained by the Cartesian product of two paths on p and q vertices, respectively. Treewidth, formally defined below, is a measure of how "tree-like" a graphs is.

Definition 9 (**Treewidth**). *A* tree decomposition *of a graph G is a pair (T, χ), where T is a tree and χ is a mapping from $V(T)$ (called* bags*) to subsets of $V(G)$, i.e., $\chi : V(T) \to 2^{V(G)}$, satisfying the following properties.*

1. *For every $uv \in E(G)$, there exists $t \in V(T)$, such that $\{u, v\} \subseteq \chi(t)$.*

2. *For every $v \in V(G)$, the subgraph of T induced by the set $T_v = \{t \in V(T) \mid v \in \chi(t)\}$ is a non-empty tree.*

The width *of a tree decomposition (T, χ) is $\max_{t \in V(T)} |\chi(t)| - 1$. The* treewidth *of G is the minimum possible width of a tree decomposition of G. The mapping χ is extended from vertices of T to subgraphs of T. In particular, for a subgraph U of T, $\chi(U) = \bigcup_{v \in V(U)} \chi(v)$.*

Planarity. A *planar graph* is a graph that can be embedded in the Euclidean plane. Whenever we consider a planar graph G, we consider an embedding of G in the plane as well, and, to simplify notation, we do not distinguish between a vertex of G and the point of the plane used in the drawing to represent the vertex or between an edge and the arc representing it. The *faces* of a planar graph are the regions bounded by the edges, including the outer infinitely large region. For a planar graph G, the *radial distance* between two vertices u and v, denoted $\mathsf{d}^R(u, v)$, is one less than the minimum length of a sequence of vertices that starts at u and ends at v, such that every two consecutive vertices in the sequence lie on a common face. As observed in [4], combining results from [22] and [38], we have the following proposition.

Proposition 10 ([4]). *Any planar graph with treewidth at least $4.5k$ has a $(k \times k)$-grid minor.*

Concentric Cycles. Let G be a planar graph. Consider a cycle C of G and let D be the closed disk associated to C. We say that D is *internally chordless* if there is no path in G whose endpoints are vertices of C and whose edges belong to the open interior of C. Let $\mathcal{C} = (C_0, \ldots, C_r)$ be a sequence of cycles in G. By D_i, we denote the closed interior of C_i (for $i \in [0, r]$). $\mathcal{C}$ is *concentric* if, for $i \in [0, r]$, C_i is contained in the open interior of D_i. Furthermore, $\mathcal{C}$ is said to be *tight* in G, if, additionally,
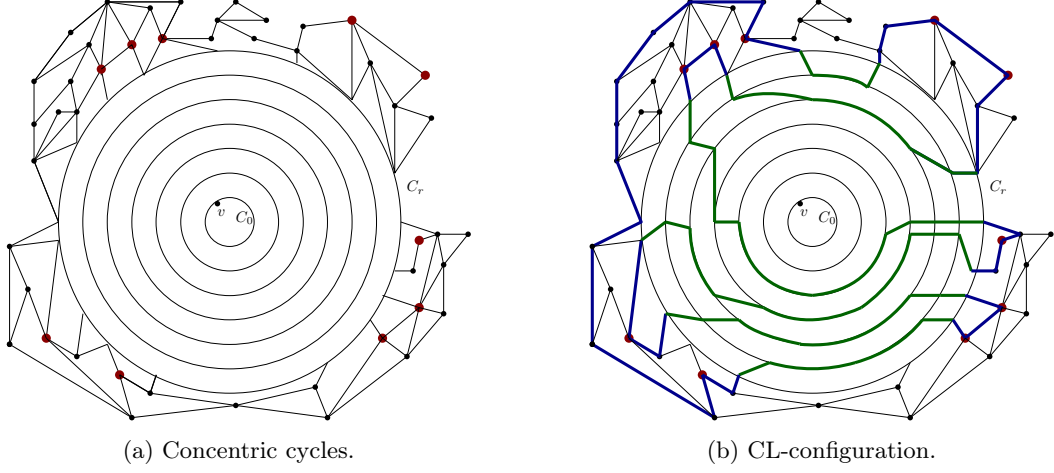
| (a) Concentric cycles. | (b) CL-configuration. |

Figure 8: (a) A sequence $\mathcal{C}$ of concentric cycles and v is r-isolated. (b) A CL-configuration $(\mathcal{C}, L)$. The segments of L are represented in green and the edges of L outside of D_r are shown in blue. Hence, the green edges along with the blue edges combine to give the loop L. In both subfigures, the terminal vertices are highlighted in red.

1. D_0 is internally chordless.

2. For $i \in [0, r]$, there is no cycle in G that is contained in $D_{i+1} \setminus D_i$ and whose closed interior D has the property $D_i \subsetneq D \subsetneq D_{i+1}$.

A vertex v is said to be ℓ-*isolated* if there exists $\ell + 1$ concentric cycles $C_0, \ldots, C_\ell$ such that v is contained in D_0 and no vertex of T is contained in D_ℓ. Intuitively, v is separated from each vertex in T by ℓ concentric cycles $C_1, \ldots, C_\ell$. See Figure 1a for an illustration. The following result from [4] shall be useful for us.

Proposition 11 ([4]). *There exists an algorithm that, in $2^{(r \cdot \sqrt{|T|})^{O(1)}} \cdot n$ time, given a n-vertex planar graph G, $T \subseteq V(G)$, and $r \in \mathbb{N}_0$, either outputs a tree decomposition G of width at most $9 \cdot (r+1) \cdot \lceil \sqrt{|T| + 1} \rceil$ or an internally chordless cycle C of G such that there exists a tight sequence of cycles $C_0, \ldots, C_r$ in G where*

- *$C_0 = C$, and*

- *all vertices of T are in open exterior of C_r.*

Segments. Let G be a planar graph and let D be the closed interior of a cycle C of G. Given a path P in G, a subpath P' of P is a D-*segment* of P, if P' is a non-empty (possibly edgeless) path obtained as a connected component of the intersection of P with D. Given a T-loop L, we say that P' is a D-segment of L if P' is a D-segment of a subpath P_0 of L.

CL-configurations. A *CL-configuration*, formally defined below, is used to capture the interactions between a sequence of concentric cycles $\mathcal{C}$ and a T-loop L. See Figure 8b for an illustration.

Definition 12 (**CL-configuration**). *Given a planar graph G, a pair $\mathcal{Q} = (\mathcal{C}, L)$ is a CL-configuration of G of depth r if $\mathcal{C} = (C_0, \ldots, C_r)$ is a sequence of concentric cycles in G, L is a T-loop in G, and D_r does not contain any vertex from T. The D_r-segments of L are said to be the segments of $\mathcal{Q}$.*

For a segment P of $\mathcal{Q}$, consider the minimum i such that $V(C_i \cap P) \neq \emptyset$. Then, the *eccentricity* of P is i. Given a cycle $C_i \in \mathcal{C}$ and a segment P of $\mathcal{Q}$, the i-*chords* of P are the connected components of $P \cap \operatorname{int}(D_i)$. Similarly, for every i-chord X of P, we define the i-*semichords* of P as the connected components of the set $X \setminus D_{i-1}$. Notice that i-chords as well as i-semichords are open arcs. Given a segment P that does not have any 0-chord, we define its *zone* as the connected component of $D_r \setminus P$ that does not contain the open-interior of D_0. (A zone is an open set.) Following [4], we define convex segments and convex CL-configurations.

Definition 13 (**Convex Segments and CL-configurations.**). *A segment P of $\mathcal{Q}$ is* convex *if the following three conditions are satisfied:*

14

(i) *P has no 0-chord.*

(ii) *For every $i \in [r]$, the following holds:*

      a. *P has at most one i-chord,*

      b. *if P has an i-chord, then $P \cap C_{i-1} \neq \varnothing$.*

      c. *Each i-chord of P has exactly two i-semichords.*

(iii) *If P has eccentricity $i < r$, there is another segment inside the zone of P with eccentricity $i+1$.*

A CL-configuration $\mathcal{Q}$ is convex *if all its segments are convex.*

Next, we define a notion of "*cost*" for a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$, which corresponds to the number of edges in L that do not belong to $\mathcal{C}$.

Definition 14 (**Cheap Loops**). *Let G be a planar graph and $\mathcal{Q} = (\mathcal{C}, L)$ be a CL-configuration of G of depth r. For a subgraph H of G, let $c(H) = |E(H) \setminus \bigcup_{i \in [0,r]} E(C_i)|$. L is $\mathcal{C}$-cheap if there is no other CL-configuration $\mathcal{Q}' = (\mathcal{C}, L')$ such that L' is a T-loop and $c(L) > c(L')$.*

4 Irrelevant Vertex Argument

Recall that a vertex v of G is irrelevant if $G - v$ has a T-loop whenever G has a T-loop. In this section, we will establish that, for some $c > 1$, if G has a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$ where $\mathcal{C} = (C_0, \ldots, C_r)$ such that $r > c \log k$, then every vertex contained in $D_{r-c\log k}$ is irrelevant, i.e., each $c \log k$-isolated vertex is irrelevant. For the rest of this section, we assume that G is a planar graph, $T \subseteq V(G)$, and $k = |T|$.

4.1 Segments of the Same Type

Let $\mathcal{Q} = (\mathcal{C}, L)$ be a depth r convex CL-configuration of G. In this section, we will establish that if L is $\mathcal{C}$-cheap, then the segments of L behave nicely and do not intersect "deeper" cycles of $\mathcal{C}$. To this end, we will use the following notion of segment types.

Definition 15 (**Segment types**). *See Figure 9a for an illustration. Let $\mathcal{Q} = (\mathcal{C}, L)$ be a depth r convex CL-configuration of G. Moreover, let S_1 and S_2 be two C_j-segments of $\mathcal{Q}$ such that S_i, for $i \in [2]$, has endpoints u_i and v_i. We say that S_1 and S_2 are* parallel*, and write $S_1 \parallel S_2$ if:*

1. *there exist paths P and P' on C_j connecting an endpoint of S_1 with an endpoint of S_2 such that these paths do not pass through the other endpoints of S_1 and S_2,*

2. *no segment of $\mathcal{Q}$ has both endpoints on P or on P', and*

3. *the closed-interior of the cycle $P \cup S_1 \cup P' \cup S_2$ does not contain the disk D_0.*

A C_j-type of segment is an equivalence class of segments of $\mathcal{Q}$ under the relation $\parallel$.

Let $\mathcal{S}$ be a set of segments of $\mathcal{Q}$ having the same C_j-type. We can define a partial order $\prec$ on $\mathcal{S}$ as follows: we say that $S_a \prec S_b$ if and only if S_a lies in the zone of S_b. See Figure 9b for a reference.

In this section, we will establish that a cheap T-loop cannot have "many" segments of the same type in a convex CL-configuration. Let P (resp., S) is a path (resp., segment) of G with endpoints u and v. Then, $int(P)$ (resp., $int(S)$) denote the set $V(P) \setminus \{u, v\}$ (resp., $V(S) \setminus \{u, v\}$). We need the following notion of *out-segments* to proceed further.

Definition 16 (**Out-segments**). *Consider a convex CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$ of depth r and let $\mathcal{S} = \{S_1, \ldots S_p\}$ be a set of segments of the same C_j-type ($j \in [r]$) such that $S_{i+1} \prec S_i$ and $p > 3$. Then, the $\mathcal{S}$-out-segments are the connected components of $L \setminus (int(S_1) \cup int(S_2) \cup int(S_3))$. Observe that, for $i \in [3]$, $|V(S_i) > 2|$ due to the definition of convex segments, and hence $|int(S_i)| > 1$. Since we remove three disjoint, connected, and non-empty subparts of a cycle, we have three $\mathcal{S}$-out-segments, and let them be denoted by O_1, O_2, O_3. When $\mathcal{S}$ is clear from the context, we refer to a $\mathcal{S}$-out-segments as simply an out-segment. An $\mathcal{S}$-out-segment O with distinct endpoints x, y is referred as x, y-out-segment. See Figure 10 for an illustration.*
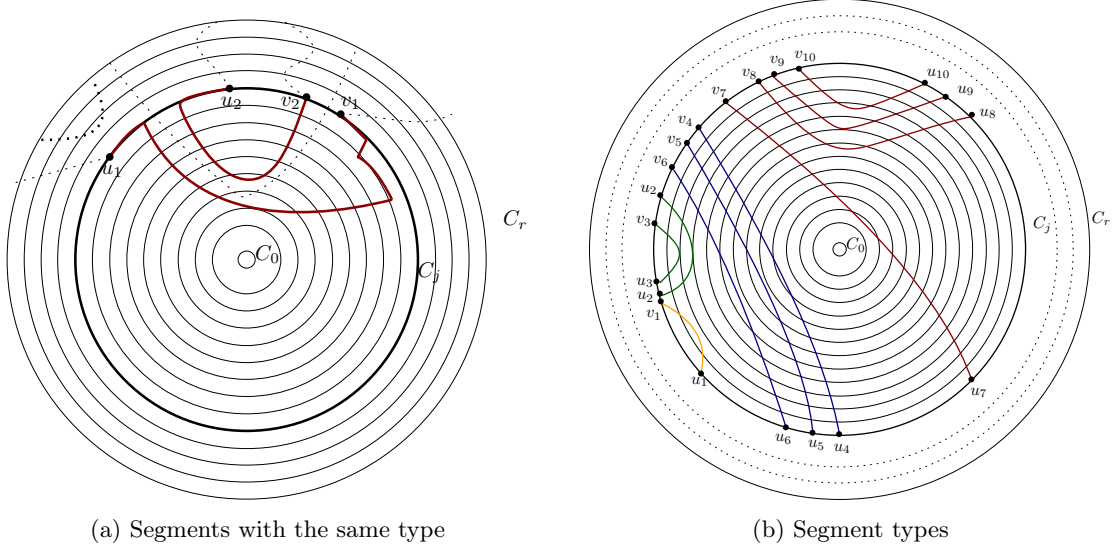
(a) Segments with the same type   (b) Segment types

Figure 9: (a) Here S_1 and S_2 are C_j-segments with endpoints u_1, v_1 and u_2, v_2, respectively. Moreover, P and P' are the (u_1, u_2)-path and (v_1, v_2)-path along C_j, respectively. Here $S_1 \parallel S_2$. (b) (i)Here, segments $S_1, \ldots, S_{10}$ are C_r-segments such that segment S_i has endpoints u_i and v_i. The segments S_4 and S_7 are not of the same C_r-type because of condition (3) in Definition 15 and S_6 and S_2 are not of the same C_r-type because of condition (2) in Definition 15. Further notice that when S_6 and S_2 are restricted to D_{r-2}, they have the same C_{r-2}-type. (ii) Here, $S_{10} \prec S_9 \prec S_8 \prec S_7$, and similarly, $S_6 \prec S_5 \prec S_4$

Now, we present some easy observations concerning segments and out-segments that will shall be helpful for us later. We begin by observing that if some cycle C contains all $\mathcal{S}$-out-segments, then C is a T-loop indeed.

Observation 17. *Let $\mathcal{Q} = (\mathcal{C}, L)$ be a depth r CL-configuration of G and let $\mathcal{S} = \{S_1, S_2, S_3 \ldots, S_p\}$ be a set of the segments of the same C_j-type $(j \in [r])$. Further, let $O_1, \ldots, O_3$ be the $\mathcal{S}$-out-segments. Then, if there exists a cycle C such that O_1, O_2, O_3 are connected subgraphs of C, then C is a T-loop.*

Proof. Since $V(L) = V(S_1) \cup V(S_2) \cup V(S_3) \cup V(O_1) \cup V(O_2) \cup V(O_3)$ (by definition of out-segments) and $T \cap (V(S_1) \cup V(S_2) \cup V(S_3)) = \emptyset$ (by definition of CL-configurations), it follows that $T \subseteq V(O_1) \cup V(O_2) \cup V(O_3)$. Hence, if some cycle C contains O_1, O_2, O_3 as subgraphs, then C is a T-loop. $\square$

Next, we observe a rather straightforward fact.

Observation 18. *Consider a convex CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$ of depth r and let $\mathcal{S} = \{S_1, \ldots, S_p\}$ be the segments of the same C_j-type such that $p > 3$ and $S_p \prec \ldots \prec S_1$. Then, $c(S_2) > 0$.*

Proof. Since S_2 and S_3 are convex segments of the same C_j-type and $S_3 \prec S_2$, it follows that S_2 contains a vertex of the cycle C_{j-1} as well (as otherwise the $\mathcal{C}$ is not tight), and hence S_2 contains at least one edge uv such that at most one of u, v is in C_j, i.e., $|\{u, v\} \cap V(C_j)| \le 1$, and hence $c(S_2) \ge c(uv) \ge 1$. $\square$

Now, we prove the main lemma of this section that establishes that in a cheap solution, there cannot be many segments of the same type.

Lemma 19. *Let $\mathcal{Q} = (\mathcal{C}, L)$ be a depth r convex CL-configuration of G. Moreover, let $\mathcal{S}$ be a set of segments of the same C_j-type, for some $j \in [r]$. If $|\mathcal{S}| > 3$, then L is not a $\mathcal{C}$-cheap T-loop.*

Proof. Suppose that $\mathcal{S} = \{S_1, \ldots, S_p\}$ such that for $i \in [p-1]$, $S_{i+1} \prec S_i$. Let S_i intersects C_j at vertices u_i and v_i, i.e., the vertices in $\{u_1, \ldots, u_p, v_1, \ldots, v_p\}$ appear on C_j in the order $(u_1, \ldots, u_p, v_p, \ldots, v_1)$ (or the reverse order). We note that possibly $u_p = v_p$, i.e., the segment S_p intersects C_j at a single vertex, say u. See Figure 9b for an illustration. If $p \le 3$, then our claim is vacuously true. So, we assume that $p > 3$. Let O_1, O_2, O_3 be the $\mathcal{S}$-out-segments. Since $\mathcal{S}$ is clear from the context, we will refer to $\mathcal{S}$-out-segments as simply out-segments to ease the notation.
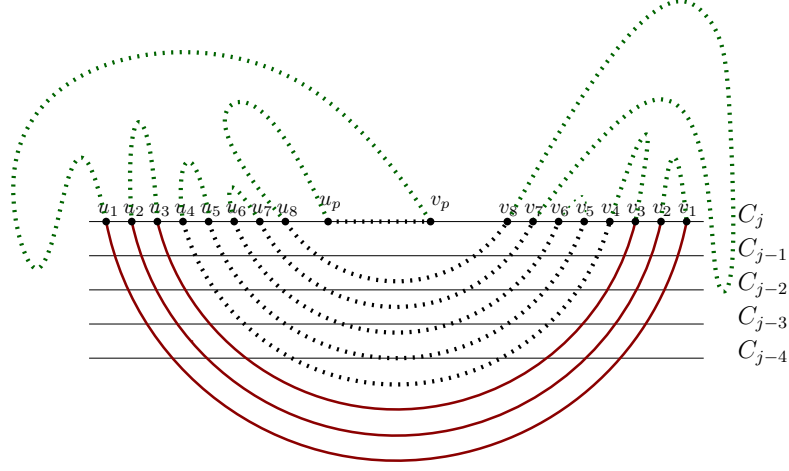
16

Figure 10: Here $\mathcal{S} = \{S_1, \ldots, S_8\}$ are segments of the same C_j-type such that $S_8 \prec \cdots \prec S_1$ and S_i has endpoints u_i and v_i. The segments S_1, S_2, and S_3 are marked in red. Other segments are marked in dotted black lines. All out-segments are represented as dotted curves and note that all segments other than S_1, S_2, S_3 are a subpart of the u_1, v_3-out-segment.

Further, we shall use the following terminology. For two consecutive vertices x, y in the sequence $(u_1, \ldots, u_p, v_p, \ldots, v_1)$, let (x, y)-C_j-path denote the (x, y)-path along C_j that does not contain any vertex from $\{u_1, \ldots, u_p, v_1, \ldots, v_p\} \setminus \{x, y\}$. We begin with establishing the following useful claim.

Claim 20. *For $i \in [3]$, no out-segment can have u_i, v_i as its endpoints, no out-segment can have u_1, v_2, and symmetrically, v_1, u_2 as its endpoints. Further, either u_1, u_2 or v_1, v_2 are endpoints of an out-segment.*

Proof of Claim. First, observe that if some out-segment, say O, is a (u_i, v_i)-out-segment, then the segment S_i along with the out-segment O induces a cycle, say C, that does not contain any segments other that S_i. Since $V(C) \subsetneq V(L)$ and $E(C) \subsetneq E(L)$, and C is a cycle, L cannot be a cycle. Second, note that if there is a (u_1, v_2)-out-segment O, then there is no way to connect v_1 to the cycle due to planarity and the definition of same segment type. The proof for the fact that there is no (v_1, u_2)-out-segment is symmetric.

Finally, observe that each vertex from $\{u_1, u_2, u_3, v_1, v_2, v_3\}$ can participate in at most one out-segment since each of these vertices is participating in a segment as well, and the degree of each vertex in a cycle is 2. Hence, there can be at most two out-segments with endpoints from $\{u_3, v_3\}$, and thus, there should be at least one out-segment, say O, with both its endpoints from $\{u_1, u_2, v_1, v_2\}$. Since O cannot have u_1, v_1 and u_2, v_2 as its endpoints, and O cannot have u_1, v_2 and v_1, u_2 as its endpoints (as proved above), the only two possible endpoint pairs for O are u_1, u_2 and v_1, v_2. Hence, O has either u_1, u_2 as endpoints or v_1, v_2 as endpoints. This completes our proof. ◇

Due to Claim 20, we know that either u_1, u_2 or v_1, v_2 are endpoints of the same out-segment, say O. Without loss of generality, let us assume that O is a (v_1, v_2)-out-segment. We will construct a T-loop L' such that $c(L') < c(L)$. See Figure 11 for an illustration. Let x be a vertex of S_3 such that (i) x is on the cycle C_{j-1}, (ii) and there exists a (x, v_3)-path P along S_3 such that each vertex of P has eccentricity at least $j - 1$. Similarly, let y be a vertex of S_1 such that (i) y is on cycle C_{j-1}, (ii) and there exists a (y, v_1)-path P' along S_1 such that each vertex of P' has eccentricity at least $j - 1$. Finally, let P'' be the (x, y)-path along C_{j-1} such that the interior of P'' lies completely in the interior of the zone of the segment S_1. We will obtain L' from L by performing the following operations (while performing the operations, the subgraphs may be disconnected). See Figure 12 for an illustration.

- Remove the subpaths corresponding to the segments S_1 and S_2.

- Add the (u_1, u_2)-C_j-path.

- Remove the path P and add the paths P' and P''.
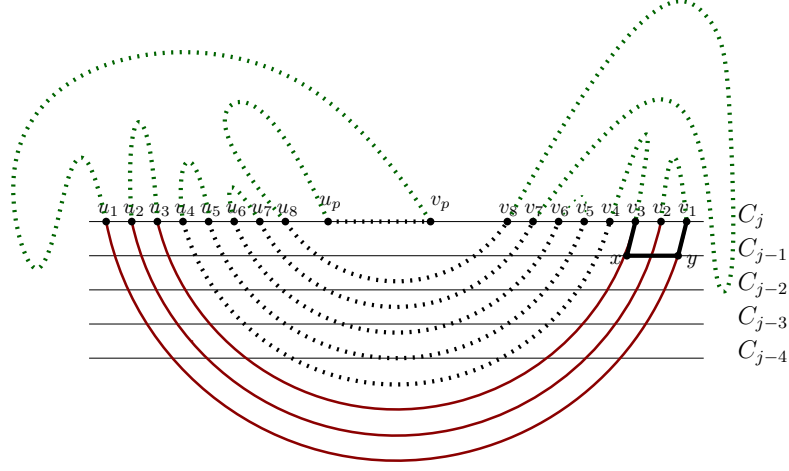
- Finally, add the (v_2, v_3)-C_j-path.

17

Figure 11: x, v_3-path marked in bold black is P, y, v_1-path in bold is P', and x, y-path in bold is P''.
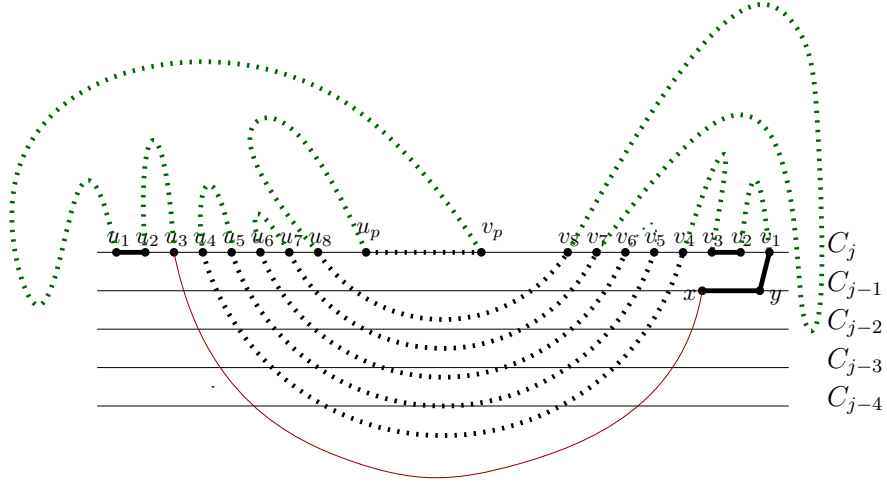

Figure 12: Construction of L' from L.

First, it is not difficult to see that L' is a cycle. To see this, first observe that L' has a path from u_1 to u_2. Also, L' has a path from v_3 to v_2, from v_2 to v_1 and from v_1 to u_3 (and hence a path from v_3 to u_3). From Claim 20, there is no out-segment from u_3 to v_3 in L. Hence, apart from the v_1,v_2-out segment, the other two out-segments in L have endpoints either (i) u_1, u_3 and (ii) u_2, v_3 or (i) u_1, v_3 and (ii) u_2, u_3, both of which are preserved by L'. If follows that every vertex in L' has degree 2 and L' is a connected graph.

Next, observe that since we do not remove any out-segment from L to obtain L', L' is a T-loop (Observation 18). Finally, observe that since we remove all the edges of S_2 that do not intersect C_{j-1}, (in particular, we remove all the edges of S_2 that contribute to $c(L)$) (Observation 17) and only add paths either along the cycle C_j ((u_1, u_2)-C_j-path and (v_2, v_3)-C_j-path) or along the cycle C_{j-1} ((x, y)-path along C_{j-1}), which have cost 0 (by definition), we have that $c(L') < c(L)$. This completes our proof. □

4.2 Logarithmic depth

The goal of this section is to show that unless there is an irrelevant vertex, the eccentricity of any segment of a CL-configuration is at least $r - O(\log k)$. To this end, we begin this section by defining ancestor-descendant and parent-child relationship between segments.

Definition 21 (Ancestor and descendant segments). *Let $\mathcal{Q} = (\mathcal{C}, L)$ be a convex CL-configuration of G of depth r, let $j \in [r]$, and let S and T be C_j-segments of $\mathcal{Q}$. We say that a segment S is an ancestor of T (equivalently, T is a descendant of S) if $T \prec S$, that is, if T is in the zone of S.*

Note that if S is the ancestor of T, then the eccentricity of S is less than or equal to that of T.

18

Definition 22 (**Chain of segments**). *Let $\mathcal{Q} = (\mathcal{C}, L)$ be a convex CL-configuration of G of depth r, and let $j \in [r]$. We say that there is a* chain *of C_j-segments of size i from segment S to segment T if there exist segments $S_0 = S, S_1, \ldots, S_i = T$ of $\mathcal{Q}$ such that for any $j, k \in [0, i]$, we have $j < k$ if and only if S_k is in the zone of S_j, and for any consecutive segments S_{k-1} and S_k for $k \in [i]$, there does not exist a segment S' of $\mathcal{Q}$ such that S' is an ancestor of S_k and S_{k-1} is an ancestor of S'.*

Definition 23 (**Generation of ancestor and descendant segments, Parent and child segments**). *Let $\mathcal{Q} = (\mathcal{C}, L)$ be a convex CL-configuration of G of depth r, and let $j \in [r]$. We say that a C_j-segment S is an (i-th generation) ancestor of a C_j-segment T if there is a chain of size i from S to T. We say that a C_j-segment P is a* parent *of a C_j-segment Q (equivalently Q is a* child *of a segment P) if P is a first generation ancestor of Q.*

Next, we define the notions of a segment forest of a CL-configuration, the height of a segment in the segment forest, and width k subtrees of the segment forest.

Definition 24 (**Segment forest**). *Given a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$ of depth r, and some $j \in [r]$, the j-th* segment forest *is a forest $F = (V, E)$, where V is the set of C_j-segments of $\mathcal{Q}$ and the edge set E models parent-child relationsip of C_j-segments.*
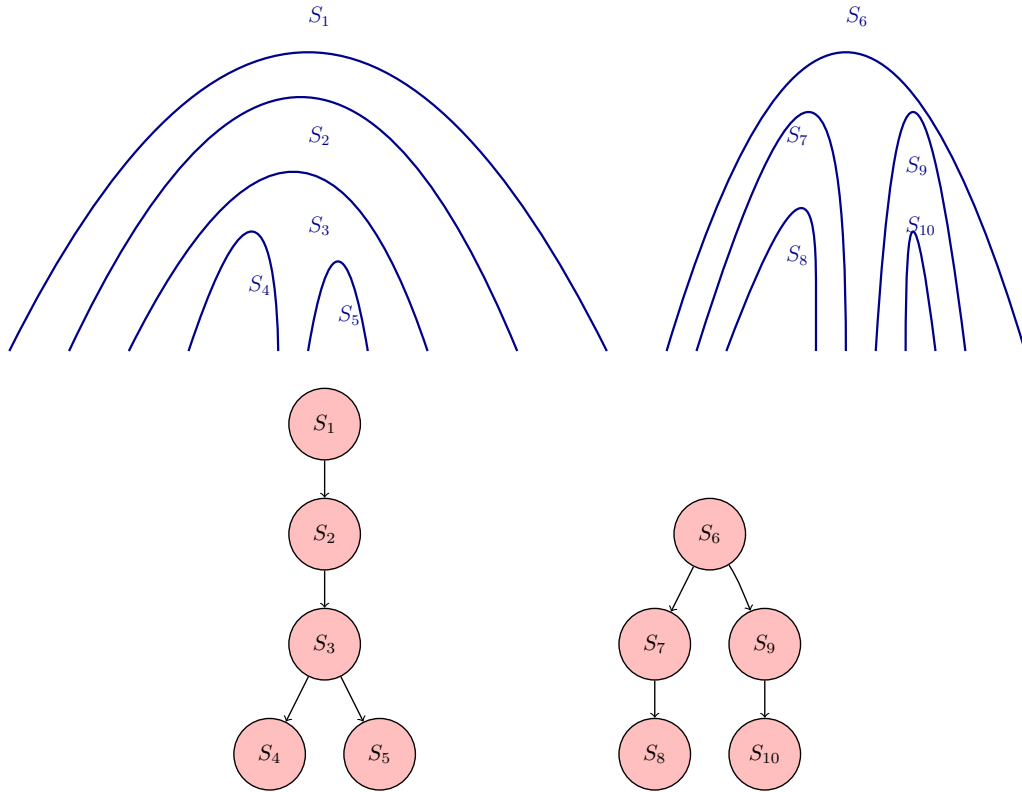

Figure 13: The top subfigure shows segments of a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$. This gives rise to the segment forest shown in the bottom subfigure. The directed edges depict parent child relationship. The height of leaves S_4, S_5, S_8 and S_{10} is 0. The height of S_1 is 3 and the height of S_6 is 2.

Definition 25 (**Height**). *Let $\mathcal{Q} = (\mathcal{C}, L)$ be a CL-configuration. Let s be the size of the maximum chain in a segment forest of $\mathcal{Q}$ from a leaf node to a segment S. Then, the* height *of S in that segment forest is equal to $s - 1$. The height of a leaf node is zero. The height of a segment forest F is the maximum height of a segment in F.*

We refer the reader to Figure 13 for an example of a segment forest.

Note that a segment has at most one parent, and every segment that is not a leaf in a segment forest has at least one child.

Definition 26 (**Width m j-subtree rooted at a segment**). *For a C_j-segment P of a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$, the subtree of the segment forest of $\mathcal{Q}$ induced by the vertex corresponding*
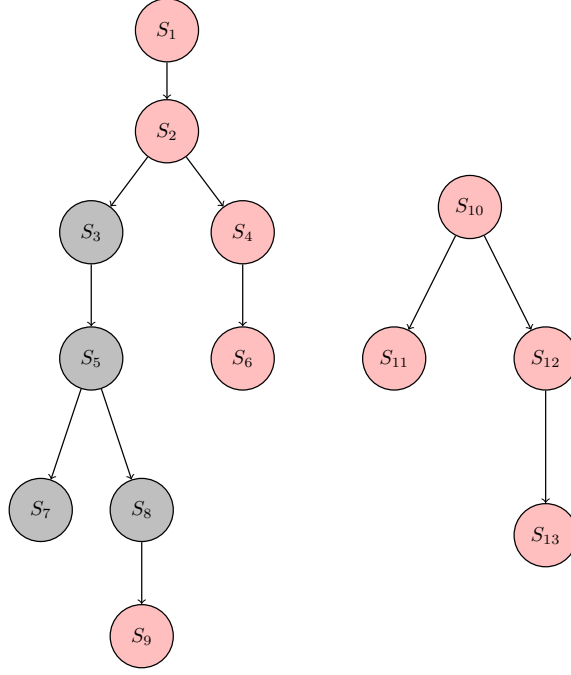
Figure 14: Here, we show a segment forest. The width 2 subtree rooted at the second generation ancestor of S_8 (which is S_3) is shown in grey.

to segment P along with all C_j-segments that are i-th generation descendants of P for all $i \in [m]$ is called the width m j-subtree rooted at P.

Lemma 27. *Let $\mathcal{Q} = (\mathcal{C}, L)$ be a convex CL-configuration of G. Let e be the eccentricity of a segment S. For any given m, suppose that S has a k-th generation ancestor that we denote by U. Let T be the width m e-subtree rooted at U. Then,*

a. either all ancestors of S in T have the same C_e-type

b. or there exists at least one other segment $R \neq S$ in T that is not an ancestor of S.

Proof. Let A be an ancestor of S that does not have the same C_e-type. Since A is an ancestor, there exist two paths P and P' on C_e connecting an endpoint of S with an endpoint of A that do not pass through the other endpoint of A. Since A does not have the same C_e type as S there exists a segment R with both endpoints in P (or both endpoints in P'). Such a segment cannot be an ancestor of S, proving the claim. $\qquad\square$

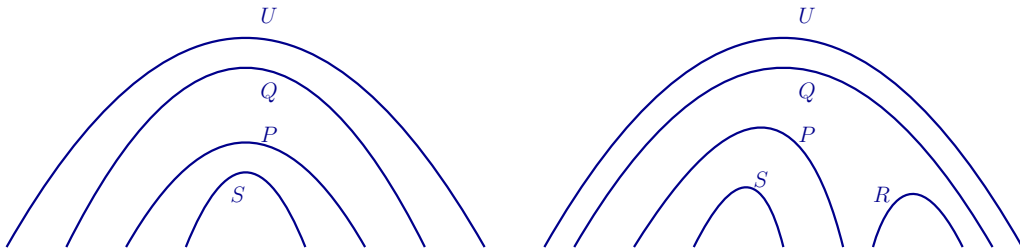The two cases of Lemma 27 are depicted in Figure 15.

Figure 15: In both subfigures, S is a segment of eccentricity e, for some e, and U is its 3-rd generation ancestor. In the left subfigure, all ancestors of S up to U have the same C_e-type, whereas in the right subfigure Q does not have the same C_e type, and this is because of the existence of another segment R in the subtree rooted at U that is not an ancestor of S and has eccentricity less than or equal to e.

Lemma 28. *Consider a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$ of a planar graph G. Let e be the eccentricity of a segment S. Suppose that S has a 3-rd generation ancestor that we denote by U. Let T be the width 3 e-subtree rooted at U. Then, there exists one other segment $R \neq S$ that is not an ancestor of S with eccentricity less than or equal to e.*

Proof. Towards a proof by contradiction, suppose that there is no such segment R in T, then we are in case (a.) of Lemma 27. Therefore, S and its three ancestors have the same C_e type. So, there exist four segments of the same C_e-type, which contradicts Lemma 19. $\qquad\square$

Theorem 29. *Let h be the height of a segment S and let N be the number of segments in the subtree rooted at S in the r-th segment forest of a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$ of depth r. Then, $h \leq \log_a N + 3$, where $a = 2^{\frac{1}{3}}$.*

Proof. We use induction to prove the claim. For the base case, for every $h \leq 3$, it follows that $h \leq \log_a N + 3$. Now, suppose that the inequality holds true for all segments of height less than $H \geq 4$. We will show that it is also true for segments of height H. Let S be a segment of height H and let N be the number of segments in the subtree rooted at S. We have $3 = \log_a 2$, which gives

$$H = \log_a(2) + H - 3$$
$$= \log_a(2a^{H-6}) + 3$$

By induction hypothesis, the subtrees rooted at segments of height $H - 3$ have at least a^{H-6} nodes. Then, by Lemma 28, there exist at least two segments of height $H - 3$ in the width 3 subtree rooted at S. Hence, $N \geq 2a^{H-6}$, proving the claim. $\qquad\square$

Corollary 30. *There exist constants c_1 and c_2 such that in a CL-configuration $\mathcal{Q} = (\mathcal{C}, L)$ of depth r, if the eccentricity of a segment is less than $r - (c_1 \log k + c_2)$, then there exists an irrelevant vertex in $\mathcal{Q} = (\mathcal{C}, L)$.*

Proof. From [4, Lemma 5], we know that a cheap solution for the DISJOINT PATHS problem has $O(k)$ different types of segments, where k is the number of terminal pairs in the DISJOINT PATHS instance. Note that the above theorem is true irrespective of the choice of the pairing $\{(s_1, t_1), \ldots, (s_k, t_k)\}$. In particular, this is true for all pairings which have $t_i = s_{i+1}$ for $i \in [k-1]$ and $t_k = s_0$. That is, it is true for all possible pairings that are needed for the T-CYCLE problem. Hence, the T-CYCLE problem has $O(k)$ different types of segments.

Since there is a constant number of segments of the same type, Theorem 29 implies that the height of the r-th segment forest is $O(\log k)$. The claim of the corollary follows. $\qquad\square$

As mentioned in Section 1, it is easy to get a $2^{O(\sqrt{k} \log k)} \cdot n^2$ time algorithm for T-CYCLE from here. Indeed, since all $c \log k$-isolated vertices are irrelevant (Corollary 30), we can, as long as possible, compute a $c \log k \times c \log k$ grid minor that does not contain any terminal, and remove its "middle-most" vertex. Since each iteration of this procedure requires time $k^{O(1)} \cdot n$, and $O(n)$ iterations are performed in total, this process will take $k^{O(1)} \cdot n^2$ time in total. Now, since the reduced graph now has treewidth $O(\sqrt{k} \log k)$ (follows from Proposition 11), a $2^{tw} \cdot n$ time algorithm for T-CYCLE coupled with above arguments, gives us a $2^{O(\sqrt{k} \log k)} \cdot n^2$ time algorithm for T-CYCLE. In the following section, we optimize the irrelevant vertices removal and remove "sufficiently many" irrelevant vertices in time linear in n.

5 Fast Removal of Irrelevant Vertices

In this section, we will provide a linear-time algorithm to compute and remove some irrelevant vertices such that the remaining graph has bounded treewidth. Our work builds on techniques developed by Reed [37] to solve k-REALIZATIONS in linear time, which were later used to obtain a linear-time algorithm for PLANAR DISJOINT PATHS [13]. To this end, we first need the notion of *punctured planes*.

Definition 31 (Punctured Plane). *A c-punctured plane $\boxdot$ is the region obtained by removing c open holes from the plane. The boundary of $\boxdot$ is the union of the boundaries of its holes. The vertices on the boundary of $\boxdot$ are called the* boundary vertices.
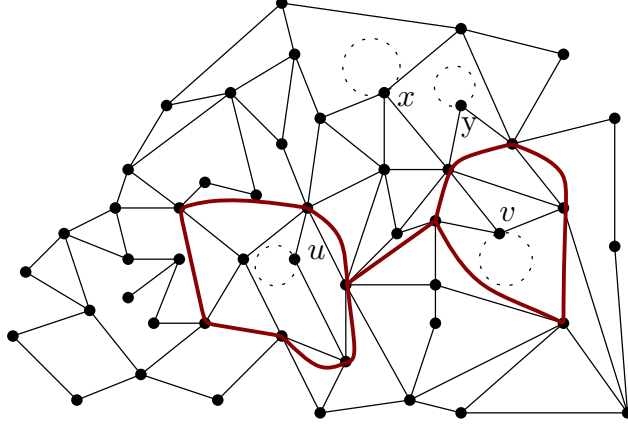
Figure 16: Here, originally G is embedded on a 4-punctured plane where each puncture is illustrated with dotted circle. The cut is illustrated in red, and after the cut reduction, we have three punctured planes: (a.) a 2-punctured plane with a puncture incident to vertex u; (b.) a 2-punctured plane with a puncture incident to vertex v; and (c.) a 3-puncture plane with punctures incident to vertices x, y. For all three punctured planes one of the punctures is given by the cut.

Let (G, T) be an instance of T-CYCLE such that G is a planar graph and $T \subseteq V(G)$ is the set of terminals. Observe that we can get a k-punctured plane $\boxdot$ from G by considering each point where a terminal lies as a (trivial) hole.

Preliminaries. A curve on the plane is called *proper* if it intersects the (planar) graph G only at its vertices. For two vertices $u, v \in V(G)$, recall that $\mathsf{d}^\mathsf{R}(u, v)$ denotes the radial distance between u and v. Similarly, for $X, Y \subseteq V(G)$, let $\mathsf{d}^\mathsf{R}(X, Y) = \min_{x \in X, y \in Y} \mathsf{d}^\mathsf{R}(x, y)$, and let $\mathsf{d}^\mathsf{R}(x, Y) = \mathsf{d}^\mathsf{R}(\{x\}, Y)$. Recall that there exists some $g(k) \in O(\log k)$ such that each $g(k)$-isolated vertex is irrelevant (Corollary 30), which can be explicitly derived from the proof of Corollary 30. A subgraph H of G embedded on a c-punctured plane $\boxdot$ is *nice* if any cycle separating a vertex v of H from the boundary of $\boxdot$ also separates v from T in G. A vertex $v \in V(H)$ is ℓ-*boundary-isolated in* $\boxdot$ if there is a sequence of ℓ concentric cycles that separates v from boundary of $\boxdot$. Observe that if a vertex $v \in V(H)$ is ℓ-boundary-isolated in $\boxdot$ and H is a nice subgraph, then v is ℓ-isolated in G (from T).

Reed [37] proved the following result, which is an essential component of our algorithm.

Proposition 32 (Lemma 2 in [37]). *Let H be a nice subgraph of G embedded on a c-punctured plane $\boxdot$, and let Y be the set of all $g(k)$-boundary-isolated vertices of H in $\boxdot$. If $c \geq 3$, then we can compute both a non-crossing proper closed curve J contained in $\boxdot$ and a (possibly empty) subset $X \subseteq Y$ in $O(|V(H)|)$ time such that*

1. *the number of vertices of $H - X$ intersected by J is at most $6g(k) + 6$, and*

2. *$\boxdot \setminus J$ contains at most three connected components each of which has less than c holes.*

In each application of Proposition 32, we get at most three components, each of which contains a reduced number of punctures than the input. We call each such application of Proposition 32 a *cut reduction*. See Figure 16 for an illustration of cut reduction. A straightforward corollary of Proposition 32 is that we can decompose G into $O(k)$ many subgraphs, each embeddable on a 2-punctured plane with a boundary of size $O(k \log k)$ in time linear in n.

Corollary 33. *Let G be a planar graph and $T \subseteq V(G)$ be a set of terminals. Then, in $O(k \cdot n)$ time, after removing some $g(k)$-isolated vertices from G, we can decompose G into $O(k)$ nice subgraphs, each embedded on either a 2-punctured or a 1-punctured plane, and the total number of boundary vertices in all of these punctured planes, combined, is $O(k \log k)$. '*

Proof. We begin with a k-punctured planar graph G, and in each application of the *cut reduction* procedure of Proposition 32 on a c-punctured plane (for $c > 3$), we obtain at most three instances, each on a $c-1$-punctured plane. As pointed out by Reed in Section 3.1 in [37], a careful analysis shows that we consider at most $4k + 8$ subproblems, and hence we apply cut reduction at most $4k + 8$ times. Hence, we decompose G into $4k + 8 \in O(k)$ nice subgraphs. Next, observe that in each application

22

of the cut reduction procedure of Proposition 32, we obtain a proper curve intersecting H in at most $6g(k) + 6 \in O(\log k)$ vertices, and hence, in each cut reduction, $O(\log k)$ vertices of G are marked as boundary vertices. Since we apply cut reduction only $4k + 8$ times, the number of total newly introduced boundary vertices are $O(k \log k)$. Finally, since each application of the cut-reduction from Proposition 32 takes $O(n)$ time and we apply this procedure $O(k)$ times, the total running time of our procedure takes $O(k \cdot n)$ time. Our claim follows. $\qquad\square$

Deleting Irrelevant Vertices on a 1-punctured plane. The next component of our algorithm is a procedure to remove all $g(k)$-boundary-isolated vertices from a graph H embedded on a 1-punctured plane in linear time. We note that this approach was also considered in [13] to obtain a linear time algorithm for PLANAR DISJOINT PATHS.[3] Let H be a nice subgraph of G embedded on a 1-punctured plane $\boxdot$, and let $V_0 \subseteq V(H)$ be the set of vertices that lie on the boundary of $\boxdot$. Further, for $i > 0$, let $V_i = \{v \mid v \in V(H)\ \&\ \mathsf{d}^{\mathsf{R}}(v, V_0) = i\}$. Note that we can partition $V(H)$ into $V_0, \dots, V_\ell$ in $O(|V(H)|)$ time using standard data structures representing an embedding for planar graphs (*doubly connected edge list*) [15]. The following result from [13] establishes that we can remove all vertices of V_j, where $j > g(k)$.

Proposition 34 ([13]). *Let H be a nice subgraph of G embedded on a 1-punctured plane $\boxdot$, and let V_0 be the set of boundary vertices of H. Moreover, for $i > 0$, let $V_i = \{x \mid x \in V(H)\ \&\ \mathsf{d}^{\mathsf{R}}(x, V_0) = i\}$. Then, for $v \in V_i$:*

1. *v is $(i-1)$-boundary-isolated in $\boxdot$,*

2. *and no sequence of i concentric cycles exists in G that separates v and V_0.*

Hence, we can use Proposition 34 to remove all $g(k)$-boundary-isolated vertices from a 1-punctured plane $\boxdot$ in linear time by computing a partition of $V(H)$ based on radial distance.

Handling 2-punctured planes. Since Corollary 33 helps us to obtain only 2-punctured planes and Proposition 34 is designed to handle 1-punctured planes, we need either a way to further apply some cut reduction on 2-punctured planes to get 1-punctured planes or directly get a way to remove irrelevant vertices from 2-punctured planes. To this end, we will use the following lemma, the proof of which was provided to us by Cho, Oh, and Oh [13] in a personal communication. We provide a proof for the sake of completeness, but we want to stress that the credit of the proof is to Cho, Oh, and Oh.

Proposition 35. *Let H be a nice subgraph of G embedded on a 2-punctured plane $\boxdot$. Then, we can remove all $4g(k)$-boundary-isolated vertices from $\boxdot$ in $O(|V(H)|)$ time.*

Proof. Let H be a nice subgraph of G embedded on a 2-punctured plane $\boxdot$. Let the two boundaries of $\boxdot$ be H_1 and H_2. We note that we can assume that H_1 is contained inside H_2 without loss of generality.

Let A be a shortest simple curve joining a vertex v_1 of H_1 and a vertex v_2 of H_2. If A intersects H at most $6g(k)$ vertices, then we can cut along A to get a new 1-punctured plane with $O(k \log k)$ boundary vertices, and then we can apply Proposition 34 to remove the $g(k)$-isolated vertices. Hence, now we assume that $|A| > 6g(k)$.

Now, let us cut along A to get a new punctured plane $\boxdot_A$. Note that the size of boundary of $\boxdot_A$ can be unbounded by any function of k. Now, in $\boxdot_A$, curve A corresponds to two curves, say A_1 and A_2. See Figure 17 for a reference. For a vertex $v \in V(H)$, let $\mathsf{d}_i^{\mathsf{R}}(v) = \mathsf{d}^{\mathsf{R}}(v, V(A_i))$, for $i \in [2]$. Let R be the set of vertices in H such that for each vertex $v \in R$, $|\mathsf{d}_1^{\mathsf{R}}(v) - \mathsf{d}_2^{\mathsf{R}}(v)| \leq 1$. It is easy to see that there is a simple curve B in $\boxdot$ connecting H_1 and H_2 such that $V(B) \subseteq R$.

Let $\boxdot_A$ and $\boxdot_B$ be the 1-punctured planes obtained by cutting $\boxdot$ along the simple curves A and B, respectively. Now, we apply the following procedure to remove some irrelevant vertices from H.

1. Use Proposition 34 on $\boxdot_A$ to remove all $g(k)$-boundary-isolated vertices from $\boxdot_A$ in H.

2. Use Proposition 34 on $\boxdot_B$ to remove all $g(k)$-boundary-isolated vertices from $\boxdot_B$ in H.

---

[3]In [13] and [37], $g(k) \in 2^{O(k)}$. We note that although we fix $g(k) \in O(\log k)$ to ease the presentation, all of our results also generalize for any $g : \mathbb{N} \to \mathbb{N}$.
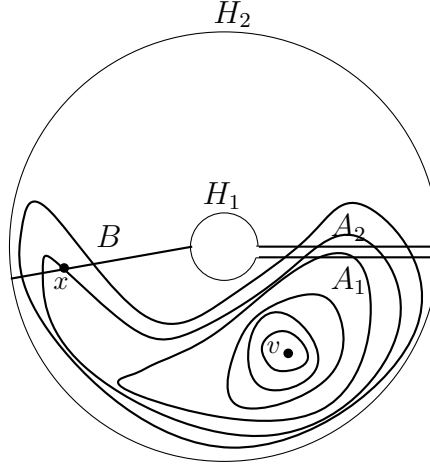
Figure 17: Illustration for the Proof of Proposition 35

Notice that each vertex removed in the above procedure is a $g(k)$-isolated vertex in $\boxdot$ in H as well (since each vertex on the boundary of $\boxdot$ is also a boundary vertex in $\boxdot_A$ as well as in $\boxdot_B$). But it might so happen that there are still many $g(k)$-boundary-isolated vertices in $\boxdot$ (i.e., the above procedure have not removed all $g(k)$-boundary-isolated vertices from $\boxdot$). We have the following crucial claim which proves that the above procedure removes all $4g(k)$-boundary-isolated vertices from $\boxdot$.

Claim 36. *Let H' be the graph embedded in $\boxdot$ obtained after removing $g(k)$-boundary-isolated vertices from $\boxdot_A$ and then removing $g(k)$-boundary-isolated vertices from $\boxdot_B$. Then, no vertex of H' is $4g(k)$-boundary-isolated in $\boxdot$.*

Proof of Claim. Targeting a contradiction, let $v \in V(H')$ be a vertex that is $4g(k)$-boundary-isolated in $\boxdot$. Then, there exists a vertex v and a sequence of $4g(k)+1$ concentric cycles, say $\mathcal{C} = \{C_0, \ldots C_{4g(k)}\}$, with the following properties:

1. $v \in D_0$ (recall that $D_i = int(C_i)$),

2. $V(A) \cap D_{g(k)} \neq \emptyset$ and $V(B) \cap D_{g(k)} \neq \emptyset$ (since no vertex in H' is $g(k)$-boundary-isolated in $\boxdot_A$ and in $\boxdot_B$), and

3. $(V(H_1) \cup V(H_2)) \cap D_{4g(k)} = \emptyset$.

First, we note that each cycle in $\mathcal{C}$ intersects A at most twice, as otherwise, it contradicts the fact that A is a shortest simple curve joining H_1 and H_2. Now, due to (2), the radial distance between v and A is less than $g(k)$, i.e., $\mathsf{d}^{\mathsf{R}}(v, V(A)) < g(k)$. Hence, WLOG, we can assume that $\mathsf{d}^{\mathsf{R}}_1(v) < g(k)$. Therefore, $\mathsf{d}^{\mathsf{R}}_2(v) > 3g(k)$ since to go from v to A_2, each radial path needs to cross all the concentric cycles in $\mathcal{C}$ (as no cycle in $\mathcal{C}$ contains H_1, no cycle intersects A more than twice). Again, due to (2), we have that the radial distance between v and some vertex, say x, of the curve B is at most $g(k)-1$, i.e., $\mathsf{d}^{\mathsf{R}}(v, x) \leq g(k) - 1$. Therefore, the radial distance between x and A_1 is at most $2g(k)-2$, i.e., $\mathsf{d}^{\mathsf{R}}_1(x) \leq 2g(k) - 2$. Thus, since x is a vertex on B, $\mathsf{d}^{\mathsf{R}}_2(x) \leq 2g(k) - 1$. Hence, $\mathsf{d}^{\mathsf{R}}_2(v)$ is at most the radial distance from v to x, and from x to A_2, i.e., $\mathsf{d}^{\mathsf{R}}_2(v) \leq 2g(k) - 1 + g(k) - 1 \leq 3g(k) - 2$. This contradicts the fact that $\mathsf{d}^{\mathsf{R}}_2(v) > 3g(k)$, completing the proof that no vertex in H' in $\boxdot$ is $4g(k)$-isolated. ◇

Hence, we have a linear-time procedure to remove all $4g(k)$-isolated vertices from $\boxdot$. □

Due to Corollary 33, we have a decomposition of G into $O(k)$ subgraphs, each embedded on either a 2-punctured plane or a 1-punctured plane. Moreover, due to Proposition 34 and Proposition 35 none of the $O(k)$ subgraphs contains a vertex inside them that is $4g(k)$-boundary-isolated. Moreover, the total number of vertices on the boundary of these punctured planes is $O(k \log k)$. The final step of our algorithm is to remove some irrelevant vertices from the boundaries of the punctured planes to reach a state where none of the vertices in G is "too much" isolated. For this purpose, we begin by proving the following lemma.
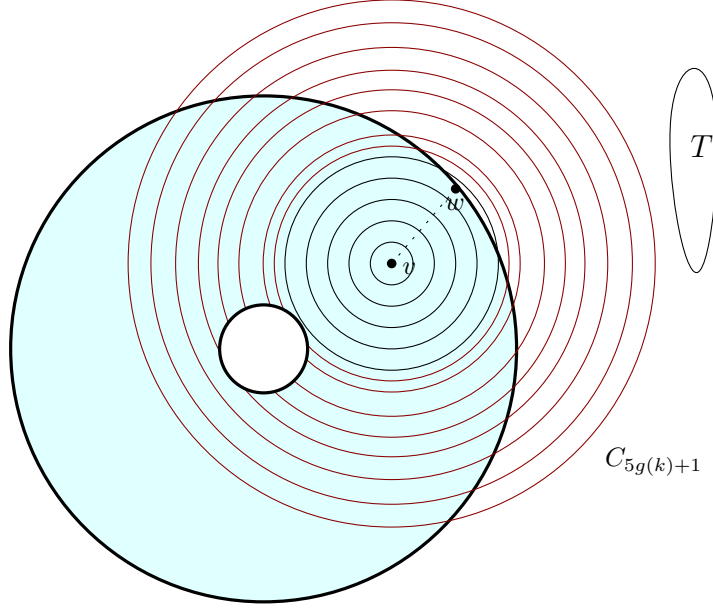
24

Figure 18: A 2-punctured plane $\boxdot$ is depicted (in blue) by its two holes illustrated in bold black, and v is a vertex in $\boxdot$ which is $5g(k) + 1$-isolated in G. A sequence $C_0, \ldots, C_{5g(k)+1}$ of concentric cycles separating v and T is illustrated such that cycles $C_0, \ldots, C_{4k}$ are illustrated in black and cycles $C_{4g(k)+1}, \ldots, C_{5g(k)+1}$ are illustrated in red.

Lemma 37. *For $v \in V(G)$, we can decide in $O(n)$ time whether v is ℓ-isolated (from T in G).*

Proof. Start a radial breadth-first search (BFS) from v. Observe that v is ℓ-isolated if and only if for each vertex $x \in T$, $\mathsf{d}^{\mathsf{R}}(v, x) > \ell$ (i.e., the radial distance between x and v is greater than ℓ). Since radial BFS from a vertex can be performed in $O(n)$ time for a planar graph [15], our proof is completed. $\qquad\square$

Finally, we have the following lemma.

Lemma 38. *Given a planar graph G and $T \subseteq V(G)$, we can remove all $5g(k) + 1$-isolated vertices in $O(k \log k \cdot n)$ time.*

Proof. First, we use Reed's algorithm (Corollary 33) to decompose G into $O(k)$ nice subgraphs in $O(k \cdot n)$ time such that each of these subgraphs is embedded either on a 2-punctured plane or a 1-punctured plane. Moreover, recall from Corollary 33 that the total number of vertices of G are on the boundary of these punctured planes is $O(k \log k)$. Then, we apply Lemma 37 on each of these vertices to see if they are $g(k)$-isolated (from T) in G, and remove if they are $g(k)$-isolated in $O(k \log k \cdot n)$ time in total. (Recall that if a vertex is $g(k)$-isolated, then it is irrelevant). Note that if all vertices of the boundary of a punctured plane $\boxdot$ that corresponds to a nice subgraph H (of G) are $g(k)$-isolated, then, by definition of nice subgraphs, all vertices of H are irrelevant, and we can remove them. Hence, after this step (removing irrelevant vertices from the boundary), we are left with $O(k)$ nice subgraphs, each embedded on either a 1-punctured or a 2-punctured plane such that none of the vertices on the boundary of these planes is $g(k)$-isolated from T in G.

Next, from each 2-punctured plane (resp., 1-punctured plane) $\boxdot$, we remove all $4g(k)$-isolated vertices in $\boxdot$ using Lemma 35 (resp., Proposition 34). Finally, we have the following crucial claim.

Claim 39. *Let H be a nice subgraph of G embedded on a c-punctured plane $\boxdot$. Moreover, suppose that none of the vertices on the boundary of $\boxdot$ is $g(k)$-isolated in G and none of the vertices in H is $4g(k)$-boundary-isolated in $\boxdot$. Then, none of the vertices in H is $5g(k) + 1$-isolated from T in G.*

Proof of Claim. Targeting a contradiction, let us assume that there exists a vertex $v \in V(H)$ such that v is $5g(k) + 1$-isolated in G. See Figure 18 for an illustration. Then, there exists a sequence of $5g(k) + 2$ concentric cycles, say $C_0, C_1, \ldots, C_{5g(k)+1}$ such that $v \in D_0$ and $T \cap D_{5g(k)+1} = \emptyset$. Further, since v is not $4g(k)$-boundary-isolated in $\boxdot$, there exists a vertex w on the boundary of $\boxdot$ such that $w \in D_{4k}$. But in this case, observe that the concentric cycles $C_{4g(k)+1}, \ldots, C_{5g(k)+1}$ separate w from

T, contradicting the fact that w is $g(k)$-isolated. Hence, each vertex in H is $5g(k) + 1$-isolated in G (from T). ◇

Due to Claim 39, and the removal of all $g(k)$-isolated vertices from the boundaries of punctured planes, and all $4g(k)$-boundary-isolated vertices from the interior of 2-punctured planes and $g(k)$-boundary-isolated (and hence, $4g(k)$-boundary-isolated) vertices from the interior of 1-punctured planes, it follows that none of the remaining vertices in G is $5g(k) + 1$-isolated. □

Now, we are ready to prove the following result.

Theorem 4. *There exists a $k^{O(1)} \cdot n$-time algorithm that, given an instance (G, T) of T-Cycle on planar graphs, outputs an equivalent instance (G', T) of T-Cycle on planar graphs along with a set of vertices U such that:*

1. *G' is a subgraph of G whose treewidth is bounded by $O(\sqrt{k} \log k)$, and*

2. *$|U| \in O(k \log k)$ and the treewidth of $G' - U$ is bounded by $O(\log k)$.*

Proof. First, we use Lemma 38 to remove all $5g(k) + 1$-isolated vertices from G to obtain an equivalent instance (G', T) such that G' is a subgraph of G (and $T \subseteq V(G')$) in $O(k \log k \cdot n)$ time. Recall that $g(k) \in O(\log k)$. Since there is no vertex in G' that is $c \log k$-isolated from T, for some constant c, G' has treewidth at most $9(c \log k + 1)\lceil \sqrt{k+1} \rceil$ (Proposition 11), i.e., $tw(G') = O(\sqrt{k} \log k)$.

Now, let U be the set of all boundary vertices of all $O(k)$ many punctured planes of G' ($|U| \in O(k \log k)$). Then, observe that there is no sequence of $4g(k) + 1$ concentric cycles in $G' - U$, as otherwise, there is a vertex in some some punctured plane ⊡ of G' that is $4g(k)$-isolated in ⊡. Finally, since there is no sequence of $4g(k)$ concentric cycles in $G' - U$, it follows from Proposition 10 that the treewidth of $G' - U$ is upper bounded by $O(g(k))$, i.e., $O(\log k)$. □

Finally, we present the main result of this section.

Theorem 1. *Restricted to planar graphs, the T-Cycle problem is solvable in time $2^{O(\sqrt{k} \log k)} \cdot n$.*

Proof. First, given an instance of T-Cycle (G, T), where G is a planar graph, we use Theorem 4 to get an equivalent instance (G', T) of T-Cycle such that G' is planar and has treewidth bounded by $O(\sqrt{k} \log k)$ in $k^{O(1)} \cdot n$ time. We can compute a tree decomposition of width $O(\sqrt{k} \log k)$ in $2^{O(\sqrt{k} \log k)} \cdot n$ time using [31]. Since we can solve T-Cycle in $2^{O(tw)} \cdot n$ time using a dynamic-programming based algorithm on tree decomposition, for example, using [17], we can solve our problem in $2^{O(\sqrt{k} \log k)} \cdot n$ time. □

We call the algorithm described in this section Reed Decomposition Algorithm. Below, we summarize the steps of Reed Decomposition Algorithm in pseudocode form for improved readability.

Algorithm Reed decomposition algorithm (G, T)

- Step 1. Given a planar graph G and a set of terminals $T \subseteq V(G)$ with $|T| = k$, decompose G into $O(k)$ nice subgraphs each embedded into either a 2-punctured plane or a 1-punctured plane using Corollary 33 so that the total number of boundary vertices is $O(k \log k)$.

- Step 2. Using Lemma 37, determine which of the boundary vertices of G are $g(k)$-isolated vertices, and remove those that are. If all the boundary vertices of some ⊡, corresponding to H, are $g(k)$-isolated, then remove $V(H)$.

- Step 3. Using Proposition 34, Remove all the $g(k)$-boundary-isolated vertices from nice subgraphs of G that are embedded in 1-punctured planes.

- Step 4. Using Proposition 35, remove all the $4g(k)$-boundary-isolated vertices from nice subgraphs of G that are embedded in 2-punctured planes.

- Step 5. Solve the T-Cycle problem using Theorem 1

6 Kernelization

In this section and the next section, we will prove T-Cycle problem admits a kernelization algorithm of size $k \log^{O(1)} k$ that runs in $k^{O(1)} \cdot n$ time. Specifically, we will prove Theorem 2. As a stepping stone to the linear time kernelization algorithm, in this section, we first design a simpler kernelization algorithm that runs in polynomial (but not linear) time. Both algorithms make use of a standard tool in parameterized complexity called *protrusion decompositions*. Towards the description of this tool, we start with a few standard definitions from [20].

Definition 40 (Boundary of a vertex set). *Given a graph G, we define the* boundary *of a vertex set $U \subseteq V(G)$, denoted by ∂U, as the set of vertices in U that have at least one neighbor outside of U.*

Definition 41 (Treewidth-η-modulator). *Given a graph G, a vertex set $U \subseteq V(G)$ is called a treewidth-η-modulator if $\text{tw}(G - U) \le \eta$.*

Definition 42 (Protrusion, protrusion decomposition). *For a graph G and an integer $q > 0$, we say that a vertex subset $U \subseteq V(G)$ is a q-protrusion if $\text{tw}(U) \le q$ and $|\partial U| \le q$.*

A partition $X_0, X_1, \ldots, X_\ell$ of vertex set $V(G)$ of a graph G is called an (α, β, γ)-protrusion decomposition of a graph G if α, β and γ are integers such that:

- *$|X_0| \le \alpha$,*

- *$\ell \le \beta$,*

- *X_i, for every $i \in [\ell]$, is a γ-protrusion of G,*

- *for every $i \in [\ell]$, $N_G(X_i) \subseteq X_0$.*

For every $i \in [\ell]$, we set $B_i = X_i^+ \setminus X_i$, where $X_i^+ = N_G[X_i]$.

Given a treewidth-η-modulator S of a planar graph G, there exists an (α, β, γ)-protrusion decomposition such that α, β and γ depend only on η and $|S|$, as the following theorem shows.

Theorem 43 (Lemmas 15.13 and 15.14 in [20]). *If a planar graph G has a treewidth-η-modulator S, then G has a set $X_0 \supseteq S$, such that*

- *each connected component of $G - X_0$ has at most 2 neighbors in S and at most 2η neighbors in $X_0 \setminus S$,*

- *G has a $((4(\eta + 1) + 1)|S|, (20(\eta + 1) + 5)|S|, 3\eta + 2)$-protrusion decomposition.*

Remark 44. *Suppose that we are given a planar graph G with a treewidth-η-modulator S. Based on the proof of [20, Lemmas 15.13 and 15.14], given a tree decomposition of width $O(\eta)$, an $(O(\eta \cdot |S|), O(\eta \cdot |S|), O(\eta))$-protrusion decomposition can be computed using the pseudocode in Algorithm 1 in Section 7.*

Instead of computing the treewidth of G exactly, we use an approximation algorithm such as the one described in [31] for computing the tree decomposition in Step 2 of Algorithm 1. Thus, Step 2 of Algorithm 1 can be executed in $2^{O(\eta)} \cdot n$ time. The remaining Steps of Algorithm 1 can be executed in $O((\eta \cdot |S|)^{O(1)} \cdot n)$ time. Thus, for our purposes, Algorithm 1 runs in $O(((\eta \cdot |S|)^{O(1)} + 2^{O(\eta)}) \cdot n)$ time.

Next, we describe some machinery we use from [44]. Following [44], we say that two graphs G_1, G_2 sharing a set of vertices B are *B-linkage equivalent w.r.t.* Disjoint Paths if for every set of pairs $\mathcal{M} \subseteq B^2$, the instances $(G_1, \mathcal{M})$ and $(G_2, \mathcal{M})$ of Disjoint Paths are equivalent. We now recall a key kernelization result from [44] that will serve as a blackbox for our algorithm.

Theorem 45 (Theorem 8 in [44]). *Let G be a planar graph of treewidth tw and $B \subseteq V(G)$ be of size c. Then, there exists a polynomial time algorithm that constructs a planar graph G' with $B \subseteq V(G')$ such that $|V(G')| = O(c^{12} \text{tw}^{12})$ and G' is B-linkage equivalent w.r.t. Disjoint Paths to G.*

Remark 46. *From the proof of Theorem 8 in [44], it can further be seen that we can construct a graph G' such that G' is a minor of G.*
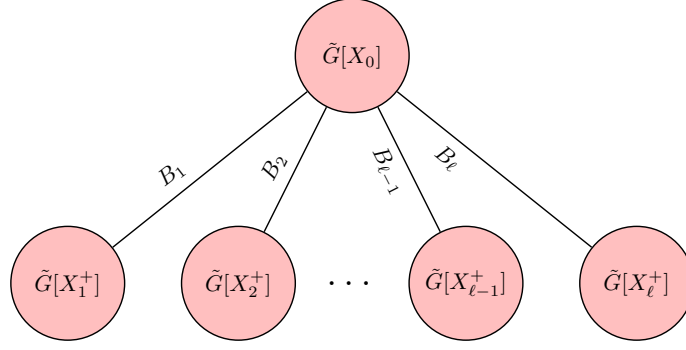
Figure 19: KERNELIZATION ALGORITHM-I first finds a graph $\tilde{G}$ such that (G, T) and $(\tilde{G}, T)$ are equivalent as T-cycle instances. It then finds a protrusion decomposition of $\tilde{G}$ which is shown in the figure. For each $i \in [\ell]$, B_i is a vertex set that is shared by graphs $G[X_0]$ and $G[X_i^+]$.

We now provide the pseudocode for a polynomial time kernelization algorithm, which we use as a springboard for our linear time kernelization algorithm in the next section.

Algorithm KERNELIZATION ALGORITHM-I (G, T)

- Step 1. From Theorem 4, we know that there exists a $k^{O(1)} \cdot n$-time algorithm that, given an instance (G, T) of T-CYCLE on planar graphs, outputs an equivalent instance $(\tilde{G}, T)$ of T-CYCLE on planar graphs and a set U such that $|U| \in O(k \log k)$ and the treewidth of $\tilde{G} - U$ is bounded by $O(\log k)$. In other words, U is a treewidth-$O(\log k)$-modulator of graph $\tilde{G}$. Our kernelization algorithm first computes such a graph $\tilde{G}$ and set U.

- Step 2. Next, using Algorithm 1, we compute an $(O(k \log^2 k), O(k \log^2 k), O(\log k))$-protrusion decomposition of $\tilde{G}$, which is a partition $\{X_0, \ldots, X_\ell\}$ of $V(\tilde{G})$. Let $G_0' = \tilde{G}[X_0]$.

- Step 3. Finally, using Theorem 45 and Corollary 47, for every $i \in [\ell]$, we compute a graph G_i' that is B_i-linkage equivalent w.r.t. DISJOINT PATHS to $\tilde{G}[X_i^+]$, where $X_i^+ = N_{\tilde{G}}[X_i]$. The kernel for the T-CYCLE instance (G, T) is given by the graph K whose vertex set is given by $V(K) = \bigcup_{i=0}^{\ell} V(G_i')$ and whose edge set is given by $E(K) = \bigcup_{i=0}^{\ell} E(G_i')$.

We refer the reader to Figure 19 for a schematic depiction of the protrusion decomposition algorithm used in Step 2 of KERNELIZATION ALGORITHM-I.

Towards the goal of showing that KERNELIZATION ALGORITHM-I is indeed correct, we first prove the following corollary.

Corollary 47. *In* KERNELIZATION ALGORITHM-I, *for every $i \in [\ell]$, we can construct, in polynomial time, a planar graph G_i' with $B_i \subseteq V(G_i')$ such that $|V(G_i')| = O(\log^{24} k)$ and G_i' is B_i-linkage equivalent w.r.t.* DISJOINT PATHS *to $\tilde{G}[X_i^+]$.*

Proof. Note that $G[X_i^+]$ is a planar graph of treewidth $O(\log k)$ and $B_i \subseteq V(G[X_i^+])$ is also of size $O(\log k)$. Applying Theorem 45, the claim follows. $\square$

Theorem 48. KERNELIZATION ALGORITHM-I *correctly computes a kernel for the T-CYCLE problem on planar graphs in polynomial time.*

Proof. We only need to justify Step 3 of KERNELIZATION ALGORITHM-I. Let SOL be the set of all solutions to an instance $(\tilde{G}, T)$ of the T-CYCLE problem. Note that for any $i \in [\ell]$ and for some $t, q \geq 0$, a solution $\mathcal{S}^j \in$ SOL to the T-CYCLE problem can be written as a union of paths in $\mathcal{P}_1 = \{S_1, \ldots, S_t\}$ and $\mathcal{P}_2 = \{O_1, \ldots, O_q\}$ where the endpoints of the paths in $\mathcal{P}_1 \bigcup \mathcal{P}_2$ are in B_i, the paths in $\mathcal{P}_1$ use edges that belong to $\tilde{G}[X_i]$ and the paths in $\mathcal{P}_2$ use edges outside of $\tilde{G}[X_i]$. Let $M_i^j = \{\{u_1, v_1\}, \ldots, \{u_q, v_q\}\}$ be the respective endpoints of the paths in $\mathcal{P}_1$, and let $\mathcal{M}_i = \{M_i^j \mid \mathcal{S}^j \in$ SOL$\}$. Then, $\mathcal{M}_i \subset B_i^2$. By construction, G_i' is B_i-linkage equivalent w.r.t. DISJOINT PATHS to $\tilde{G}[X_i^+]$. So, there is also a way to connect all the pairs in $\mathcal{M}_i$ in G_i'. Thus, after replacing each graph $\tilde{G}[X_i^+]$ by an equivalent subgraph G_i', we obtain a new instance of the T-CYCLE problem whose solution set is non-empty if and only if $|$SOL$| \neq 0$. Using Corollary 47, the claim follows. $\square$

The size of the vertex set $V(K)$ for the kernel K computed by KERNELIZATION ALGORITHM-I is $k \log^{O(1)} k$. This follows from Corollary 47 along with the fact that ℓ and X_0 determined in Step 2 of KERNELIZATION ALGORITHM-I are both bounded by $O(k \log^2 k)$.

7 Linear time Kernelization Algorithm

Building on the results from the previous section, in this section, we provide a linear time procedure for providing a $k \log^{O(1)} k$ sized kernel. Before we get to the algorithm, we need to adapt the techniques of Section 5 to B-linkage equivalence for T-CYCLE. Indeed, this is the content of Corollary 54. With an eye towards this goal, we provide some definitions.

Definition 49 ($\mathcal{M}$-cycle). *Given a planar graph G, a vertex set $B \subseteq V(G)$, and a set of pairs $\mathcal{M} \subseteq B^2$, the graph $G_{\mathcal{M}}$ obtained from G by inserting for every pair $\{u_i, v_i\} \in \mathcal{M}$ a pair of edges $\{u_i w_i, w_i v_i\}$ in $G_{\mathcal{M}}$ is called the $\mathcal{M}$-subdivided graph of G. We say that there is an $\mathcal{M}$-cycle in G if $G_{\mathcal{M}}$ has a $T_{\mathcal{M}}$-loop, where the set of vertices $T_{\mathcal{M}} = \{w_i | i \in [|\mathcal{M}|]\}$ are called subdivision vertices.*

Definition 50 (B-linkage equivalence w.r.t. T-CYCLE). *Let G_1 and G_2 be two graphs with a common set of vertices B. That is, $B \subseteq V(G_1)$ and $B \subseteq V(G_2)$. Then, we say that G_1 and G_2 are B-linkage equivalent w.r.t. T-CYCLE if for every $\mathcal{M} \subseteq B^2$, there is an $\mathcal{M}$-cycle in G_1 implies that there is also an $\mathcal{M}$-cycle in G_2 and vice versa.*

Definition 51 (B-linkage irrelevant vertices). *Given a planar graph G, and a vertex set $B \subseteq V(G)$, we say that a vertex $v \in V(G) \setminus B$ of a graph G is B-linkage irrelevant if G and $G - \{v\}$ are B-linkage equivalent w.r.t. T-CYCLE.*

We now prove a lemma for B-linkage irrelevant vertices that is analogous to Corollary 30 in Section 4.2.

Lemma 52. *Given a planar graph G, and a vertex set $B \subseteq V(G)$, let $\ell = |B|$. Then, for some $g(\ell) \in O(\log \ell)$, all $g(\ell)$-isolated vertices are B-linkage irrelevant.*

Proof. Let c_1 and c_2 be the constants determined by Corollary 30. For $g(\ell) = c_1 \log \ell + c_2$, we want to show that all $g(\ell)$-isolated vertices are B-linkage irrelevant. Towards a proof by contradiction, assume that this is not the case. Then, there exists an $\mathcal{M} \subseteq B^2$ such that in the $\mathcal{M}$-subdivided graph $G_{\mathcal{M}}$, there exists a $g(\ell)$-isolated vertex that is not B-linkage irrelevant. Applying Corollary 30 directly to the T-CYCLE instance $(G_{\mathcal{M}}, T_{\mathcal{M}})$, where $T_{\mathcal{M}}$ is the set of subdivision vertices, we get a contradiction. $\square$

Remark 53. *One way to interpret Lemma 52 is that while the set of irrelevant vertices for T-cycle instances $(G_{\mathcal{M}_1}, T_{\mathcal{M}_1})$ $(G_{\mathcal{M}_2}, T_{\mathcal{M}_2})$ for distinct $\mathcal{M}_1, \mathcal{M}_2 \subseteq B^2$ may not be the same, all the $g(\ell)$-isolated vertices are B-linkage irrelevant.*

Note that Propositions 32, 34 and 35 and Corollary 33 are not specific to the T-CYCLE problem as such. In particular, given a planar graph G, and a set of boundary vertices $B \subseteq V(G)$, we can now apply the first three steps of the REED DECOMPOSITION ALGORITHM from Section 5.

Algorithm REED DECOMPOSITION ALGORITHM FOR B-LINKAGE EQUIVALENCE (G, B)

- Step 1. Given a planar graph G and a set of boundary vertices $B \subseteq V(G)$ with $|B| = m$, decompose G into $O(m)$ nice subgraphs each embedded into either a 2-punctured plane or a 1-punctured plane using Corollary 33 so that the total number of vertices in the (new) boundary $B' \supset B$ is $O(m \log m)$.

- Step 2. Using Proposition 34, remove all the $g(m)$-boundary-isolated vertices from nice subgraphs of G that are embedded in 1-punctured planes.

- Step 3. Using Proposition 35, remove all the $4g(m)$-boundary-isolated vertices from nice subgraphs of G that are embedded in 2-punctured planes.

As a result of applying REED DECOMPOSITION ALGORITHM FOR B-LINKAGE EQUIVALENCE, we have the following corollary.

Corollary 54. *Given a planar graph G and a set of vertices $B \subseteq V(G)$ with $|B| = m$, one can compute a set of vertices $B' \supset B$ such that $|B'| \in O(m \log m)$ and remove all vertices that are (at least) $4g(m)$-boundary-isolated in linear time to obtain a graph $\tilde{G}$ that is B-linkage equivalent w.r.t. T-CYCLE to G. Moreover, B' is a treewidth-$O(g(m))$-modulator of $\tilde{G}$.*

Next, we provide a detailed description of KERNELIZATION ALGORITHM-II.
Algorithm KERNELIZATION ALGORITHM-II (G, T)

- Step 1. As in the case of KERNELIZATION ALGORITHM-I, use the $k^{O(1)} \cdot n$-time algorithm from Theorem 4, that given an instance (G, T) of T-CYCLE on planar graphs, outputs an equivalent instance $(\tilde{G}, T)$ and a set U such that $|U| \leq O(k \log k)$ and the treewidth of $\tilde{G} - U$ is bounded by $O(\log k)$. That is, the kernelization algorithm first computes such a graph $\tilde{G}$ and a treewidth-$O(\log k)$-modulator U of graph $\tilde{G}$.

- Step 2. Using Algorithm 1, we compute an $(O(k \log^2 k), O(k \log^2 k), O(\log k))$-protrusion decomposition of $\tilde{G}$, which is a partition $\{X_0, \ldots, X_\ell\}$ of $V(\tilde{G})$. Let $G'_0 = \tilde{G}[X_0]$.

- Step 3. For every $i \in [\ell]$, use the linear-time algorithm from Corollary 54 to obtain a graph G'_i that is B_i-linkage equivalent w.r.t. T-CYCLE to $\tilde{G}[X_i^+]$ and a set $B'_i \subseteq V(G'_i)$ such that $|B'_i|$ is $O(\log k \log \log k)$ and the treewidth of $G'_i - B'_i$ is bounded by $O(\log \log k)$. That is, the kernelization algorithm first computes such a graph G'_i and a treewidth-$O(\log \log k)$-modulator B'_i of graph G'_i.

- Step 4. For every $i \in [\ell]$, using Algorithm 1, compute an $(O(\log k \log^2 \log k), O(\log k \log^2 \log k), O(\log \log k))$-protrusion decomposition of G'_i, which gives a partition $\{Y_{i,0}, \ldots, Y_{i,m_i}\}$ of $V(G'_i)$. For $i \in [\ell]$, $j \in [m_i]$, let G'_{ij} be the subgraph of G'_i induced by vertices in $Y_{i,j}^+ = N_{G'_i}[Y_{i,j}]$. That is, let $G'_{ij} = G'_i[Y_{i,j}^+]$. Also, let $G'_{i,0} = G'_i[Y_{i,0}]$ for $i \in [\ell]$. Finally, let $B_{i,j} = Y_{i,j}^+ \setminus Y_{i,j}$, for $i \in [\ell]$, $j \in [m_i]$.

- Step 5. For every $i \in [\ell], j \in [m_i]$, do the following:
    - Step 5.1 Generate all possible distinct graphs H with $|V(H)| = O(\log^{24} \log k)$ and $B_{i,j} \subseteq V(H)$ in a set $\mathcal{G}_{i,j}$.
    - Step 5.2 For each graph $H \in \mathcal{G}_{i,j}$,
        * Use the dynamic programming in [1, 2] to check if H is a minor of $G'_{i,j}$.
        * For every pairing of terminals $\mathcal{M} \subseteq B_{i,j}^2$, check if $(H, \mathcal{M})$ and $G'_{i,j}, \mathcal{M})$ are equivalent as disjoint path instances, that is, check if the yes / no answers to these instances are the same using the algorithm in [13].

- Step 6 For every $i \in [\ell], j \in [m_i]$, let $H'_{i,j} \in \mathcal{G}_{i,j}$ denote a graph that is $B_{i,j}$-linkage equivalent w.r.t. DISJOINT PATHS to $G''_{i,j}$ in Step 5.2. The kernel for the T-CYCLE problem is given by the graph K' where

$$V(K') = V(G'_0) \bigcup_{i \in [\ell]} V(G'_{i,0}) \bigcup_{\substack{i \in [\ell] \\ j \in [m_i]}} V(H'_{i,j}) \text{ and } E(K') = E(G'_0) \bigcup_{i \in [\ell]} E(G'_{i,0}) \bigcup_{\substack{i \in [\ell] \\ j \in [m_i]}} E(H'_{i,j}).$$

Please see Figure 20 for a schematic depiction of the nested protrusion decomposition algorithm used in KERNELIZATION ALGORITHM-II.

Remark 55. *Note that we use two different notions of B-linkage equivalence in Step 3 and Step 6 of KERNELIZATION ALGORITHM-II respectively. First, it is each to check that given two subgraphs G_1 and G_2 of G with $B \in V(G_1) \cap V(G_2)$, if they are B-linkage equivalent w.r.t. DISJOINT PATHS, then they are also B-linkage equivalent w.r.t. T-CYCLE. For our purposes, checking only B-linkage equivalent w.r.t. T-CYCLE would suffice in all cases. In Step 6, we check linkage equivalent w.r.t. DISJOINT PATHS, simply because of the simplicity of formulation.*

Likewise, in Step 3 of KERNELIZATION ALGORITHM-I, we replace a graph $\tilde{G}[X_i^+]$ with a B_i-linkage equivalent w.r.t. DISJOINT PATHS graph G'_i because we are using a blackbox.

Steps 1 and 2 of KERNELIZATION ALGORITHM-II are in common with Steps 1 and 2 of KERNELIZATION ALGORITHM-I. We now prove that Step 3 of KERNELIZATION ALGORITHM-II is also correct.
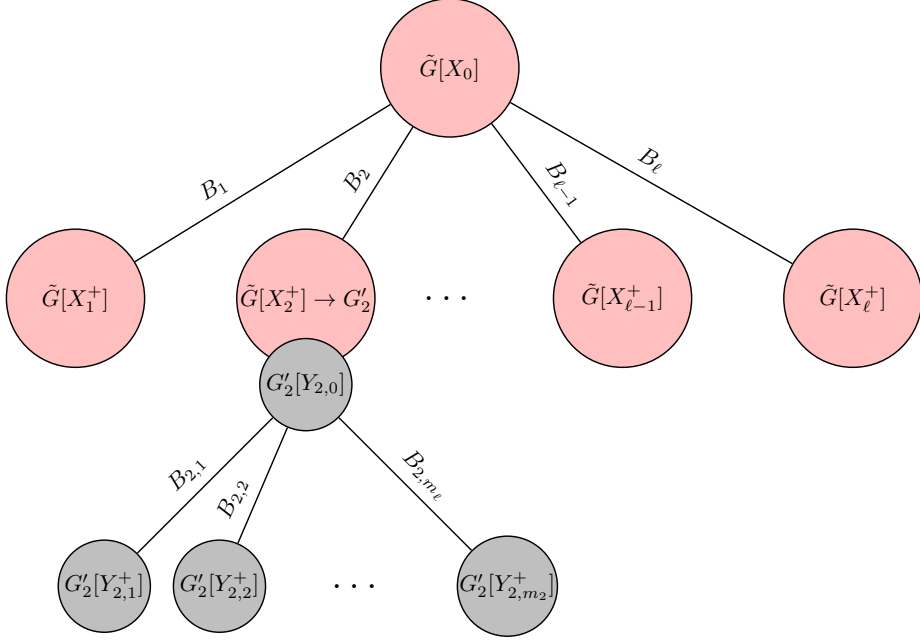
Figure 20: KERNELIZATION ALGORITHM-II first finds a graph $\tilde{G}$ such that (G, T) and $(\tilde{G}, T)$ are equivalent as T-cycle instances. It then finds a protrusion decomposition of $\tilde{G}$ which is shown by the nodes in red. For each $i \in [\ell]$, B_i is a vertex set that is shared by graphs $\tilde{G}[X_0]$ and $\tilde{G}[X_i^+]$. For every $i \in [\ell]$, we obtain a graph G_i' that is B_i-linkage equivalent w.r.t. T-CYCLE to $\tilde{G}[X_i^+]$. For clarity, we show this transition only in the node of $\tilde{G}[X_2^+]$. Next, for every $i \in [\ell]$, we find the protrusion decomposition of G_i'. In the figure we only show the protrusion decomposition of G_2' with nodes in grey. For each $j \in [m_2]$, $B_{2,j}$ is a vertex set that is shared by graphs $G_2'[Y_{2,0}]$ and $G_2'[Y_{2,j}^+]$.

Lemma 56. *The vertices in $\tilde{G}[X_i^+]$ that are B_i-linkage-irrelevant in Step 3 of* KERNELIZATION ALGORITHM-II *are also irrelevant for the T-CYCLE problem in G.*

Proof. This follows from the fact that a solution to the T-CYCLE problem in G can be written as a union of paths in $\mathcal{P}_1 = \{S_1, \ldots, S_t\}$ and $\mathcal{P}_2 = \{O_1, \ldots, O_q\}$ where the endpoints of the paths in $\mathcal{P}_1 \bigcup \mathcal{P}_2$ are in B_i, the paths in $\mathcal{P}_1$ use edges that belong to $\tilde{G}[X_i]$ and the paths in $\mathcal{P}_2$ use edges outside of $\tilde{G}[X_i]$. The paths in $\mathcal{P}_2$ induce a matching $\mathcal{M} \subset B_i^2$. If a vertex v is B_i-linkage-irrelevant, then there exists a way to connect the pairs in $\mathcal{M}$ without using the vertex v. Hence, there is a solution to the T-CYCLE problem that does not use v, proving the claim. $\qquad\square$

In the following lemma, we show that the instance K' computed by KERNELIZATION ALGORITHM-II is indeed a kernel. We end the section with Theorem 2 where we provide bounds on the size of the kernel and show that it runs in linear time.

Lemma 57. *After the conclusion of Step 6 of* KERNELIZATION ALGORITHM-II, *the T-CYCLE instance (G, T) is equivalent to the T-CYCLE on the graph K' where*

$$V(K') = V(G_0') \bigcup_{i \in [\ell]} V(G_{i,0}') \bigcup_{\substack{i \in [\ell] \\ j \in [m_i]}} V(H_{i,j}') \text{ and } E(K') = E(G_0') \bigcup_{i \in [\ell]} E(G_{i,0}') \bigcup_{\substack{i \in [\ell] \\ j \in [m_i]}} E(H_{i,j}').$$

Proof. It follows from Lemma 56 that after the conclusion of Step 3 of KERNELIZATION ALGORITHM-II, the T-CYCLE instance (G, T) is equivalent to the T-CYCLE instance (K, T) on the graph K whose vertex set is given by $V(K) = \bigcup_{i=0}^{\ell} V(G_i')$ and whose edge set is given by $E(K) = \bigcup_{i=0}^{\ell} E(G_i')$.

Moreover, we know from Corollary 54 that B_i' is a treewidth-$O(\log \log k)$-modulator of graph G_i'. For every $i \in [\ell]$, this allows us to use Algorithm 1, to compute an $O(\log k \log^2 \log k)$, $O(\log k \log^2 \log k)$, $O(\log \log k)$)-protrusion decomposition of G_i', giving us a partition $\{Y_{i,0}, \ldots, Y_{i,m_i}\}$ of $V(G_i')$. As in KERNELIZATION ALGORITHM-II, we use the notation $G_{i,j}' = G_i'[Y_{i,j}^+]$ and $B_{i,j} = Y_{i,j}^+ \setminus Y_{i,j}$ for $i \in [\ell], j \in [m_i]$, whereas $G_{i,0}' = G_i'[Y_{i,0}]$ for $i \in [\ell]$. Now, we use Theorem 45 as an existential rather than an algorithmic result. That is, instead of kernelizing each of the graph G_{ij}' using the

algorithm from [44], we simply generate all possible graphs H such that $|V(H)| = O(\log^{24} \log k)$ and $B_{i,j} \subseteq V(H)$. Using Theorem 45, we know that one of the generated graphs (denoted by $H'_{i,j}$) will be found to be $B_{i,j}$-linkage equivalent w.r.t. DISJOINT PATHS to $G'_{i,j}$ since $G'_{i,j}$ is a planar graph of treewidth $O(\log \log k)$ and $B_{i,j} \subseteq V(G'_{i,j})$ is also of size $O(\log \log k)$.

A solution to the T-CYCLE instance of (K, T) can be written as a union of paths in $\mathcal{P}_1 = \{S_1, \ldots, S_t\}$ and $\mathcal{P}_2 = \{O_1, \ldots, O_q\}$ where the endpoints of the paths in $\mathcal{P}_1 \bigcup \mathcal{P}_2$ are in $B_{i,j}$, the paths in $\mathcal{P}_1$ use edges that belong to $G[Y_{i,j}]$ and the paths in $\mathcal{P}_2$ use edges outside of $G[Y_{i,j}]$. Let $M = \{\{u_1, v_1\}, \ldots, \{u_q, v_q\}\}$ be the respective endpoints of the paths in $\mathcal{P}_1$, and let $\mathcal{M}$ be the collection of all such sets M found by varying the solutions to the T-CYCLE instance (K, T). Then, $\mathcal{M} \subset B_{i,j}^2$. By the preceding exhaustive algorithm, $H'_{i,j}$ is $B_{i,j}$-linkage equivalent w.r.t. DISJOINT PATHS to $G'_{i,j}$. So, there is also a way to connect all the pairs in $\mathcal{M}$ in $H'_{i,j}$. Inductively, after replacing each graph $G'_{i,j}$ by an equivalent subgraph $H'_{i,j}$, for every $i \in [\ell]$ and $j \in [m_i]$, we obtain a new instance of the T-CYCLE problem whose solution set is non-empty if and only if the solution set of the T-CYCLE instance (K, T) is non-empty. The claim of the lemma follows. $\square$

In order to establish runtime bounds for the algorithm, we use two results from prior works.

Theorem 58 (Theorem 4 in [1]). *Given a host graph G with $|V(G)| = n$ embedded in a fixed surface, a pattern H with $|V(H)| = h$, and treewidth at most k, we can decide whether G contains a minor isomorphic to H in $2^{O(k+h)+2k\cdot\log h} \cdot n$ time.*

Theorem 59 (Theorem 36 in [13]). *The PLANAR DISJOINT PATHS problem can be solved in $2^{O(k^2)}n$ time, where n is the size of the graph and k is the number of terminals.*

Algorithm 1: Computation of an (α, β, γ)-protrusion decomposition of a planar graph G

Input : Treewidth-η-modulator S of planar graph G
Output: An $(O(\eta \cdot |S|), O(\eta \cdot |S|), O(\eta))$-protrusion decomposition of G

1 **Procedure** `Protrusion-Decomoposition(S,G)`
2   Compute a nice tree decomposition (T, χ) of $G - S$.
3   Choose an arbitrary root node r for T.
4   Let M denote the set of marked nodes. Initialize $M \leftarrow \emptyset$.
5   **repeat**
6                 $\triangleright$ For a node $v \in V(T)$, let T_v denote the subtree of T rooted at v.
7   Let v be a lowermost node in T such that some component C of $G[\chi(T_v - M)]$ has at least three neighbors in S.
8        $M \leftarrow M \bigcup \{v\}$.
9   **until** *every component C of $G[\chi(T - M)]$ has at most two neighbors in S*;
10     $L \leftarrow$ LCA-CLOSURE(M).
11     $X_0 \leftarrow S \bigcup \chi(L)$.
12   Find the connected components $C_1, C_2, \ldots, C_t$ of $G - X_0$.
13   Find a partition $X_1, \ldots, X_\ell$ of $G - X_0$ by grouping components $C_1, C_2, \ldots, C_t$ with the same neighborhood in S. That is, $C_i \subseteq X_k$ and $C_j \subseteq X_k$ if and only if $N(C_i) = N(C_j)$.
14   RETURN $X_0, \ldots, X_\ell$.

Finally, we are ready to prove the main result of this section. The pseudocode for the Protrusion decomposition is provided in Algorithm 1.

Theorem 2. *Restricted to planar graphs, the T-CYCLE problem admits a $k^{O(1)} \cdot n$-time kernelization algorithm of size $k \cdot \log^{O(1)} k$.*

Proof. From Lemma 57, we know that (K', T) is equivalent as a T-CYCLE to the pair (G, T). That is, K' is indeed a kernel.

First, we provide a bound on the size of the kernel. Note that $\tilde{G}[X_0]$ has $O(k \log^2 k)$ vertices, and $\ell = O(k \log^2 k)$ by using Theorem 43 in Step 2. By using Theorem 43 in Step 4, $G'_{i,0} = G'_i[Y_{i,0}]$ has $O(\log k \log^2 \log k)$ vertices. So, $\bigcup_{i \in [\ell]} V(G'_{i,0})$ has $O(k \log^3 \log^2 \log k)$ vertices. Moreover, using Theorem 43 once again in Step 4, each m_i for each $i \in [\ell]$ is $O(\log k \log^2 \log k)$

and by construction $V(H'_{i,j})$ has $O(\log^{24} \log k)$ vertices for each $i \in [\ell]$ and $j \in [m_i]$. Therefore, $\bigcup_{\substack{i \in [\ell] \\ j \in [m_i]}} V(H'_{i,j})$ has at most $k \log^3 k \log^{O(1)} \log k$ vertices. Therefore, the kernel K', where $V(K') = V(G'_0) \bigcup_{i \in [\ell]} V(G'_{i,0}) \bigcup_{\substack{i \in [\ell] \\ j \in [m_i]}} V(H'_{i,j})$, has $O(k \log^4 k)$ vertices.

We now prove that the algorithm runs in linear time.

(1.) Using Theorem 4, Step 1 runs in $k^{O(1)} \cdot n$ time.

(2.) From Remark 44, it follows that Step 2 runs in $k^{O(1)} \cdot n$ time.

(3.) The algorithm from Corollary 54 runs in $\log^{O(1)} k \cdot n$ time since B'_i is $O(\log k \log \log k)$ for each $i \in [\ell]$. As noted before $\ell \in O(k \log^2 k)$. Hence, Step 3 runs in $\log^{O(1)} k \cdot n$ time.

(4.) For Step 5.1 guessing a graph in $H \in \mathcal{G}_{i,j}$ involves guessing the edges of H. This means that there are $O(2^{\log^{48} \log k})$ (which is $O(k)$) number of graphs in $\mathcal{G}_{i,j}$ for every $i \in [\ell]$ and $j \in [m_i]$. Hence, the total cost of Step 5.1 is at most $k^2 \log^{O(1)} k$.

(5.) Using Theorem 4 in [1], the dynamic programming algorithm for MINOR CONTAINMENT in Step 5.2 takes $2^{O(\log \log k + \log^{24} \log k) + 2 \log \log k \cdot \log \log^{24} \log k} \cdot n$ which is $O(k) \cdot n$ time.

Using Theorem 36 in [13], a single run of the dynamic programming algorithm for DISJOINT PATHS costs $2^{O(\log^2 \log k)} \cdot n$ which is $O(k) \cdot n$ time. In Step 5.2, $B_{i,j}$ is in $O(\log^2 \log k)$. Thus, there are $2^{O(\log^4 \log k)} = O(k)$ different pairings of terminals for which the DISJOINT PATHS algorithm needs to run. Running the over all pairs, the cost is $O(k^2) \cdot n$ time.

Finally, the two substeps above are executed for every $i \in [\ell]$ and $j \in [m_i]$ and for each of the subgraphs H found in Step 5.2. Nonetheless, the total cost of Step 5.2 over the course of the algorithm is bounded by $k^{O(1)} \cdot n$ time.

Combining the cost of all the steps, the algorithm runs in $k^{O(1)} \cdot n$ time. $\qquad \square$

8 Conclusion

In this paper, we considered the T-CYCLE problem on planar graphs. We begin by providing an FPT algorithm with running time $2^{O(\sqrt{k} \log k)} \cdot n$. Most subexponential time FPT algorithms on planar graphs use Bidimensionality in a naive fashion: If the treewidth of the input graph is large, then we trivially know whether our instance is a yes or no instance, and when the treewidth is small (sublinear), we use dynamic programming using the tree decomposition. Unfortunately, this fails on T-CYCLE (and related terminal-based routing problems) as large treewidth neither blocks a T-loop nor guarantees it. We bypass this roadblock using a clever rerouting argument and new variations of several techniques in the literature to deal with planar graphs. Further, to the best of our knowledge, T-CYCLE is the first natural terminal-based routing problem that admits a subexponential algorithm for planar graphs. We believe that this may pave the road for new subexponential algorithms for planar graphs for problems that cannot be solved by naive application of bidimensionality. Further, we perform non-trivial work to reduce the running time dependence on n to linear.

Next, we provided a linear time kernelization algorithm for T-CYCLE on planar graphs of size $k \cdot \log^{O(1)} k$. As pointed in the Introduction section, the kernelization complexity of T-CYCLE on general graphs is one of the biggest open problems in the field. Our result shows that for planar graphs, we have an almost linear kernel. This raises the natural and important question as to whether our kernel for T-CYCLE can be lifted to more general non-planar graphs, or even to general graphs. To obtain our kernel, we use Reed's technique of plane cutting. This is perhaps the first application of this technique in kernelization. Further, we obtain the kernelization to be almost-optimal combining Theorem 4 with a "nested" protrusion decomposition technique, which is novel to this paper. Perhaps these techniques can be combined to obtain new/improved kernelization results for other terminal-based problems.

As discussed in [34], algorithms for terminal-based routing problems on planar graphs serve as a crucial step in designing algorithms for general case. Indeed, this was the case for DISJOINT PATHS. All known algorithms for DISJOINT PATHS are based on distinguishing the cases when the graph contains a large clique as a minor and when it does not. Furthermore, when the input graph does not contain a large clique as a minor, the graph either has low treewidth (leading towards a DP) or

contains a large *flat wall* that necessitates the study of these problems on planar and "almost-planar" graph classes. As mentioned in the Introduction, it would be interesting to see if it is possible to extend our results for "almost-planar graphs".

Finally, we wish to point out that our arguments may be relevant for the resolution of the problem on general graphs as well. In this direction, the first step should be to generalize our results to flat walls towards the resolution of T-CYCLE on minor-free graphs.

Acknowledgments

References

[1] Isolde Adler, Frederic Dorn, Fedor V. Fomin, Ignasi Sau, and Dimitrios M. Thilikos. Fast minor testing in planar graphs. In Mark de Berg and Ulrich Meyer, editors, *Algorithms – ESA 2010*, pages 97–109, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[2] Isolde Adler, Frederic Dorn, Fedor V. Fomin, Ignasi Sau, and Dimitrios M. Thilikos. Faster parameterized algorithms for minor containment. *Theoretical Computer Science*, 412(50):7018–7028, 2011.

[3] Isolde Adler, Stavros G Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios Thilikos. Tight bounds for linkages in planar graphs. In *Automata, Languages and Programming: 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I 38*, pages 110–121. Springer, 2011.

[4] Isolde Adler, Stavros G Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M Thilikos. Irrelevant vertices for the planar disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 122:815–843, 2017.

[5] Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014.

[6] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017.

[7] Andreas Björklund, Thore Husfeldt, and Nina Taslaman. Shortest cycle through specified elements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1747–1753, 2012.

[8] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[9] Hans L Bodlaender, Fedor V Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M Thilikos. (meta) kernelization. *Journal of the ACM (JACM)*, 63(5):1–69, 2016.

[10] John Adrian Bondy and László Lovász. Cycles through specified vertices of a graph. *Combinatorica*, 1:117–140, 1981.

[11] Liming Cai, Jianer Chen, Rodney G Downey, and Michael R Fellows. Advice classes of parameterized tractability. *Annals of pure and applied logic*, 84(1):119–138, 1997.

[12] Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *J. ACM*, 63(5):40:1–40:65, 2016.

[13] Kyungjin Cho, Eunjin Oh, and Seunghyeok Oh. Parameterized algorithm for the disjoint path problem on planar graphs: Exponential in k^2 and linear in n. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3734–3758. SIAM, 2023.

[14] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[15] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational geometry algorithms and applications*. Spinger, 2008.

[16] Gabriel Andrew Dirac. In abstrakten graphen vorhandene vollständige 4-graphen und ihre unterteilungen. *Mathematische Nachrichten*, 22(1-2):61–85, 1960.

[17] Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos. Catalan structures and dynamic programming in h-minor-free graphs. *J. Comput. Syst. Sci.*, 78(5):1606–1622, 2012.

[18] Péter L Erdős and Ervin Győri. Any four independent edges of a 4-connected graph are contained in a circuit. *Acta Mathematica Hungarica*, 46:311–313, 1985.

[19] Michael R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, pages 276–277, 2006.

[20] Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

[21] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[22] Qian-Ping Gu and Hisao Tamaki. Improved bounds on the planar branchwidth with respect to the largest grid minor size. *Algorithmica*, 64(3):416–453, 2012.

[23] Roland Häggkvist and Carsten Thomassen. Circuits through specified edges. *Discrete mathematics*, 41(1):29–34, 1982.

[24] Derek A. Holton, Brendan D. McKay, Michael D. Plummer, and Carsten Thomassen. A nine point theorem for 3-connected graphs. *Combinatorica*, 2:53–62, 1982.

[25] Ken-ichi Kawarabayashi. One or two disjoint circuits cover independent edges: Lovász–woodall conjecture. *Journal of Combinatorial Theory, Series B*, 84(1):1–44, 2002.

[26] Ken-ichi Kawarabayashi. Cycles through a prescribed vertex set in n-connected graphs. *Journal of Combinatorial Theory, Series B*, 90(2):315–323, 2004.

[27] Ken-ichi Kawarabayashi. An improved algorithm for finding cycles through elements. In *Integer Programming and Combinatorial Optimization: 13th International Conference, IPCO 2008 Bertinoro, Italy, May 26-28, 2008 Proceedings 13*, pages 374–384. Springer, 2008.

[28] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory B*, 102(2):424–435, 2012.

[29] Ken-ichi Kawarabayashi, Zhentao Li, and Bruce Reed. Recognizing a totally odd k 4-subdivision, parity 2-disjoint rooted paths and a parity cycle through specified elements. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 318–328. SIAM, 2010.

[30] Yusuke Kobayashi and Ken-ichi Kawarabayashi. Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1146–1155. SIAM, 2009.

[31] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. *SIAM Journal on Computing*, (0):FOCS21–174, 2023.

[32] Tuukka Korhonen, Michał Pilipczuk, and Giannos Stamoulis. Minor containment and disjoint paths in almost-linear time. *FOCS*, 2024.

[33] Andrea S. LaPaugh and Ronald L. Rivest. The subgraph homeomorphism problem. *J. Comput. Syst. Sci.*, 20(2):133–149, 1980.

[34] Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Efficient graph minors theory and parameterized algorithms for (planar) disjoint paths. In *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, pages 112–128, 2020.

[35] Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Efficient computation of representative weight functions with applications to parameterized counting (extended version). In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 179–198, 2021.

[36] Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. On subexponential parameterized algorithms for steiner tree and directed subset TSP on planar graphs. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 474–484, 2018.

[37] Bruce Reed. Rooted routing in the plane. *Discrete Applied Mathematics*, 57(2-3):213–227, 1995.

[38] Neil Robertson and Paul D Seymour. Graph minors. x. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

[39] Neil Robertson and Paul D. Seymour. Graph minors .xiii. the disjoint paths problem. *J. Comb. Theory B*, 63(1):65–110, 1995.

[40] Daniel P Sanders. On circuits through five edges. *Discrete Mathematics*, 159(1-3):199–215, 1996.

[41] Carsten Thomassen. Note on circuits containing specified edges. *Journal of Combinatorial Theory, Series B*, 22(3):279–280, 1977.

[42] Dekel Tsur. Faster deterministic parameterized algorithm for k-path. *Theor. Comput. Sci.*, 790:96–104, 2019.

[43] Magnus Wahlström. Abusing the tutte matrix: An algebraic instance compression for the k-set-cycle problem. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, pages 341–352, 2013.

[44] Michal Wlodarczyk and Meirav Zehavi. Planar disjoint paths, treewidth, and kernels. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 649–662, 2023.

[45] Douglas R Woodall. Circuits containing specified edges. *J. Comb. Theory, Ser. B*, 22(3):274–278, 1977.