Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 35 (2012)

Finding Cycles and Trees in Sublinear Time

Artur Czumaj Oded Goldreich Dana Ron C. Seshadhri[*] Asaf Shapira
Christian Sohler

June 18, 2012

Abstract

We present sublinear-time (randomized) algorithms for finding simple cycles of length at least $k \geq 3$ and tree-minors in bounded-degree graphs. The complexity of these algorithms is related to the distance of the graph from being C_k-minor free (resp., free from having the corresponding tree-minor). In particular, if the graph is $\Omega(1)$-far from being cycle-free (i.e., a constant fraction of the edges must be deleted to make the graph cycle-free), then the algorithm finds a cycle of polylogarithmic length in time $\widetilde{O}(\sqrt{N})$, where N denotes the number of vertices. This time complexity is optimal up to polylogarithmic factors.

The foregoing results are the outcome of our study of the complexity of *one-sided error* property testing algorithms in the bounded-degree graphs model. For example, we show that cycle-freeness of N-vertex graphs can be tested with one-sided error within time complexity $\widetilde{O}(\text{poly}(1/\epsilon) \cdot \sqrt{N})$, where ϵ denotes the proximity parameter. This matches the known $\Omega(\sqrt{N})$ query lower bound for one-sided error cycle-freeness testing, and contrasts with the fact that any minor-free property admits a *two-sided error* tester of query complexity that only depends on ϵ. We show that the same upper bound holds for testing whether the input graph has a simple cycle of length at least k, for any $k \geq 3$. On the other hand, for any fixed tree T, we show that T-minor freeness has a one-sided error tester of query complexity that only depends on the proximity parameter ϵ.

Our algorithm for finding cycles in bounded-degree graphs extends to general graphs, where distances are measured with respect to the actual number of edges. Such an extension is not possible with respect to finding tree-minors in $o(\sqrt{N})$ complexity.

Keywords: Sublinear-Time Algorithms, Property Testing, Bounded-Degree Graphs, One-Sided vs Two-Sided Error Probability.

[*]Employee of Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

ISSN 1433-8092

Contents

1 Introduction

Consider the algorithmic problem of finding a (simple) cycle in a bounded degree graph (assuming one exists), where the aim is to find such a cycle in (randomized) sublinear time. In general, finding a cycle in sublinear time may not be possible, since the graph may contain only cycles of length $\Omega(n)$. This may also be the case if one needs to remove a constant *number* of the edges of the graph in order to make it cycle-free. But suppose one needs to remove a constant *fraction* of the graph's edges in order to make it cycle free. Can we then devise a sublinear time algorithm? One of our results in this paper is an affirmative answer to this question. Furthermore, the running time of that algorithm is (essentially) optimal.

1.1 Our main results

As we have mentioned above, we consider graphs of bounded degree d with N vertices. We say that a graph is ϵ-far from being cycle-free if one has to remove at least ϵdN edges from G in order to make it cycle free.[1] In all our results, vertex manipulation operations are counted at unit cost. We can now formally state our first result.

Theorem 1.1 (finding cycles): *There exists a randomized algorithm that, on input an N-vertex graph G of degree bound d that is ϵ-far from being cycle-free, finds a simple cycle in G in expected time $\widetilde{O}(\text{poly}(d/\epsilon) \cdot \sqrt{N})$. Furthermore, the cycle found has length at most $\text{poly}(\epsilon^{-1} d \log N)$.*

Using the connection to one-sided error property testing (detailed in Section 1.2), we infer that the algorithm of Theorem 1.1 is optimal; that is, no randomized $o(\sqrt{N})$-time algorithm can find cycles in (bounded-degree) graphs that are $\Omega(1)$-far from being cycle-free. Furthermore, one cannot expect to find simple cycles of length $o(\log N)$, since such may not exist (even if the graph is far from being cycle-free). The result of Theorem 1.1 can be extended to finding a simple cycle of length at least k, for any fixed $k > 3$ (where the case $k = 3$ is covered by Theorem 1.1).

Theorem 1.2 (finding cycles of length at least k): *For every constant $k > 3$, there exists a randomized algorithm that, on input an N-vertex graph G of degree bound d that is ϵ-far from having no simple cycles of length at least k, finds such a cycle in expected time $\widetilde{O}(\text{poly}(d^k/\epsilon) \cdot \sqrt{N})$. Furthermore, the cycle found has length at most $\text{poly}(d^k \cdot \epsilon^{-1} \log N)$.*

Again, the algorithm obtained is optimal in the sense discussed above.

We note that our results can be stated in terms of finding graph minors. A graph G has an H-minor if H can be obtained from G through a series of vertex removals, edge removals, and edge contractions. A graph G is H-minor free, if it contains no H-minor. Note that cycles of length at least k in G correspond to C_k-minors of G, where C_k denotes the k-vertex cycle.

We next turn from finding cycles to finding tree-structures in graphs; that is, finding tree-minors. Consider the following interesting special case. For any constant k, we want to find a tree with at least k leaves. One of our results is a randomized algorithm that finds such trees in expected time that is polynomially related to k and to the distance of the input graph from a graph

[1] In some sources, being ϵ-far from a property means that and ϵ fraction of the function's values should be changed so to obtain a function that has the property. In our case, such a definition would translate to an omission of $\epsilon dN/2$ edges, since each edge appears twice (i.e., once in each of its endpoints). Nevertheless, for sake of simplicity, we chose to measure distance in terms of dN (rather than in terms of $dN/2$).

having no such trees. This problem corresponds to finding minors that are k-vertex stars. More generally, we prove the following result.

Theorem 1.3 (finding tree minors): *For any fixed tree T with k vertices, there exists a randomized algorithm that, on input an N-vertex graph G of constant degree bound d that is ϵ-far from being T-minor free, finds a T-minor in expected time* $\text{poly}(d^D)$, *where* $D = k(16d/\epsilon)^{4k+2}$.

We highlight the fact that finding tree minors can be done within complexity that does not depend on the size of the graph (but rather depends only on (d, k and) ϵ), whereas finding cycles requires $\Omega(\sqrt{N})$ time (also for constant $\epsilon > 0$). In fact, we show that Theorem 1.3 extends to any cycle-free graph (forest) H, and on the other hand we prove that *for any H that contains a cycle finding H-minors requires $\Omega(\sqrt{N})$ queries* (see Theorem 6.1).Thus, we obtain the following characterization:

Corollary 1.4 (finding graph minors, a dichotomy): *Finding H-minors in a constant degree graph that is ϵ-far from being H-minor free can be done in complexity that only depends on ϵ if and only if H is cycle-free.*

1.2 The property testing connection

Loosely speaking, property testing refers to sublinear time probabilistic algorithms for deciding whether a given object has a predetermined property or is far from any object having this property (see the surveys [Fis01, Ron10, Ron08]). Such algorithms, called testers, obtain local views of the object by making suitable queries; that is, the object is seen as a function and the tester gets oracle access to this function (and thus may be expected to work in time that is sublinear in the size of the object).

Randomization is essential to natural testers (i.e., testers of natural properties that have sublinear query-complexity) [GS07]. The same holds also for error probability, at least on some instances, but the question is whether a (small) error probability must appear on all instances. In particular, *should we allow* (small) *error probability both on instances that have the property and on instances that are far from having it?*[2]

Indeed, testers come in two basic flavors referring to the foregoing question: two-sided error testers allow (small) error probability both on instances that have the property and on instances that are far from having it, whereas one-sided error testers only allow (small) error probability on instances that are far from having the property. That is, in one-sided error testers, any instance that has the property is accepted with probability 1.

An important observation regarding one-sided error testers is that *whenever such a tester rejects some instance, it always has a certificate that this instance does not have the property, where this certificate is the partial view of the instance as obtained by the tester.* Indeed, in the case of one-sided error, rejecting an instance based on a specific partial view means that there exists no instance that has the property and is consistent with this partial view. Furthermore, in some cases (as those addressed in the current work), this partial view contains some natural structures (e.g., a cycle or a tree of interest).

Consider, for example, the case of testing cycle-freeness (with one-sided error). In this case, whenever the tester rejects, its partial view must contain a cycle. Thus, *any one-sided tester of*

[2]Recall that, in any case, the basic paradigm of property testing allows arbitrary error in case the instance neither has the property nor is far from having it.

cycle-freeness may be used for finding cycles in graphs that are far from being cycle-free. A similar observation applies to finding T-minors, for any fixed tree T.

We mention that in most of the property testing literature, one-sided error is viewed as a secondary feature that some testers have and others may lack. The foregoing connection demonstrates the fundamental advantage of one-sided error testers over standard (two-sided error) testers. (Other advantages are discussed in Section 1.5.)

Lower bounds on the complexity of one-sided error testers that significantly exceeds the performance guarantees of known two-sided error testers have been observed, starting with [GGR98, Sec. 10.1.6]. However, so far, no study has been devoted to providing a *one-sided error tester of optimal complexity, in the case where this complexity significantly exceeds that of the corresponding two-sided error tester.*

To the best of our knowledge, the text that seems closest to addressing this issue is the discussion in [AS03, Sec. 2] that refers to the complexity of testing $K_{t,t}$-freeness in the *adjacency matrix model* (introduced in [GGR98]). Specifically, [AS03, Clm. 2.2] asserts a two-sided tester of $K_{t,t}$-freeness having query complexity $O(1/\epsilon)$, whereas [AS03, Clm. 2.3] (combined with [GT03, Thm. 2]) asserts that one-sided error testing of $K_{t,t}$-freeness requires $\Omega(\epsilon^{-t/4})$ queries. As noted at the end of [AS03, Sec. 2], this is tight up to a polynomial function (i.e., there exists two-sided tester of $K_{t,t}$-freeness having query complexity $\epsilon^{-O(t)} = \text{poly}(\epsilon^{-t/4})$). It is telling that [AS03, Sec. 2] leaves the complexity of one-sided error testing undetermined (at the "polynomial slackness" level). Indeed, like other prior works that address the complexity of one-sided error testers, their interest is in demonstrating the gap between the complexities of two-sided and one-sided error testing, and not in determining the latter.

In contrast, our work is aimed at providing one-sided error testers of (almost) optimal complexity, in cases in which this complexity significantly exceed the complexity of the corresponding two-sided error tester. For example, recall that Goldreich and Ron provided a two-sided error tester for cycle-freeness of $\text{poly}(1/\epsilon)$ query complexity [GR02, Thm. 4.2], where ϵ denotes the desired proximity parameter (i.e., the tester distinguishes cycle-free graphs from graphs that are ϵ-far from being cycle-free). In contrast, [GR02, Prop. 4.3] asserts that cycle-freeness has no one-sided error tester that makes $o(\sqrt{N})$ queries (even for $\epsilon = 1/3$), where N denotes the number of vertices in the input graph. In that context, Theorem 1.1 is equivalent to

Theorem 1.5 (one-sided error tester for cycle-freeness): *Cycle-freeness of constant degree N-vertex graphs can be tested with one-sided error within time complexity $\widetilde{O}(\text{poly}(d/\epsilon) \cdot \sqrt{N})$. Furthermore, whenever the tester rejects, it outputs a simple cycle of length* $\text{poly}(\epsilon^{-1} d \log N)$.

Indeed, by the foregoing discussion, whenever the tester asserted in Theorem 1.5 rejects, it is the case that it explored a subgraph that is not cycle-free. Moreover, the furthermore clause of Theorem 1.5 asserts that in this case the explored subgraph actually contains a simple cycle of length $\text{poly}(\epsilon^{-1} d \log N)$. Thus, Theorem 1.5 implies Theorem 1.1. Similarly, *Theorem 1.5 extends to testing C_k-minor freeness, for any $k > 3$, which in turn is equivalent to Theorem 1.2.* And, similarly, Theorem 1.3 is equivalent to the existence of a *tester for T-minor freeness of query complexity that only depends on the proximity parameter, for any tree T.*

1.3 Techniques

As stated at the end of Section 1.1, all our results are obtained via the study of the complexity of one-sided error testers for the corresponding properties.

Our testers for C_k-minor freeness are all obtained by local reductions. Specifically, our cycle-freeness (i.e., C_3-minor freeness) tester is obtained by a *randomized* reduction to testing bipartiteness, whereas our C_k-minor freeness tester is obtained by a *deterministic* reduction to testing cycle-freeness.

1.3.1 Testing cycle-freeness

We mention that the two-sided error cycle-freeness tester of [GR02] does not even try to find a simple cycle. It just estimates the number of edges in the graph and rejects if this estimate exceed the number of edges that correspond to any forest that spans the set of connected components of the graph.[3] We also mention that as observed by Bollobás and Thomason [BT97, Thm. 5], a "girth versus edge-density" lower bound implies that any graph $G = ([N], E)$ that is ϵ-far from being cycle-free (and hence contains $N + \Omega(\epsilon N)$ edges) must have a simple cycle of length $O(\log N + 1/\epsilon)$. The problem, however, is finding such a cycle in sublinear time.

Our one-sided error tester of cycle-freeness finds a cycle in the original graph by randomly reducing this problem to the problem of finding an odd-length cycle in an auxiliary graph. Specifically, the input graph $G = ([N], E)$ is randomly transformed into an auxiliary graph such that each edge $e \in E$ is replaced, with probability 1/2 by a 2-vertex path (with an auxiliary vertex), and remains intact otherwise. Thus, with probability 1/2, each cycle in G is transformed into an odd-length cycle. Furthermore, we show that *if G is ϵ-far from being cycle-free, then* (w.h.p.) *the resulting graph is $\Omega(\epsilon)$-far from being bipartite.*

A crucial feature of the foregoing randomized reduction is that it is local in the sense that each operation on the transformed graph can be implemented by a constant number of operations on the original graph. Thus, we can emulate the execution of a bipartite tester (i.e., the one of [GR99]) on the transformed graph. This allows us to establish Theorem 1.5.

1.3.2 Testing C_k-minor freeness, for any $k > 3$

Recall that the set of C_k-minor-free graphs coincides with the set of graphs that have no simple cycle of length at least k. Theorem 1.2 is proved by a (local) reduction of testing C_k-minor-freeness to testing cycle-freeness. For example, in the case of $k = 4$ we replace each triangle by a 3-vertex star; that is, we omit the original edges of this triangle, and introduce an auxiliary vertex that is connected to the three corresponding vertices. We then prove that if the original graph is C_4-minor-free then the resulting graph is cycle-free, whereas *if the original graph is ϵ-far from being C_4-minor-free then the resulting graph is $\Omega(\epsilon)$-far from being cycle-free.*

For larger values of k, a more sophisticated local replacement is used; that is, replacing all small cycles by auxiliary vertices will not do. To illustrate the difficulty of dealing with $k > 4$, note that, unlike in the case $k = 4$, a C_k-minor free graph may contain cycles of length smaller than k that share some common edges, and so the simple replacement will not yield a cycle-free graph.

[3]Note that any cycle-free graph is a forest, and if the number of trees in this forest is t, then the difference between the number of vertices and the number of edges in the graph equals t. The two-sided error tester of [GR02] estimates the number of edges and the number of connected components in the graph, and conducts the adequate computation. The number of connected components is estimated by the number of connected components that have $O(1/\epsilon)$ vertices, whereas the latter number is approximated by exploring the neighborhood of a few randomly selected vertices.

1.3.3 Testing H-minor freeness, for any cycle-free H

The main challenge for this problem is testing T-minor freeness, where T is an arbitrary tree. The simple case in which T is a k-vertex star, for some $k \geq 2$, provides a good illustration to the underlying main idea. In this case we may select a random vertex and start a Breadth First Search (BFS) at this vertex, stopping whenever either we encounter a layer with at least k vertices or we explored more than $4k/\epsilon$ layers (or we explored the entire connected component). In the first case, we found a desired minor and can safely reject, whereas in the second case we found a set of at least $4k/\epsilon$ vertices that is separated from the rest of the graph by less than dk edges. Thus, if the graph $G = ([N], E)$ contains at least $(1 - \epsilon/4) \cdot N$ start vertices that do not lead the algorithm to reject, then G can be decomposed to connected components that are each T-minor free by omitting at most $\epsilon dN/2$ edges (i.e., the edges that are incident at the $\epsilon N/4$ exceptional vertices and the edges of the aforementioned small cuts).

The case of a general tree T is much more complex, but the governing principle remains a tight relation between having few start vertices that contain a T-minor at their vicinity and the ability to decompose the graph to connected components with few edges between them. This relation is captured by the following result, which may be of independent interest.

Theorem 1.6 ("local expansion" and tree minors): *For every d and k there exists an $r = r(d, k)$ such that if the r-neighborhood of a vertex s in a graph of degree bound d does not contain a T-minor of some tree T with at most k vertices, then this neighborhood contains a set S that is separated from the rest of the graph by less than $\epsilon d|S|/4$ edges.*

In other words, if all "sub-neighborhoods" of the r-neighborhood of s are "expanding" (i.e., are not separated from the rest by small cuts), then this r-neighborhood contains a T-minor of every tree T with at most k vertices. (We mention that the problem of finding small trees in locally expanding graphs has been studied before (cf., e.g. [FP87]). However, our Theorem 1.6 seems incomparable, since we seek specific *tree minors* rather than specific trees, whereas our expansion condition is very weak.)

Finally, we reduce finding H-minors, where H is an arbitrary cycle-free graph (i.e., a forest), to finding disjoint tree minors. Again, the reduction is local, and in this case it is almost straightforward, where the subtlety is related to the fact that we refer to one-sided error. Specifically, if H consists of the connected components $H_1, \ldots, H_m$, then it does not necessarily hold that G is H-minor free if and only if G is H_i-minor free for all $i \in [m]$. Still, this is "almost true" and so a small modification of the straightforward reduction will do.

1.4 Another perspective: Finding arbitrary forbidden minors

Our results may be viewed as progress in resolving an open problem, posed by Benjamini, Schramm, and Shapira [BSS08], that refers to one-sided error testing of H-minor-freeness, for any finite graph H (or even a finite family of such graphs). Specifically, Benjamini *et al.* [BSS08] proved that, for any H, the property of being H-minor-free can be tested within query complexity that only depends on the proximity parameter,[4] *when allowing two-sided error.* They conjectured that for any non-forest H, there exists an H-minor-freeness tester with query complexity $O(\sqrt{N})$. Viewed from that

[4]The query complexity obtained in [BSS08] is triple-exponential in $1/\epsilon$. The complexity was improved to exponential in $1/\epsilon$ [HKNO09].

perspective, our results prove the aforementioned conjecture in the special case of $H = C_k$, for every $k \geq 3$.

We note that finding cycles seems the "hard" part of finding minors; that is, cycles are the source of the $\Omega(\sqrt{N})$ query lower bound. Specifically, recall that [GR02, Prop. 4.3] establishes an $\Omega(\sqrt{N})$ query lower bound for any algorithm that finds C_3-minors (or, in other words, a one-sided property tester for cycle-freeness). In [BSS08] it was suggested that this lower bound can be deduced by adapting the lower bound argument from [GR02]. We present a proof of this fact, thus establishing *an $\Omega(\sqrt{N})$ query lower bound for any algorithm that finds minors that contain cycles.* Recall that this stands in contrast to Theorem 1.3 (which asserts that finding cycle-free minors can be done in a number of queries that is independent of the size of the graph).

A wider perspective on finding forbidden minors. The first result dealing with graph minors is the well known Kuratowski-Wagner theorem [Kur30, Wag37] that states that any non-planar graph contains a K_5 or $K_{3,3}$ minor. Consider a property $\mathcal{P}$ such that if $G \in \mathcal{P}$, then, for any minor H of G, it holds that $H \in \mathcal{P}$. Such a property is *minor-closed.* It was conjectured by Wagner that for *any* minor-closed property $\mathcal{P}$, there is a finite set of graphs $\mathcal{H}_{\mathcal{P}}$ such that $G \in \mathcal{P}$ if and only if G is H-minor free, for all $H \in \mathcal{H}_{\mathcal{P}}$. Robertson and Seymour had a long series of papers, which culminated in the proof of this conjecture [RS04], called the Graph-Minor Theorem. From an algorithmic perspective, one of the milestones in this series was a polynomial time algorithm that checked H-minor freeness, for any (constant-size) graph H [RS95]. (We will use a recent improvement on that by Kawarabayashi, Kobayashi, and Reed [KKR12], which gives a quadratic time algorithm for this problem.)

It is natural to consider a sublinear variant of the above algorithmic question; that is, given a graph G that is far from being minor-free, can we find an H-minor by looking at a sublinear portion of the graph? An affirmative answer would, in particular, imply that such a graph contains sublinear sized H-minors, which is an interesting combinatorial conjecture. Needless to say, this paper provides an affirmative answer in the special case that H is a cycle.

1.5 Further reflections regarding one-sided error

The relative power of two-sided versus one-sided error randomized decision procedures has been the focus of considerable study in many settings, including in the context of property testing. Indeed, in any setting, one-sided error procedures offer the advantage of never rejecting yes-instances. However, as we already saw in Section 1.2, this advantage has a special appeal in the context of property testing, since it yields algorithms for very efficiently finding some desired structures (whenever the graph is far from being "free of them"). Additional benefits of one-sided error testers are discussed next.

Firstly, we note that property testing is asymmetric in nature: It refers to distinguishing objects that *perfectly* satisfy a predetermined property from objects that are *far* from satisfying this property. Indeed, property testing is a relaxation of the original decision task (which refers to distinguishing objects that satisfy the property from objects that do not satisfy it), where the relaxation is applied to one type of instances but not to the other. In this context, it is natural to apply the probabilistic relaxation also to one type of instances (i.e., the far-away instances) but not to the other.

Secondly, we note that one of the main applications of property testers is their potential use as a preliminary "fast but crude" decision step, which when coupled with an exact (but slow) decision procedure yields a procedure that is always correct and often very fast. That is, we envision using a property tester as a "sieve" that rejects "on the spot" (i.e., "fast") very bad instances (i.e., those that are far from satisfying this property), while passing the rest of the instances for further examination. In such a context, we can afford passing very bad instances for further examination (since all this means is a waste of time), but we cannot afford not passing a good instance.

Lastly, we consider the relationship between property testing and local structures in the tested property. Intuitively, the existence of a property tester means that a global structure (i.e., distance of the object to the property) is reflected in (or co-related with) a local structure (i.e., the part of the object being probed by the tester). In the general case (of two-sided error), this co-relation is statistical, whereas in the case of one-sided error this correlation is actually a ("robust") characterization.

The last aspect is particularly clear in the current study. Firstly, the notion of local structure is most appealing in the bounded-degree model, where it refers to graph neighborhoods. Secondly, the different types of local structures underlying the two-sided and one-sided error testers is most striking in the case of cycle-freeness. The two-sided error tester of [GR02] relies on the fact that distance from cycle-freeness in connected graphs is reflected by the difference between the number of edges and the number of vertices, whereas these numbers can be estimated (with two-sided error) by sampling the graph's vertices. Note that such *estimates* cannot yield a characterization (let alone a robust one) of the cycle-free graphs. In contrast, our one-sided error tester relies on the fact that distance from cycle-freeness is reflected in the density of short simple cycles in the graph, whereas such cycles can be found by an appropriate randomized exploration of the graph. Indeed, this yields a (robust) characterization of the set of cycle-free graphs (i.e., a graph is cycle-free if and only if it contains no simple cycle, and the farther the graph is from being cycle-free the shorter and more abundant these cycles are).

1.6 The general (unbounded-degree) graph model

Although our upper bounds (e.g., Theorem 1.1) state the dependence of the complexities on the degree bound, d, so far we thought of d as being a constant (or at least as being extremely small in comparison to N). Indeed, an upper bound as stated in Theorem 1.1 (i.e., an arbitrary polynomial dependence on d) is not meaningful, when $d = N$ (or even $d = \sqrt{N}$). Nevertheless, it is possible to obtain a better result than stated in Theorem 1.1 – specifically, eliminate the dependence on d. That is, there exists a randomized algorithm that, on input an N-vertex graph G of degree bound d that is ϵ-far from being cycle-free, finds a simple cycle (of length $\text{poly}(\epsilon^{-1} \log N)$) in G in expected time $\widetilde{O}(\text{poly}(1/\epsilon) \cdot \sqrt{N})$.

The foregoing algorithm can be extended to the general graphs model (i.e., the model in [PR02]), where distances are measured with respect to the actual number of edges (see Section 8).[5] This follows by an alternative presentation of the basic randomized reduction, which may be viewed as reducing cycle-freeness to a generalization of 2-colorability. In this generalization, edges of the graph are labeled by either `eq` or `neq`, and a legal 2-coloring (of the vertices) is one in which every two vertices that are connected by an edge labeled `eq` (resp. `neq`) are assigned the same color

[5]Algorithms in this model use the same type of incidence queries as in the main (bounded-degree) model we consider. The difference is that a graph $G = ([N], E)$ is said to be ϵ-far from H-minor-freeness if $2\epsilon|E|$ edges (rather than ϵdN edges) must be removed from G in order to obtain an H-minor-free subgraph. The point is that the number of edges is related to the average degree of G rather than to its degree (upper) bound, which may be significantly smaller. Thus, distances under this model are possibly larger, and thus the testing requirement is possibly harder.

(resp., opposite colors). We observe that the (one-sided error) Bipartite testers of [GR99, KKR04] extend to this generalization of 2-colorability.

We mention that analogous extensions do not work for testing C_k-minor freeness, for $k > 3$, nor for testing tree-minor-freeness. In fact, in the general graph model, it is not possible to find tree-minors (or even test tree-minor freeness with two-sided error) by using $o(\sqrt{N})$ queries.

1.7 Organization

Section 2 contains a formal statement of the relevant definitions and terminology. The testers of C_k-minor freeness are presented in Sections 3–5. Our first result (i.e., the one-sided error tester of cycle-freeness) is presented in Section 3. The reduction of testing C_k-minor freeness to testing cycle-freeness is presented in Section 5, but Section 4 provides an adequate warm-up by treating the case of $k = 4$.

In Section 6, we prove the lower bound claimed in [BSS08] regarding the query complexity of one-sided error testing H-minor freeness, when H contains a cycle. In contrast, in Section 7 we consider the case that H is cycle-free, and present the improved testers for H-minor freeness in this case (i.e., when H is a forest).

Finally, in Section 8 we consider the unbounded-degree model, discussed in Section 1.6, and in Section 9 we compile a list of open problems that are scattered throughout the paper.

2 Preliminaries

This work refers mainly to the bounded-degree model (introduced in [GR02]). The only exception is Section 8, where we consider the unbounded-degree model, also discussed in Section 1.6. The bounded-degree model refers to a fixed degree bound, denoted d, where a tester is given oracle access to an N-vertex graph $G = ([N], E)$ of maximum degree d. Specifically, for any $v \in [N]$ and $1 \leq i \leq d$, the tester can ask for the ith neighbor of vertex v. If v has less than i neighbors, then the answer returned is 0 (and no assumption is made on the order of the neighbors of any vertex).

Definition 2.1 (testers in the bounded-degree model): *Let $d \in \mathbb{N}$ be fixed and Π be a property of graphs with maximum degree at most d. We denote the restriction of Π to N-vertex graphs by Π_N. A randomized oracle machine T is called a* tester for Π *if the following two conditions hold:*

1. *For every $N \in \mathbb{N}$ and $\epsilon \in [0, 1]$, on input (N, ϵ) and when given oracle access to any $G \in \Pi_N$ the machine T accepts with probability at least $2/3$; that is, $\Pr[T^G(N, \epsilon) = 1] \geq 2/3$.*

2. *For every $N \in \mathbb{N}$ and $\epsilon \in [0, 1]$, and every N-vertex graph G that is ϵ-far from Π_N, it holds that $\Pr[T^G(N, \epsilon) = 1] \leq 1/3$, where $G = ([N], E)$* is ϵ-far from Π_N *if for every $G' = ([N], E') \in \Pi_N$ it holds that the symmetric difference of E and E' contains more than $\epsilon \cdot dN$ elements.*

In case the first condition holds with probability 1, we say that T has one-sided error*. Otherwise, we say that T has* two-sided error*.*

Throughout our study, the degree bound d is a constant, and sometimes O/Omega-notions hide a dependence on d. The query and time complexities of testers are stated as functions of the graph size, N, and the proximity parameter, ϵ. When discussing time complexity, basic vertex-manipulation operations are counted at unit cost. We may assume without loss of generality that

$d \geq 3$, where in order to obtain a result for $d = 2$, we can simply run the algorithm with $d = 3$ and a proximity parameter of $2\epsilon/3$ (and for $d = 1$ all problems become trivial).

Notation. For a graph $G = ([N], E)$, we denote the set of neighbors of $v \in [N]$ (in G) by $\Gamma_G(v)$; that is, $\Gamma_G(v) = \{u \in [N] : \{u, v\} \in E\}$.

Terminology. By a cycle in a graph $G = ([N], E)$ we mean a sequence of vertices $(v_1, \ldots, v_t, v_{t+1})$ such that $v_1 = v_{t+1}$ and for every $i \in [t]$ it holds that $\{v_i, v_{i+1}\} \in E$; that is, (u, v, w, v, u) (or even (u, v, u)) is considered a cycle. A simple cycle is a cycle as above in which $t \geq 3$ and $|\{v_i : i \in [t]\}| = t$.

A useful bound. For any positive integer a and fraction $0 < \alpha < 1/2$ we have:

$$\binom{a}{\alpha a} < 2^{H_2(\alpha) \cdot a} \tag{1}$$

where $H_2(\alpha) = \alpha \log(1/\alpha) + (1 - \alpha) \log(1/(1 - \alpha))$ is the binary entropy function.

3 Testing Cycle-Freeness

As stated in the introduction, we reduce testing cycle-freeness to testing bipartiteness. Recall that we consider bounded-degree graphs, where the degree bound d is assumed to be a constant (for the general case, see Section 8). The reduction is randomized and local so that operations in the resulting graph are easily implemented via operations in the original graph. Wishing to avoid a general definition of (randomized) local reductions, we explicitly present the tester obtained by it.

For a fixed graph $G = ([N], E)$ and function $\tau : E \to \{1, 2\}$, we denote by G_τ the graph obtained from G by replacing each edge $e \in E$ such that $\tau(e) = 2$ by a 2-edge path (with an auxiliary intermediate vertex). Each edge $e \in E$ such that $\tau(e) = 1$ remains an edge in G_τ. That is, the graph $G_\tau = (V_\tau, E_\tau)$ is defined as follows:

$$V_\tau \overset{\text{def}}{=} [N] \cup \{a_e : e \in E \wedge \tau(e) = 2\} \tag{2}$$

$$E_\tau \overset{\text{def}}{=} \{e : e \in E \wedge \tau(e) = 1\} \cup \{\{u, a_e\}, \{a_e, v\} : e = \{u, v\} \in E \wedge \tau(e) = 2\} \tag{3}$$

Note that $|V_\tau| \leq (d+1) \cdot N$ and that G_τ preserves the degree bound, d. We first establish the next lemma concerning features of the transformation from G to G_τ, and later turn to discuss the tester in detail.

Lemma 3.1 (analysis of the randomized transformation):

1. *If G is cycle-free, then, for every choice of $\tau : E \to \{1, 2\}$, the graph G_τ is bipartite.*

2. *If G is not cycle-free, then, with probability at least $1/2$ over the random choice of $\tau : E \to \{1, 2\}$, the graph G_τ is not bipartite.*

3. *There exist universal constants $c_1 > 1$ and $c_2, c_3 > 0$ such that, for every $\epsilon \geq c_1/(dN)$, if G is ϵ-far from being cycle free, then, with probability at least $1 - \exp(-c_2 \epsilon d N)$ over the random choice of $\tau : E \to \{1, 2\}$, the graph G_τ is $c_3 \cdot \epsilon/2d$-far from being bipartite.*

Proof: The first item follows from the fact that if G is cycle-free, then, for every $\tau : E \to \{1, 2\}$, the graph G_τ is also cycle-free, and thus bipartite. The second item follows by observing that any cycle in G is transformed with probability 1/2 to an odd-length cycle in G_τ. Turning to the last item, we consider an arbitrary graph G that is not cycle-free. Denoting by Δ the actual number of edges (not its fraction) that should be omitted from G in order to obtain a cycle-free graph, we shall show the following. For Δ that is at least some constant (i.e., $\Delta \geq c_1$), with probability $1-\exp(-\Omega(\Delta))$, the number of edges that should be omitted from G_τ in order to obtain a bipartite graph is $\Omega(\Delta)$. (Note that the second item in the lemma holds for any $\Delta \geq 1$, which may be below this constant.)

We start by considering the case that the graph G is connected. We later address the case in which G contains more than one connected component. We may assume without loss of generality that G has no vertices of degree 1, since removing such vertices maintains the value of Δ (i.e., the absolute distance from being cycle-free) as well as (the distribution of) the number of edges that have to be removed to make G_τ bipartite. We also observe that except in the case that G is a simple cycle, which is covered by the second item in the lemma, we may assume that there are no vertices of degree 2. This is true since we can contract paths that only contain intermediate vertices of degree 2 to a single edge, while again preserving Δ as well as (the distribution of) the number of edges that have to be removed to make G_τ bipartite. The latter assertion follows from the fact that the distribution of the parity of the path-lengths in G_τ is maintained (i.e., both the original path and the contracted path in G_τ have odd/even length with probability 1/2). We also mention that the contracted graph G may contain self-loops and parallel edges, but the rest of the argument holds in this case too. We stress that the contracted graph is merely a mental experiment for proving the current lemma.

In light of the foregoing, we consider a connected graph $G = ([N], E)$, which may have self-loops and parallel edges, in which each vertex has degree at least 3. It follows that $\Delta = |E| - (N-1) > N/2$. We shall prove that, with high probability over the choice of τ, for some constant $c_3 > 0$, more than $c_3 \cdot \Delta \geq c_3 \epsilon d N$ edges must be omitted from the graph G_τ in order to obtain a bipartite graph. Since the number of vertices in G_τ is upper bounded by $(d+1)N$ (and its degree bound is d), we get that G_τ is at least $(c_3\epsilon/2d)$-far from bipartite, since $\frac{c_3 \epsilon d N}{d(d+1)N} > \frac{c_3\epsilon}{2d}$.

For each $E' \subset E$ of size $c_3\Delta$, let G'_τ denotes the subgraph of G_τ obtained by applying the foregoing randomized reduction to the graph $G' = ([N], E \setminus E')$ rather than to $G = ([N], E)$. We consider the probability that G'_τ is bipartite. Note that G_τ is at (absolute) distance at most $c_3\Delta$ from being bipartite if and only if there exists a set E' of size $c_3\Delta$ such that G'_τ is bipartite. Thus, the probability that G_τ is at distance at most $c_3\Delta$ from being bipartite is given by

$$\begin{aligned}
p &\stackrel{\text{def}}{=} \Pr_\tau[\exists E' \subset E \text{ such that } |E'| = c_3\Delta \text{ and } G'_\tau \text{ is bipartite}] \\
&\leq \sum_{E' \subset E :\, |E'| = c_3\Delta} \Pr_\tau[G'_\tau \text{ is bipartite}] \\
&\leq \binom{|E|}{c_3\Delta} \cdot 2^{N-1} \cdot 2^{-(|E| - c_3\Delta)}
\end{aligned}$$

where the second inequality is due to considering all possible 2-partitions of $[N]$, and noting that for each edge e in $E \setminus E'$ and each 2-partition π, with probability 1/2 over the choice of $\tau(e) \in \{1, 2\}$, the partition π is inconsistent with the value of $\tau(e)$. (In such a case we say that e violates the 2-partition π.) Specifically, if $\pi(u) = \pi(v)$ and $\tau(\{u, v\}) = 1$, then the edge $\{u, v\}$ violates the

2-partition π, and ditto if $\pi(u) \neq \pi(v)$ and $\tau(\{u,v\}) = 2$. Note that the hypothesis that G is (connected and is) at (absolute) distance Δ from being cycle-free implies that $|E| = (N-1) + \Delta$. Now, substituting $|E|$ by $(N-1)+\Delta$, using $\Delta \geq N/2$ and Equation (1) we get

$$\begin{aligned} p &\leq \binom{N-1+\Delta}{c_3\Delta} \cdot 2^{-(\Delta - c_3\Delta)} \\ &< \binom{3\Delta}{c_3\Delta} \cdot 2^{-(1-c_3)\Delta} \\ &< 2^{H_2(c_3/3)\cdot 3\Delta - (1-c_3)\Delta} \end{aligned}$$

which vanishes exponentially in Δ provided that $c_3 > 0$ is a sufficiently small constant.

It remains to address the case in which G is not connected. Let $C_1, \ldots, C_t$ be the connected components of G where $t > 1$. For each $1 \leq i \leq t$, let N_i be the number of vertices in C_i, and let Δ_i be the number of edges that should be removed from C_i in order to make it cycle-free. Thus, $\sum_{i=1}^t N_i = N$ and $\sum_{i=1}^t \Delta_i = \Delta$. Let τ_i be the restriction of τ to the edges in C_i, let $C_{i,\tau}$ be the graph obtained by applying the transformation defined by τ_i to C_i, and let $\Delta_{i,\tau}$ be the number of edges that should be removed from C_i to make it bipartite.

By applying the argument detailed above to each C_i separately, we get that $\Pr[\Delta_{i,\tau} < c_3\Delta_i] \leq 2^{-c_4\Delta_i}$ (for constants $0 < c_3, c_4 < 1$). We would like to infer that

$$\Pr\left[\sum_{i=1}^t \Delta_{i,\tau} < c_3' \sum_{i=1}^t \Delta_i\right] \leq \exp\left(-\Omega\left(\sum_{i=1}^t \Delta_i\right)\right) = \exp(-\Omega(\Delta)) \tag{4}$$

for some constant c_3'. To this end, for each C_i we define $m_i = c_3\Delta_i$ independent 0/1 random variables, $X_{i,1}, \ldots, X_{i,m_i}$, such that $\Pr[X_{i,j} = 1] = 2^{-c_4/c_3}$. Observe that $\Pr[\sum_{j=1}^{m_i} X_{i,j} \leq m_i] = 1$ and $\Pr[\sum_{j=1}^{m_i} X_{i,j} = 0] = 2^{-c_4\Delta_i}$. Thus, for any $\beta > m_i$, we have $\Pr[\sum_{j=1}^{m_i} X_{i,j} < \beta] = 1$, whereas for any $\beta \leq m_i$ we have $\Pr[\sum_{j=1}^{m_i} X_{i,j} < \beta] \geq 2^{-c_4\Delta_i} \geq \Pr[\Delta_{i,\tau} < \beta]$. This implies that $\Pr[\sum_{j=1}^{m_i} X_{i,j} < \beta] \geq \Pr[\Delta_{i,\tau} < \beta]$ for every threshold β, which means that the random variables $\sum_{j=1}^{m_i} X_{i,j}$ and $\Delta_{i,\tau}$ can be coupled (i.e., defined over the same sample space) such that the value of the first is always upper bounded by the value of the second. Hence, in order to prove Equation (4), it suffices to bound the probability that $\sum_{i=1}^t \sum_{j=1}^{m_i} X_{i,j} < c_3' \sum_{i=1}^t \Delta_i = c_3'\Delta$. But since these ($\sum_{i=1}^t m_i = c_3\Delta$) random variables are independent, we can apply a multiplicative Chernoff bound, which gives us that the probability that $\sum_{i=1}^t \sum_{j=1}^{m_i} X_{i,j} < c_3'\Delta$ for $c_3' = c_3 \cdot 2^{-c_4/c_3 - 1}$ (i.e., half the expected value of the sum), is $\exp(-\Omega(\Delta))$. ■

The Tester For Cycle-Freeness. The tester emulates the execution of the bipartiteness testing algorithm [GR99] on G_τ by performing queries to G. We next state the main theorem proved in [GR99].

Theorem 3.2 [GR99] *There exist an algorithm* Test-Bipartite *for testing bipartiteness of bounded-degree graphs whose query complexity and running time are* $\mathrm{poly}((\log \tilde{N})/\tilde{\epsilon}) \cdot \sqrt{\tilde{N}}$ *where* $\tilde{N}$ *denotes the number of vertices in the graph and* $\tilde{\epsilon}$ *is the given proximity parameter. The algorithm uniformly selects random vertices and performs random walks from them. Whenever the algorithm rejects a graph it outputs a* certificate *to the non-bipartiteness of the graph in form of an odd-length cycle of length* $\mathrm{poly}(\tilde{\epsilon}^{-1} \log \tilde{N})$.

(Indeed, the foregoing complexities are independent of the degree bound; cf. [KKR04].) As stated in Theorem 3.2, algorithm Test-Bipartite performs two types of operations: (1) selecting a vertex uniformly at random, and (2) taking random walks by querying vertices on their neighbors.[6] Thus the execution of the cycle-freeness tester boils down to emulating these operations, as described next.

Algorithm 3.3 (the cycle-freeness tester): *Given input graph $G = ([N], E)$, the tester selects uniformly at random a function $\tau : E \to \{1, 2\}$ and invokes* Test-Bipartite *on the graph G_τ with the proximity parameter set to $c_3\epsilon/2d$* (where c_3 is the constant from the last item in Lemma 3.1), *emulating its operations as follows.*[7]

1. *If* Test-Bipartite *wishes to select a random vertex in G_τ, then the tester first selects uniformly a vertex $v \in [N]$. It then outputs v with probability $1/(d + 1)$, and otherwise it selects each neighbor of v with probability $1/2d$ and outputs $a_{\{u,v\}}$ if $\tau(\{u, v\}) = 2$, where u denotes the selected neighbor. Thus, v is output with probability $\frac{1}{d+1}$, and each $a_{\{u,v\}}$ such that $\tau(\{u, v\}) = 2$ is output* (here) *with probability $(1 - \frac{1}{d+1}) \cdot \frac{1}{2d} = \frac{1/2}{d+1}$.*

   Indeed, the foregoing process outputs a vertex in G_τ with probability at least $1/(d+1)$, and in case no vertex is output, the procedure is repeated (up to $O(\log N)$ times).

2. *If* Test-Bipartite *queries for the i^{th} neighbor of vertex $v \in [N] \subseteq V_\tau$, then the tester queries for the i^{th} neighbor of v in G, and answers accordingly. Specifically, if the answer to this query was u* (i.e., u is the i^{th} neighbor of v in G), *then u is given to* Test-Bipartite *if $\tau(\{u, v\}) = 1$ and otherwise $a_{\{u,v\}}$ is given.* (If the answer was 0, indicating that v has less than i neighbors, then 0 is returned as answer to Test-Bipartite.)

   Finally, if Test-Bipartite *queries for the i^{th} neighbor of a vertex $a_{\{u,v\}}$ such that $u < v$, then the tester answer with u if $i = 1$, with v if $i = 2$, and with 0 if $i > 2$.*

When Test-Bipartite *halts, the current tester halts with the same verdict.*

Furthermore, if Test-Bipartite provides an odd-length cycle in G_τ, then we can easily obtain a corresponding cycle in G (by contracting the 2-vertex paths that appear on it into single edges).

Note that in each iteration of the process detailed in Step 1, each vertex of G_τ (regardless if it is an original vertex of G or an auxiliary vertex) is output with probability exactly $\frac{1}{N} \cdot \frac{1}{d+1}$ (and with probability $1 - |V_\tau| \cdot \frac{1}{N} \cdot \frac{1}{d+1}$ no vertex is output), Thus, conditioned on a vertex being selected in Step 1 (which happens with very high probability since the process is repeated sufficiently many times), Step 1 implements a uniform random selection of vertices in G_τ.

Conclusion. Combining Lemma 3.1 with Theorem 3.2 we conclude that Algorithm 3.3 is a one-sided error tester for cycle-freeness. Its complexity is $\widetilde{O}(\text{poly}(d/\epsilon) \cdot \sqrt{N})$ and if it rejects the graph G then it outputs a cycle of length $\text{poly}(\epsilon^{-1} d \log N)$. This establishes Theorem 1.5.

[6] Actually, Test-Bipartite requires also a rough estimate of the number of vertices in the graph, since such an estimate is used to determine a couple of parameters (i.e., the number of random walks performed and their length). It is clear that our reduction provides such an estimate, since $|V_\tau| = \Theta(N)$.

[7] Actually, the function $\tau : E \to \{1, 2\}$ is selected on-the-fly; that is, whenever the tester needs the value of τ on some edge in E, it retrieves it from its memory in case it was determined already and selects it at random (and stores it for future use) otherwise.

4 Testing C_4-Minor-Freeness

As a warm-up towards testing C_k-minor-freeness, for any $k \geq 3$, we present the treatment of the special case of $k = 4$. We actually reduce the task of *testing C_4-minor-freeness* to the task of *testing C_3-minor-freeness*. Loosely speaking, the reduction replaces each triangle $\{u, v, w\}$ in the input graph by an auxiliary vertex (denoted $\bigtriangledown_{\{u,v,w\}}$) that is connected to the corresponding three vertices. The reduction is summarized in the following construction.

Construction 4.1 (the reduction): *Given a graph $G = ([N], E)$ (of max degree d), we* (locally) *construct the auxiliary graph $G' = ([N] \cup T, E')$ such that T contains the vertex $\bigtriangledown_{\{u,v,w\}}$* (referred to as a "triangle" vertex) *if and only if $\{u, v\}, \{v, w\}, \{w, u\} \in E$ and*

$$E' = \left(E \setminus \left(\bigcup_{u,v,w:\bigtriangledown_{\{u,v,w\}} \in T} \{u, v\}\right)\right) \cup \left\{\{u, \bigtriangledown_{\{u,v,w\}}\} : \bigtriangledown_{\{u,v,w\}} \in T\right\}. \tag{5}$$

Specifically, the set of neighbors of $v \in [N]$ in G', denoted $\Gamma_{G'}(v)$, consists of the following elements of $[N] \cup T$.

1. *Neighbors of v in G that do not reside in G on a triangle together with v; that is, $u \in \Gamma_G(v)$ is in $\Gamma_{G'}(v)$ if and only if $\Gamma_G(u) \cap \Gamma_G(v) = \emptyset$.*
2. *Each triangle that contains v in G; that is, $\bigtriangledown_{\{u,v,w\}}$ is in $\Gamma_{G'}(v)$ if and only if $u, w \in \Gamma_G(v)$ and $\{w, u\} \in E$.*

The set of neighbors of $\bigtriangledown_{\{u,v,w\}} \in T$ equals $\{u, v, w\}$. Noting that $d + \binom{d}{2} \leq d^2$, we view G' as a graph of maximal degree d^2.

For an illustration of Construction 4.1 see Figure 1. Note that given any $v \in [N]$, we can easily

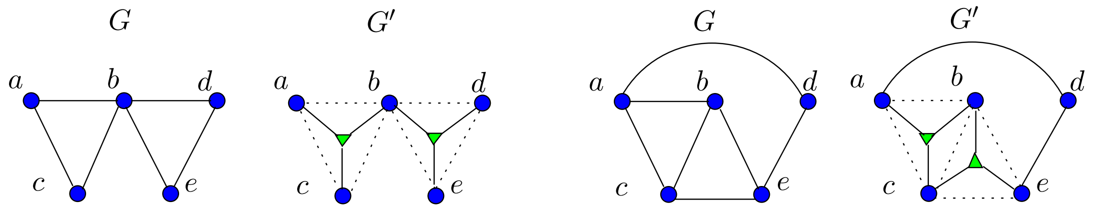

Figure 1: An illustration for Construction 4.1. On the left, G is C_4-minor free, and indeed G' is cycle-free; while on the right, G is not C_4-minor free, and G' contains cycles (but no cycles of length 3 (triangles).)

determine its neighbors in G' by checking the foregoing conditions. Similarly, for every u, v, w, we can easily determine whether $\bigtriangledown_{\{u,v,w\}}$ is in G'. Lastly, note that we can select a vertex of G' uniformly by using the following procedure.

1. Select uniformly $v \in [N]$.
2. Select one of the following two instructions at random with equal probability.

(a) (Generating a vertex of G):
Output v with probability d^{-2}.

(b) (Generating a triangle):
Query all neighbors of v to obtain $\Gamma_G(v)$, and select uniformly $u, w \in \Gamma_G(v)$ such that $u \neq w$. If $\{u, w\} \in E$, then output $\triangledown_{\{u,v,w\}}$ with probability $p_v = d^{-2} \cdot \binom{|\Gamma_G(v)|}{2}/3$.

In all the other cases, there is no output.

Thus, this process outputs each vertex of G with probability $N^{-1} \cdot 0.5 \cdot d^{-2} = d^{-2}/2N$, and outputs each $\triangledown_{\{u,v,w\}} \in T$ with probability $\sum_{x\in\{u,v,w\}} N^{-1} \cdot 0.5 \cdot \binom{|\Gamma_G(x)|}{2}^{-1} \cdot p_x = d^{-2}/2N$. Since there are at least N vertices in G', the probability that the process does not output *any* vertex in G' is at most $(1-d^{-2})$. If we repeat the process $\Theta(\log N)$ times (recall that d is assumed to be a constant), then the probability that we get no output is $1/\text{poly}(N)$. Since the total size of the sample needed is $o(N)$, by a union bound, the probability that this occurs at any step of the algorithm, is negligible, and this can be accounted for in the one-sided error probability by letting the algorithm accept in case sampling fails.

Algorithm 4.2 (the C_4-minor-freeness tester): *Given input graph $G = ([N], E)$, the tester emulates the execution of Algorithm 3.3 on the graph $G' = ([N] \cup T, E')$ as defined in Construction 4.1. In the emulation, vertices of G' are selected at random and their neighbors are explored on the fly, as detailed above.*

The analysis of Algorithm 4.2 reduces to an analysis of Construction 4.1.

Claim 4.3 *If G is C_4-minor-free, then G' is cycle-free.*

Proof: We first give a high-level idea of the proof and then give a detailed argument. By the hypothesis, the only simple cycles in G are triangles, and they are replaced in G' by stars centered at auxiliary vertices. Specifically, the triangle $\{u, v, w\}$ (i.e., the edges $\{u, v\}, \{v, w\}, \{w, u\}$) is replaced by a star-tree centered at $\triangledown_{\{u,v,w\}}$ and having the leaves u, v, w. Note that this replacement can form no simple cycles in G', because the simple paths in G' correspond to simple paths in G (where the sub-path v—$\triangledown_{\{u,v,w\}}$—$w$ corresponds to the edge v—w).

The corresponding detailed argument proceeds as follows. Assume, contrary to the claim, that there exists a simple cycle $\psi' = v_1$—v_2–$\cdots$–v_t—$v_{t+1} = v_1$ in G'. Consider replacing each length-2 subpath u—$\triangledown_{\{u,w,x\}}$—$w$ in ψ' by the edge (in G) between u and w (where this edge exists because u and w belong to a common triangle and $u \neq w$). Since, by construction of G', there are no edges in G' between *triangle* vertices, this way we obtain a cycle in G, which we denote by ψ. We next show that ψ is a simple cycle of length greater than 3, and we reach a contradiction to the hypothesis that G is C_4-minor-free.

We first verify that the length of ψ is greater than 2. This is true because otherwise, the cycle ψ' is either of the form u—$\triangledown_{\{u,w,x\}}$—$w$—$u$, or it is of the form u—$\triangledown_{\{u,w,x_1\}}$—$w$—$\triangledown_{\{u,w,x_2\}}$—$u$. In the first case ψ' contains an edge $\{w, u\}$ of a triangle in G, which is not possible by construction of G'. In the second case, since ψ' is simple (so that $x_1 \neq x_2$), there is a simple 4-cycle u—x_1—w—x_2—u in G (contradicting the hypothesis that G is C_4-minor-free). It follows that ψ is a simple cycle and it remains to verify that its length is greater than 3.

Suppose that the length of ψ is 3, that is, $\psi = u$—w—v—u is a triangle in G. It follows that none of the edges $\{u, w\}, \{w, v\}, \{v, u\}$ belong to G' and therefore, $\psi' = u$— $\bigtriangledown_{\{u,w,x_1\}}$ —w— $\bigtriangledown_{\{w,v,x_2\}}$ —v— $\bigtriangledown_{\{v,u,x_3\}}$ —u, where the triangles are distinct and hence at least one of them does not equal $\bigtriangledown_{\{u,w,v\}}$. But this implies that there exists a simple 4-cycle in G (contradicting the hypothesis that G is C_4-minor-free). ■

Claim 4.4 *If G is ϵ-far from being C_4-minor-free, then G' is $\Omega(\epsilon)$-far from being cycle-free, where the Omega-notation hides a polynomial in $1/d$.*

Proof: Suppose that G' is δ-close to being cycle-free, where the distance refers to the degree bound of G', which is d^2 as well as the number of vertices in G' which is $N + |T|$. Let R' be a set of at most $\delta \cdot d^2 \cdot (N + |T|)$ edges such that removing R' from G' yields a cycle-free graph, $([N] \cup T, E' \setminus R')$. Let $R \subseteq E$ be a set of edges that consists of (1) all edges of E that are in R', and (2) each edge $\{u, v\} \in E$ such that $\{u, \bigtriangledown_{\{u,v,w\}}\}$ is in R'. Hence, $|R| \leq 2|R'| < 2\delta \cdot d^4 N$, where we use $|T| \leq \binom{d}{2} \cdot N$. We next prove that removing R from G yields a graph that is C_4-minor-free, and it follows that G is $2d^3\delta$-close to being C_4-minor-free.

Assume, contrary to the claim, that for some $t \geq 4$ there exists a *simple* cycle v_1—v_2–$\cdots$–v_t—v_1 in the resulting graph (i.e., in the graph $([N], E\setminus R)$). We consider the corresponding (not necessarily simple) cycle in the graph $([N] \cup T, E' \setminus R')$:

Case 1: If the edge $\{v_i, v_{i+1}\} \in E \setminus R$ is not a part of any triangle in G, then $\{v_i, v_{i+1}\} \in E' \setminus R'$, because $\{v_i, v_{i+1}\}$ is an edge of G' and it cannot be in R' (since this would imply that $\{v_i, v_{i+1}\} \in R$). In this case, we just use the edge $\{v_i, v_{i+1}\}$ on the cycle in the graph $([N] \cup T, E' \setminus R')$.

Case 2: If the edge $\{v_i, v_{i+1}\} \in E \setminus R$ is part of a triangle v_i, v_{i+1}, w (in G), then $\{v_i, \bigtriangledown_{\{v_i,v_{i+1},w\}}\} \in E'\setminus R'$ and $\{v_{i+1}, \bigtriangledown_{\{v_i,v_{i+1},w\}}\} \in E'\setminus R'$, because both pairs are edges of G' and cannot be in R' (since this would imply that $\{v_i, v_{i+1}\} \in R$). In this case, we replace the edge $\{v_i, v_{i+1}\} \in E\setminus R$ by the length-two-path v_i— $\bigtriangledown_{\{v_i,v_{i+1},w\}}$ —v_{i+1} (in the graph $([N] \cup T, E' \setminus R')$).

Observe that the "triangle" vertices used in Case (2) need not be distinct, but they can collide only when they refer to three consecutive vertices on the original t-cycle (i.e., if $\bigtriangledown_{\{v_i,v_{i+1},w_1\}} = \bigtriangledown_{\{v_j,v_{j+1},w_2\}}$, for $i < j$, then $v_j = v_{i+1}$ must hold, and $w_1 = v_{j+1} = v_{i+2}$ follows). Such collisions can be eliminated at the cost of omitting a single "non-triangle" vertex (i.e., the path v_i—$\bigtriangledown_{\{v_i,v_{i+1},v_{i+2}\}}$ —v_{i+1}— $\bigtriangledown_{\{v_i,v_{i+1},v_{i+2}\}}$ —v_{i+2} is replaced by the path v_i— $\bigtriangledown_{\{v_i,v_{i+1},v_{i+2}\}}$ —v_{i+2}). Thus, we derive a simple cycle of length at least $t \geq 4$ in the graph $([N]\cup T, E' \setminus R')$ (since we have a "triangle" vertex per each omitted "non-triangle" vertex). This contradicts the hypothesis that $([N] \cup T, E' \setminus R')$ is cycle-free, and so the claim follows. ■

Conclusion. Combining Claims 4.3 and 4.4 with Theorem 1.5 and the fact that the number of vertices in G' is linear in N (and polynomial in d), we conclude that *there exists a one-sided error tester of complexity $\widetilde{O}(\text{poly}(d/\epsilon) \cdot \sqrt{N})$ for C_4-minor-freeness.*

5 Testing C_k-Minor-Freeness, for any $k \geq 4$

In this section we show that, for any $k \geq 4$, the task of *testing C_k-minor-freeness* reduces to the task of *testing C_3-minor-freeness*. The reduction extends the ideas underlying the reduction of *testing C_4-minor-freeness* to *testing C_3-minor-freeness* (as presented in Section 4).

The basic idea of the reduction is replacing simple cycles that have length smaller than k by stars. Actually, we replace certain subgraphs that contain such cycles by stars. We start by defining the class of (induced) subgraphs that we intend to replace by stars. These subgraphs (or rather their vertex sets) will be called *spots*. Below, the term 2-connectivity means *2-vertex connectivity*; that is, a graph is called 2-connected if every two vertices in the graph can be connected by two vertex-disjoint paths.

Definition 5.1 (spots): *A set $S \subseteq [N]$ is called a k-spot of the graph $G = ([N], E)$ if the following three conditions hold:*

1. *The subgraph induced by S, denoted G_S, contains no simple cycle of length at least k; that is, G_S is C_k-minor-free.*
2. *The subgraph induced by S is 2-connected and $|S| \geq 3$.*
3. *For every $u, v \in S$ such that $u \neq v$, either u and v are not connected by any path that is external to G_S or the length of every such external path is at least $\ell(k) \overset{\text{def}}{=} 2k$. Here, by a path external to G_S we mean a path that does not use any edge that is incident to a vertex in S with the exception of the endpoints u and v* (i.e., all intermediate vertices of the path belong to $[N] \setminus S$).

For example, every 4-spot of G induces a triangle in G, whereas the set of possible subgraphs induced by 5-spots of G consists of the following graphs: the 4-cycle (i.e., C_4), the 4-cycle augmented by a chord, the 4-clique (i.e., K_4), and the graphs $K_{2,n}$ and $K'_{2,n}$ for every $n \geq 3$, where $K'_{2,n}$ is the graph $K_{2,n}$ augmented by a single edge that connects the two vertices on the small side.[8] (Indeed, in Section 4 we essentially used a relaxed notion of a 4-spot in which the third condition was not required.)

5.1 Some basic facts regarding spots

Since k is fixed throughout the rest of our discussion, we may omit it from the notations and refer to k-spots as spots. A few basic properties of spots are listed below.

Claim 5.2 *If S is a k-spot of G, then the diameter of G_S is smaller than $k/2$.*

It follows from the claim that for every k-spot S where $k \geq 4$,

$$|S| \;<\; \sum_{i=0}^{k/2} d^i \;<\; 2d^{k/2} \;<\; d^{k-1} \tag{6}$$

[8] Recall that $K_{m,n}$ denotes the complete bipartite graph with m vertices on one side and n vertices on the other side; that is, $K_{m,n} = ([m+n], \{\{i, m+j\} : i \in [m], j \in [n]\})$.

(since $d \geq 3$).[9]

Proof: Assume, contrary to the claim that the diameter of G_S is at least $k/2$ and consider $u, v \in S$ such that the distance between u and v in G_S is at least $k/2$. Since G_S is 2-connected, there exists a simple cycle in G_S that passes through both u and v, and it follows that this cycle has length at least k, which contradicts the hypothesis that G_S is C_k-minor-free. ■

Note that, for any spot S and every three distinct vertices $u, v, w \in S$, the subgraph G_S contains a simple path that goes from u to v via w. This holds by the very fact that G_S is 2-connected (i.e., the second condition in Definition 5.1). By Claim 5.2 the length of this path is less than d^{k-1}. As we shall show next, a much better bound follows by using the fact that G_S is C_k-minor-free (i.e., the first condition in Definition 5.1),

Claim 5.3 *For every k-spot S and distinct vertices $u, v, w \in S$, the subgraph G_S contains a simple path of length at most $2k - 1$ that goes from u to v via w.*

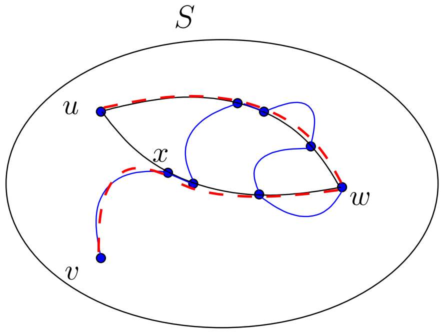

Figure 2: An illustration for the proof of Claim 5.3. The jotted line is the path between u and v that passes through w.

Proof: We just take a closer look at the standard proof that the fact that a graph is 2-connected implies the existence of a $u\text{–}\cdots\text{–}w\text{–}\cdots\text{–}v$ path (for every three vertices u, v, w in the graph). For an illustration of the argument that follows, see Figure 2. The proof starts by considering two different vertex-disjoint $u\text{–}\cdots\text{–}w$ paths, and an arbitrary path between v and w. In the current case (i.e., by C_k-minor-freeness), we may assume that the total length of the first two paths is smaller than k. Similarly, without loss of generality, the length of the third path is smaller than k. Proceeding as in the standard proof, we ask whether the third path (i.e., the $v\text{–}\cdots\text{–}w$ path) intersects both the $u\text{–}\cdots\text{–}w$ paths. If the answer is negative, then we are done (as we obtain the desired simple path by concatenating the path $v\text{–}\cdots\text{–}w$ to the $w\text{–}\cdots\text{–}u$ path that does not intersect it).

Otherwise, let x be the "closest to v" vertex on the path $v\text{–}\cdots\text{–}w$ that appear on either of the $u\text{–}\cdots\text{–}w$ paths; that is, x is on one of the $u\text{–}\cdots\text{–}w$ paths and the sub-path $v\text{–}\cdots\text{–}x$ (of the path $v\text{–}\cdots\text{–}w$) contains no vertex from either the $u\text{–}\cdots\text{–}w$ paths. Note that $x = v$ is possible (but

[9]We mention that there may exists spots of size $d^{(k-1)/2}$. Consider, for example, a graph that consists of two copies of a depth $(d-1)$-ary tree of depth $(k-1)/2$ such that each vertex in one tree is connected to its mirror vertex in the second tree. To see that this graph is C_k-minor-free, consider the correspondence between cycles on this graphs and traversals of parts of the original tree, and note that simple cycles correspond to traversals in which each edge is used at most twice. Since such traversals have length at most twice the depth of the tree, the claim follows.

$x = w$ is not), and assume, w.l.o.g., that x resides on the first $u\text{–}\cdots\text{–}w$ path. Then, consider the path obtained by combining the following three path segments: (1) the segment $v\text{–}\cdots\text{–}x$ of the path $v\text{–}\cdots\text{–}w$, (2) the segment $x\text{–}\cdots\text{–}w$ of the first $u\text{–}\cdots\text{–}w$ path, and (3) the second $u\text{–}\cdots\text{–}w$ path. Note that the total length of this path is at most $2(k-1)$ (i.e., the total length of the three paths), and that the three segment do not intersect (since the $v\text{–}\cdots\text{–}x$ segment does not intersect the $x\text{–}\cdots\text{–}w$ segment nor the $u\text{–}\cdots\text{–}w$ path by the choice of x). ■

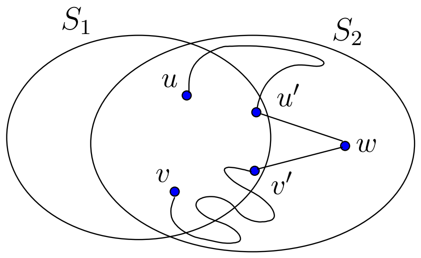

Figure 3: An illustration for the proof of Claim 5.4.

Claim 5.4 *If $S_1 \neq S_2$ are k-spots of G, then $|S_1 \cap S_2| \leq 1$.*

Proof: Assume, contrary to the claim that $|S_1 \cap S_2| > 1$ for two k-spots $S_1 \neq S_2$. Consider (w.l.o.g.) $u, v \in S_1 \cap S_2$ such that $u \neq v$ and $w \in S_2 \setminus S_1$ (as in Figure 3). By Claim 5.3, the subgraph G_{S_2} contains a simple path of length at most $2k-1$ that goes from u to v via w. Let u' (resp., v') be the last (resp., first) vertex of S_1 that appears on this path before reaching w (resp., after leaving w). Then, we get a simple path (in G) from $u' \in S_1$ to $v' \in S_1 \setminus \{u'\}$ such that this path contains only intermediate vertices of $S_2 \setminus S_1$. Recalling that this path has length at most $2k-1$, we reach a contradiction to the hypothesis that S_1 is a k-spot (specifically to the third condition of Definition 5.1). ■

As a corollary of Claim 5.4 we get:

Corollary 5.5 *Every vertex v may belong to at most $|\Gamma(v)|/2$ spots and hence the number of spots in a graph G is upper-bounded by the number of edges in G.*

Proof: The second part of the corollary follows directly from the first, and so we only need to establish the first part. Since by the definition of a spot, it must contain at least 3 vertices, every spot S that contains a vertex v must also contain at least two of v's neighbors. However, by Claim 5.4, spots that contain v may not share any other vertex. ■

Claim 5.6 *Each simple cycle in any C_k-minor-free graph G is a subset of some k-spot of G.*

Proof: Consider the following iterative process of constructing a spot S that contains the aforementioned cycle. Initially, we set S to equal the set of vertices that reside on this cycle. Clearly, this set S satisfies the first two conditions of the definition of a spot (i.e., Definition 5.1), which is an invariant that we shall maintain throughout the iterative process. The process ends once all three conditions are satisfied. Since the size of the spot increases in each iteration, the process must eventually end. Thus, at the start of each iteration of the process we have a set S that satisfies the first two conditions in Definition 5.1 but does not satisfy the third condition. That is, there exists a simple path external to S that connects two of its vertices $u, v \in S$. Adding this path to S we obtain a new set that satisfies Condition 1 (since G is C_k-minor-free). To see that the new set satisfies Condition 2, we need to show that there exist two disjoint paths between each pair of vertices that are not both in S. For an illustration of the argument that follows, see Figure 4.

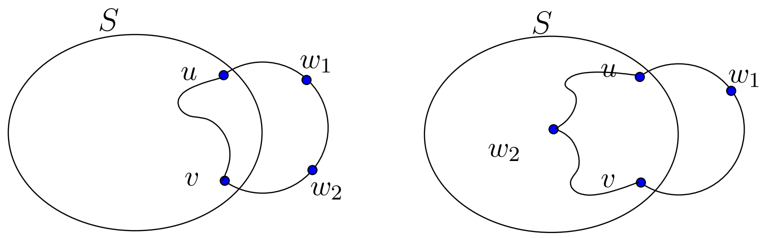

Figure 4: An illustration for the proof of Claim 5.6.

In the case that w_1 and w_2 are both new vertices (which reside on the aforementioned S-external path), we connect them by the direct path that resides outside of S as well as by a simple path that (without loss of generality) connects w_1 to u (via the external path), connects u and v via S, and connects v and w_2 (via the external path). In the case that w_1 is new but $w_2 \in S$, we use the external path to connect w_1 to u and v, respectively, and use the fact that there are vertex disjoint paths in G_S that connect u and v to w_2. Thus the new set satisfies the first two conditions in Definition 5.1, as desired. ■

5.2 The actual reduction

Using these facts, we are ready to present our reduction.

Construction 5.7 (the reduction): *Given a graph $G = ([N], E)$ (of max degree d), we* (locally) *construct the auxiliary graph $G' = ([N] \cup \{\langle S\rangle : S \in \mathcal{S}\}, E')$ such that $\mathcal{S}$ is the set of all spots of G and*

$$E' = \Big(E \setminus \Big(\bigcup_{S\in\mathcal{S}} \{\{u,v\} : u, v \in S\}\Big)\Big) \cup \Big\{\{v, \langle S\rangle\} : S \in \mathcal{S}, v \in S\Big\}. \tag{7}$$

Specifically, the set of neighbors of $v \in [N]$ in G', denoted $\Gamma_{G'}(v)$, consists of the following elements of $[N] \cup \{\langle S\rangle : S \in \mathcal{S}\}$.

1. *Neighbors of v in G that do not reside in any spot together with v; that is, $u \in \Gamma_G(v)$ is in $\Gamma_{G'}(v)$ if and only if $\{u, v\}$ is not a subset of any $S \in \mathcal{S}$.*
2. *Each spot that contains v in G; that is, $\langle S\rangle$ is in $\Gamma_{G'}(v)$ if and only if $S \in \mathcal{S}$ and $v \in S$.*

For any $S \in \mathcal{S}$, the set of neighbors of $\langle S\rangle$ in G' equals S. Recalling that by Equation (6) each $S \in \mathcal{S}$ has size at most d^{k-1}, we view G' as a graph of maximal degree d^{k-1}.

Observe that the set of spots that contain a vertex $v \in [N]$ is determined by the $(k + \ell(k))$-neighborhood of v in G, where the t-neighborhood of v contains all vertices that are at distance at most t from v. Thus, we can determine the set of neighbors of each vertex in G'. We note that the process of determining the spots that contain a vertex may fail if a cycle of length at least k is encountered. In such a case the algorithm can clearly reject. Lastly, note that we can select a vertex of G' uniformly by using the following procedure.

1. Select uniformly $v \in [N]$.
2. Select one of the following two instructions at random with equal probability.
   (a) (Generating a vertex of G):
   Output v with probability $1/d$.
   (b) (Generating a spot):
   Select uniformly a spot S that contain v (i.e., $S \in \mathcal{S}_v$), and output $\langle S \rangle$ with probability $p_v(S) = \frac{|\mathcal{S}_v|}{d|S|}$, where $\mathcal{S}_v \stackrel{\rm def}{=} \{S \in \mathcal{S} : v \in S\}$. (Recall that by Corollary 5.4, $|\mathcal{S}_v| \leq d/2$, so that $p_v(S) < 1$).

   In all the other cases, there is no output.

Thus, this process outputs each vertex of G with probability $N^{-1} \cdot 0.5 \cdot d^{-1} = 1/(2dN)$, and outputs each spot $\langle S \rangle \in \mathcal{S}$ with probability $\sum_{v \in S} N^{-1} \cdot 0.5 \cdot |\mathcal{S}_v|^{-1} \cdot p_v(S) = 1/(2dN)$.

Algorithm 5.8 (the C_k-minor-freeness tester): *Given input graph $G = ([N], E)$, the tester emulates the execution of Algorithm 3.3 on the graph G' as defined in Construction 5.7. In the emulation, vertices of G' are selected at random and their neighbors are explored on the fly, as detailed above.*

The analysis of Algorithm 5.8 reduces to an analysis of Construction 5.7.

Claim 5.9 (yes-instances): *If G is C_k-minor-free, then G' is cycle-free.*

Proof: Suppose, contrary to the claim, that v_1—v_2–$\cdots$–v_t—v_1 is a simple cycle in G'. We consider two cases.

Case 1: *All v_i's are vertices of G.* In this case, the edges $\{v_i, v_{i+1}\}$ in G' must be edges of G (since the only edges in G' that are not edges in G are incident to spot-vertices). On the other hand $t < k$ must hold, because G is C_k-minor-free. But this yields a contradiction, because, by Claim 5.6, the set $\{v_i : i \in [t]\}$ must be a subset of some spot of $\mathcal{S}$, which means that none of the edges $\{v_i, v_{i+1}\}$ may exist in G'.

Case 2: *Some v_i represents a spot of G.* Let $v_i = \langle S \rangle$, for some $S \in \mathcal{S}$. By the definition of the neighborhood relations in G' we have that $v_{i+1}, v_{i-1} \in S$. Now, consider a minimal sub-path of $v_{i+1}, \ldots, v_t, v_1, \ldots, v_{i-1}$ that starts in a vertex of S, denoted u, and ends in a vertex of S, denoted v. That is, we consider a sub-path that starts and ends in vertices of S, but has no intermediate vertices in S. This sub-path (in G') cannot consist of a single edge (because the edge $\{u, v\} \subset S$ cannot appear in G'), it cannot contain the vertex $\langle S \rangle$ (because $\langle S \rangle$ already appears as v_i), and it cannot be a path of length 2 that goes through another spot (because, by Claim 5.4, no other spot may contain both u and v). Since this path may not contain intermediate vertices in S, and since spot-vertices cannot be adjacent in G', it follows that this path must contain a vertex $w \in [N] \setminus S$. That is, we get a path in G' that goes from u to v via w, without passing through any vertex in S.

We now obtain a corresponding path in G; that is, a path in G that goes from u to v via w, without passing through any vertex in S. This is done by replacing any length-2 subpath

u'—$\langle S'\rangle$—v' (in G') by a sub-path u'–$\cdots$–v' (in G) that does not pass through S, where the latter path exists by the fact that $u', v' \in S'$ are connected by vertex-disjoint paths (internal to S') such that their intersection with S contains at most a single vertex (see Claim 5.4). It follows that G itself contains a path between u and v that passes through w and does not pass through S, where $u, v \in S$ but $w \notin S$. Thus, G itself contains a simple (non-edge) path between u and v that does not pass through S (i.e., an external path). By the third condition in Definition 5.1, the length of this external path is at least $\ell(k) > k$, but this contradicts the hypothesis that G is C_k-minor-free (because u and v are connected in G_S and $\ell(k) \geq k$, yielding a simple cycle of length at least k).

The claim follows. ■

Claim 5.10 (no-instances): *If G is ϵ-far from being C_k-minor-free, then G' is $\Omega(\epsilon)$-far from being cycle-free, where the Omega-notation hides a d^{-k} factor.*

Proof: Suppose that G' is δ-close to being cycle-free, where the distance refers to the degree bound of G', which is d^{k-1}. Recall that by Corollary 5.5 $|\mathcal{S}| \leq |E| \leq dN/2$. Let R' be a set of at most $\delta \cdot d^{k-1}(N + |\mathcal{S}|) < \delta \cdot d^k N$ edges such that removing R' from G' yields a cycle-free graph. Let $R \subseteq E$ be a set of edges that consists of (1) all edges of E that are in R', and (2) each edge $\{v, w\} \in E$ such that $\{v, \langle S\rangle\}$ is in R'. Hence, $|R| \leq d|R'| < \delta \cdot d^{k+1}N$. We next prove that removing R from G yields a graph that is C_k-minor-free, and it follows that G is $\delta \cdot d^k$-close to being C_k-minor-free.

Suppose, contrary to the claim, that for $t \geq k$ there exists a simple cycle v_1—v_2–$\cdots$–v_t—v_1 in the resulting graph (i.e., in the graph $([N], E \setminus R)$). We first show that there exists a corresponding (not necessarily simple) cycle in $E' \setminus R'$. Specifically, for each $\{v_i, v_{i+1}\} \in E \setminus R$, we consider two cases.

Case 1: *This edge is not a subset of any spot in G.* In this case, $\{v_i, v_{i+1}\} \in E' \setminus R'$, because this edge is in E' and cannot be in R' (or else it would have been in R). So we just use this edge in the cycle (in $E' \setminus R'$).

Case 2: *This edge is a subset of a spot S in G.* In this case, $\{v_i, \langle S\rangle\}, \{v_{i+1}, \langle S\rangle\} \in E' \setminus R'$, because both these edges are in E' and cannot be in R' (or else $\{v_i, v_{i+1}\}$ would have been in R). In this case, we replace the edge $\{v_i, v_{i+1}\} \in E \setminus R$ by the length-two-path v_i—$\langle S\rangle$—v_{i+1}.

Thus, we obtain a cycle in $([N] \cup \{\langle S\rangle : S \in \mathcal{S}\}, E' \setminus R')$ that contains the vertices $v_1, \ldots, v_t \in [N]$ as well as (possibly) some elements in $\{\langle S\rangle : S \in \mathcal{S}\}$. Since the latter elements may appear in multiple copies, the foregoing cycle is not necessarily simple. Note that a simple cycle in $([N] \cup \{\langle S\rangle : S \in \mathcal{S}\}, E' \setminus R')$ yields a contradiction to the hypothesis that this graph is cycle-free, and thus establishes our claim that the graph $([N], E \setminus R)$ is C_k-minor-free. We obtain a simple cycle, in two steps, as follows.

First, we replace every maximal sub-path of the form v_i—$\langle S\rangle$—v_{i+1}—$\langle S\rangle$–$\cdots$–$\langle S\rangle$—v_j, where $j \neq i$ (or else S contains a t-cycle for $t \geq k$), by a length-two path v_i—$\langle S\rangle$—v_j. If the resulting cycle contains distinct spot (representative) vertices, then we are done (since we obtain a simple cycle). Otherwise, we obtain a cycle of the form

$$u_1\text{–}\cdots\text{–}u_{t_1}\text{—}\langle S_1\rangle\text{—}u_{t_1+1}\text{–}\cdots\text{–}u_{t_1+t_2}\text{—}\langle S_2\rangle\text{—}u_{t_1+t_2+1}\text{–}\cdots\text{–}u_{t_1+t_2+t_3}\text{—}\langle S_3\rangle \cdots \langle S_m\rangle\text{—}u_1$$

where the u_i's are all distinct and adjacent S_i's are distinct (but non-adjacent S_i's may be identical). Next, we consider a sub-path of the foregoing cycle such that the endpoints of this sub-path are two copies of the same spot S and no other spot appears more than once on this sub-path. This sub-path cannot have length two (because adjacent S_i's are distinct), which means that it is actually a simple cycle, and we are done. ■

Conclusion. Combining Claims 5.9 and 5.10 with Theorem 1.5 and the fact that the number of vertices in G' is linear in N (for constant d and k), we conclude that *Algorithm 5.8 is a one-sided error tester for C_k-minor-freeness, and its complexity is $\widetilde{O}(\text{poly}(d^k/\epsilon) \cdot \sqrt{N})$.* This establishes Theorem 1.2.

6 Proof of the Lower Bound

Recall that Goldreich and Ron proved a $\Omega(\sqrt{N})$ query lower bound on the complexity of one-sided error testers for cycle-freeness [GR02, Prop. 4.3]. As stated in the introduction, Benjamini, Schramm, and Shapira [BSS08] suggested that this lower bound can be extended to testing H-minor freeness for any H that is not a forest. This is indeed the case, as proved next.

Theorem 6.1 (lower bound for one-sided error testing of H-minor freeness, for any H that contains cycles): *For any fixed H that contains a simple cycle, the query complexity of* one-sided error *testing of H-minor freeness is $\Omega(\sqrt{N})$.*

Indeed, as can been seen in the case that H is a single edge, the lower bound does not hold in case H contains no simple cycles. A general study of testing H-minor freeness for any cycle-free H is initiated in Section 7.

Proof: Following the proof of [GR02, Prop. 4.3], we show that for sufficiently large N, with high probability, the random N-vertex graphs considered in [GR02, Sec. 7] are far from being H-minor free. Once this is done, the theorem follows, since it was shown in [GR02, Sec. 7] that a probabilistic algorithm that makes $o(\sqrt{N})$ queries is unlikely to find a cycle in such a random graph (and this algorithm must accept whenever it fails to see a cycle, because otherwise it will reject some H-minor free graph with positive probability). Furthermore, it suffices to show that, for any fixed k and sufficiently large N, with high probability, such a random graph is far from being K_k-minor free, because containing a minor of K_k implies containing a minor of any k-vertex graph H.

The random graphs considered in [GR02, Sec. 7] are graphs uniformly chosen in the family $\mathcal{G}_N$ (which is denoted $\mathcal{G}_1^N$ in [GR02]). Each (N-vertex) graph in $\mathcal{G}_N$ consists of the union of a simple N-vertex (Hamiltonian) cycle and a perfect matching of these N vertices. (Indeed, each graph in $\mathcal{G}_N$ is 3-regular.) Furthermore, the cycle is fixed to be $(1, 2, \ldots, N, 1)$ and so a random graph in $\mathcal{G}_N$ corresponds to a random choice of a perfect matching. Our aim is to prove that, with high probability, such a random graph is $\epsilon/3$-far from being K_k-minor free, where $\epsilon = 1/ck^2$ for a sufficiently large constant c (to be determined below). We first show that any graph having a specific property (which is stated in the conditions of the following claim) is far from being K_k-minor free.

Claim 6.2 *Suppose that the vertices of the N-vertex graph G can be partitioned into $\widehat{N} = 2\epsilon N$ equal-sized sets, denoted $S_1, S_2, \ldots, S_{\widehat{N}}$, such that the following conditions hold:*

1. *The subgraph induced by each S_i is connected.*

2. *For every two disjoint collections of sets, C and C', such that $|C| = |C'| \geq \widehat{N}/6k$, there are at least $\epsilon N + 1$ edges between vertices in $U = \bigcup_{i \in C} S_i$ and vertices in $U' = \bigcup_{i \in C'} S_i$.*

Then, removing any set of ϵN edges from G, yields a graph that contains an K_k-minor (i.e., G is $\epsilon/3$-far from being K_k-minor free).

Proof: We prove the claim by contradiction. Suppose that there is a subset E' of at most ϵN edges whose removal from G results in a graph, denoted G', that is K_k-minor free. First, note that at most ϵN of the sets S_i can become disconnected. Thus, (1) at least $\widehat{N} - \epsilon N = \widehat{N}/2$ of the S_i's induced connected subgraphs in G'. Furthermore, (2) for every U and U' as defined in the claim, there exists at least one edge in G' between U and U'.

Starting with (1), assume, w.l.o.g., that for $i = 1, ..., \widetilde{N} \stackrel{\text{def}}{=} \widehat{N}/2$, the subgraph of G' induced by S_i is connected. We partition these sets into k equal-sized parts; that is, for $1 \leq i \leq k$, let $T_i = \bigcup_{r=((i-1)\widetilde{N}/k)+1}^{i\widetilde{N}/k} S_r$ and G'_i be the subgraph of G' induced by T_i.

We first show that each G'_i has a connected component that contains at least $\widetilde{N}/3k$ sets S_r (which are contained in T_i). Let $W_1, ..., W_t$ denote the connected components of G'_i, and suppose towards the contradiction that each of them contains less than $\widetilde{N}/3k$ sets S_r. Then, there exists $I \subset [t]$ such that both $W = \bigcup_{i \in I} W_i$ and $W' = \bigcup_{i \in [t] \setminus I} W_i$ contain at least $\widetilde{N}/3k = \widehat{N}/6k$ sets S_r. But, then a contradiction is reached, because by (2) there must be an edge in G' between some vertex of W and some vertex of W'.

Hence, each G'_i contain a connected component, denoted C_i, that has at least $\widetilde{N}/3k$ sets S_r. Applying (2) again, we infer that there must be an edge in G' going between any two of the C_i's. By contracting each C_i to a node, we obtain a K_k-minor in G', which contradicts the hypothesis that G' is K_k-minor free. ■

It remains to show that, with high probability, a graph G drawn from $\mathcal{G}_N$ satisfies the two conditions stated in Claim 6.2. Recalling that graph G in $\mathcal{G}_N$ consists of a Hamiltonian cycle augmented by a matching, we obtain the desired sets $S_1, \ldots, S_{\widehat{N}}$, by partitioning the Hamiltonian cycle into $\widehat{N} = 2\epsilon N$ contiguous segments, each of length $1/2\epsilon$. Clearly, these S_i's satisfy the first condition (i.e., the subgraph of G induced by each S_i is connected). We shall show that the second condition holds too, by considering all relevant sets U and U', showing that, with probability $1 - \exp(-\Omega(N/k^2))$ (over the choice of a random matching), there are $\Omega(N/k^2) > \epsilon N + 1$ edges going between U and U', and applying a union bound.

Specifically, we fix two arbitrary disjoint collections of sets, C and C', such that $|C| = |C'| = \widehat{N}/6k$, and consider the sets $U = \bigcup_{i \in C} S_i$ and $U' = \bigcup_{i \in C'} S_i$. Since $|U| = |U'| = (\widehat{N}/6k) \cdot (1/2\epsilon) = N/6k$, we expect the number of edges between U and U' to be $N/(36k^2)$. Intuitively, with very high probability (i.e., with probability $1 - \exp(-\Omega(N/k^2))$), the number of edges is a constant fraction of its expected size; in fact, this is the case as shown in Claim 6.3 (below).

Applying a union bound over all possible choices of C and C' (which underlie the choice of U and U'), we infer that the second condition stated in Claim 6.2 is satisfied with probability at least $1 - \binom{2\epsilon N}{2\epsilon N/(6k)}^2 \cdot \exp(-\Omega(N/k^2))$, which is lower bounded by $1 - 2^{4\epsilon N} \cdot \exp(-\Omega(N/k^2)) = 1 - \exp(-\Omega(N/k^2))$, since $\epsilon = 1/ck^2$ for a sufficiently large c (which is determined at this point to make assertion hold). Thus, modulo Claim 6.3 (below), the theorem follows.

Claim 6.3 *For some universal constant $c' > 0$, the following holds for all N and t. Consider selecting a matching between N vertices uniformly at random, and let T_1 and T_2 be two disjoint* *sets of N/t vertices each. Then, with probability at least $1 - \exp(-c' \cdot N/t^2)$, over the choice of the matching edges, there exist $c'N/t^2$ edges going between these sets.*

Proof: A random matching can be selected in $N/2$ steps, where at each step we pick an arbitrary yet-unmatched vertex v, and select, uniformly at random, another yet-unmatched vertex u to match v to. In particular, we can start by matching the vertices in T_1, and once they are all matched we continue in an arbitrary order with the remaining unmatched vertices.

Observe that the number of steps it takes to match all vertices in T_1 is at least $N' \stackrel{\text{def}}{=} N/(2t)$, and we shall lower bound the number of edges obtained between T_1 and T_2 (only) in the first N' steps. Let $X_1, \ldots, X_{N'}$ be $0/1$ random variables, where $X_i = 1$ if and only if the selected edge in the i^{th} step has as its second endpoint a vertex in T_2. By the definition of the matching process, $\Pr[X_1 = 1] = N'/(N-1)$. More generally, $\Pr[X_i = 1]$ equals the fraction of yet-unmatched vertices in T_2 at the start of the i^{th} step over $N - 2(i-1) - 1$. Since we consider only the first N' steps, which means that at most half of the vertices in T_2 can be matched, we have that $\Pr[X_i = 1] \geq \frac{N'}{N-2i+1} > \frac{1}{2t}$ for every $i \in \{1, 2, ..., N'\}$. Hence, the expected value of $\sum_{i=1}^{N'} X_i$, which is a lower bound on the expected number of edges between T_1 and T_2, is at least $N'/2t = N/(4t^2)$.

We would like to show that

$$\Pr\left[\sum_{i=1}^{N'} X_i < \frac{N}{8t^2}\right] < \exp(-\Omega(N/t^2)) \ . \tag{8}$$

Since the X_i's are not independent random variables (as the probability that $X_i = 1$ depends on $X_1, \ldots, X_{i-1}$), we cannot simply apply a multiplicative Chernoff bound in order to obtain Equation (8). However, we shall define a related sequence of independent random variables that will give us the bound in the equation.

For every $x_1, ..., x_{N'} \in \{0,1\}$ and every $i \in [N']$, let $f(x_1 \cdots x_{i-1})$ denote the probability that $X_i = 1$ conditioned on $X_j = x_j$ for every $j \in [i-1]$ (i.e., $f(x_1 \cdots x_{i-1}) = \Pr[X_i = 1 | X_1 \cdots X_{i-1} = x_1 \cdots x_{i-1}]$). Recall that $f(x_1 \cdots x_{i-1}) \geq 1/2t$. Define random variables $Y_1, ..., Y_{N'}$ such that Y_i depends on $X_1, ..., X_{i-1}$ and $\Pr[Y_i = 1] = \frac{1/2t}{f(X_1 \cdots X_{i-1})}$. Lastly, define $Z_1, ..., Z_{N'}$ such that $Z_i = 1$ if and only if $X_i = Y_i = 1$. Observe that for every $x_1, ..., x_{i-1} \in \{0,1\}$ and $z_1, ..., z_{i-1} \in \{0,1\}$ it holds that

$$\begin{aligned}
&\Pr[Z_i = 1 | X_1 \cdots X_{i-1} = x_1 \cdots x_{i-1} \wedge Z_1 \cdots Z_{i-1} = z_1 \cdots z_{i-1}] \\
&= \Pr[X_i = 1 | X_1 \cdots X_{i-1} = x_1 \cdots x_{i-1}] \cdot \frac{1/2t}{f(x_1 \cdots x_{i-1})} \\
&= \frac{1}{2t}
\end{aligned}$$

Hence, the Z_i's are independent random variables (since independence in presence of an auxiliary condition implies independence without it).[10] Also, $\sum_{i=1}^{N'} Z_i \leq \sum_{i=1}^{N'} X_i$ always holds (since $Z_i = 1$ occurs only when $X_i = 1$). Now the claim (or rather Equation (8)) follows by applying a multiplicative Chernoff bound to the sum of the Z_i's. ■

As stated above, the proof of Claim 6.3 completes the proof of the theorem. ■

[10]That is, if $\Pr[A|B\&X = x] = \Pr[A|X = x]$ for every x, then $\Pr[A|B] = \sum_x \Pr[X = x] \cdot \Pr[A|B\&X = x] = \sum_x \Pr[X = x] \cdot \Pr[A|X = x] = \Pr[A]$.

7 Testing Tree-Minor Freeness

As noted in Section 6, the $\Omega(\sqrt{N})$ lower bound of Theorem 6.1 does not hold in the case that the forbidden minor is a tree. This is easiest to see in the case that the forbidden minor is a single edge. We show that, for any cycle-free graph H, the set of H-minor free graphs can be tested with one-sided error with query complexity that is independent of the input graph's size (and that only depends on the proximity parameter and on H).

To begin, we provide a reduction of the case where H is a forest to the case where H is a tree. Actually, this reduction works for any H (regardless of cycle-freeness) allowing to focus on the connected components of H. Next, we turn to two special cases (which are easy to handle): the case that H is a k-path and the case that H is a k-star. Since these cases correspond to the two possible extremes, it is tempting to hope that all cases can be treated easily. We warn, however, that the extreme cases have simple characterizations, which are not available in non-extreme cases. Nevertheless, the case of stars provides some intuition towards the more complicated treatment of general trees. Further intuition can be obtained from the case of depth-two trees, treated in Section 7.5, where we also obtain better complexity than in the general case.

7.1 A reduction of unconnected H to connected H

Let H be a graph with connected components $H_1, \ldots, H_m$. Then, essentially (but not exactly), a graph G is H-minor free if and only if for some $i \in [m]$ the graph G is H_i-minor free; in other words, G has an H-minor if and only if for every $i \in [m]$ the graph G contains an H_i-minor. The alternative formulation reveals the small inaccuracy: it may be that the H_i-minors contained in G are not disjoint (and in such a case G does not necessarily have an H-minor). Still, for our purposes (of studying one-sided error testers of sublinear query complexity), this problem can be overcome (as done next).

Indeed, we focus on one-sided error testers of sublinear query complexity. Given such testers for H_i-minor freeness, we present the following one-sided error tester for H-minor freeness.

Algorithm 7.1 (the H-minor-freeness tester for cycle-free H): *On input $G = ([N], E)$ and proximity parameter ϵ, set $G_0 = G$ and proceed in m iterations, as follows. For $i = 1$ to m,*

1. *Invoke the H_i-minor tester on input G_{i-1}, using error parameter $1/3m$ and proximity parameter $\epsilon/2$.*
2. *If the answer is positive then accept.*
3. *Otherwise, omit from G_{i-1} all vertices that were visited by the tester, obtaining a residual graph G_i.*

If all iterations rejected, then reject.

If Algorithm 7.1 rejects, then (by the one-sided error feature of the tests) the m exploration contain corresponding (disjoint) H_i-minors, and so G contains an H-minor. Thus, Algorithm 7.1 satisfies the one-sided error condition. On the other hand, if G is ϵ-far from being H-minor free, then, for every $i \in [m]$, the graph G must be ϵ-far from being H_i-minor free (because otherwise G is ϵ-close to an H_i-minor free graph, which in turn is H-minor free). Furthermore, for every $i \in [m]$, the graph G_{i-1} is $\epsilon/2$-far from being H_i-minor free, because G_{i-1} is obtained from G by omitting $o(N)$

edges (since all testers have sublinear query complexity). Thus, in each iteration i, with probability at least $1-(1/3m)$, the corresponding tester rejects. It follows that Algorithm 7.1 rejects G with probability at least $2/3$ (as required). We thus get the following result.

Proposition 7.2 *Let H have connected components $H_1,\ldots,H_m$, and suppose that H_i-minor freeness can be tested by a one-sided error tester of query complexity $q_i(N,\epsilon)$. Suppose that $q_i(N,\epsilon)$ is monotonically non-decreasing with N. Then, H-minor freeness can be tested by a one-sided error tester of query complexity $q(N,\epsilon)=O(\log m)\cdot\sum_{i=1}^{n} q_i(N,\epsilon/2)$.*

(The $O(\log m)$ factor is due to error reduction that is employed on each of the testers.)

Detour. For sake of elegance, it would be nice to prove a similar reduction also for the case of two-sided error testers. Naturally, for testing H-minor freeness with two-sided error, we may just run all H_i-minor freeness tests (with error probability parameter set to $1/3m$) and accept if and only if at least one of these tests accepted (i.e., reject if and only if all these tests rejected). Clearly, if G is ϵ-far from being H-minor free, then, for every i, the graph G must be ϵ-far from being H_i-minor free (see above), and so in this case, with probability at least $2/3$, all tests will reject, and so will we. But what is missing is proving that if G is H-minor free, then the above tester accepts with high probability. (Indeed, it is not necessarily the case that if G is H-minor free then for some i it holds that G is H_i-minor free).

7.2 Testing that the graph contains no simple k-length path

Here we consider the special case where $H=P_k$, where P_k denotes the k-length path. Note that a graph G is P_k-minor free if and only if G contains no simple path of length k. Thus, we just search for such a path at random. Specifically, we select uniformly a start vertex and take a random k-step walk, rejecting if and only if the walk corresponds to a simple path. Clearly, we never reject a P_k-minor free graph.

Claim 7.3 *If G is ϵ-far from being a P_k-minor free graph, then we reject with probability at least $\epsilon/2d^k$.*

Thus, P_k-minor freeness can be tested by a one-sided error tester of query complexity $q \stackrel{\rm def}{=} O(d^k k/\epsilon)$ and time complexity $O(q)$. We note that it may be possible to reduce the query complexity to $\mathrm{poly}(dk/\epsilon)$, but an analogous improvement of the time complexity is unlikely (because finding k-long paths in graphs is NP-Hard, when k is part of the input). We mention that, subsequent to our initial posting of this work, Reznik [Rez11] presented a $\mathrm{poly}(dk/\epsilon)$-time algorithm for the special case that the input graph is cycle-free.

Proof: We call a vertex v bad if there is a simple path of length k starting at v. Let ρ denote the density of bad vertices in G. Then, on the one hand, we reject G with probability at least ρ/d^k. On the other hand, $\rho\geq\epsilon/2$, because omitting all bad vertices (or rather their incident edges) from G we obtain a graph that has no simple k-length paths. ■

7.3 Testing that the graph contains no k-star as a minor

Here we consider the special case where $H = T_k$, where T_k denotes the k-star (i.e., the $(k+1)$-vertex tree that has k leaves). The key observation here is that a graph $G = ([N], E)$ is T_k-minor free if and only if for every set S such that G_S is connected it holds that the set S has less than k neighbors (in $[N] \setminus S$). This implies that if for every connected set S of size at most k/ϵ, the set S has less than k neighbors in $[N] \setminus S$, then the graph is ϵ-close to being T_k minor free. The reasoning (which is detailed in the proof of Claim 7.5) is that if the premise of the statement (the small-cuts condition) holds, then by removing less than $k \cdot d \cdot (N/(k/\epsilon)) = \epsilon d N$ edges we can partition the graph into connected components such that each is T_k-minor free. Another useful observation is that searching for sets that violate the condition can be done efficiently by performing a BFS with a bounded depth (and width) and running a polynomial-time procedure on the subgraph induced by the BFS.

Algorithm 7.4 (the k-star-minor-freeness tester): *On input $G = ([N], E)$ and proximity parameter ϵ, proceed as follows.*

1. *Select uniformly a start vertex $s \in [N]$.*
2. *Perform a BFS starting at s and stopping as soon as either $2k/\epsilon$ layers were explored or a layer with at least k vertices was encountered.*

   Note that it may also be that the BFS terminates before either of these conditions hold; this can only happen if s resides in a connected component of size smaller than $2k^2/\epsilon$.
3. *Accept if and only if the explored graph is T_k-minor free.*

Clearly, Algorithm 7.4 never rejects a T_k-minor free graph. The query complexity of Algorithm 7.4 is $q(k, \epsilon) = O(k^2/\epsilon)$ (the maximum depth of the BFS times the maximum number of vertices in each level, assuming the degree d is a constant). By Corollary 1.2 in [KKR12], the time complexity is of the form $f(k) \cdot q(k, \epsilon)^2$ for some function f (which is not explicitly specified in [KKR12]). Thus, all that is left is to prove the following claim.

Claim 7.5 *If G is ϵ-far from being a T_k-minor free graph, then Algorithm 7.4 rejects with probability at least $\epsilon/2$.*

Thus, T_k-minor freeness can be tested by a one-sided error tester that invokes Algorithm 7.4 for $O(1/\epsilon)$ times. This tester has query complexity $O(q(k, \epsilon)/\epsilon) = O(k^2/\epsilon^2)$ and time complexity $f'(k) \cdot O(1/\epsilon^3)$ for some function f'.

Proof: We call a vertex v bad if there exists a set S containing v such that (i) G_S is connected and has radius at most $2k/\epsilon$ from v (i.e., all vertices in S are at distance at most $2k/\epsilon$ from v), and (ii) the set S has at least k neighbors in G (i.e., $|\{u \in [N] \setminus S : \exists w \in S \text{ s.t. } \{u, w\} \in E\}| \geq k$). Note that if a bad vertex is chosen in Step 1, then Algorithm 7.4 rejects in Step 3 (because either a $2k/\epsilon$-step BFS of G starting at v reaches a layer with at least k vertices, or it reaches all vertices in the witness set S). Let ρ denote the fraction of bad vertices in G. By the above, Algorithm 7.4 rejects with probability at least ρ. We next show that G must be $(\rho + (\epsilon/2))$-close to T_k-minor free, and so $\rho \geq \epsilon/2$ follows.

Let $G^{(0)}$ denote the graph obtained from G by omitting all the edges that are incident at bad vertices. Indeed, $G^{(0)}$ is ρ-close to G. The rest of our analysis proceeds in iterations. If the current graph $G^{(i-1)}$ is T_k-minor free, then we are done. Otherwise, we pick an arbitrary vertex $s^{(i)}$ that resides in some T_k-minor. Since $s^{(i)}$ is not bad, it must reside in a connected component of $G^{(i-1)}$ that has radius at least $2k/\epsilon$ from $s^{(i)}$ (because otherwise the existence of a T_k-minor containing $s^{(i)}$ contradicts the hypothesis that v is not bad). Consider an arbitrary set $S^{(i)} \ni s^{(i)}$ of $2k/\epsilon$ vertices such that $G^{(i-1)}_{S^{(i)}}$ is connected. Since $s^{(i)}$ is not bad, it follows that $S^{(i)}$ has less than k neighbors (in $G^{(i-1)}$). We now obtain $G^{(i)}$ by omitting from $G^{(i-1)}$ the (less than kd) edges of the cut $(S^{(i)}, [N] \setminus S^{(i)})$, and observe that $G^{(i)}_{S^{(i)}}$ is T_k-minor free (and that $S^{(i)}$ will not intersect with any future $S^{(j)}$). When the process ends, we have a T_k-minor free graph. In total, we omitted at most $tk \cdot d$ edges (from $G^{(0)}$), where $t \leq N/(2k/\epsilon)$ denotes the number of iteration. Noting that $tdk \leq (\epsilon/2)dN$, we conclude that $G^{(0)}$ is $\epsilon/2$-close to $G^{(t)}$ and thus G is $(\rho+(\epsilon/2))$-close to T_k-minor free. ■

7.4 The general case: Testing T-minor freeness for any tree T

Following is a presentation of the main result of this section: a one-sided tester for T minor-freeness, where T is an arbitrary rooted tree with k vertices. The algorithm is an extension of the algorithm for stars: We perform a BFS from a random starting vertex (but for more levels) and check if we find a T-minor.

The analysis of this algorithm, in the current (general) case, is far more involved; nonetheless, the basic intuition remains the same. Suppose our procedure is typically unable to find a T-minor in G. We shall show that we can split up the graph into many small pieces, each being T-minor free and having few edges leaving it. Removing the few edges going between these pieces, we get a T-minor free graph, which proves that G is close to being T-minor free.

The main challenge is to perform the foregoing decomposition. For that, we will define an auxiliary procedure, called `find`, that attempts to find T-minors. This procedure will not be used by our algorithm; it will be used solely in the analysis. But, first, let us detail the alleged tester. In all that follows we assume that $\epsilon \leq \epsilon_0$ for some sufficiently small constant ϵ_0 (or else we can run the algorithm with ϵ set to ϵ_0).

Algorithm 7.6 (the tree-minor-freeness tester): *Given as input a proximity parameter ϵ and given query access to a graph $G = ([N], E)$ with maximum degree at most d, set $D = k \cdot (4d/\epsilon)^{4k+2}$ and proceed as follows.*

1. *Select uniformly, independently at random, $8/\epsilon$ start vertices in $[N]$.*
2. *For each selected start vertex s, perform a BFS starting at s and stop as soon as D layers are explored (or the BFS reaches all the vertices of a connected component in G).*
3. *Accept if and only if all explored subgraphs are T-minor free.*

Clearly, Algorithm 7.6 never rejects a T-minor free graph. Its query complexity is exponential in D, and its time complexity is polynomial in its query complexity (for constant k, by Corollary 1.2 of [KKR12]). The correctness of the algorithm thus follows from the next theorem.

Theorem 7.7 *If G is ϵ-far from being a T-minor free graph, then Algorithm 7.6 rejects with probability at least $2/3$.*

As noted in the introduction, one of the byproducts of our analysis is a combinatorial theorem claiming that any graph with "local" expansion must contain all tree minors of some constant size.

Definition 7.8 *Let G be a graph of maximum degree d and s be a vertex of G. We say that the* R-neighborhood of s in G is ϵ-expanding, *if for every vertex set S such that* $\max_{u\in S}\{\text{dist}(s,u)\} \leq R$, *it holds that the number of edges in the cut $(S, [N] \setminus S)$ is at least $\epsilon|S|d$.*

Theorem 7.9 *For any k and d, if the $k(4d/\epsilon)^{4k+2}$-neighborhood of s in G is ϵ-expanding, then this neighborhood contains a T-minor of any tree T of at most k vertices.*

Both theorems are proved using a procedure called `find`, which tries to find small T-minors. When invoked at a certain vertex and failing to find a small T-minor, the procedure provides us with a sort of "explanation for its failure" in the form of a sparse cut, that is, a cut with relatively few edges crossing it. Thus, if the graph G is accepted by the tester with high probability, then we can use this procedure to get the desired decomposition. As may be expected, the procedure `find` is designed by a (tedious, but not obvious) induction on the size of T. Following is an overview of our approach.

Consider the tree T and remove an edge so as to obtain two trees T_1 and T_2. Let the roots of these trees be the endpoints of the edge removed. A T-minor can be broken up into a T_1-minor and T_2-minor with a path connecting the two respective roots. So, it seems that we should try to find "rooted minors", where we specify a vertex v that must be present in the connected component that is the root. Inductively, assume that we have a procedure `find` for T_1 and T_2. We can use `find` to get these minors and try to connect the roots by a path. The problem is that we have to get *disjoint* minors to get a T-minor. Suppose we find a T_1-minor in the original graph. Because we want to find a disjoint T_2-minor, we make the vertices in this minor a *forbidden* set F (and effectively remove them from G). This means that `find` is not allowed to use the vertices of F in the T_2-minor. But now, `find` may return a sparse cut, instead of a T_2-minor, in the modified graph. This cut, between a set S and the rest of the vertices, is only sparse in the modified graph (without F), but it is possible that it is not sparse in the graph G. That is, there may be many edges between S and the rest of the vertices in G, which include the vertices in F. To get around this, we somehow need to ensure that whenever a cut is found, the number of vertices in the smaller side of the cut is much larger than $|F|$. Then, a sparse cut in the modified graph remains sparse in the original. We will give an indication of how this is done when we describe the parameters of `find`.

7.4.1 Setting the stage

We introduce some definitions and notation, after which we can formally express the claim about the procedure `find`. Given that claim, we will prove Theorem 7.7 and Theorem 7.9. In the next subsection, we will prove the claim about `find`. For a graph $H = (V(H), E(H))$ and a subset of vertices $S \subseteq V(H)$, we use the standard notation H_S to denote the subgraph of H that is induced by S.

Definition 7.10 (Distances) *Let $H = (V(H), E(H))$ be a fixed graph. For any pair of vertices $v, u \in V(H)$, let $\text{dist}_H(v,u)$ be the shortest-path distance between u and v in H. Given a set of vertices $T \subset V(H)$ and a vertex $v \in V(H)$, let* $\Delta_H(v,T) \overset{\text{def}}{=} \max_{u\in T}\{\text{dist}_H(v,u)\}$. *More generally, for two sets of vertices $S, T \subseteq V(H)$, let* $\Delta_H(S,T) \overset{\text{def}}{=} \max_{u\in T}\min_{v\in S}\text{dist}_H(v,u)$.

Definition 7.11 (Sparse Cuts) *For a graph $H = (V(H), E(H))$ with degree bound d, a cut $(S, V(H) \setminus S)$ is ζ-sparse with respect to H, if the number of edges in $E(H)$ that cross the cut is at most $\zeta|S|d$. We denote the cut $(S, V(H) \setminus S)$ by $cut_H(S)$.*

The parameters of `find`: The procedure `find` takes as input a vertex v in a graph $G' = ([N], E')$, a set of vertices U containing v, a rooted tree T with k nodes, and a set of *forbidden* vertices F (not containing v). There is also a proximity parameter ζ, but since it is fixed for our discussion, we will not consider it as an input parameter. We will set $\zeta = \Theta(\epsilon)$ for the final analysis. For now, it will be convenient to have it as a separate parameter.

Let $f = \max\{|F|, k(4d/\zeta)^{4k+2}\}$, and $G'' = G'_{[N]\setminus F}$. The procedure works under the conditions that U is disjoint from F, $|U| \geq 4f/\zeta$, and $\Delta_{G''}(v, U) \leq (4/\zeta)\ln(f/\zeta)$. The procedure `find`$(v, U, T, F)$ outputs a pair (σ, S) such that $\sigma \in \{\texttt{minor}, \texttt{cut}\}$ and $S \subseteq [N] \setminus F$, where there is a path in G'' between v and every vertex in S. It will be convenient to express quantities in terms of $\hat{k} = 4k - 2$.

The requirement from `find`: The output (σ, S) of `find`(v, U, T, F) should satisfy the following conditions.

- $\Delta_{G''}(v, S) \leq (4d/\zeta)^{\hat{k}} \ln(f/\epsilon)$.
- **If** $\sigma =$ **`minor`,** then the graph G'_S contains a T-minor not involving F that is rooted at v (i.e., v resides in the connected component that is contracted to fit the root r of T).

  If $\sigma =$ **`cut`,** Then the cut $cut_{G'}(S)$ is ζ-sparse.

Intuitively, the set U acts as a kind of large buffer around v. This deals with the issue that we raised earlier. When we try to find a T_2-minor by making the vertices of the T_1-minor forbidden, we could get a sparse cut in this modified graph. The buffer U ensures that this cut contains sufficiently many vertices.

Claim 7.12 *There exists a procedure `find` that satisfies the foregoing requirements.*

We next show how Theorem 7.7 and Theorem 7.9 can be proved based on this claim. In what follows, when we say we perform a BFS in a graph $H = (V(H), E(H))$ from a subset of vertices M, we mean the following. Consider the graph $H'(M)$ whose vertex set is $(V(H) \setminus M) \cup \{v(M)\}$ (so that M is replaced by a single vertex $v(M)$), and whose edge set is $\{(u, w) \in E(H) : u, w \in V(H) \setminus M\} \cup \{(u, v(M)) : u \notin M \text{ and } \exists w \in M \text{ s.t. } (u, w) \in E(H)\}$. A BFS from M in H corresponds to a BFS in $H'(M)$ that starts from $v(M)$. We first state a fairly simple claim that gives one of the basic "dichotomies" that we will repeatedly use: Either a BFS starting from some set M finds a ζ-sparse cut, or it leads to a level set that is quite large. We will introduce a forbidden set F, and make all our arguments with respect to $G'_{[N]\setminus F}$.

Claim 7.13 *Let F and M be two disjoint subsets of vertices in G' such that $|M| \geq (2/\zeta)|F|$. Suppose we perform a BFS up to depth t in $G'_{[N]\setminus F}$, starting from M, and let ℓ be the size of the last level reached. Then either there exists a subset of vertices R that are reached by the BFS and such that $cut_{G'}(R)$ is ζ-sparse, or $\ell \geq |M| \cdot e^{(\zeta/3)t}$.*

Proof: Consider some intermediate level in the BFS, and let R be the set of vertices reached up to that level (including it). Suppose that the next level has at most $\zeta|R|/2$ vertices. All edges in $\mathtt{cut}_{G'}(R)$ are either incident to vertices in the next level (which contains at most $\zeta|R|/2$ vertices) or to F. Since $|R| \geq |M| \geq 2|F|/\zeta$, the size of the cut is at most $\zeta|R|d$, and hence it is ζ-sparse.

Otherwise, the size of the levels keeps expanding by a factor of at least $(1+\zeta/2)$. Since the depth of the BFS is t, the size of the last level is at least $|M| \cdot (1+\zeta/2)^t \geq |M| \cdot e^{(\zeta/3)t}$. ■

Proof of Theorem 7.7. Recall that $D = k \cdot (8d/\epsilon)^{4k+2}$, and that Algorithm 7.6 performs a BFS from $4/\epsilon$ start vertices, up to depth D for each, and rejects if any of the subgraphs observed contains a T minor. We call a vertex v $\mathtt{bad}$ if its D-neighborhood (i.e., the subgraph induced by all vertices at distance at most D from v) contains a T-minor, and denote the fraction of bad vertices (in G) by ρ. We shall show that G is $(\rho+\epsilon/2)$-close to being T-minor free. The lemma follows since this implies that if G is ϵ-far from being T-minor free, then $\rho > \epsilon/2$. In such a case, the probability that no bad vertex is selected as a start vertex by the algorithm is at most $(1-\epsilon/2)^{4/\epsilon} < e^{-2} < 1/3$.

In order to prove that G is $(\rho+\epsilon/2)$-close to being T-minor free, we will remove at most $(\rho+\epsilon/2)dN$ edges from G to make it T-minor free. We start by removing all edges incident to bad vertices, so that the number of edges removed at this stage is at most ρdN. Let the resulting graph be $G^{(0)}$. The rest of our analysis proceed in iterations in which we invoke the procedure $\mathtt{find}$ with proximity parameter $\zeta = \epsilon/2$. Note that $D = k(4d/\zeta)^{4k+2}$. At the start of each iteration we have a current graph $G^{(i-1)}$ where some connected components are marked "minor free". These components are certified to have no T-minor. If all the components are marked, then we are done. Otherwise, consider some unmarked component C. Suppose there is $v \in C$, such that $\Delta_{G^{(i-1)}}(v, C) \leq D$. If C contains a T-minor, then v must be bad. This contradicts that fact that C is a connected component containing v. Therefore C has no T-minor, and can be marked. We proceed in this fashion till we get a component C that cannot be marked.

For such an unmarked component C we take an arbitrary vertex $s^{(i)} \in C$ and observe that $\Delta_{G^{(i-1)}}(s^{(i)}, C) > D$. Let F be initialized to $\emptyset$. Set $f = \max(|F|, k(4d/\zeta)^{4k+2})$. We perform a BFS from $s^{(i)}$ up to depth $D_0 = (3/\zeta)\ln(4f/\zeta)$ steps, and invoke Claim 7.13 with $M = \{s^{(i)}\}$, $F = \emptyset$, and $t = D_0$. We have $f = k(4d/\zeta)^{4k+2}$ (and $k \geq 1$), so

$$\begin{aligned} D_0 = (3/\zeta)\ln(4f/\zeta) &\leq (4/\zeta)\ln(4k(4d/\zeta)^{4k+2}/\zeta) \\ &= (4/\zeta)\ln(4k/\zeta) + ((16k+8)/\zeta)\ln(4d/\zeta) \leq k(4d/\zeta)^{4k+2} = D\ . \end{aligned} \tag{9}$$

Suppose we get a set $S^{(i)}$ such that $cut_{G^{(i-1)}}(S^{(i)})$ is ζ-sparse. By the bound in Equation (9), $\Delta_{G^{(i-1)}}(s^{(i)}, S^{(i)}) = D_0 \leq D$. Hence, the subgraph $G_{S^{(i)}}$ cannot contain a T-minor. We remove all edges in the cut $cut_{G^{(i-1)}}(S^{(i)})$ and mark the connected components in $G_{S^{(i)}}$ as minor free. This gives us the graph $G^{(i)}$, and we continue with the next iteration.

Otherwise (by Claim 7.13), the BFS gives a set U, such that $|U| \geq e^{(\zeta/3)D_0} = 4f/\zeta$, and $\Delta_{G^{(i-1)}}(s^{(i)}, U) \leq D_0 \leq (4/\zeta)\ln(f/\zeta)$. We therefore call $\mathtt{find}(s^{(i)}, U, T, F)$ (on the graph $G' = G^{(i-1)}$). If it outputs $(\mathtt{minor}, S^{(i)})$, then $s^{(i)}$ must be bad. This is a contradiction, and hence the output must be $(\mathtt{cut}, S^{(i)})$. We have $\Delta_{G^{(i-1)}}(s^{(i)}, S^{(i)}) \leq D$, where $cut_{G^{(i-1)}}(S^{(i)})$ is ζ-sparse. We proceed as before by removing all edges in $cut_{G^{(i-1)}}(S^{(i)})$ to get $G^{(i)}$.

When the process ends, we have a T-minor free graph. Since all the $S^{(i)}$'s considered are disjoint, in total, we omitted at most $\sum_i \zeta d|S^{(i)}| \leq \epsilon dN/2$ edges (from $G^{(0)}$), and thus G is $(\rho+\epsilon/2)$-close to T-minor freeness. ■

Proof of Theorem 7.9. The theorem follows from Claim 7.12, where the key observation is that `find` works for any k-vertex tree T and that `find` may not return a ζ-sparse cut (because no such cut exists by the hypothesis). Specifically, set $\zeta = \epsilon$, $F = \emptyset$, and $f = k(4d/\epsilon)^{4k+2}$. Let U be a set such that $|U| \geq 4f/\epsilon$ and $\Delta_G(v, U) \leq (4/\epsilon)\ln(f/\epsilon)$ (which exists since the said neighborhood contains no ζ-sparse cuts). Now, for any k-vertex tree T, we run $\texttt{find}(v, U, T, F)$ and get the output (σ, S), where $\sigma \neq \texttt{cut}$. Thus, we get the desired T-minor. ■

7.4.2 The procedure `find`

We first introduce the notion of a boundary.

Definition 7.14 (Boundaries) *Given sets of vertices S and F, let $\partial_F(S)$ denote the* boundary *of S in $G'_{[N]\setminus F}$. That is, $\partial_F(S) \overset{\text{def}}{=} \{u \in S : \exists w \in [N] \setminus (S \cup F) \text{ s.t. } (u, w) \in E(G')\}$. We use $\overline{\partial}_F(S)$ to denote the set $S \setminus \partial_F(S)$.*

We build a little on the basic dichotomy of Claim 7.13. Given a starting set M and forbidden set F, we try to find a ζ-sparse cut that is not too far from the boundary $\partial_F(M)$. If we fail, then we can find a vertex $v \in \partial_F(M)$ and a set $U_v \ni v$ disjoint from F, such that U_v is large, but vertices in U_v are quite close to v.

Claim 7.15 *Let F and M be two disjoint subsets of vertices such that $|M| \geq (2/\zeta)|F|$, and let $\widetilde{F} = \overline{\partial}_F(M) \cup F$. There is a procedure that, given a graph G' and an integer parameter t, outputs one of the following:*

- *A set R such that the $\text{cut}_{G'}(R)$ is ζ-sparse and $\Delta_{G'_{[N]\setminus F}}(\partial_F(M), R) \leq t$.*
- *A vertex $v \in \partial_F(M)$ and a set U_v disjoint from $\widetilde{F}$ such that $v \in U_v$, $|U_v| \geq e^{(\zeta/3)t}$, and $\Delta_{G'_{[N]\setminus \widetilde{F}}}(v, U_v) \leq t$*

Proof: We start by performing a BFS from M in $G'' = G'_{[N]\setminus F}$ up to depth t. By the definition of the BFS, all the vertices reached in levels $1, \ldots, t$ are disjoint from M and F. Applying Claim 7.13, in the process of this BFS either we find a ζ-sparse cut, thus satisfying the first condition, or the size of the last level is at least $|M| \cdot e^{(\zeta/3)t}$. In the latter case, for each vertex $v \in \partial_F(M)$, perform a BFS in $G'_{[N]\setminus \widetilde{F}}$ up to depth t, and let U_v be the set of vertices reached. Since the last level of the original BFS is contained in $\bigcup_v U_v$, we have that $\sum_{v \in \partial_F(M)} |U_v| \geq |M| \cdot e^{(\zeta/3)t}$. Therefore, there exists a vertex $v \in \partial_F(M)$ such that $|U_v| \geq |M| \cdot e^{(\zeta/3)t}/|\partial_F(M)| \geq e^{(\zeta/3)t}$. ■

With these tools in hand, we are ready to describe the procedure `find`.

Proof of Claim 7.12. We prove the claim by induction over the size of the tree T. For the base case, let T be a singleton vertex. Then, the procedure `find` just outputs the pair $(\texttt{minor}, U)$. Now for the induction step.

Take an edge e of T that is incident to the root r. Removing this edge gives us two trees T_1 and T_2 with roots r_1 and r_2 (these are the respective endpoints of e). We let T_1 be the tree still rooted at r (so that $r_1 = r$). Using subscripts to denote the respective size parameters of these trees, we have $\hat{k} = \hat{k}_1 + \hat{k}_2 + 2$ (recall that $\hat{k} = 4k - 2$). We also have that $\hat{k}_1, \hat{k}_2 \geq 2$.

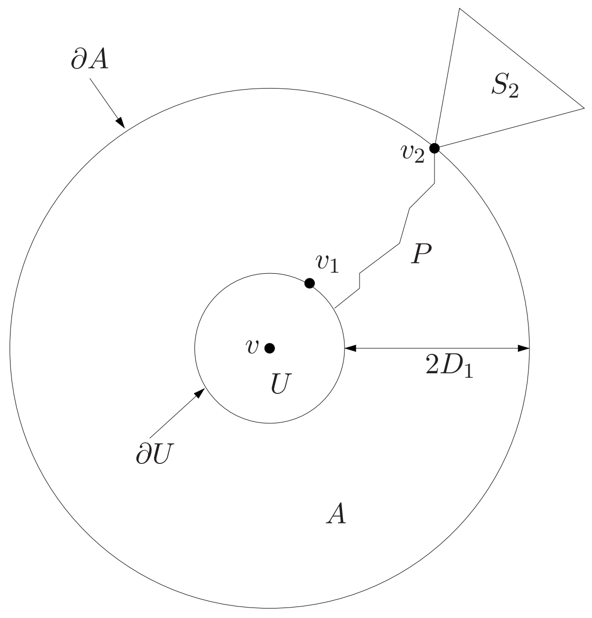

Figure 5: The various sets in `find`

We will describe the procedure $\texttt{find}(v, U, T, F)$ using the respective procedures for T_1 and T_2. We set $D_1 = (4d/\zeta)^{\hat{k}_1} \ln(6f/\zeta^2)$ (recall that $f = \max\{|F|, k(4d/\zeta)^{4k+2}\}$). We will be dealing mainly with the graph $G'' = G'_{[N]\setminus F}$ and hence all our boundaries are in this graph. Recall that the procedure is required to work under the conditions that U is disjoint from F, $|U| \geq 4f/\zeta$, and $\Delta_{G''}(v, U) \leq (4/\zeta)\ln(f/\zeta)$. We may actually assume that $|U| = 4f/\zeta$. Suppose this is not the case. Take the vertex in U farthest from v and remove it from U. We keep repeating this until $|U| = 4f/\zeta$. Note that the upper bound on $\Delta_{G''}(v, U)$ remains.

We now describe the steps of the procedure `find`. We will encapsulate each step through a different claim. These claims will be stated here, and their proofs shall be given after we describe `find` (and provide guarantees on its behavior on the basis of these claims). These proofs will depends heavily on the notation used here as well as the induction hypotheses. Nonetheless, we defer their proofs so that the reader can more easily follow the flow of our argument. Refer to Figure 5 to understand the various claims. For a set R, it will be convenient to refer to the following as Condition (*): $R \ni v$, $\texttt{cut}_{G'}(R)$ is ζ-sparse, and $\Delta_{G''}(v, R) \leq (4d/\zeta)^{\hat{k}} \ln(f/\zeta)$. If R satisfies this, we will say "R satisfies Condition (*)".

Claim 7.16 *We can either find a set R satisfying Condition (*) or a set A with the following properties. The set A is disjoint from F and exactly contains all vertices (outside F) at distance at most $2D_1$ from U. Also, $|U| \cdot e^{(\zeta/2)D_1} \leq |\partial_F(A)|$ and $|A| \leq |U| \cdot e^{2D_1 \ln d}$.*

Claim 7.17 *Suppose we have a set A satisfying the conditions given in Claim 7.16. Then, we can either find a set R satisfying Condition (*), or a vertex $v_2 \in \partial_F(A)$ and set $S_2 \ni v_2$ with the following properties. The set S_2 is disjoint from $F \cup \overline{\partial}_F(A)$ and contains a T_2-minor such that v_2 belongs to the set whose contraction is the root. Furthermore, $\Delta_{G''}(v, S_2) \leq (4d/\zeta)^{\hat{k}} \ln(f/\zeta)$.*

Claim 7.18 *Let A, v_2 and S_2 have the properties stated in the previous claims. We can either find a set R satisfying Condition (*), or a path $P \subseteq A$, a vertex $v_1 \in \partial_F(U)$, and set $S_1 \ni v_1$ satisfying the following. The sets P and S_1 are disjoint from F and disjoint from each other. The path P connects v_2 to U in G''. The set S_1 contains a T_1-minor such that v_1 belongs to the set whose contraction is the root, and $\Delta_{G''}(v, S_1) < 2D_1$.*

With the claims, the proof becomes fairly direct. The procedure `find` invokes Claim 7.16. If we find a ζ-sparse cut, we are done. Otherwise, we have a set A with the properties stated in Claim 7.16. Now, `find` invokes Claim 7.17. Again, if we do not find a ζ-sparse cut, we have a vertex v_2 and set S_2. Applying Claim 7.18, we either find a ζ-sparse cut, or get a path P, vertex v_1 and set S_1.

We show how to construct a T-minor using U, S_1, S_2, and P. Refer to Figure 5 to see how these sets are laid out. Note that all these sets are disjoint from F. Furthermore, since $\Delta_{G''}(v, S_1) < 2D_1$ and all vertices in S_2 have distance at least $2D_1$ from U, the set S_1 is disjoint from S_2. Also, S_1 is disjoint from P (Claim 7.18). Our aim is to connect v_1 to v_2 (in G'') by a path that is disjoint from $S_1 \cup S_2$. If this path contains v, we will get a T-minor rooted at v that involves no vertex of F. Take the path P in G'' that connects $\partial_F(U)$ to v_2. This path is disjoint from $S_1 \cup S_2$. The vertex v_1 is in $\partial_F(U)$ and connected to all of U in G''. We take a path from v to P and a path from v to v_1. This connects v_1 to v_2 (via v) in G'' and completes the construction of the T-minor. ■

In the proofs of Claims 7.16, 7.17 and 7.18, the following bounds will be repeatedly used. By assumption, $\Delta_{G''}(v, U) \leq (4/\zeta)\ln(f/\zeta)$, $|U| = 4f/\zeta$ and $D_1 = (4d/\zeta)^{\hat{k}_1}\ln(6f/\zeta^2)$. Since $\hat{k} = \hat{k}_1 + \hat{k}_2 + 2$ and $\hat{k}_1, \hat{k}_2 \geq 2$, $\max(\hat{k}_1, \hat{k}_2) \leq \hat{k} - 4$. We first state a technical claim.

Claim 7.19

$$\Delta_{G''}(v, U) + (2 + 6(\ln d)/\zeta)D_1 + (4d/\zeta)^{\hat{k}_2}\ln(|U|/\zeta) + (3/\zeta)\ln(4|U|/\zeta) \;\; \leq \;\; (1/2)(4d/\zeta)^{\hat{k}}\ln(f/\zeta)\ .$$

Proof: As argued earlier, $|U| = 4f/\zeta$, so $(3/\zeta)\ln(4|U|/\zeta) \leq (4/\zeta)\ln(16f/\zeta^2)$. Assuming ζ is at most a sufficiently small constant, we can bound:

$$\begin{aligned}\Delta_{G''}(v, U) + (3/\zeta)\ln(4|U|/\zeta) &\leq (4/\zeta)(\ln(f/\zeta) + \ln(16f/\zeta^2)) \\ &= (4/\zeta)\ln(16f^2/\zeta^3) \\ &\leq (4/\zeta)^4\ln(f/\zeta)\ .\end{aligned}$$

We trivially bound $2 + 6(\ln d)/\zeta$ by $8d/\zeta$. We now bound the summation in the claim as follows. We use the fact that $\hat{k}_1 + 2 \geq 4$, and that $\hat{k} - 2 \geq \max(\hat{k}_1 + 2, \hat{k}_2 + 2)$.

$$\begin{aligned}&(4/\zeta)^4\ln(f/\zeta) + 8dD_1/\zeta + (4d/\zeta)^{\hat{k}_2}\ln(|U|/\zeta) \\ &\quad= (4/\zeta)^4\ln(f/\zeta) + 2(4d/\zeta)^{\hat{k}_1+1}\ln(6f/\zeta^2) + (4d/\zeta)^{\hat{k}_2}\ln(4f/\zeta^2) \\ &\quad\leq (4d/\zeta)^{\hat{k}_1+2}\ln(f/\zeta) + (4d/\zeta)^{\hat{k}_1+2}\ln(f/\zeta) + (4d/\zeta)^{\hat{k}_2+2}\ln(f/\zeta) \\ &\quad\leq (1/2)(4d/\zeta)^{\hat{k}}\ln(f/\zeta)\ ,\end{aligned}$$

and the proof is completed. ■

Proof of Claim 7.16. Initiate a BFS in the residual graph $G'' = G'_{[N]\setminus F}$ starting from U for $2D_1$ steps. Let A denote the set of all vertices reached (including U). We now invoke Claim 7.13

with F, $M := U$, and $t := 2D_1$. Suppose we find a ζ-sparse cut $cut_{G'}(R)$. We have $\Delta_{G''}(v, R) \leq \Delta_{G''}(v, U) + 2D_1$. By Claim 7.19, this is at most $(4d/\zeta)^{\hat{k}} \ln(f/\zeta)$. Hence, R satisfies Condition (*). Otherwise, the BFS reaches $2D_1$ levels, and (by Claim 7.13), $|U| \cdot e^{(\zeta/2)D_1} \leq |\partial_F(A)|$. Since the degree bound is d, the size (after $2D_1$ steps of a BFS) can only blow up by d^{2D_1}. Hence, $|A| \leq |U| \cdot e^{2D_1 \ln d}$. ■

Proof of Claim 7.17. Note that $|\partial_F(A)| \geq |U| = 4f/\zeta \geq |F|$. Set $F_2 = \overline{\partial}_F(A) \cup F$, so that

$$|F_2| = |\overline{\partial}_F(A) \cup F| = |\overline{\partial}_F(A)| + |F| \leq |\overline{\partial}_F(A)| + |\partial_F(A)| = |A| \leq |U| \cdot e^{2D_1 \ln d} .$$

We invoke Claim 7.15 with F, $M := A$, and $t := (3/\zeta)\ln(4|F_2|/\zeta)$. Suppose we get a ζ-sparse cut $cut_{G'}(R)$.

$$\begin{aligned} \Delta_{G''}(v, R) &\leq \Delta_{G''}(v, U) + \Delta_{G''}(U, \partial_F(A)) + t \\ &\leq \Delta_{G''}(v, U) + \Delta_{G''}(U, \partial_F(A)) + (3/\zeta)\ln(4|F_2|/\zeta) \\ &\leq (4/\zeta)\ln(f/\zeta) + 2D_1 + (3/\zeta)\ln(4|U|/\zeta) + 6D_1(\ln d)/\zeta \end{aligned}$$

An application of Claim 7.19 proves that this is at most $(4d/\zeta)^{\hat{k}} \ln(f/\zeta)$. So R satisfies Condition (*). By Claim 7.15, if we do not get a set R, we get a vertex $v_2 \in \partial_F(A)$ and a set U_2 disjoint from F_2 such that $v_2 \in U_2$, $|U_2| \geq e^{(\zeta/3)t}$, and $\Delta_{G'_{[N]\setminus F_2}}(v_2, U_2) \leq (3/\zeta)\ln(4|F_2|/\zeta)$. By choice of t, $|U_2| \geq 4|F_2|/\zeta$. Let $f_2 = \max\{|F_2|, k_2(4d/\zeta)^{4k_2+2}\}$. To call the procedure $\texttt{find}(v_2, U_2, F_2, T_2)$, we need to argue that $|U_2| \geq 4f_2/\zeta$ and $\Delta_{G'_{[N]\setminus F_2}}(v, U_2) \leq (4/\zeta)\ln(f_2/\zeta)$. We have chosen $f = \max\{|F|, k(4d/\zeta)^{4k+2}\}$, and $|F_2| \geq f$. Hence, $f_2 = |F_2|$. So $|U_2| \geq 4|F_2|/\zeta = 4f_2/\zeta$ and $\Delta_{G'_{[N]\setminus F_2}}(v_2, U_2) \leq (3/\zeta)\ln(4|F_2|/\zeta) \leq (4/\zeta)\ln(4f_2/\zeta)$.

Let (σ, S^*) be the output of this call to find. We have,

$$\Delta_{G''}(v, S^*) \leq \Delta_{G''}(v, U) + \Delta_{G''}(U, \partial_F A) + \Delta_{G''}(v_2, S^*) .$$

By the induction hypothesis $\Delta_{G''}(v_2, S^*) \leq (4d/\zeta)^{\hat{k}_2} \ln(f_2/\zeta)$. Since $f_2 \leq |U|e^{2D_1 \ln d}$, this is at most $(4d/\zeta)^{\hat{k}_2} \ln(|U|/\zeta) + (2\ln d)(4d/\zeta)^{\hat{k}_2} D_1$. We now get

$$\Delta_{G''}(v, S^*) \leq \Delta_{G''}(v, U) + \Delta_{G''}(U, \partial_F A) + (4d/\zeta)^{\hat{k}_2} \ln(|U|/\zeta) + (2\ln d)(4d/\zeta)^{\hat{k}_2} D_1$$

Barring the last term, we have that $\Delta_{G''}(v, U) + 2D_1 + (4d/\zeta)^{\hat{k}_2} \ln(|U|/\zeta)$ is at most $(1/2)(4d/\zeta)^{\hat{k}} \ln(f/\zeta)$ by Claim 7.19. Turning to the last term,

$$\begin{aligned} (2\ln d)(4d/\zeta)^{\hat{k}_2} D_1 &= (2\ln d)(4d/\zeta)^{\hat{k}_2} \cdot (4d/\zeta)^{\hat{k}_1} \ln(6f/\zeta^2) \\ &= (2\ln d)(4d/\zeta)^{\hat{k}_1+\hat{k}_2} \ln(6f/\zeta^2) \\ &= (2\ln d)(4d/\zeta)^{\hat{k}-2} \ln(6f/\zeta^2) \\ &< (1/2)(4d/\zeta)^{\hat{k}} \ln(f/\zeta) . \end{aligned}$$

Putting it all together, $\Delta_{G''}(v, S^*) \leq (4d/\zeta)^{\hat{k}} \ln(f/\zeta)$. If the output is a cut, then S^* is the desired R (satisfying Condition (*)). Otherwise (by the induction hypothesis), the set S^* (disjoint from F_2) contains a T_2-minor such that v_2 belongs to the subset whose contraction corresponds to the root r_2 of T_2. We have $F_2 = \overline{\partial}_F(A) \cup F$, where the set A contains all vertices whose distance (in

G'') from U is at most $2D_1$. Since S^* is disjoint from F_2, the distance in G'' of any vertex in S^* from U is at least $2D_1$. So S^* is the desired set S_2. ■

Proof of Claim 7.18. Consider the shortest path P from U to v_2 (in G''). Since $v_2 \in \partial_F(A)$, the path $P \setminus \{v_2\}$ is entirely contained in $\overline{\partial}_F(A)$. Hence, S_2 is disjoint from $P \setminus \{v_2\}$. By construction, $|P| \leq 2D_1$.

We have $f = \max\{|F|, k(4d/\zeta)^{4k+2}\}$. By some elementary calculations (see Fact 7.20 (below)), we have $f \geq (4d/\zeta)^{\hat{k}} \ln(6f/\zeta^2) \geq 2D_1 \geq |P|$. Hence, $|F \cup P| \leq 2f$. Let $F' := F \cup P$, $F_1 := \overline{\partial}_F(U) \cup F'$ and $f_1 = \max\{|F_1|, k_1(4d/\zeta)^{4k_1+2}\}$. We invoke Claim 7.15 with F' as the forbidden set, $M := U$, and $t := (3/\zeta)\ln(4f_1/\zeta)$. Since $|F_1| \leq |U| + 2f \leq 4f/\zeta + 2f < 6f/\zeta$, we have that $t = (3/\zeta)\ln(4f_1/\zeta) < (4d/\zeta)^{\hat{k}_1} \ln(6f/\zeta^2) = D_1$. If we get a ζ-sparse cut $cut_{G'}(R)$, we end with the desired R.

Otherwise, we get a vertex $v_1 \in \partial_F(U)$ and a set $U_1 \ni v_1$ disjoint from F_1 such that $|U_1| \geq e^{(3/\zeta)t} = 4|f_1|/\zeta$ and $\Delta_{G'_{[N]\setminus F_1}}(v_1, U_1) \leq t \leq 4\ln(f_1/\zeta)/\zeta$. We thus have the necessary conditions to call $\texttt{find}(v_1, U_1, F_1, T_1)$. By the induction hypothesis, for the set S^* returned, $\Delta_{G'_{[N]\setminus F_1}}(v_1, S^*) \leq (4d/\zeta)^{\hat{k}_1} \ln(f_1/\zeta) \leq (4d/\zeta)^{\hat{k}_1} \ln(6f/\zeta^2) \leq D_1$ (using the bound $f_1 < 6f/\zeta$). Hence $\Delta_{G''}(v, S^*) \leq \Delta_{G''}(v, U) + D_1 < 2D_1 \leq (4d/\zeta)^{\hat{k}} \ln(f/\zeta)$. Regardless of whether S^* is output as a cut or a minor, we complete the proof. ■

Fact 7.20 *If $x \geq k(4d/\zeta)^{4k+2}$, then $x \geq (4d/\zeta)^{\hat{k}} \ln(6x/\zeta^2)$, where $\hat{k} = 4k - 2$.*

Proof: Consider the function $x/\ln(\alpha x)$. The derivative is given by $\frac{d}{dx}\left(\frac{x}{\ln(\alpha x)}\right) = \frac{1}{\ln(\alpha x)} - \frac{1}{\ln^2(\alpha x)}$. Hence, this function is increasing when $x \geq e/\alpha$. We set $\alpha = 6/\zeta^2$, and note that for small enough ζ, $k(4d/\zeta)^{4k+2} > e\zeta^2/6$. For $x \geq k(4d/\zeta)^{4k+2}$,

$$\frac{x}{\ln(6x/\zeta^2)} \geq \frac{k(4d/\zeta)^{4k+2}}{\ln(6k/\zeta^2) + (4k+2)\ln(4d/\zeta)} \geq (4d/\zeta)^{4k-2},$$

and the proof is completed. ■

7.5 Testing T-minor freeness for any depth-two tree T

Let T be an arbitrary depth-two tree with k vertices; that is, T consists of a root, denoted r, and m stars, denoted $T_1, \ldots, T_m$, that are rooted at neighbors of r, where here we consider also the singleton vertex as a star (with 0 leaves). Denote the m corresponding roots by $r_1, \ldots, r_m$, and denote the number of leaves in these stars by $k_1, \ldots, k_m$ (i.e., $k = 1 + m + \sum_{i \in [m]} k_i$). The following algorithm is tailored for this tree T.

Algorithm 7.21 (tailored for the foregoing T): *On input $G = ([N], E)$ and proximity parameter ϵ, set $D = (5d^2k/\epsilon)^2$ and proceed as follows.*

1. *Select uniformly a start vertex $s \in [N]$.*

2. *Perform a BFS starting at s and stopping as soon as D layers are explored (or the BFS reaches all the vertices of a connected component in G).*

3. *Accept if and only if the explored graph is T-minor free.*

Clearly, Algorithm 7.21 never rejects a T-minor free graph. Its query complexity is exponential in D, and its time complexity is polynomial in its query complexity (by Corollary 1.2 of [KKR12]).

Lemma 7.22 *If G is ϵ-far from being a T-minor free graph, then Algorithm 7.21 rejects with probability at least $\epsilon/4$.*

Proof: We call a vertex v bad if its D-neighborhood (i.e., the vertices that are at distance at most D from v) contains a T-minor, and denote the fraction of bad vertices (in G) by ρ. As in the proof of Claim 7.5, it suffices to show that G is $(\rho + (\epsilon/2))$-close to being T-minor free, and we again start by omitting all edges incident at bad vertices and considering the resulting graph, denoted $G^{(0)}$. Indeed, $G^{(0)}$ is ρ-close to G.

The rest of our analysis proceed in iterations. If the current graph $G^{(i-1)}$ is T-minor free, then we are done. Otherwise, we pick an arbitrary vertex $s^{(i)}$ that resides in (the root of) some T-minor. Since $s^{(i)}$ is not bad, it must reside in a connected component of $G^{(i-1)}$ that has radius at least D from $s^{(i)}$. We shall show how to identify a set $S^{(i)}$ such that $G^{(i-1)}_{S^{(i)}}$ has radius at most D and the cut $(S^{(i)}, [N] \setminus S^{(i)})$ has less that $\epsilon d|S^{(i)}|/2$ edges. Omitting these cut edges yields a graph $G^{(i)}$ such that $G^{(i)}_{S^{(i)}}$ is T-minor free (and $S^{(i)}$ will not intersect with any future $S^{(j)}$). When the process ends, we have a T-minor free graph. In total, we omitted at most $\sum_i \epsilon d|S^{(i)}|/4 \leq \epsilon dN/2$ edges (from $G^{(0)}$), and thus G is $(\rho + (\epsilon/2))$-close to T-minor free.

The crux of the proof is indeed the process of identifying a suitable set $S' = S^{(i)}$ in $G' \stackrel{\text{def}}{=} G^{(i-1)}$. The identification procedure is initiated at $s' = s^{(i)}$ and proceeds in two stages. In the first stage, the procedure tries to find either a set S_0 of size at least $4m/\epsilon$ such that the cut $(S_0, [N] \setminus S_0)$ has less than m edges or a set S_0 of size at most $2dm/\epsilon$ such that G'_{S_0} contains an m-star as a minor rooted at s'. Clearly, in the first case we are done. In the second case, we get to the second stage of the procedure, which explores G' (somewhat) beyond S_0 in an attempt to extend the m-star minor into a T-minor, but this attempt is bound to fail, and this failure will allow finding the desired cut. Loosely speaking, this second stage proceeds by trying to find disjoint T_j-minors, for $j = 1, \ldots, m$. This is done by invoking a "k'-star-minor finding" procedure, denoted $\texttt{FS}_{k'}$, which generalizes the procedure that is described in the proof of Claim 7.5. The procedure $\texttt{FS}_{k'}$ is invoked on a vertex, v, and a set of forbidden vertices, denoted F, and tries to either find a k'-star rooted at v in $G'_{[N]\setminus F}$ or find a good cut. Indeed, F will contain the set S_0 as well as adequate sets that will prevent the current search from entering any of the previously found star minors. We first provide a specification of FS, and then turn to its actual implementation.

Specification of the procedure FS. On input a vertex v and a forbidden set F, the procedure $\texttt{FS}_{k'}$ outputs a triplet (σ, R', F') such that $\sigma \in \{\texttt{minor}, \texttt{cut}, \texttt{free}\}$ and $F' \subseteq R' \subseteq [N] \setminus F$ such that $|F'| < dk'$ and $|R'| < (4dk'/\epsilon) \cdot (|F| + 1)$. In addition, it always holds that all vertices of $G'_{R'}$ are connected to v, and one of the following cases holds.

$\sigma = \texttt{minor}$. The graph $G'_{R'}$ contains a k'-star as a minor that is rooted at v (i.e., v resides in the connected component that is contracted to fit the root of the k'-star). Furthermore, all edges of the cut $(R' \setminus F', [N] \setminus (R' \setminus F'))$ are incident at $F \cup F'$.

$\sigma = \texttt{cut}$. The cut $(R', [N] \setminus R')$ contains less that $\epsilon d|R'|/2$ edges.

$\sigma = \texttt{free}$. All edges of the cut $(R', [N] \setminus R')$ are incident at F.

Let T' denote a generic k'-star, where we may assume that $k' \geq 1$.

Implementing the procedure FS. Our aim is to either find a (relatively small) T'-minor or find a set with a relatively small cut from the rest of the graph. This is done by initiating a BFS in the residual graph $G'_{[N]\setminus F}$ starting at v, and stopping as soon as one of the following three cases occurs.

Case 1: *A layer containing at least k' vertices is found before $4(|F|+k')/\epsilon$ vertices are encountered.* In this case the procedure returns $(\texttt{minor}, R', F')$, where R' is the set of encountered vertices and F' is the set of vertices in the last BFS layer.

Note that in this case $G'_{R'}$ contains a T'-minor rooted at v, and that $|F'| < dk'$ (as otherwise the BFS would have terminated in a previous layer). Furthermore, by structure of the BFS, all edges of the cut $(R' \setminus F', [N] \setminus (R' \setminus F'))$ are incident at $F \cup F'$.

Case 2: *The search encountered at least $4(|F| + k')/\epsilon$ vertices, while Case 1 does not hold.* In this case the procedure returns $(\texttt{cut}, R', \emptyset)$, where R' is the set of encountered vertices.

Note that in this case the cut $(R', [N] \setminus R')$ contains less than $(|F| + k') \cdot d \leq \epsilon d|R'|/4$ edges.

Case 3: *The search cannot be extended any further, while Cases 1 and 2 do not hold.* In this case the procedure returns $(\texttt{free}, R', \emptyset)$, where R' is the set of encountered vertices.

Note that in this case the cut $(R', [N] \setminus R')$ contains only edges that are incident at F.

Observe that in the first case $|R'| < 4(|F| + k')/\epsilon$, in the second case $|R'| < 4d(|F| + k')/\epsilon$ (since otherwise the BFS could have stopped in the previous level), and in the third case $|R'| < 4(|F| + k')/\epsilon$. Hence, in all cases

$$|R'| \ < \ 4dk'(|F| + 1)/\epsilon \ . \tag{10}$$

Thus, this implementation satisfies the specification. We note that the above description applies also in case $k' \in \{0, 1\}$, where $k' = 0$ is trivial[11] (i.e., always return $(\texttt{minor}, \{v\}, \{v\})$) and $k' = 1$ is almost trivial (i.e., return $(\texttt{minor}, \{v, w\}, \Gamma_{G'}(v) \setminus F)$ if v has a neighbor w in $G'_{[N]\setminus F}$ and $(\texttt{free}, \{v\}, \emptyset)$ otherwise).

Using the star finding procedure FS, we now turn to the main identification procedure, which is invoked on input vertex $s' = s^{(i)}$ and aims at finding an adequate set $S' = S^{(i)}$. Recall that r denotes the root of T, and $r_1, \ldots, r_m$ denote the roots of the subtrees $T_1, \ldots, T_m$, where T_j is a k_j-star. The main procedure operates as follows.

1. It initiates a BFS in the graph G' starting at s', stopping as soon as at least $B = 4dk/\epsilon$ vertices are encountered. Let S_0 denote the set of encountered vertices. Note that $|S_0| \geq B$ must hold, because $s' = s^{(i)}$ resides in a set of vertices that can be contracted to the root of some T-minor having radius greater than D.

   Note that necessarily $|S_0| < dB$ (because otherwise we would have stopped at the previous BFS-layer).

2. Let F_0 denote the last layer in the BFS performed in the previous step. If $|F_0| < m$, then we just use S_0 as the desired set (i.e., let $S^{(i)} = S_0$).

[11] Actually, this case never occurs; that is, we never invoke $\texttt{FS}_0$. The case $k' = 1$ may occur, but we could have avoided it too, by a direct treatment.

Note that, in this case, the cut $(S_0, [N] \setminus S_0)$ contains less than $m \cdot d$ edges, whereas by the case hypothesis $|S_0| \geq B > 4m/\epsilon$ (as $k > m$). So the conditions regarding this set are satisfied.

We continue to the next step only if $|F_0| \geq m$.

3. (The purpose of the current step is to generate calls to `FS` that will eventually lead to returning a set as in the second output case (i.e., `cut`), which can serve as $S^{(i)}$ (see above). The presentation, however, pretends that we attempt to find a T-minor as in the first output case (i.e., `minor`). Observing that $S_0 \setminus F_0$ can serve as a contraction of the root of T, we attempt to find disjoint sets S_j that contain T_j-minors rooted at some $v_j \in F_0$.)

   For $j = 1, \ldots, m$, we try to find S_j as follows. Let $F' = \bigcup_{a \in [j-1]} F_a$ and $V' = \{v_1, \ldots, v_{j-1}\}$. For every $v \in F_0 \setminus V'$, we proceed as follows.

   We invoke $\texttt{FS}_{k_j}$, letting $(\sigma, X, Y) \leftarrow \texttt{FS}_{k_j}((F_0 \setminus \{v\}) \cup F', v)$.

   We note that by the specification of $\texttt{FS}_{k_j}$ and Equation (10), we have that $|X| \leq (4dk_j/\epsilon) \cdot (|F_0| + |F'| + 1)$ and $|Y| \leq dk_j$. Recall that $|F_0| < |S_0| < dB = 4d^2k/\epsilon$ and $|F'| = \sum_{a \in [j-1]} |F_a| < d \sum_{a \in [j-1]} k_a < d(k - m)$, where $k = 1 + m + \sum_{a \in [m]} k_a$. Thus, $|X| < (5d^2k/\epsilon)^2$.

   We consider the following three cases regarding σ.

   $\sigma =$ **`minor`.** In this case we set $v_j \leftarrow v$ and $(S_j, F_j) \leftarrow (X, Y)$, and proceed to the next value of j (i.e., $j \leftarrow j + 1$); see comment below.

   Note that $|S_j| < (5dk/\epsilon)^2$. In fact, the same upper bound can be proved for $\sum_{a=0}^{j} |S_a|$.

   Note that this case cannot occur when $j = m$, because this would yield a small T-minor rooted in s' in contradiction to the hypothesis that $s' = s^{(i)}$ is not bad.

   $\sigma =$ **`cut`.** In this case we just use X as the desired set (i.e., let $S^{(i)} = X$).

   Note that, by the specification of `FS`, the cut $(S^{(i)}, [N] \setminus S^{(i)})$ contains relatively few edges.

   $\sigma =$ **`free`.** In this case we do nothing, and continue to the next candidate v.

   Note that we halted with a desired cut if either Step 2 found such a cut or any of the invocations of `FS` returned a `cut`-value. Furthermore, as noted in the above discussion concerning the case $\sigma =$ `minor`, it cannot be the case that in Step 3 we obtained a `minor`-value for each $j \in [m]$. Thus, we remain with the case that, for some $j \in [m]$, all invocations of `FS` returned a `free`-value. In this case, we let X' be the union of all sets X that were returned in the corresponding $|F_0| - (j - 1)$ invocations, and use $S_0 \cup X'$ as the desired set (i.e., let $S^{(i)} = S_0 \cup X'$). In this case, the size of the cut $(S^{(i)}, [N] \setminus S^{(i)})$ is at most $d \cdot |F'| < d^2k$, because for each X all edges of the cut $(X, [N] \setminus X)$ are incident at $F_0 \cup F' \subseteq S_0 \cup F'$. Thus, the cut is sufficiently small, because $|S^{(i)}| \geq |S_0| \geq B = 4dk/\epsilon$. On the other hand, the size of $S_0 \cup X'$ is at most $|F_0| \cdot (4dk/\epsilon) \cdot |F'| < (4dk/\epsilon)^2$.

This completes the description of the operation of the procedure I as well as the showing that it satisfies its specification. It follows that for any $s^{(i)}$ that reside in the root of some T-minor in $G^{(i-1)}$, we obtain a set $S^{(i)}$ such that the cut $(S^{(i)}, [N] \setminus S^{(i)})$ has less than $4d|S^{(i)}|/\epsilon$ edges. Using the fact $|S^{(i)}| < D$, it follows that $G^{(i-1)}_{S^{(i)}}$ is T-minor free, and the lemma follows. ■

8 The unbounded-degree graph model

In this section we consider (one-sided error) testing cycle-freeness (and tree-minor freeness) in what we shall refer to as the unbounded-degree incidence-lists model. In this model, introduced in [PR02], the maximum degree d may be as large as $N-1$, so there is effectively no degree-bound, and a graph G is represented by a function $g : [N] \times [N-1] \to \{0, \ldots, N\}$. Similarly to the bounded-degree model, the algorithm may ask for the identity of the i^{th} neighbor of a vertex v, for any $v \in [N]$ and $i \in [N-1]$ of its choice, by querying the function g. (If v has less than i neighbors, then the answer returned is '0'). For the sake of simplicity, we assume that the algorithm can also query the degree of any vertex of its choice (where such a query can, of course, be replaced by $O(\log N)$ neighbor queries).

The main and crucial difference between the unbounded-degree model and the bounded-degree model is in the *distance measure between graphs*. Rather than measuring distance between graphs in terms of the size of the domain of g, as done in the bounded-degree model, we measure it with respect to the number of edges $|E|$ in $G = ([N], E)$. That is, we shall say that a graph G is ϵ-far from being cycle-free (in the unbounded-degree model), if the number of edges that must be removed in order to make it cycle-free is greater than $2\epsilon|E|$ (see Footnote 1). Letting d_{avg} denote the average degree in G (and assuming that G is connected), this is equivalent to saying that the number of edges in G is greater than $(N-1) + \epsilon d_{\text{avg}} N$.

We note that while the bounded-degree model is appropriate for testing graphs in which the maximum degree is of the same order as the average degree (and in particular for constant-degree graphs), the unbounded-degree model is appropriate for testing graphs in which the maximum degree may be much larger than the average degree. We mention that the model considered in [KKR04] (see also Section 8.3) also allows adjacency queries (as in [GGR98]), but such queries are useless for us when the degree is smaller than $\sqrt{N}$.

A necessary assumption. Throughout this section, we assume that the number of edges in the graph is at least linear in the number of vertices. As noted in [KKR04], without this assumption the model becomes intractable for sublinear algorithms, because one can always hide a tiny graph (which may be far from having the property) inside a huge graph that contains mostly isolated vertices. Also, unless explicitly stated otherwise, we also assume that the graphs are simple (i.e., have no self-loops or parallel edges).

8.1 Testing cycle-freeness

In this subsection, we show that the result of Theorem 1.5 (and thus also Theorem 1.1) extends to the unbounded-degree (incidence lists) model. Furthermore, the extended result is actually stronger than stated in Theorem 1.5, since we eliminate the dependence of the complexities on the average (let alone maximum) degree of vertices in the input graph. This is done by viewing the randomized reduction that underlies Algorithm 3.3 in a slightly different manner, which actually yields an alternative tester (which is closely related to but different from Algorithm 3.3). We then show that this algorithm extends easily to the unbounded-degree model.

The pivot of our exposition is the following generalization of 2-colorability in which edges of the graph are labeled by either `eq` or `neq`. That is, an instance of this problem is a graph $G = ([N], E)$ along with a labeling $\Lambda : E \to \{\texttt{eq}, \texttt{neq}\}$. We say that $\chi : [N] \to \{0, 1\}$ is a legal 2-coloring of this instance if for every $\{u, v\} \in E$ it holds that $\chi(u) = \chi(v)$ if and only if $\Lambda(\{u, v\}) = \texttt{eq}$. That is, a legal 2-coloring (of the vertices) is one in which every two vertices that are connected by an edge labeled `eq` (resp. `neq`) are assigned the same color (resp., opposite colors). Note that the standard notion of 2-colorability corresponds to the case in which all edges are labeled `neq`.

We first observe that the (one-sided error) Bipartite testers of [GR99] and [KKR04] can be extended to test this generalization of 2-colorability.[12] Next, we observe that the randomized reduction that underlies Algorithm 3.3 can be viewed as a randomized reduction of cycle-freeness to generalized 2-coloring, while keeping the graph intact. Combining these two observations, we obtain:

Theorem 8.1 (Theorems 1.5 and 1.1, generalized to the unbounded-degree model): *Cycle-freeness in N-vertex graphs can be tested with one-sided error within time complexity* $\widetilde{O}(\text{poly}(1/\epsilon) \cdot \sqrt{N})$, *where ϵ denotes the proximity parameter that refers to the number of* (omitted) *edges as a fraction of the total number of edges in the input graph. Furthermore, whenever the tester rejects, it outputs a simple cycle of length* $\text{poly}(\epsilon^{-1} \log N)$.

Proof: We first present a (local) randomized reduction of testing cycle-freeness to testing the foregoing generalization of 2-colorability. Unlike the reduction that underlies Algorithm 3.3, the current reduction does not modify the input graph $G = ([N], E)$; it merely introduces a random labeling $\Lambda : E \to \{\mathtt{eq}, \mathtt{neq}\}$. Specifically, the graph $G = ([N], E)$ is mapped to a random instance of the generalized 2-coloring problem such that the graph equals G itself and the labeling is selected uniformly among all possible $\Lambda : E \to \{\mathtt{eq}, \mathtt{neq}\}$. Thus, invoking any generalized 2-coloring (one-sided error) tester on the resulting instance, we are done. (Indeed, unlike in the case of Algorithm 3.3, here the emulation of the generalized 2-coloring tester is straightforward. Also note that here we obtained a true local reduction, whereas in Section 3 we only showed how to emulate bipartite testers of a special type (i.e., such that only select uniformly distributed vertices and make random-neighbor queries).)

The analysis of the forgoing randomized reduction is analogous to the proof of Lemma 3.1. For sake of clarity, we spell out what this means.

1. If G is cycle-free, then, for every choice of $\Lambda : E \to \{\mathtt{eq}, \mathtt{neq}\}$, the graph G is 2-colorable with respect to the labeling Λ (i.e., the exists a legal 2-coloring of the instance (G, Λ)).

2. If G is not cycle-free, then, with probability at least $1/2$ (over the random choice of $\Lambda : E \to \{\mathtt{eq}, \mathtt{neq}\}$), the graph G is not 2-colorable with respect to the labeling Λ.

3. There exist universal constants $c_1 > 1$ and $c_2, c_3 > 0$ such that, for every $\epsilon \geq c_1/|E|$, if G is ϵ-far from being cycle free, then, with probability at least $1 - \exp(-c_2 \epsilon d N)$ (over the random choice of $\Lambda : E \to \{\mathtt{eq}, \mathtt{neq}\}$), at least $c_3 \epsilon \cdot |E|$ edges must be omitted from G in order to obtain a graph that is 2-colorable with respect to the labeling Λ.

Each of the above three claims may be proved by mimicking the proof of the corresponding item of Lemma 3.1. In particular, note that in the proof of the third item, we actually established that (w.v.h.p.) $c_3 \cdot (|E| - N + 1)$ edges must be omitted from G_τ in order to obtain a bipartite graph,[13]

[12] A similar observation refers to the k-colorability testers of [GGR98], which operate in the dense graph model. Thus, for every $k \geq 2$, the foregoing generalization of k-colorability can be tested (with one-sided error) in the dense graph model by using $\text{poly}(1/\epsilon)$ queries.

[13] The stated distance of $c_3\epsilon/2d$ in the bounded degree model, was obtained by using $|E| - N + 1 = \epsilon d N$ (which holds there), and dividing the result by $d \cdot (d+1)N$ (which represents the product of the degree of G_τ by an upper bound on the number of its vertices).

which means that these many edges must be omitted from G in order to obtain a graph that is 2-colorable with respect to the labeling Λ.

We turn to presenting a generalized 2-coloring (one-sided error) tester for the unbounded-degree model. One way of doing so it to observe that the 2-coloring tester of [KKR04] (which builds upon [GR99]) extends to the generalized version. We shall *not* follow this way, but rather present an alternative one, but before doing so we comment that all that is needed in order to support this observation is to (fictitiously) *define* edges labeled `eq` as having even length (say, length zero or two), whereas edges labeled `neq` are defined as having odd length (say, length one). Modulo this definition, the entire analysis of [GR99] remains intact. Specifically, all references in [GR99] to the length of paths and cycles are re-interpreted as referring to the foregoing definition. In particular, an odd length cycle (under this label-dependent definition of length) indicates that the graph cannot be 2-colored (under the corresponding labeling of edges), whereas the non-existence of odd length cycles enables such a 2-coloring. (The same holds for [KKR04], which operates by a (local) reduction to [GR99].)

Seeking a self-contained presentation, we present an alternative way of deriving a generalized 2-coloring (one-sided error) tester for the unbounded-degree model. Specifically, we shall reduce generalized 2-coloring testing to standard 2-coloring testing as follows. Given an instance (G, Λ) of the generalized 2-coloring problem, where $G = ([N], E)$ and $\Lambda : E \to \{\mathtt{eq}, \mathtt{neq}\}$, we consider the following multi-graph $G' = ([2N], E')$, where a multi-graph is a graph with parallel edges: Each vertex $v \in [N]$ of G is replaced by two vertices, v and $N + v$, which are connected by $2|\Gamma_G(v)|$ parallel edges, and the edges of G are replaced by edges among the corresponding copies such that edges labeled `neq` are replaced by edges between "matching copies" and edges labeled `eq` are replaced by edges between opposite copies. Specifically, if the edge $\{u, v\} \in E$ is labeled `neq` (i.e., $\Lambda(\{u, v\}) = \mathtt{neq}$), then E' contains the edges $\{u, v\}$ and $\{N + u, N + v\}$, otherwise (i.e., $\Lambda(\{u, v\}) = \mathtt{eq}$) the edges $\{u, v + N\}$ and $\{N + u, N\}$ are placed in E'.

Note that the degree of each vertex v in G' is thrice the degree of the original vertex $v' \overset{\text{def}}{=} (v - 1) \bmod N + 1$ in G, where the first $2|\Gamma_G(v)|$ edges go to the other copy of v' and the rest are connected to the adequate copies of neighbors of v'. Clearly, this reduction is local; specifically, given oracle access to G and Λ, one can implement oracle access to G' by issuing a single query to G per each query to G'. The reader can easily verify that the number of edges that must be omitted from G in order to obtain a graph that is 2-colorable with respect to Λ equals half the number of edges that must be omitted from G' in order to obtain a bipartite graph. (The key observation is that when 2-coloring G', the number of edges that are violated is minimized by assigning the two copies of each vertex of G different colors).

It seems that we are done, except that the multi-graph obtained in the reduction has parallel edges, whereas the algorithm of [KKR04] only refers to simple graphs (i.e., no parallel edges). This assumption is used in [KKR04] for a single purpose – for the construction of a randomized subroutine that samples vertices with probability that is (approximately) proportional to their degree. Other than that, both the algorithm and the analysis of [KKR04] are oblivious to whether the input is a simple graph or a multi-graph. The algorithm and its analysis refer to the behavior of random walks taken on an auxiliary graph, and at that level it does not matter whether the original graph (or the auxiliary graph) have parallel edges or not. Thus, we merely need to address the problem of sampling vertices in the graph G' obtained via our reduction such that vertices are sampled with probability that is (approximately) proportional to their degree.

The latter problem is easily solved by noticing that if we drop all parallel edges from G', we obtain a graph G'' such that the degree of each vertex in G'' is one third of its degree in G'. Hence, we may just as well invoke the vertex-sampling procedure of [KKR04] on G'', which is a simple graph. The theorem follows. ■

Digest. Note that combining the two reductions presented in the foregoing proof, we obtain a randomized reduction of (one-sided error) testing cycle-freeness to (one-sided error) testing bipartiteness (of graphs having parallel edges). The advantage of this reduction over the one presented in Section 3 is that the number of vertices in the resulting graph is only a constant factor larger than the number of vertices in the original graph. (The same holds for the number of edges, but this was true also for the reduction in Section 3.) The disadvantage of the current reduction (in comparison to the one presented in Section 3) is that it yields graphs with parallel edges. Fortunately, the number of parallel edges incident at each vertex is a fixed fraction of the degree of this vertex, which in turn allows for an easy adaptation of the Bipartiteness tester presented in [KKR04] so that it can be applied to these graphs.

Finding longer cycles. We were not able to extend our results regarding finding C_k-minors for $k > 3$ to the unbounded-degree model. Our reductions to finding simple cycles (i.e., to the case of $k = 3$) increase the size of the graph by a factor of at least d^{k-3}, where d is the initial degree bound, putting aside the difficulty of handing the case that the average degree is significantly smaller than d. Indeed, this topic is left for further study.

8.2 Testing tree-minor-freeness

In contrast to Section 8.1, we show that the result of Theorem 1.3 cannot be extended to the unbounded-degree model. This follows by considering an N-vertex graph G that consists of a cycle of length $N - \sqrt{N}$ and a clique of size $\sqrt{N}$ (i.e., $G = C_{N-\sqrt{N}} + K_{\sqrt{N}}$). Denoting the 3-star by T_3, note that G is $\Omega(1)$-far from being T_3-minor-free (since we must omit $\sqrt{N} - 3$ edges from each vertex of the $\sqrt{N}$-clique in order to eliminate all copies of T_3 itself). On the other hand, no $o(\sqrt{N})$-query algorithm can find a T_3-minor in a random isomorphic copy of G, except with probability $o(1)$. Furthermore, any algorithm of query complexity $o(\sqrt{N})$ cannot distinguish a random copy of G from a random copy of a N-vertex graph that consists of a cycle of length $N - \sqrt{N}$ and $\sqrt{N}$ isolated vertices. Thus, in this model, even two-sided error testing of T_3-minor freeness requires $\Omega(\sqrt{N})$ queries.

We mention that an $O(\sqrt{N})$-query one-sided error tester for T_k-minor-freeness does exist for any k, where T_k denotes the k-star. This tester may be obtained by combining the tester for the bounded-degree model (for $d = k-1$, as presented in Section 7.3) with an $O(\sqrt{N})$-query procedure for finding a vertex of degree at least k. The argument is detailed in the proof of the following result.

Theorem 8.2 (Testing T_k-minor freeness in the unbounded-degree model): *For every $k \geq 3$, testing T_k-minor freeness of N-vertex graphs, in the unbounded-degree model, has query complexity $\widetilde{\Theta}(\sqrt{N})$, where the upper bound hides factors that are polynomial in k/ϵ. Furthermore, the lower bound holds also for two-sided error testers, whereas the upper bound holds with respect to a one-sided error tester that outputs a T_k-minor of size $O(k^2/\epsilon)$ whenever it rejects.*

Proof: The lower bound follows from the foregoing discussion. Recall that we consider an N-vertex graph G that consists of a cycle of length $N-\sqrt{N}$ and a clique of size $\sqrt{N}$, which is $\Omega(1)$-far from being T_k-free. We claim that any algorithm of query complexity $o(\sqrt{N})$ cannot distinguish a random copy of G from a random copy of a N-vertex graph that consists of a cycle of length $N-\sqrt{N}$ and $\sqrt{N}$ isolated vertices. This claim is proved by noting that, conditioned on the case that all the prior queries refer to vertices that are on the long cycle, the probability that the next query refers to a vertex not on this cycle is at most $\sqrt{N}/N$, where equality holds if the next query refers to a vertex that was not seen before (as either part of a query or an answer).

We now turn to the upper bound. The suggested tester invokes an arbitrary (one-sided error) T_k-minor free tester for the bounded-degree model (e.g., Algorithm 7.4), where the degree bound, d, is set to $k-1$. Indeed, this algorithm is invoked on a graph that may not satisfy the degree bound, and so while emulating this algorithm we check whether each encountered vertex has degree at most $k-1$. If (during the emulation) we ever encounter a vertex of degree at least k, then we reject (while outputting this vertex and k of its neighbors). If the emulation terminates outputting a T_k-minor, then we just output this minor. Otherwise, using the edge sampling algorithm of [KKR04, Fig. 5] (which has query complexity $\widetilde{O}(\sqrt{N/\epsilon})$ per sample), we take an almost uniform sample of $O(1/\epsilon)$ edges and check the degrees of the endpoints of all these edges. Needless to say, if any of these vertices is found to have degree at least k, then we reject (while outputting this vertex and k of its neighbors), otherwise we accept.

Suppose that $G=([N],E)$ is ϵ-far from being T_k-minor free, which means that at least $2\epsilon|E|$ edges have to be removed from G in order to obtain a T_k-minor free graph. Let H denote the set of vertices in G that have degree at least k, let $G'=(V',E')$ be the subgraph of G that is induced by $V'\stackrel{\rm def}{=}[N]\setminus H$, and $G''=([N],E')$ be G' augmented by $|H|$ isolated vertices. If at least $\epsilon|E|$ edges have to be removed from G' in order to obtain a T_k-minor free graph, then G'' is $\epsilon/4k$-far from being a T_k-minor free (with respect to the bounded-degree model with $d=k-1$, while assuming $|E|>N/2$). Thus, Algorithm 7.4 invoked on G'' (with proximity parameter $\epsilon/4k$), will output a T_k-minor with probability at least 2/3. The same would happen if Algorithm 7.4 is invoked on G, because as long as vertices in H are not visited both graphs are indistinguishable, whereas visiting any graph in H will cause the algorithm to output a T_k-minor.

So we are left with the complimentary case in which G can be made T_k-minor free by omitting at least $\epsilon|E|$ edges that have at least one endpoint in H. In this case, the edge sampling algorithm of [KKR04] (invoked with proximity parameter $\epsilon/2$), will hit such an edge with probability at least $\Omega(\epsilon)$ per each invocation (see [KKR04, Lem. 6]). (Needless to say, hitting any edges with an endpoint in H will cause the algorithm to output a T_k-minor.) The theorem follows by noting that we invoked this edge sampling algorithm $O(1/\epsilon)$ times, whereas its query complexity is $\widetilde{O}(\sqrt{N/\epsilon})$ (and the query complexity of Algorithm 7.4 is $O(k^2/\epsilon)$). ■

Finding other tree-minors. Our star-minor free (one-sided error) tester to the unbounded-degree model, begs the question of whether similar results can be obtained with respect to other tree-minors. Indeed, this question is left for further study.

8.3 Testing with adjacency queries

Here we consider an augmentation of the model with adjacency queries. This augmentation was first considered in [KKR04], and it was shown to be useful (for testing bipartiteness) when the average degree, $d_{\rm avg}$, exceeds $\sqrt{N}$. We observe that the same holds with respect to testing cycle-freeness (see details below). (In contrast, recall that in the bare model (i.e., without adjacency queries) the upper bound presented in Section 8.1 are optimal.)

We note that the reduction presented in Section 8.1 remains valid, except that in this case the generalized 2-coloring tester (derived from [KKR04]) may use adjacency queries. In this case, the resulting cycle-freeness tester will have complexity $\min(\widetilde{O}(\sqrt{N}), \widetilde{O}(N)/d_{\rm avg}) \cdot {\rm poly}(1/\epsilon)$ (just as the 2-coloring tester of [KKR04]).

We also note that the results regarding testing tree-minors (see Section 8.2) extend similarly. Specifically, for any $k \geq 3$, the complexity of testing T_k-minor freeness is $\min(\widetilde{\Theta}(\sqrt{N}), \widetilde{\Theta}(N)/d_{\rm avg}))$.

9 Open Problems

The current paper leaves open many questions regarding the complexity of one-sided error testing of H-minor-freeness, for arbitrary (fixed) graphs H. While some significant progress has been done regarding this general question, much is left to be desired (even if we restrict ourselves to the bounded-degree model).

Let us denote by $\mathtt{QT}_H : \mathbb{N} \times [0,1] \to \mathbb{N}$ the query complexity of one-sided error testing of H-minor-freeness in the bounded-degree model. (For simplicity, we fix the degree bound and consider the number of queries made as a function of the number of vertices and the proximity parameter.) The most begging open problem is whether or not, for any fixed $\epsilon_0 > 0$ and graph H, it holds that $\mathtt{QT}_H(N, \epsilon_0) = o(N)$. We showed that $\mathtt{QT}_H(N, \epsilon_0) = \widetilde{\Theta}(\sqrt{N})$ for any H that is a cycle, and $\mathtt{QT}_H(N, \epsilon_0) = O(1)$ for any cycle-free H, but for other graphs H that have cycles we only know that $\mathtt{QT}_H(N, \epsilon_0) = \Omega(\sqrt{N})$. The only step beyond this state of knowledge is represented by the following result, which we discovered recently.

Proposition 9.1 *Let H be a 4-vertex graph consisting of a triangle and an additional edge. Then, for any fixed $\epsilon_0 > 0$, it holds that $\mathtt{QT}_H(N, \epsilon_0) = \widetilde{O}(\sqrt{N})$.*

There seems to be hope to obtain a similar result for any graph H that contains exactly one cycle (along with some additional edges), but we do not see how to do this at the moment.[14] Another natural case to consider is the one in which H is the 4-vertex clique (or the 4-vertex clique with one edge omitted).

Proof: The lower bound follows as a special case of Theorem 6.1, and so our focus is on the upper bound. The idea is to invoke a one-sided tester for cycle-freeness (e.g., Algorithm 3.3), and scan the simple cycles in the subgraph that this algorithm has explored. Our goal is to find a cycle that contains a vertex of degree greater than two, since any such cycle yields a desired H-minor. (If this vertex has a neighbor outside the cycle, then the cycle combined with the corresponding edge yields an H-minor. Otherwise the non-cycle edge is a chord, and we can use this chord together with half of the cycle and some other edge to form an H-minor.)

Note that if the current cycle that we scan does not yield an H-minor, then it resides on an isolated cycle in the input graph. In this case, we omit it from the subgraph explored by the algorithm and continue looking for other cycles (which may still exist) in this subgraph.

[14]Even the two cases of 5-vertex graphs that contain a single cycle and a total of five edges cannot be handled by the idea that underlies the proof of Proposition 9.1.

To analyze this algorithm assume that G is ϵ-far from being H-minor free, and consider the isolated cycles (of vertices having degree G) that may exist in G. Let G' be a graph obtained from G by omitting a single edge from each such isolated cycle. Then, G' is also ϵ-far from being H-minor free (and so also ϵ-far from being cycle-free), whereas invoking our algorithm on G' will yield an H-minor with probability at least $2/3$ (since any cycle in G' contains a vertex of degree exceeding 2). But then the same must happen also when we invoke our algorithm on G, because the manner in which Algorithm 3.3 explores one connected component is independent of its exploration of a different connected component. ■

Finer bounds. Focusing on the case that H is cycle-free, we know that $\texttt{QT}_H(N,\epsilon) = F_H(\epsilon)$ for some function F_H. A natural question refers to the exact dependence of $F_H(\epsilon)$ on the graph H. Specific questions include:

1. We know that if H is a k-path, then $F_H(\epsilon)$ is at most exponential in k. *Can $F_H(\epsilon)$ be polynomial in k?*

   (Partial progress is reported in [Rez11], which deals with the special case that the input graph is a tree.)

2. We know that if H is a k-vertex tree, then $F_H(\epsilon)$ is at most double-exponential in k. *Can $F_H(\epsilon)$ be at most exponential in k?*

   (In case H has at most depth two, the answer is positive, as reported in [CGR$^+$10, Sec. 7.5]. Also recall that if H has depth one (i.e., is a star), then $F_H(\epsilon)$ is polynomial in k.)

Also recall that when H is a k-cycle, we have $\texttt{QT}_H(N,\epsilon) = \widetilde{O}(\sqrt{N}) \cdot \text{poly}(2^k/\epsilon)$, and one may ask whether $\texttt{QT}_H(N,\epsilon) = \widetilde{O}(\sqrt{N}) \cdot \text{poly}(k/\epsilon)$ is possible.

The unbounded degree model. Turning to the unbounded degree model, which was considered in Section 8, we let $\texttt{QT}'_H : \mathbb{N} \times \mathbb{N} \times [0,1] \to \mathbb{N}$ denote the corresponding query complexity (assuming only neighbor queries), which is now a function of the number of vertices, the (approximate) average degree, and the proximity parameter. We showed that, for every fixed $\epsilon_0 > 0$ and H that is either a 3-cycle or a star, it holds that $\texttt{QT}'_H(N,d,\epsilon_0) = \widetilde{\Theta}(\sqrt{N})$. The open questions here refer even to other cycles and trees (for which the complexity in the bounded-degree model is known).

Acknowledgments

We are grateful to Aviv Reznik for useful comments on a prior version, and to the anonymous reviewers for their numerous comments, corrections, and suggestions.

References

[AS03] N. Alon and A. Shapira. Testing satisfiability. *Journal of Algorithms*, 47:87–103, 2003.

[BSS08] I. Benjamini, O. Schramm, and A. Shapira. Every minor-closed property of sparse graphs is testable. In *Proceedings of the Fourtieth Annual ACM Symposium on the Theory of Computing*, pages 393–402, 2008.

[BT97] B. Bollobás and A. Thomason. On the girth of Hamiltonian weakly pancyclic graphs. *Journal of Graph Theory*, 26(3):165–173, 1997.

[CGR$^+$10] A. Czumaj, O. Goldreich, D. Ron, C. Seshadhri, A. Shapira, and C. Sohler. Finding cycles and trees in sublinear time. Technical report, arXiv paper number 1007.4230 [cs.DS, cs.DM], 2010. Also available as ECCC report TR12-035.

[Fis01] E. Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of the European Association for Theoretical Computer Science*, 75:97–126, 2001.

[FP87] J. Friedman and N. Pippenger. Expanding graphs contain all small trees. *Combinatorica*, 7:71–76, 1987.

[GGR98] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[GR99] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.

[GR02] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002.

[GS07] O. Goldreich and O. Sheffet. On the randomness complexity of property testing. In *Proceedings of the Eleventh International Workshop on Randomization and Computation (RANDOM)*, pages 296–310, 2007.

[GT03] O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Structures and Algorithms*, 23(1):23–57, 2003.

[HKNO09] A. Hassidim, J. Kelner, H. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Proceedings of the Fiftieth Annual Symposium on Foundations of Computer Science (FOCS)*, 2009.

[KKR04] T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004.

[KKR12] K. Kawarabayashi, Y. Kobayashi, and B. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory Series B*, 102(2):424–435, 2012.

[Kur30] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematica*, 15:271–283, 1930.

[PR02] M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, 20(2):165–183, 2002.

[Rez11] A. Reznik. Finding k-paths in cycle-free graph. Master's thesis, Weizmann Institute of Science, December 2011.

[Ron08] D. Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008.

[Ron10] D. Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5, 2010.

[RS95] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory Series B*, 63(1):65–110, 1995.

[RS04] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory Series B*, 92(1):325–357, 2004.

[Wag37] K. Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114:570–590, 1937.