Testing Hamiltonicity (And Other Problems) in Minor-Free Graphs

Reut Levi ✉ 🆔
Efi Arazi School of Computer Science, The Interdisciplinary Center Herzliya, Israel

Nadav Shoshan ✉
Efi Arazi School of Computer Science, The Interdisciplinary Center Herzliya, Israel

──── **Abstract** ────

In this paper we provide sub-linear algorithms for several fundamental problems in the setting in which the input graph excludes a fixed minor, i.e., is a minor-free graph. In particular, we provide the following algorithms for minor-free unbounded degree graphs.

1. A tester for Hamiltonicity with two-sided error with poly($1/\epsilon$)-query complexity, where ϵ is the proximity parameter.

2. A local algorithm, as defined by Rubinfeld et al. (ICS 2011), for constructing a spanning subgraph with almost minimum weight, specifically, at most a factor $(1 + \epsilon)$ of the optimum, with poly($1/\epsilon$)-query complexity.

Both our algorithms use partition oracles, a tool introduced by Hassidim et al. (FOCS 2009), which are oracles that provide access to a partition of the graph such that the number of cut-edges is small and each part of the partition is small. The polynomial dependence in $1/\epsilon$ of our algorithms is achieved by combining the recent poly(d/ϵ)-query partition oracle of Kumar-Seshadhri-Stolman (ECCC 2021) for minor-free graphs with degree bounded by d.

For bounded degree minor-free graphs we introduce the notion of *covering partition oracles* which is a relaxed version of partition oracles and design a poly(d/ϵ)-time covering partition oracle for this family of graphs. Using our covering partition oracle we provide the same results as above (except that the tester for Hamiltonicity has one-sided error) for minor-free bounded degree graphs, as well as showing that any property which is monotone and additive (e.g. bipartiteness) can be tested in minor-free graphs by making poly(d/ϵ)-queries.

The benefit of using the covering partition oracle rather than the partition oracle in our algorithms is its simplicity and an improved polynomial dependence in $1/\epsilon$ in the obtained query complexity.

1 Introduction

The family of minor-free graphs has been at the focus of attention ever since the theory of graph minors began many decades ago and has been drawing much attention in the field of computer science as well. Aside from being an important family that includes natural families of graphs such as planar graphs, it also has the appeal that some hard graph problems become easy when restricted to this family of graphs (e.g. Graph Isomorphism [15]).

Minor-free graphs have been extensively studied also in the realm of sublinear algorithms and in particular property testing (see e.g. [4, 5, 29, 21, 17, 3, 10, 18, 7, 6, 19, 20]). In particular, in the general-graph model [30] where there is much less body of work, compared to the bounded-degree graph model [11] and the dense graph model [13], these graphs draw attention as they allow for better characterization compared to general unbounded degree graphs. A notable example is the result by Czumaj and Sohler [7] who recently gave a full characterization of properties that can be tested with one-sided error with query complexity that is independent of the size of the graph (i.e. *testable*). They showed that the latter is possible if and only if testing the property can be reduced to testing for a finite family of finite forbidden subgraphs. This raises the question regarding the testability of properties that can not be reduced to testing for a finite family of finite forbidden subgraphs when we allow the tester to have two-sided error. A well known example of such property is the property of being Hamiltonian.

Another question, which is also relevant for bounded degree graphs, is whether we can obtain algorithms with query complexity which is only polynomial in $1/\epsilon$ where ϵ is the proximity parameter. Newman and Sohler [29] showed that any property of bounded degree hyperfinite graphs and in particular minor-free graphs is testable. Their algorithm learns the graph up to modifications of ϵdn edges with query complexity which is super-polynomial in d and ϵ. While this approach works for all properties of graphs, more efficient testers can be obtained for specific properties of graphs. In particular, properties of graphs which are monotone and additive can be tested by using $O(d/\epsilon)$ queries to a partition oracle, a tool introduced by Hassidim et al. [14]. Thus, an implication of the recent poly(d/ϵ)-query partition oracle of Kumar-Seshadhri-Stolman [20] is that monotone and additive properties are testable with poly(d/ϵ)-queries. Thus the question of designing testers with poly$(1/\epsilon)$-queries remains open for properties which are not monotone or not additive, as the property of being Hamiltonian.

An additional motivation for studying Hamiltonicity in minor-free graphs is, as shown by Yoshida-Ito [33] and Goldreich [12], that it can not be tested with sublinear query complexity in general bounded degree graphs.

1.1   Our Results

All our algorithms work under the promise that the input graph is minor-free.

1.1.1   Testing Hamiltonicity

In the general graph model, we provide an algorithm for approximating the distance from Hamiltonicity up to an additive error of ϵn where n denotes the number of vertices in the input graph with query complexity which is poly$(1/\epsilon)$ and time complexity which is exponential in poly$(1/\epsilon)$. This also implies a tolerant tester with two-sided error for testing Hamiltonicity with the same complexities.

In the bounded-degree graph model, we provide an algorithm for testing Hamiltonicity with one-sided error with query complexity which is poly(d/ϵ), where d denotes the bound on the degree, and time complexity which is exponential in poly(d/ϵ).

1.1.2 Local algorithm for constructing spanning subgraphs of almost optimum weight

In the general graph model, we provide a local algorithm for constructing a sparse spanning subgraph of weight at most $(1 + \epsilon)$OPT where OPT denotes the weight of the MST of the input graph. The algorithm receives as parameters ϵ and an upper bound, W, on the maximum weight of an edge in the graph. Moreover, the number of edges of the output graph that do not belong to the MST of the input graph[1] is $O(\epsilon n/W)$. The query complexity and time complexity of the algorithm is poly(W/ϵ). We note that in addition to incidence queries our algorithm also use random neighbour queries.

In the bounded-degree graph model, we provide a simpler algorithm with the same guarantees whose query complexity and time complexity is poly(dW/ϵ), where the polynomial of the complexity is somewhat improved compared to the algorithm for graphs of unbounded degree.

1.1.3 Testing monotone and additive properties of graphs

We prove that any property which is monotone (closed under removal of edges and vertices) and additive (closed under the disjoint union of graphs) can be tested in the bounded degree model with poly(d/ϵ)-query complexity under the promise that the input graph is minor-free. The same result was recently shown independently by Kumar-Seshadhri-Stolman [20]. While in [20] they use partition oracles in the proof, we use a relaxed notion of partition oracles (which we introduce in this paper) and consequently obtain a somewhat improved polynomial dependence in the query complexity and a simpler algorithm.

1.2 Our algorithms for minor-free unbounded degree graphs

1.2.1 Testing Hamiltonicity

We begin by proving that the distance from Hamiltonicity of any graph $G = (V, E)$ equals the size of the minimum path cover of G minus 1, where a path cover of a graph is a set of disjoint paths such that each $v \in V$ belongs to exactly one path (see Claim 11).

We then prove that if we remove $O(\epsilon|V|)$ edges from G as well as edges that are incident to $O(\epsilon|V|)$ vertices in G then the distance from Hamiltonicity may be increased by (at most) $O(\epsilon|V|)$ (see Claims 12 and 13).

Thus, in order to obtain an approximation, with an additive error of $O(\epsilon|V|)$, to the size of the minimum path cover of G (and hence to its distance from Hamiltonicity) it suffices to obtain such approximation to the size of the minimum path cover of $\hat{G}$ where $\hat{G}$ is defined as follows. We obtain $\hat{G}$ from G by first removing the edges incident to vertices of high degree, which we refer to as *heavy* vertices, then running the partition oracle on the resulting graph and then removing the cut-edges of the partition. An approximation to the size of the minimum path cover of $\hat{G}$ can be obtained by sampling vertices u.a.r. from V and computing the size of the minimum path cover of their connected component in $\hat{G}$.

Since we obtain an approximation algorithm for the distance from being Hamiltonian we also obtain a tolerant tester with two-sided error for this property.

---

[1] Without loss of generality we assume that the weights of the edges are distinct and hence that there is a unique MST.

1.2.2 Constructing spanning subgraphs with almost optimum weight

We present our algorithm as a global algorithm and prove its correctness. Thereafter, we describe the local implementation of this global algorithm.

Our global algorithm proceeds as follows. In the first step, the algorithm adds all the edges between *heavy* vertices to the edges of the constructed spanning subgraph, E', where a heavy vertex is defined to be a vertex of degree greater than some threshold.

It then runs the partition oracle on the graph induced on the vertices that are not heavy, i.e., *light* vertices and adds all the cut-edges of the partition to E'.

In the second step, each part of the partition is partitioned into subparts by running a controlled variant of Borůvka's algorithm for finding an MST on each part independently. The edges spanning the sub-parts are then added to E'. Then, for each each sub-part, the algorithm adds a single edge to a single heavy vertex which is adjacent to the sub-part (assuming there is one).

We prove that all the edges added in the second step belong to the minimum spanning forest (MSF) of a graph which is $O(\epsilon/W_G)$-close to G, where W_G denotes the maximum weight of an edge in G. Additionally we prove that if we remove $O(\epsilon|V|/W_G)$ edges from G, then the weight of the minimum spanning forest (MSF) may increase by (at most) $O(\epsilon|V|)$.

The second step partitions the vertices of the graph into *clusters* and *isolated parts*, where isolated parts are parts that are not adjacent to any heavy vertex, and the clusters are defined as follows. Each cluster contains a single heavy vertex, which we refer to a the *center* of the cluster and sub-parts that are adjacent in the constructed graph to the center (each sub-part is adjacent to at most a single center).

In the third step the algorithm adds edges to E' between pairs of cluster that are adjacent to each other. For every edge $\{u, v\}$ which is adjacent to two different clusters, A and B, the algorithm runs another algorithm that samples edges incident to A and B and returns the lightest one. The edge $\{u, v\}$ is added to E' if it is lighter than the returned edge.

We note that the algorithm that samples edges incident to a pair of specific clusters, A and B may not sample sufficient number of edges or may not return any edge (this is likely when the degree of both centers is large compared to the number of edges which are incident to both A and B). In the analysis which is adapted from [28], we show that nonetheless, w.h.p. the number of edges added in the third step is sufficiently small. The main idea is to consider the graph in which each cluster is contracted into a single vertex and then to analyse the sampling algorithm with respect to this graph which is also minor-free and hence has bounded arboricity [2]. The bounded arboricity of the contracted graph ensures that w.h.p. the sampling algorithm samples enough edges which are incident to A and B as long as the cut between these clusters is sufficiently large. On the other hand, if this is not the case then we show that we can afford to add to E' all the edges in the cut.

The local implementation of the above-mention global algorithm is quite straightforward and is presented in Section B.7.

1.3 Our algorithms for minor-free bounded degree graphs

1.3.1 Covering partition oracles

We introduce a relaxed version of partition oracles which we call *covering partition oracles* and design such an oracle for minor-free graphs with query complexity $\text{poly}(d/\epsilon)$. Given query access to a graph $G = (V, E)$ and parameter ϵ a partition oracle provides access to

---

[2] The arboricity of a graph is the minimum number of forests into which its edges can be partitioned.

a partition of V, $\mathcal{P}$, such that the size of each part of $\mathcal{P}$ is small (usually poly$(1/\epsilon)$), the number of cut-edges of $\mathcal{P}$ is at most $\epsilon|V|$ (w.h.p.) and $\mathcal{P}$ is determined only by G and the randomness of the oracle. On query $v \in V$ the oracle returns the part of v in $\mathcal{P}$.

A covering partition oracle has the same guarantees only that the requirement to return the part of v on query $v \in V$ is relaxed as follows. On query $v \in V$ the oracle is required to return a (small) subset S such that S contains the part of v in $\mathcal{P}$.

Our covering partition oracle builds on a central theorem from the recent work of [19]. The theorem states that for any minor-free bounded degree graph there exists a partition of the graph into small parts with small number of cut-edges such that for each part of the partition, P, there exists a vertex, $s \in V$, such that if we preform sufficiently many (polynomial in $1/\epsilon$) lazy random walks from s then w.h.p. we encounter all the vertices in P. Building on this theorem we prove that the simple algorithm that on query $v \in V$ performs a set of lazy random walks from v (of different lengths) and then performs a set of lazy random walks from each endpoint of these walks is a covering partition oracle.

The algorithms described in Subsections 1.3.2-1.3.4, use our covering partition oracle.

We note that since a partition oracle is a special case of a covering partition oracle (with stronger guarantees) all our algorithms for bounded degree graphs also work when one replaces calls to the covering partition oracle by calls to a partition oracle.

As mentioned above, the use of covering partition oracles has two benefits. The first benefit is that the implementation of the covering partition oracles is much simpler and the second benefit is that the query complexity per oracle query is better (though both our covering partition oracle and the partition oracle of [20] have query complexity which is poly(d/ϵ)), which consequently affects the query complexity of the algorithms.

Another conceptual benefit in introducing covering partition oracles is that for some families of graphs the gap in the query complexity can be more dramatic. To give a concrete example consider $(\epsilon, \rho(\epsilon))$-hyperfinite graphs [3]. It is straightforward to obtain a covering partition oracle with query complexity $O(d^{\rho(\epsilon)})$ for this family of graphs while the best known partition oracle for this family has query complexity which is $O(2^{d^{\rho(c\epsilon^3)}})$ [14], where c is some constant. We note that all our algorithms for bounded degree graphs work for any family of graphs for which there is a covering partition oracle (including $(\epsilon, \rho(\epsilon))$-hyperfinite graphs).

1.3.2 Testing Hamiltonicity

In addition to relating the distance from Hamiltonicity of any graph $G = (V, E)$ to the size of its minimum path cover, as mentioned above, we also prove that given a subset $S \subset V$, if the size of the minimum path cover of $G[S]$ is greater than the number of edges in the cut of S and $V \setminus S$ then G is not Hamiltonian. Using this claim it becomes straightforward to design a one-sided error tester for Hamiltonicity that uses $O(1/\epsilon)$ queries to a partition oracle. We prove that it suffices to use the same number of queries to the covering partition oracle. We note that in this case there is a trade-off between the query complexity and time complexity. In particular while using the covering partition oracle rather than the partition oracle results in a better polynomial dependence of the query complexity it also results in a worse polynomial dependence in the exponent of the running time (in both cases the running time is exponential in ϵ^{-1} since we find the size of the minimum path cover by brute force [4]).

---

[3] Let ρ be a function from $\mathbb{R}_+$ to $\mathbb{R}_+$. A graph $G = (V, E)$ is $(\epsilon, \rho(\epsilon))$-hyperfinite if for every $\epsilon > 0$ it is possible to remove $\epsilon|V|$ edges of the graph such that the remaining graph has connected components of size at most $\rho(\epsilon)$.

[4] Finding the minimum path cover is APX-hard since (as noted by Chandra Chekuri at stackexchange.com [16]) we can reduce the TSP-path problem in metrics with distances 1 and 2 to it. The latter problem is APX-hard [9]).

1.3.3 Constructing spanning subgraphs with almost optimum weight

As mentioned-above, given a weighted graph $G = (V, E, w)$ if we remove $O(\epsilon|V|/W_G)$ edges from G then the weight of the MSF of the resulting graph may increase by at most $O(\epsilon|V|)$ compared to the weight of G. Thus given access to a partition of V such that the subgraph induced on each part is connected and the number of cut-edges is $O(\epsilon|V|/W_G)$ we can proceed as follows. For each part of the partition we add to E' the edges of the MST of the subgraph induced on this part. In addition, we add to E' the cut-edges of the partition. Consequently, the total weight of the edges in E' is greater than the weight of the MST of G by at most $O(\epsilon|V|/W_G)$. Hence, if on query $\{u, v\}$ we query the partition oracle on u and v then it is possible to determine whether $\{u, v\} \in E'$ where E' is constructed as described above with respect to the partition of the oracle. We prove that the same approach works when we preform the same queries to the covering partition oracle.

1.3.4 Testing monotone and additive properties

One of the main applications of the partition oracle is a general reduction for testing monotone and additive properties of bounded degree minor-free graphs. The idea of the reduction (from testing to the partition oracle) is to sample $O(d/\epsilon)$ vertices and for each vertex v in the sample to test whether the subgraph induced on the part of v has the properties. The tester accepts iff all sampled parts pass the test. We prove that the same reduction works when we replace the queries to the partition oracle by queries to the covering partition oracle.

1.4 Organization

Due to space limitations, an extended section on related work as well as omitted proofs and details of Section 3 appear in the appendix.

2 Preliminaries

In this section we introduce several definitions and some known results that will be used in the following sections. Unless stated explicitly otherwise, we consider simple graphs, that is, with no self-loops and no parallel edges.

Let $G = (V, E)$ be a graph over n vertices. Each vertex $v \in V$ has an id, $id(v)$, where there is a full order over the ids.

The total order over the vertices induces a total order (ranking) ρ over the edges of the graph in the following straightforward manner: $\rho((u, v)) < \rho((u', v))$ if and only if $\min\{u, v\} < \min\{u', v'\}$ or $\min\{u, v\} = \min\{u', v'\}$ and $\max\{u, v\} < \max\{u', v'\}$ (recall that $V = [n]$). Thus, given a weighted graph, we may assume without loss of the generality that the weights of the edges are unique by breaking ties according to the order over the edges.

For a subset of vertices X, we let $G[X]$ denote the subgraph of G induced by X.

When we consider bounded degree graphs, we consider the bounded-degree graph model [11]. The graphs we consider have a known degree bound d, and we assume we have query access to their incidence-lists representation. Namely, for any vertex v and index $1 \leq i \leq d$ it is possible to obtain the ith neighbor of v (where if v has less than i neighbors, then a special symbol is returned). If the graph is edge-weighted, then the weight of the edge is returned as well. When we consider graphs with unbounded degree we consider the general graph model [30] equipped with an additional type of query: random neighbor query. Namely, when we query any given vertex v a random neighbor of v is returned [5].

---

[5] We note that we do not use the random neighbor query in our tester for Hamiltonicity.

For a graph $G = (V, E)$ and two sets of vertices $V_1, V_2 \subseteq V$, we let $E^G(V_1, V_2)$ denote the set of edges in G with one endpoint in V_1 and one endpoint in V_2. That is $E(V_1, V_2) \overset{\text{def}}{=} \{(v_1, v_2) \in E : v_1 \in V_1, v_2 \in V_2\}$. If G is clear from the context we may omit it from the notation.

A tester with two-sided error for a property $\mathcal{P}$ receives a parameter ϵ and query access to a graph $G = (V, E)$ and has the following guarantees. If $G \in \mathcal{P}$ then the tester accepts with probability at least $2/3$. If G is ϵ-far [6] from $\mathcal{P}$ then the tester rejects with probability at least $2/3$. A tester with one-sided error accepts $G \in \mathcal{P}$ with probability 1.

2.1 Partition oracles and covering partition oracles

▶ **Definition 1.** *For $\epsilon \in (0, 1]$, $k \geq 1$ and a graph $G = (V, E)$, we say that a partition $\mathcal{P} = (V_1, \ldots, V_t)$ of V is an (ϵ, k)-partition (w.r.t. G), if the following conditions hold:*
1. *For every $1 \leq i \leq t$ it holds that $|V_i| \leq k$;*
2. *For every $1 \leq i \leq t$ the subgraph induced by V_i in G is connected;*
3. *The total number of edges whose endpoints are in different parts of the partition is at most $\epsilon|V|$ (that is, $|\{(v_i, v_j) \in E : v_i \in V_j, v_j \in V_j, i \neq j\}| \leq \epsilon|V|$).*

Let $G = (V, E)$ be a graph and let $\mathcal{P}$ be a partition of V. We denote by $g_{\mathcal{P}}$ the function from $v \in V$ to 2^V (the set of all subsets of V), that on input $v \in V$, returns the subset $V_\ell \in \mathcal{P}$ such that $v \in V_\ell$. We denote the set of cut-edges of $\mathcal{P}$ by $E_{\mathcal{P}}^G = \{(u, v) \in E : g_{\mathcal{P}}(v) \neq g_{\mathcal{P}}(u)\}$ (we may omit G from the notation when it is clear from the context).

▶ **Definition 2** ([14])**.** *An oracle $\mathcal{O}$ is a* partition oracle *if, given query access to the incidence-lists representation of a graph $G = (V, E)$, the oracle $\mathcal{O}$ provides query access to a partition $\mathcal{P} = (V_1, \ldots, V_t)$ of V, where $\mathcal{P}$ is determined by G and the internal randomness of the oracle. Namely, on input $v \in V$, the oracle returns $g_{\mathcal{P}}(v)$ and for any sequence of queries, $\mathcal{O}$ answers consistently with the same $\mathcal{P}$. An oracle $\mathcal{O}$ is an (ϵ, k)-partition oracle with respect to a class of graphs $\mathcal{C}$ if the partition $\mathcal{P}$ it answers according to has the following properties.*
1. *For every $V_\ell \in \mathcal{P}$, $|V_\ell| \leq k$ and the subgraph induced by V_ℓ in G is connected.*
2. *If G belongs to $\mathcal{C}$, then $|E_{\mathcal{P}}| \leq \epsilon|V|$ with high constant probability, where the probability is taken over the internal coin flips of $\mathcal{O}$.*

We consider the following relaxation of Definition 2.

▶ **Definition 3.** *An oracle $\mathcal{O}$ is a* covering partition oracle *if, given query access to the incidence-lists representation of a graph $G = (V, E)$, the oracle $\mathcal{O}$, on input $v \in V$, returns a subset $S \subseteq V$ such that $g_{\mathcal{P}}(v) \subseteq S$ where $\mathcal{P}$ is a partition of V determined by G and the internal randomness of the oracle. For any sequence of queries, $\mathcal{O}$ answers consistently according to the same $\mathcal{P}$. An oracle $\mathcal{O}$ is an (ϵ, k)-covering partition oracle with respect to a class of graphs $\mathcal{C}$ if the following conditions hold.*
1. *On every query, the subgraph induced by the subset returned by $\mathcal{O}$, S, is connected and $|S| \leq k$.*
2. *If G belongs to $\mathcal{C}$, then w.h.p., $|E_{\mathcal{P}}| \leq \epsilon|V|$, where the probability is taken over the internal coin flips of $\mathcal{O}$.*

---

[6] In the bounded degree model, a graph $G = (V, E)$ is said to be ϵ-far from a property $\mathcal{P}$ if for every graph $G' = (V, E') \in \mathcal{P}$ of maximum degree d it holds that the symmetric difference between E and E' has cardinality which is greater than $\epsilon \cdot d|V|/2$. In the general graph model, a graph $G = (V, E)$ is said to be ϵ-far from a property $\mathcal{P}$ if for every graph $G' = (V, E') \in \mathcal{P}$ it holds that the symmetric difference between E and E' has cardinality which is greater than $\epsilon \cdot \max\{|E|, |E'|\}$.

2.2   Graph minors

Recall that a graph R is called a *minor* of a graph G if R is isomorphic to a graph that can be obtained by zero or more edge contractions on a subgraph of G. A graph G is R-*minor-free* if R is not a minor of G. We next quote two results that will play a central role in this work.

▶ **Fact 4.** *Let R be a fixed graph with r edges. For every R-minor-free graph $G = (V, E)$ it holds that:*
1. *$|E| \leq r \cdot |V|$;*
2. *E can be partitioned into at most r forests.*

Unless stated otherwise, in all our algorithms, we assume that the input graph is R-minor-free graph where R is a fixed graph with r edges (we could receive r as a parameter but we make this assumption for the sake of brevity).

2.3   Hamiltonian path and minimum path cover

▶ **Definition 5** (Hamiltonian path). *A Hamiltonian path in $G = (V, E)$ is a path between two vertices of G that visits each vertex of G exactly once.*

▶ **Definition 6** (minimum path cover). *Given an undirected graph $G = (V, E)$, a path cover is a set of disjoint paths such that every vertex $v \in V$ belongs to exactly one path. The minimum path cover of G is a path cover of G having the least number of paths.*

2.4   Local algorithms for constructing sparse spanning subgraphs

▶ **Definition 7** ([28]). *An algorithm $\mathcal{A}$ is a local sparse spanning graph (LSSG) algorithm if, given $n \geq 1$, $\epsilon > 0$, and query access to the incidence-lists representation of a connected graph $G = (V, E)$ over n vertices, it provides oracle access to a subgraph $G' = (V, E')$ of G such that:*
1. *G' is connected.*
2. *$|E'| \leq (1 + \epsilon) \cdot n$ with high constant probability [7], where E' is determined by G and the internal randomness of $\mathcal{A}$.*

More specifically, on query $u, v \in E$, $\mathcal{A}$ returns whether $(u, v) \in E'$, and for any sequence of edges, $\mathcal{A}$ answers consistently with the same G'.

An algorithm $\mathcal{A}$ is an LSSG algorithm for a family of graphs $\mathcal{C}$ if the above conditions hold, provided that the input graph G belongs to $\mathcal{C}$.

▶ **Definition 8** ([26]). *A local algorithm for $(1 + \epsilon)$-approximating the minimum weight spanning graph of a graph $G = (V, E, w)$ with positive weights and $\min_{e \in E} w(e) \geq 1$, is a local algorithm for $(1 + \epsilon)$-sparse spanning graph of $G = (V, E, w)$ for which the following holds: $\sum_{e \in E'} w(e) \leq (1 + \epsilon)\alpha$, where α is the weight of a minimum weight spanning tree of G.*

For a graph $G = (V, E, w)$ we define $W_G = \max_{e \in E} w(e)$ (when it is clear from the context, we sometimes omit the subscript G). We denote by $\mathrm{MSF}(G)$ the set of edges the minimum-spanning-forest of G (as mentioned above we assume without loss of generality that all weights are distinct and thus the minimum-spanning-forest is unique). For a connect weighted graph we denote by $\mathrm{MST}(G)$ the set of edges the minimum-spanning-forest of G. For a subset of edges $S \subseteq E$, we define $w(S) \stackrel{\text{def}}{=} \sum_{e \in S} w(e)$.

---

[7] In some papers the required success probability is high, i.e. at least $1 - 1/\Omega(n)$.

Our algorithms build on the following rules.
1. The *cut rule* states that for any cut of the graph (a cut is a partition of the vertices into two sets), the lightest edge that crosses the cut must be in the MST.
2. The *cycle rule* states that if we have a cycle, the heaviest edge on that cycle cannot be in the MST.

3 Algorithms for minor-free graphs with unbounded degrees

3.1 Testing Hamiltonicity

In this section we prove the following Theorem.

▶ **Theorem 9.** *Given query access to an input graph $G = (V, E)$ where G is a minor-free unbounded degree graph and parameters ϵ and $|V|$, there exists an algorithm that accepts G with probability at least $2/3$ if G is $\epsilon/2$-close to being Hamiltonian and rejects G with probability at least $2/3$ if G is ϵ-far from being Hamiltonian. The query complexity of the algorithm is $\mathrm{poly}(1/\epsilon)$ and the running time is exponential in $\mathrm{poly}(1/\epsilon)$.*

Theorem 9 is a direct consequence of following claim (which is proved in the sequel).

▷ **Claim 10.** Given an input graph $G = (V, E)$ which is a minor-free graph, and parameter ϵ, Algorithm 1 outputs a value x such that with high constant probability $\delta_{\mathrm{HAM}}(G) - \epsilon|V| \leq x \leq \delta_{\mathrm{HAM}}(G) + \epsilon|V|$.

We next state a couple of claims which we use in the proof of Claim 10.

▷ **Claim 11.** Let $G = (V, E)$ be a graph and let k be the size of a minimum path cover of G. Then the distance of G for being Hamiltonian, $\delta_{\mathrm{HAM}}(G)$, is $k - 1$.

▷ **Claim 12.** Let $G = (V, E)$ be a graph and let $F \subseteq E$ be a subset of edges. Then

$$\delta_{\mathrm{HAM}}(G) \leq \delta_{\mathrm{HAM}}(G') \leq \delta_{\mathrm{HAM}}(G) + |F|,$$

where $G' = (V, E')$ and $E' = E \setminus F$.

▷ **Claim 13.** Let $G = (V, E)$ be a graph and let $S \subseteq V$ be a subset of vertices. Then

$$\delta_{\mathrm{HAM}}(G) \leq \delta_{\mathrm{HAM}}(G') \leq \delta_{\mathrm{HAM}}(G) + 2|S|,$$

where $G' = (V, E')$ and E' is the set of edges in E that are not incident to vertices in S.

Proof of Claim 10. Let $\mathcal{P}$ denote the partition for which the partition oracle executed in Step 3(a)i answers according to. With high constant probability $|E_{\mathcal{P}}| \leq \frac{\epsilon|V|}{4}$. Let E_1 denote this event.

Let $G' = (V, E')$ be the graph such that E' is the set of edges that are not incident to vertices in H and are not in $E_{\mathcal{P}}$. By Claims 12 and 13,

$$\delta_{\mathrm{HAM}}(G) \leq \delta_{\mathrm{HAM}}(G') \leq \delta_{\mathrm{HAM}}(G) + |E_{\mathcal{P}}| + 2|H|.$$

By Markov's inequality and Fact 4, $|H| \leq \frac{\epsilon|V|}{4}$. We prove that, conditioned that E_1 occurs, Algorithm 1 outputs with high constant probability a $(1 + \epsilon)$-approximation to $\delta_{\mathrm{HAM}}(G')$.

For each $v \in V$ define the random variable x_v as defined in Step 3 of Algorithm 1. Let $T \in \mathcal{P}$ be a part in $\mathcal{P}$, then $\sum_{v \in T} x_v = k$ where k is the minimum path cover of $G[T]$. Thus $\sum_{v \in V} x_v$ is the minimum path cover of G'. Since for every $v \in V$, $x_v \in (0, 1]$, it follows by

■ **Algorithm 1** Approximating the distance to Hamiltonicity in minor-free, unbounded degree, graphs.

Input: Oracle access to a minor-free, unbounded-degree, graph $G = (V, E)$
Output: $(1 + \epsilon)$-approximation to $\delta_{\mathrm{HAM}}(G)$

1. Define $\Delta \overset{\text{def}}{=} 8c(h)/\epsilon$, H to be the set of vertices of degree greater than Δ, and $L \overset{\text{def}}{=} V \setminus H$.
2. Sample a set S of $y = \Theta(1/\epsilon^2)$ vertices u.a.r.
3. For each vertex $v \in S$:
   a. If $v \in L$ then:
      i. Query the partition oracle on v with parameter $\epsilon/4$ w.r.t. the graph $G[L]$. Let S_v denote the returned set.
      ii. Set $x_v = k/|S_v|$ where k is the size of the minimum path cover of $G[S_v]$.
   b. Otherwise, set $x_v = 1$.
4. Output $\frac{\sum_{v \in S} x_v}{|S|} \cdot |V|$.

---

the additive Chernoff's bound that with high constant probability $\left| \frac{\sum_{v \in S} x_v}{|S|} - \frac{\sum_{v \in V} x_v|}{|V|} \right| \leq \frac{\epsilon}{4}$. Thus, with high constant probability,

$$\delta_{\mathrm{HAM}}(G) - \frac{\epsilon|V|}{4} \leq \frac{\sum_{v \in S} x_v}{|S|} \cdot |V| \leq \delta_{\mathrm{HAM}}(G) + \epsilon|V|,$$

as desired.   ◁

3.2   A Local algorithm for constructing a spanning subgraph with almost optimum weight

In this section we prove the following theorem.

▶ **Theorem 14.** *There exists a local algorithm for $(1+\epsilon)$-approximating the minimum weight spanning graph for the family of unbounded degree minor-free graphs, with positive weights and minimum weight which is at least 1. The query complexity and time complexity of the algorithm is $\mathrm{poly}(W/\epsilon)$ where W is an upper bound on the maximum weight. The algorithm receives ϵ and W as parameters.*

▷ **Claim 15.**   Let $G = (V, E, w)$ be a weighted graph and let $G' = (V, E', w)$ be a graph such that $E' = E \setminus S$ where $S \subseteq E$. Then $w(\mathrm{MSF}(G')) \leq w(\mathrm{MSF}(G)) + |S|W_G$. Moreover, $|\mathrm{MSF}(G') \setminus \mathrm{MSF}(G)| \leq |S|$.

We next describe our algorithm from a global point of view.

3.2.1   The global algorithm

Our global algorithm, which is listed in Algorithm 2, proceeds as follows. In Step 2, the algorithm adds all the edges between *heavy* vertices to E' where a heavy vertex is defined to be a vertex of degree greater than $\Delta \overset{\text{def}}{=} 6r^2W/\epsilon$.

It then runs the partition oracle on the graph induced on the *light* vertices, namely vertices that are not heavy, and adds all the cut-edges of the partition, $\mathcal{P}$, to E' (Step 4).

■ **Algorithm 2** Global algorithm for approximated-MST in unbounded-degree minor-free graphs.

Input: parameters ϵ and W and access to a minor-free graph $G = (V, E)$.

Output: $G = (V, E')$ which is an approximated-MST of G.

1. Let H denote the set of all vertices of degree greater than Δ and let $L = V \setminus H$.
2. Add all the edges of $G[H]$ to E'.
3. Run the partition oracle with parameter $\epsilon/(6W)$ on all vertices in $G[L]$. Let $\mathcal{P}$ denote the resulting partition.
4. Add all the edges in $E_{\mathcal{P}}$ to E'.
5. For each $S \in \mathcal{P}$:
   a. Run Algorithm 4 and let $F = (S, A)$ denote the returned graph.
   b. Add the edges in A to E'.
   c. For each connected component of F, C:
      i. Add to E' the lightest edges which is adjacent to C and H (if such edge exists).
6. For each $v \in H$, define the *cluster* of v, denoted by $\mathcal{C}(v)$, to be subset of vertices that contains v and all the vertices in sub-parts B such that there exists an edge in E' that is incident to v and a vertex in B. v is referred to as the *center* of the cluster.
7. For each edges $\{u, v\} \in E$ such that u and v belong to different clusters:
   a. Run Algorithm 3 and add $\{u, v\}$ to E' if it is lighter than the edge returned by the algorithm or if it returned null.
8. Return $G' = (V, E')$.

Step 5 consists of two parts. In Sub-step 5a the algorithm runs Algorithm 4 on each part in $\mathcal{P}$. Algorithm 4 partitions the parts into connected sub-parts by preforming a controlled variant of Borůvka's algorithm. The edges added in Sub-step 5a are the edges that span the sub-parts. In Sub-step 5c, for each sub-part, the algorithm adds a single edge to a single vertex in H which is adjacent to the sub-part (assuming there is one).

This partitions the vertices of the graph into *clusters* and *isolated parts*, namely, parts in $\mathcal{P}$ that are not adjacent to any vertex in H. Each cluster contains a single heavy vertex, which we refer to as the *center* of the cluster and sub-parts that are connected by a single edge to the center.

In Step 7 the algorithm adds edges to E' between pairs of clusters that are adjacent to each other in $G' = (V, E \setminus E_{\mathcal{P}})$. For every edge $\{u, v\}$ which is adjacent to two different clusters, A and B, the algorithm runs Algorithm 3 which samples edges incident to A and B and returns the lightest one. The edge $\{u, v\}$ is added to E' if it is lighter than the edge returned by Algorithm 3 (or if the algorithm returned null).

We note that we do not have direct access to uniform samples of edges incident to a pair of specific clusters, A and B. In fact, it could be the case that Algorithm 3 does not return any edge (this is likely when the degree of both centers is large compared to the number of edges which are incident to both A and B). By adapting the analysis in [28], we show that nonetheless, w.h.p. the number of edges added in Step 7 is sufficiently small.

This concludes the description of the global algorithm. The proof of correctness as well as the local implementation (Algorithm 8) are differed to the appendix.

■ **Algorithm 3** Return sampled lightest edge.

Input: $a \in H$ and $b \in H$.
Output: The sampled lightest edge between $\mathcal{C}(a)$ and $\mathcal{C}(b)$
1. Initially $A = \emptyset$ and let x be an upper bound on the size of the parts returned by the partition oracle when executed with parameter $\epsilon/(6W)$.
2. Sample a set, S, of $\tilde{\Theta}(W^2 r^3 x \Delta / \epsilon^2)$ edges incident to a.
3. For each $\{a, u\} \in S$ do:
    a. If $u \in H$ then add it to A if and only $u = b$.
    b. Otherwise, find the part of u and the sub-parts of this part.
    c. For each sub-part find its center.
    d. Add to A all the edges between the sub-part of u and sub-parts that belong to $\mathcal{C}(b)$.
4. Repeat Steps 2-3 where a and b switch roles.
5. Return the lightest edge in A (if $A = \emptyset$ then return null).

■ **Algorithm 4** Partition into sub-parts.

Input: Access to the input graph $G = (V, E)$ and a subset $S \subseteq L$ such that $G[S]$ is connected.
Output: A graph $F = (S, A)$ such that each connected component of F is a sub-part of S
1. Initially every vertex in S is a in its own sub-part (a singleton) and $A = \emptyset$.
2. We say a sub-part B is *active* if the
    lightest in $E^G(B, S \setminus B)$ is lighter than the lightest edge in $E^G(B, H)$ or if $E^G(B, H) = \emptyset$.
3. While there are still active sub-parts do:
    a. Each active sub-part B selects the lightest edge in $E^G(B, S \setminus B)$, denoted by e_B.
    b. For each active sub-part, B, add e_B to A
    c. Update the new sub-parts to be the connected components of the graph $F = (S, A)$ (each connected component is a sub-part).
4. Return F.

---

4   Algorithms for minor-free graphs with bounded degrees

4.1   Covering partition oracle

In this section we prove the following theorem.

▶ **Theorem 16.** *Algorithm 5 is an $(\epsilon, \text{poly}(\epsilon^{-1}))$-covering-partition oracle for minor-free bounded degree graphs with query complexity $\text{poly}(\epsilon^{-1})$. Specifically, the size of the sets returned by the oracle is $O(\epsilon^{-640} \log^2(1/\epsilon))$.*

We begin with a couple of definitions and lemmas from [20] that we build on.

▶ **Definition 17** ([20]). *Given $\mathbf{x} \in (\mathbb{R}^+)^{|V|}$ and parameter $\xi \in [0, 1)$, the ξ-clipped vector $\text{cl}(\mathbf{x}, \xi)$ is the lexicographically least vector $\mathbf{y}$ optimizing the program:* $\min \|\mathbf{y}\|_2$, *subject to $\|\mathbf{x} - \mathbf{y}\|_1 \le \xi$ and $\forall v \in V, \mathbf{y}(v) \le \mathbf{x}(v)$.*

▶ **Lemma 18** ([20]). *There is an absolute constant α such that the following holds. Let H be a graph on r vertices. Suppose G is a H-minor-free graph. Then for any $h \ge \alpha r^3$, there exists at least $(1 - 1/h)n$ vertices such that $\|\text{cl}(\mathbf{p}_{v,\ell}, 3/8)\|_2^2 \ge h^{-7}$.*

Given two parameters $\epsilon \in [0, 1/2]$, and a graph R on $r \geq 3$ vertices. The length of the random walk is $\ell = \alpha r^3 + \lceil \epsilon^{-20} \rceil$ where α is some absolute constant.

▶ **Theorem 19** ([20]). *Suppose there are at least $(1-1/\ell^{1/5})n$ vertices s s.t. $\|\mathrm{cl}(\mathbf{p}_{s,\ell}, 1/4)\|_2^2 > \ell^{-c}$. Then, there is a partition $\{P_1, P_2, \ldots, P_b\}$ of the vertices s.t.:*
1. *For each P_i, there exists $s \in V$ such that: $\forall v \in P_i$, $\sum_{t < 10\ell^{c+1}} p_{s,t}(v) \geq 1/8\ell^{c+1}$.*
2. *The total number of edges crossing the partition is at most $8dn\sqrt{c\ell^{-1/5}\log \ell}$.*

▶ **Corollary 20.** *Let $G = (V, E)$ be a graph which is R-minor-free where R is a graph on r vertices. There exists a partition $\{P_1, P_2, \ldots, P_b\}$ of the V such that:*
1. *For each P_i, there exists $s \in V$ such that: $\forall v \in P_i$, $\sum_{t < 10\ell^8} p_{s,t}(v) \geq 1/8\ell^8$.*
2. *The total number of edges crossing the partition is at most ϵdn.*

Proof. We first note that $8dn\sqrt{c\ell^{-1/5}\log \ell} \leq \epsilon dn$ for sufficiently large constant α.

By Lemma 18 there exist at least $(1 - 1/\ell)n$ vertices such that $\|\mathrm{cl}(\mathbf{p}_{v,\ell}, 3/8)\|_2^2 \geq \ell^{-7}$. Thus the corollary follows from the facts that $(1-1/\ell)n \geq (1-1/\ell^{1/5})n$ and $\|\mathrm{cl}(\mathbf{p}_{s,\ell}, 1/4)\|_2^2 \geq \|\mathrm{cl}(\mathbf{p}_{s,\ell}, 3/8)\|_2^2$. ◄

■ **Algorithm 5** Covering-partition oracle.

Input: $v \in V$.
Output: A subset S which covers the part of v.
1. For every $t < 10\ell^8$ perform $x \stackrel{\text{def}}{=} \Theta(\ell^8 \log \ell)$ random walks of length t from v.
2. Let R denote the endpoints of the random walks preformed in the previous step.
3. For every vertex $r \in R$, for every $t < 10\ell^8$, perform x random walks of length t from r.
4. Let S denote the set of all vertices encountered by the random walks performed in Step 1 and Step 3.
5. Return S.

Proof of Theorem 16. Let $G = (V, E)$ be a graph which is R-minor-free where R is a graph over r vertices. Consider the partition of V, $\mathcal{P} = \{P_1, P_2, \ldots, P_b\}$ as defined in Corollary 20 when we take the proximity parameter to be $\epsilon/2$. We shall define another partition $\mathcal{P}'$ which is a refinement of $\mathcal{P}$ such that Algorithm 5 returns for every $v \in V$, a subset S such that $P' \subseteq S$ where P' denotes the part of v in $\mathcal{P}'$. Thereafter, we shall prove that, w.h.p., the number of cut-edges of $\mathcal{P}'$ is not much greater than the number of cut-edges of $\mathcal{P}$.

For every $v \in V$, we say that v *fails* if Algorithm 5, when queried on v, does not return S such that $P^v \subseteq S$, where P^v denotes the part of v in $\mathcal{P}$. We define $\mathcal{P}'$ as follows. For every $v \in V$, if there exists $u \in P^v$ such that u fails, then the part of v in $\mathcal{P}'$ is defined to be the singleton $\{v\}$ (namely, the entire part P^v is partitioned into singletons in $\mathcal{P}'$). Otherwise, it is defined to be P^v.

We next show that w.h.p. the cut-edges of $\mathcal{P}'$ is at most $\epsilon d|V|$. For every v, the probability that v fails is at most $p \stackrel{\text{def}}{=} \ell^{-c_1}$ for an appropriate setting of x (with accordance to the Theta-notation), where c_1 is a constant that will be determined later. Let $y = \ell^{c_2}$ be an upper bound on the number of vertices in the parts of $\mathcal{P}$, where c_2 is a constant. [8]

---

[8] Clearly, since for each P_i, there exists $s \in V$ such that: $\forall v \in P_i$, $\sum_{t < 10\ell^8} p_{s,t}(v) \geq 1/8\ell^8$, it follows that $y \leq 10\ell^8 \cdot 8\ell^8$.

For every $v \in V$, define the random variable X_v as follows. If v fails then $X_v = |P_v|/y$ and otherwise $X_v = 0$. Clearly $y \cdot \sum_{v \in V} X_v \geq |\mathcal{P}'| - |\mathcal{P}|$. Note that $\{X_v\}_{v \in V}$ are independent random variables ranging in $[0, 1]$.

For every $v \in V$, we define the random variable Y_v as follows. With probability p, $Y_v = 1$ and otherwise $X_v = 0$. Clearly, Y_v dominates X_v. Since $\{Y_v\}_{v \in V}$ are identical independent random variables, it follows by the multiplicative Chernoff's bound that w.h.p. $y \cdot \sum_{v \in V} Y_v \leq y \cdot |V| \cdot 2p$.

Since for sufficiently large c_1, $p \leq \frac{\epsilon}{4y}$, it follows that w.h.p. $|\mathcal{P}'| - |\mathcal{P}| \leq \epsilon|V|/2$. Thus, the number of cut-edges in $\mathcal{P}'$ is greater than the number of cut-edges in $\mathcal{P}$ by at most $\epsilon d|V|/2$ (recall that the algorithm refines $\mathcal{P}$ by decomposing entire parts into singletons). ◀

4.2 Testing Hamiltonicity

In this section we prove the following theorem.

▶ **Theorem 21.** *Given query access to an input graph $G = (V, E)$ where G is a minor-free bounded degree graph and a parameters ϵ and $|V|$, Algorithm 6 accepts G with probability 1 if G is Hamiltonian and rejects G with probability at least $2/3$ if G is ϵ-far from being Hamiltonian. The query complexity of the algorithm is $\mathrm{poly}(d/\epsilon)$ and the running time is exponential in $\mathrm{poly}(d/\epsilon)$.*

The correctness of Algorithm 6 builds on the following claim which, given $S \subset V$, bounds the size of a minimum path cover in $G[S]$ by the size of the cut of S.

▷ Claim 22.   Let $G = (V, E)$ be a graph and let $S \subset V$ be a subset of vertices of G. Let k be the size of a minimum path cover in $G[S]$. If $k - 1 > |E(S, V \setminus S)|/2$, then there is no Hamiltonian path in G. Moreover, any Hamiltonian path in G must include at least $2(k-1)$ edges from $E(S, V \setminus S)$.

Proof.   Let $G = (V, E)$ be a graph. Assume toward contradiction that there exists Hamiltonian path in G, $\mathcal{H} = (v_1, v_2, \ldots, v_{|V|})$ and a subset $S \subset V$ such that $k - 1 > |E(S, V \setminus S)|/2$, where k is the size of a minimum path cover in $G[S]$. Let $\mathcal{P}'_S$ denote the set of all maximal sub-paths of $\mathcal{H}$ in $G[S]$. In order to connect the sub-paths in $\mathcal{P}'_S$ it must hold that $\mathcal{H}$ leaves and returns to $G[S]$ at least $2(\mathcal{P}'_S - 1)$ times, each time using a different edge. Thus, $|E(S, V \setminus S)| \geq 2(|\mathcal{P}'_S| - 1)$. Since $\mathcal{P}'_S$ is a path cover of $G[S]$ it follows that $|\mathcal{P}'_S| \geq k$, thus, $|E(S, V \setminus S)| \geq 2(k - 1)$, in contradiction to our assumption. Hence the claim follows.   ◁

We next list our algorithm and prove its correctness.

Proof of Theorem 21. By Claim 22, Algorithm 6 never rejects graphs which are Hamiltonian.

Let G be a minor-free bounded degree graph which is ϵ-far from being Hamiltonian. We shall prove that Algorithm 6 rejects G with probability at least $2/3$. Let $\mathcal{P}$ denote the partition that the oracle, executed in Step 2a, answers according to. With high constant probability, it holds that $|E_{\mathcal{P}}| \leq \frac{\epsilon|V|}{6}$. Let E_1 denote the event that this conditions holds.

Let $\mathcal{F}$ denote the set of parts, S, in $\mathcal{P}$, such that $E(S, V \setminus S) = \emptyset$ or for which the size of the minimum path cover of $G[S]$ is greater $|E(S, V \setminus S)|/2 + 1$

Assume towards contradiction that E_1 occurs and that $|\mathcal{F}| < \epsilon|V|/(2x)$ where x is an upper bound on the number of vertices in each part of $\mathcal{P}$. We next show that $\delta_{\mathrm{HAM}}(G) \leq \epsilon|V|$ in contradiction to our assumption.

For each part $S \in \mathcal{F}$ we construct a path over S that visits each vertex in S exactly once by adding at most $|S| - 1 \leq x - 1$ edges to G.

■ **Algorithm 6** Testing Hamiltonicity in minor-free, bounded degree, graphs.

Input: Oracle access to a minor-free, bounded-degree, graph $G = (V, E)$

Output: Tests if G is Hamiltonian with one-sided error.

1. Sample a subset, $S \subseteq V$, of $y \stackrel{\text{def}}{=} \Theta(x/\epsilon)$ vertices, uniformly at random, where x is an upper bound on the size of the sets returned by the covering partition oracle when execute with parameter $\epsilon/6$.

2. For each $v \in S$ do:
   a. Query the covering partition oracle on v with parameter $\epsilon/6$, and let S_v denote the returned set.
   b. If $E(S_v, V \setminus S_v) = \emptyset$ then return REJECT.
   c. For each subset $T \subseteq S_v$ such that $G[T]$ is connected, find the size of the minimum path cover of T and return REJECT if it is greater than $|E(T, V \setminus T)|/2 + 1$.

For each part $S \in \mathcal{P} \setminus \mathcal{F}$, let $\mathcal{C}_S$ denote a minimum path cover of $G[S]$. We construct a path over S that visits each vertex in S exactly once by adding at most $|\mathcal{C}_S| - 1 \leq |E(S, V \setminus S)|/2$ edges to G.

We then connect all the paths induced on the different parts of $\mathcal{P}$ by adding at most $|\mathcal{P}| = |\mathcal{F}| + |\mathcal{P} \setminus \mathcal{F}|$ edges.

Overall the number of edges added is at most:

$$|\mathcal{F}| \cdot (x - 1) + |E_\mathcal{P}| + |\mathcal{F}| + |\mathcal{P} \setminus \mathcal{F}| \leq |\mathcal{F}| \cdot x + 3|E_\mathcal{P}| < \epsilon|V|,$$

where the first inequality follows from the fact that $|\mathcal{P} \setminus \mathcal{F}| \leq 2|E_\mathcal{P}|$ as for each $S \in \mathcal{P} \setminus \mathcal{F}$ it holds that $E(S, V \setminus S) \cap E_\mathcal{P} \neq \emptyset$ and each edge in $E_\mathcal{P}$ is adjacent to at most 2 parts in $\mathcal{P}$.

Thus, if $\delta_{\text{HAM(G)}} > \epsilon|V|$ and E_1 occurs then $|\mathcal{F}| \geq \epsilon|V|/(2x)$. Thus, the number of vertices in parts that belong to $\mathcal{F}$ is at least $\epsilon|V|/(2x)$, which implies that G is rejected w.h.p. either in Step 2b or in Step 2c of Algorithm 6, as desired.   ◀

4.3   Local algorithms for constructing a spanning subgraph with almost optimum weight

In this section we prove the following theorem.

▶ **Theorem 23.** *Algorithm 7 is a local algorithm for $(1 + \epsilon)$-approximating the minimum weight spanning graph for minor-free graphs, with high constant success probability and time and query complexity* $\text{poly}(W, d, \epsilon^{-1})$.

Proof. Let $\mathcal{P}$ denote the partition that the oracle, executed in Step 1, answers according to. With high constant probability, it holds that $|E_\mathcal{P}| \leq \epsilon|V|/W$. Let E_1 denote the event that this conditions holds. We claim that the number edges for which Algorithm 7 returns YES for which both endpoints belong to the same part is at most $|V| - 1$. To see this consider a part $T \in \mathcal{P}$ and a cycle C in $G[T]$. Let $\{u, v\}$ denote the heaviest edge in the cycle. When queried on u and v the covering partition oracle returns sets S_u and S_v such that $T \subseteq S_u \cup S_v$. Thus the cycle C is contained in $G[S_u \cup S_v]$. Therefore the algorithm returns NO on $\{u, v\}$ in Step 3. By the cycle rule the number of edges in $G[T]$ for which the algorithm returns YES is exactly $|T| - 1$.

Hence conditioned on E_1, the total number of edges for which Algorithm 7 returns YES is at most $(|V| - 1) + \epsilon|V|/W$.

By the cycle rule, any edge, e, for which Algorithm 7 returns NO does not belong to the MST of G. Since the MST consists of exactly $|V| - 1$ edges, it follows that, conditioned on E_1, the number of edges that do not belong to the MST and for which Algorithm 7 returns YES is at most $\epsilon|V|/W$ as desired.   ◄

◼ **Algorithm 7** Local algorithm for approximated-MST in bounded-degree minor-free graphs.

Input: $\{u, v\} \in E$ and parameters ϵ and W.
Output: YES if $\{u, v\}$ belongs to the approximated-MST and NO otherwise.
1. Perform a query u and a query v to the covering-partition oracle with parameter ϵ/W. Let S_u and S_v denote the subsets returned by the oracle, respectively.
2. Find the subgraph induced on $S_u \cup S_v$, denoted by $G[S_u \cup S_v]$.
3. Return NO if and only if $\{u, v\}$ is the heaviest edge on any cycle in $G[S_u \cup S_v]$.

4.4   Testing monotone and additive properties

▶ **Theorem 24.** *Any property of graphs which is monotone (closed under removal of edges and vertices) and additive (closed under the disjoint union of graphs) can be tested with one-sided error in minor-free graphs with bounded degree d with query complexity which is* $\mathrm{poly}(d/\epsilon)$ *where ϵ is the proximity parameter.*

Proof. Let $\mathcal{T}$ be a property of graphs which is monotone and additive. We propose the following algorithm for testing $\mathcal{P}$ on an input graph G which is a minor-free graphs of degree bounded by d. Sample a set of $O(d/\epsilon)$ vertices, S, uniformly at random and run the covering partition oracle on each $v \in S$ with parameter $\epsilon/2$.

For each $v \in S$, let S_v denote the set returned by the covering partition oracle when queried on v. Return ACCEPT iff for all $v \in S$, $G[S_v]$ has the property $\mathcal{T}$.

If G has the property $\mathcal{T}$ then since $\mathcal{T}$ is monotone it follows that for all $v \in S$, $G[S_v]$ has the property $\mathcal{T}$ as well.

Let $\mathcal{P}$ denote the partition that the covering partition oracle answers according to. With high constant probability $|E_{\mathcal{P}}| \leq \epsilon|V|/2$. Let E_1 denote the event that this condition holds. If G is ϵ-far from having the property $\mathcal{T}$ then, conditioned on E_1, $G' = (V, E \setminus E_{\mathcal{P}})$ is $(\epsilon/2)$-far from having the property $\mathcal{T}$.

Thus we need to remove at least $\epsilon|V|/2$ edges from G' to obtain the property $\mathcal{T}$. By the additivity and monotonicity of $\mathcal{T}$ it follows that G' has the property $\mathcal{T}$ if and only if for every $T \in \mathcal{P}$, $G[T]$ has the property $\mathcal{T}$. Thus, there are at least $\epsilon|V|/2$ edges, and hence at least $\epsilon|V|/(2d)$ vertices, that belong to parts, $T \in \mathcal{P}$ such that $G[T]$ does not have the property $\mathcal{T}$. Hence, with high constant probability the algorithm sample one of these vertices and rejects G. This concludes the proof.   ◄

───── **References** ─────

1   Isolde Adler and Noleen Köhler. An explicit construction of graphs of bounded degree that are far from being hamiltonian, 2021. `arXiv:2008.05801`.

2   N. Alon, R. Rubinfeld, S. Vardi, and N. Xie. Space-efficient local computation algorithms. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1132–1139, 2012.

3   Jasine Babu, Areej Khoury, and Ilan Newman. Every property of outerplanar graphs is testable. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPIcs*, pages 21:1–21:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. `doi:10.4230/LIPIcs.APPROX-RANDOM.2016.21`.

4   Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 393–402. ACM, 2008. `doi:10.1145/1374376.1374433`.

5   Artur Czumaj, Oded Goldreich, Dana Ron, C. Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Struct. Algorithms*, 45(2):139–184, 2014. `doi:10.1002/rsa.20462`.

6   Artur Czumaj, Morteza Monemizadeh, Krzysztof Onak, and Christian Sohler. Planar graphs: Random walks and bipartiteness testing. *Random Struct. Algorithms*, 55(1):104–124, 2019. `doi:10.1002/rsa.20826`.

7   Artur Czumaj and Christian Sohler. A characterization of graph properties testable for general planar graphs with one-sided error (it's all about forbidden subgraphs). In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1525–1548. IEEE Computer Society, 2019. `doi:10.1109/FOCS.2019.00089`.

8   Alan Edelman, Avinatan Hassidim, Huy N. Nguyen, and Krzysztof Onak. An efficient partitioning oracle for bounded-treewidth graphs. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, volume 6845 of *Lecture Notes in Computer Science*, pages 530–541. Springer, 2011. `doi:10.1007/978-3-642-22935-0_45`.

9   Lars Engebretsen and Marek Karpinski. Approximation hardness of TSP with bounded metrics. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 2001. `doi:10.1007/3-540-48224-5_17`.

10   Hendrik Fichtenberger, Reut Levi, Yadu Vasudev, and Maximilian Wötzel. A sublinear tester for outerplanarity (and other forbidden minors) with one-sided error. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.ICALP.2018.52`.

11   O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. `doi:10.1007/s00453-001-0078-7`.

12   Oded Goldreich. On testing hamiltonicity in the bounded degree graph model. *Electron. Colloquium Comput. Complex.*, 27:109, 2020. URL: `https://eccc.weizmann.ac.il/report/2020/109`.

13   Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. `doi:10.1145/285055.285060`.

14   A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Proceedings of the Fiftieth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 22–31, 2009. `doi:10.1109/FOCS.2009.77`.

15   John E. Hopcroft and Robert Endre Tarjan. Isomorphism of planar graphs. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research*

Center, Yorktown Heights, New York, USA, The IBM Research Symposia Series, pages 131–152. Plenum Press, New York, 1972. `doi:10.1007/978-1-4684-2001-2_13`.

16   Bell (https://cstheory.stackexchange.com/users/48832/bell). Hardness of approximation for minimum path cover in an undirected graph? Theoretical Computer Science Stack Exchange. URL:https://cstheory.stackexchange.com/q/40344 (version: 2018-03-08). `arXiv:https://cstheory.stackexchange.com/q/40344`.

17   Hiro Ito. Every property is testable on a natural class of scale-free multigraphs. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 51:1–51:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. `doi:10.4230/LIPIcs.ESA.2016.51`.

18   Akash Kumar, C. Seshadhri, and Andrew Stolman. Finding forbidden minors in sublinear time: A n^1/2+o(1)-query one-sided tester for minor closed properties on bounded degree graphs. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 509–520. IEEE Computer Society, 2018. `doi:10.1109/FOCS.2018.00055`.

19   Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors II: a poly($d\,\epsilon^{-1}$)-query tester for minor-closed properties of bounded degree graphs. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 559–567. ACM, 2019. `doi:10.1145/3313276.3316330`.

20   Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors III: poly(d/?)-time partition oracles for minor-free graph classes. *Electron. Colloquium Comput. Complex.*, 28:8, 2021. URL: `https://eccc.weizmann.ac.il/report/2021/008`.

21   Mitsuru Kusumoto and Yuichi Yoshida. Testing forest-isomorphism in the adjacency list model. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 763–774. Springer, 2014. `doi:10.1007/978-3-662-43948-7_63`.

22   C. Lenzen and R. Levi. A centralized local algorithm for the sparse spanning graph problem. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 87:1–87:14, 2018.

23   R. Levi and M. Medina. A (centralized) local guide. *Bulletin of the EATCS*, 122, 2017.

24   R. Levi, G. Moshkovitz, D. Ron, R. Rubinfeld, and A. Shapira. Constructing near spanning trees with few local inspections. *Random Struct. Algorithms*, 50(2):183–200, 2017.

25   R. Levi and D. Ron. A quasi-polynomial time partition oracle for graphs with an excluded minor. *ACM Trans. Algorithms*, 11(3):24:1–24:13, 2015.

26   R. Levi, D. Ron, and R. Rubinfeld. Local algorithms for sparse spanning graphs. In *Proceedings of the Eighteenth International Workshop on Randomization and Computation (RANDOM)*, pages 826–842, 2014.

27   R. Levi, D. Ron, and R. Rubinfeld. Local algorithms for sparse spanning graphs. *CoRR*, abs/1402.3609, 2014. URL: `http://arxiv.org/abs/1402.3609`.

28   Reut Levi, Dana Ron, and Ronitt Rubinfeld. Local algorithms for sparse spanning graphs. *Algorithmica*, 82(4):747–786, 2020. `doi:10.1007/s00453-019-00612-6`.

29   Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM J. Comput.*, 42(3):1095–1112, 2013. `doi:10.1137/120890946`.

30   Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Struct. Algorithms*, 20(2):165–183, 2002. `doi:10.1002/rsa.10013`.

31   M. Parter, R. Rubinfeld, A. Vakilian, and A. Yodpinyanee. Local computation algorithms for spanners. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 58:1–58:21, 2019.

32   R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. Fast local computation algorithms. In *Proceedings of The Second Symposium on Innovations in Computer Science (ICS)*, pages 223–238, 2011.

33   Yuichi Yoshida and Hiro Ito. Query-number preserving reductions and linear lower bounds for
        testing. *IEICE Trans. Inf. Syst.*, 93-D(2):233–240, 2010. `doi:10.1587/transinf.E93.D.233`.

A   Related Work

A.1   Partition Oracles

Partition oracles were introduced by Hassidim et al. [14] as a tool for approximating para-
meters and testing properties of minor-free bounded degree graphs. The query complexity of
the partition oracle of [14] is exponential in $1/\epsilon$. The query complexity was later improved
in [25] to be quasi-polynomial in $1/\epsilon$. Very recently, Kumar-Seshadhri-Stolman [20] obtained
a partition oracle with query complexity which is polynomial in $1/\epsilon$.

   Edelman et al. [8] obtained a partition oracle with query complexity polynomial which is
in $1/\epsilon$ for graphs with bounded treewidth.

A.2   Testing Hamiltonicity

Yoshida and Ito [33] and more recently Goldreich [12] proved a linear (in the number of
vertices) lower bound for testing Hamiltonicity (even with two-sided error) in bounded degree
graphs. Adler and Köhler [1] provided a deterministic construction of families of graphs for
which testing Hamiltonicity with one-sided error requires linear number of queries.

A.3   Testing properties of minor-free graphs

Newman and Sohler [29] showed that any property of hyperfinite graphs and in particular
minor-free graphs can be tested with query complexity that depends only on $1/\epsilon$ and d
where d is a bound on the maximum degree. In fact, they proved a stronger claim, that a
minor-free graph can be learned up to a precision of ϵdn edges with such query complexity.
However, although the query complexity of their canonical tester is independent of n it is
super-polynomial in d/ϵ.

   For minor-free graphs of unbounded degrees, Czumaj et al. [6] obtained an algorithm
whose query complexity depends only on $1/\epsilon$ for testing Bipartiteness. More recently, this
result was generalized by Czumaj ans Sohler [7] who proved that any property of minor-free
graphs can be tested with one-sided error with query complexity that depends only on $1/\epsilon$
if and only if it can be reduced to testing for a finite family of finite forbidden subgraphs.
Czumaj et al. [6] also provide a canonical tester for testing H-subgraph freeness for any fixed
H with query complexity that is independent of n, however super polynomial in $1/\epsilon$.

   It was shown that for other restrictive families of graphs of unbounded degree that every
property is testable with query complexity which is at most polylogarithmic in n [21, 17, 3].
Specifically, Kusumoto and Yoshida [21] proved that any property of forests can be tested
with query complexity poly$(\log n)$ and that testing Isomorphism of forests requires $\Omega(\sqrt{\log n})$.
This result was generalized in Babu-Khoury-Newman [3] for k-outerplanar graphs.

A.4   Local algorithms for constructing sparse spanning subgraphs

The model of *local computation algorithms* as considered in this work, was defined by
Rubinfeld et al. [32] (see also Alon et al. [2] and survey in [23]). The problem of constructing
sparse spanning subgraphs in this model was studied in several papers [26, 24, 25, 22, 31, 28].
This problem is a special case of constructing an ϵ-almost MST in which the weights of all
the edges are identical.

For restricted families of graphs, it was shown that the complexity of the problem is independent of n. Specifically, it was shown in [24] that for families of graph that are, roughly speaking, sufficiency non-expanding, one can provide an algorithm with query complexity that is independent of n (however, super-exponential in $1/\epsilon$). This is achieved by simulating a localized version of Kruskal's algorithm. On the negative side, it was also shown in [24] that for graphs with expansion properties that are a little better, there is *no local algorithm* that inspects a number of edges that is independent of n.

In [28] there is an algorithm for locally constructing sparse spanning subgraphs in minor-free, unbounded degree, graphs with query complexity and time complexity which are polynomial in d and $1/\epsilon$. Thus our algorithm for unbounded degree graphs generalizes this result for the weighted case.

In [26, 27] it was shown that a spanning subgraph of almost optimum weight can be constructed locally in minor-free graph with degree bounded by d with query complexity and time complexity which are quasi-polynomial in d, $1/\epsilon$ and W where W is the maximum weight of an edge. Thus our algorithm for unbounded degree graphs generalizes this result to unbounded degree graphs and improves the complexity of the upper bound from quasi-polynomial to polynomial in d, $1/\epsilon$ and W.

B   Omitted proofs and details

B.1   Proof of Claim 11

Let $G = (V, E)$ be a graph and let $\mathcal{C} = \{P_1, \ldots, P_k\}$ be a minimum path cover of G.

We first prove that $\delta_{\mathrm{HAM}}(G) \leq k - 1$. For every $1 \leq i \leq k - 1$ we add an edge which connects the end-vertex of P_i to the start-vertex of P_{i+1}. Thus, by adding $k - 1$ edges we constructed a Hamiltonian path in G.

We next prove that $\delta_{\mathrm{HAM}}(G) \geq k - 1$. By definition, there exist $\delta_{\mathrm{HAM}}(G)$ edges such that when added to G, G becomes Hamiltonian. Let E' denote a set of $\delta_{\mathrm{HAM}}(G)$ such edges and let $G' = (V, E \cup E')$ be the graph resulting from adding these edges to G. Let $\mathcal{H} = (s_1, \ldots, s_{|V|})$ denote a Hamiltonian path in G'. After we remove back the edges in E' we break $\mathcal{H}$ into $|E'| + 1$ connected components (each edge we remove adds an additional connected component), i.e. into $|E'| + 1$ paths. Thus the size of the minimum path cover of G is at most $|E'| + 1 = \delta_{\mathrm{HAM}}(G) + 1$. Thus $k \leq \delta_{\mathrm{HAM}}(G) + 1$ and so $\delta_{\mathrm{HAM}}(G) \geq k - 1$ as desired.

B.2   Proof of Claim 12

The claim that $\delta_{\mathrm{HAM}}(G) \leq \delta_{\mathrm{HAM}}(G')$ follows from the fact that the distance from being Hamiltonian can not decrease when we remove edges.

Let $\mathcal{C}$ be a minimum path cover of G. By Claim 11, $\delta_{\mathrm{HAM}}(G) = |\mathcal{C}| - 1$. Now consider removing the edges in F one by one and how this affects the number of paths in $\mathcal{C}$. After removal of a single edge, the number of paths may increase by at most one. Thus, after removing all the edges in F the paths in $\mathcal{C}$ break into at most $|\mathcal{C}| + |F|$ paths. Thus the size of the minimum path cover of G' is at most $|\mathcal{C}| + |F|$. By claim 11, $\delta_{\mathrm{HAM}}(G') \leq |\mathcal{C}| + |F| - 1 = \delta_{\mathrm{HAM}}(G) + |F|$, as desired.

B.3   Proof of Claim 13

The proof of this claim is similar to the proof of Claim 12.

The claim that $\delta_{\mathrm{HAM}}(G) \leq \delta_{\mathrm{HAM}}(G')$ follows from the fact that the distance from being Hamiltonian can not decrease when we remove edges.

Let $\mathcal{C}$ be a minimum path cover of G. By Claim 11, $\delta_{\mathrm{HAM}}(G) = |\mathcal{C}| - 1$. Now consider removing the edges adjacent to vertices in S vertex by vertex and how this affects the number of paths in $\mathcal{C}$. After removal of edges incident to a specific vertex, the number of paths may increase by at most two. This follows from the fact that each vertex v belongs to exactly one path, P, and the fact that when the edges incident to v are removed, P may break into at most 3 different paths. Thus, after removing all the edges incident to vertices in S the paths in $\mathcal{P}$ break into at most $|\mathcal{C}| + 2|S|$ paths. Thus the size of the minimum path cover of G' is at most $|\mathcal{C}| + 2|S|$. By claim 11, $\delta_{\mathrm{HAM}}(G') \leq |\mathcal{C}| + 2|S| - 1 = \delta_{\mathrm{HAM}}(G) + 2|S|$, as desired.

B.4 Proof of Claim 15

We claim that $\mathrm{MSF}(G) \subseteq \mathrm{MSF}(G') \cup S$. To see this observe that for every edge $e \in E' \setminus \mathrm{MSF}(G')$, it holds, by the cycle rule, that there exists a cycle in G' such that e is the heaviest edges in this cycle. Thus, these edges are not in $\mathrm{MSF}(G)$ either (because all the cycles in G' exist in G as well).

Since the number of connected components in G is at most the number of connected components in G' it holds that $|\mathrm{MSF}(G')| \leq |MSF(G)|$. Thus,

$$|\mathrm{MSF}(G) \setminus \mathrm{MSF}(G')| \geq |\mathrm{MSF}(G') \setminus \mathrm{MSF}(G)| . \tag{1}$$

Since $\mathrm{MSF}(G) \subseteq \mathrm{MSF}(G') \cup S$ it holds that $\mathrm{MSF}(G) \setminus MSF(G') \subseteq S$. Thus,

$$|\mathrm{MSF}(G) \setminus \mathrm{MSF}(G')| \leq |S| . \tag{2}$$

It follows from Equations 1 and 2 that $|\mathrm{MSF}(G') \setminus \mathrm{MSF}(G)| \leq |S|$. Thus, the claim follows from the bound on the maximum weight of an edge in G.

B.5 Correctness of Algorithm 2

▷ **Claim 25.** With high constant probability, the number of edges added to E' in steps 2 and 4 of Algorithm 2 is at most $\epsilon|V|/(3W)$.

Proof. With high constant probability $|E_{\mathcal{P}}| \leq \epsilon|V|/(6W)$. Since G is minor-free it follows by Fact 4 and Markov's inequality the number of edges in $G[H]$ is at most $\epsilon|V|/(6W)$. The claim follows. ◁

▷ **Claim 26.** All edges added to E' in step 5 of Algorithm 2 belong to $\mathrm{MSF}(\hat{G})$ where $\hat{G} = (V, E \setminus E_{\mathcal{P}})$.

Proof. Let $S \in \mathcal{P}$ and let $F = (S, A)$ denote the graph returned by Algorithm 4.

We first prove that all the edges in A belong to $\mathrm{MSF}(\hat{G})$. Since each sub-part B of S is active (see Step 2 of Algorithm 4) as long as the lightest edge in $E^G(B, S \setminus B)$ is lighter than the lightest edge in $E^G(B, H)$ it follows that e_B (see Step 3a of Algorithm 4) is the lightest edge in $E^{\hat{G}}(B, V \setminus B)$. Thus, all the edges of A belong to $\mathrm{MSF}(\hat{G})$ by the cut rule (see Subsection 2.4).

We next prove that for each connected component of F, C, the lightest edge which is adjacent to C and H (if such edge exists) is in $\mathrm{MSF}(\hat{G})$. We first note that we only need to consider S such that $E^G(S, H) \neq \emptyset$. In this case each connected component of F, C, is adjacent to at least one vertex in H. Since C is not active it follows that lightest edge which is adjacent to C and H is the lightest edge in the cut of C in $\hat{G}$. Thus the claim follows from the cut rule. This concludes the proof of the claim. ◁

In the proof of the following claim we closely follow the analysis in [28] (see proof in Subsection B.6).

▷ Claim 27.   Let U denote the set of edges in E' that are incident to two different clusters (namely, each endpoint belongs to a different cluster). With probability $1 - 1/\Omega(|V|)$, $|U| \leq \epsilon|V|/(3W)$.

▷ Claim 28.   Let $G = (V, E)$ be a connected minor-free graph, then with high constant probability the graph returned by Algorithm 2, $G' = (V, E')$, is connected and $\sum_{e \in E'} w(e) \leq (1 + \epsilon)\mathrm{OPT}$, where OPT is the weight of a minimum weight spanning tree of G.

Proof. We begin by proving G' is connected. To see this observe that each vertex either belongs to a cluster or to a subset $S \in \mathcal{P}$ such that $E^G(S, H) = \emptyset$.

By construction, the subgraph induced on each cluster in G' is connected. For any two clusters which are connected by an edge the lightest edge that connects the clusters belongs to E'. This follows from Step 7 in Algorithm 2. If $E^G(S, H) = \emptyset$ then $G'[S]$ is connected by Algorithm 2, as all sub-parts of S remain active throughout the entire execution of the algorithm. Moreover, all the edges in $E^G(S, V \setminus S)$ are in E' as well. Thus G' is connected. The claim regrading the weight of the edges of G' follows from Claim 15 and Claims 25-27.

◁

B.6   Proof of Claim 27

For each cluster B, we charge to B a subset of the edges incident to B so that the union of all the charged edges (over all clusters) contains U. Our goal is to show that with probability $1 - 1/\Omega(n)$, the total number of charged edges is at most $\epsilon|V|/(3W)$.

Let $\hat{G} = (V, \hat{E})$ be such that $\hat{E} = E \setminus E_{\mathcal{P}}$. Consider the auxiliary graph, denoted $\widetilde{G}$, that results from contracting each cluster B and isolated parts in $\hat{G}$ into a *mega-vertex* in $\widetilde{G}$, which we denote by $v(B)$. For each pair of clusters B and B' such that $E^{\hat{G}}(B, B')$ is non-empty, there is an edge $(v(B), v(B'))$ in $\widetilde{G}$, which we refer to as a *mega edge*, and whose weight is $|E^{\hat{G}}(B, B')|$. Since G is minor-free, so is $\widetilde{G}$. By Fact 4, which bounds the arboricity of minor-free graphs, we can partition the mega-edges of $\widetilde{G}$ into r forests. Consider orienting the mega-edges of $\widetilde{G}$ according to this partition (from children to parents in the trees of these forests), so that each mega-vertex has at most r outgoing mega-edges. For cluster B and a cluster B' such that $(v(B), v(B'))$ is an edge in $\widetilde{G}$ that is oriented from $v(B)$ to $v(B')$, we shall charge to B a subset of the edges in $E^{\hat{G}}(B, B')$, as described next.

Let x be an upper bound on the size of parts returned by the partition oracle when executed with parameter $\epsilon/(6W)$. Thus x is an upper bound on the size of part in the partition $\mathcal{P}$ of $G[\mathcal{L}]$. Let $E^b(B, B')$ denote the subset of edges in $E^{\hat{G}}(B, B')$ that are the $(\epsilon/(9rW)) \cdot |E(B, B')|$ lightest edges of $E(B, B')$. We charge all the edges in $E^b(B, B')$ to B. The rationale is that for these edges it is likely that the algorithm won't sample an edge in $E^{\hat{G}}(B, B')$ which is lighter. The total number of such edges is at most $(\epsilon/(9rW)) \cdot |E| \leq \epsilon|V|/(9W)$.

For a light vertex y let $subpart(y)$ denote the subpart of y. Let u be the center of the cluster B, and let $N^b(u, B')$ be the set of vertices, $y \in N(u)$, such that:

$$\big(y \in B' \text{ and } (u, y) \in E^b(B, B')\big) \text{ or } \big(\exists(y', z) \in E^b(B, B') \text{ s.t. } y' \in subpart(y)\big) .$$

That is, $N^b(u, B')$ is the subset of neighbors of u such that if Algorithm 3 selects one of them in Step 2, then it obtains an edge in $E^b(B, B')$. We consider two cases.

First case: $|N^b(u, B')|/|N(u)| < \epsilon^2/(162W^2r^3x\Delta)$. In this case we charge all edges in $E^{\hat{G}}(B, B')$ to B. For each part $subpart(y)$ such that $y \in N^b(u, B')$ there are at most $|subpart(y)| \cdot \Delta$ edges $(y', z) \in E^b(B, B')$ for which $y' \in subpart(y)$. Therefore, in this case

$|E^b(B, B')| \leq x\Delta \cdot |N^b(u, B')| \leq N(u) \cdot \epsilon^2/(162W^2r^3)$ and hence $|E(B, B')| < (9rW/\epsilon) \cdot \epsilon^2/(162W^2r^3) \cdot N(u)$. It follows that the total number of charged edges of this type is at most $(\epsilon/(18rW)) \cdot 2|E| \leq \epsilon n/(9W)$.

Second case: $|N^b(u, B')|/|N(u)| \geq \epsilon^2/(162W^2r^3x\Delta)$. For each u and B' that fall under this case we define the set of edges $Y(u, B') = \{(u, v) : v \in N^b(u, B')\}$ and denote by Y the union of all such sets (over all such pairs u and B'). Edges in Y are charged to B if and only if they belong to U and are incident to a vertex in B. Fix an edge in Y that is incident to B, and note that the selection of neighbors of u is done according to a t-wise independent distribution for $t > 4q$, where q is the sample size set in Step 2 of the Algorithm 3. Therefore, the probability that the edge belongs to U is upper bounded by $(1 - \epsilon^2/(162W^2r^3x\Delta))^q$, which by the setting of q, is at most $p = \epsilon/(18Wr)$ (for sufficiently large constant w.r.t. the Theta notation).

We next show, using Chebyshev's inequality, that w.h.p., the number of edges in Y that are in U is at most $2p|E|$. For $y \in Y$, define J_y to be an indicator variable that is 1 if and only if $y \in F$. Then for a fixed $y \in Y$, $\mathrm{E}[J_y] \leq p$ and $\{J_y\}$ are pairwise independent (this is due to the fact that the samples of every pair of edges are pairwise independent). Therefore, by Chebyshev's inequality,

$$\Pr\left[\sum_{y \in Y} J_y \geq 2p|E|\right] \leq \frac{\mathrm{Var}[\sum_{y \in Y} J_y]}{(p|E|)^2} = \frac{\sum_{y \in Y} \mathrm{Var}(J_y)}{(p|E|)^2} \leq \frac{p(1-p)|E|}{(p|E|)^2} = \frac{1-p}{p|E|} = \frac{1}{\Omega(n)} ,$$

and the proof of Claim 27 is completed.

▶ **Remark 29.** The random seed that Algorithm 2 uses consists of two parts. The first part is for running the partition oracle. The second part is for selecting random neighbors in Step 2 of Algorithm 3. Since the selection of neighbors is according to a t-wise independent distribution we obtain that a random seed of length $\tilde{O}(\log n)$ is sufficient.

B.7 The local implementation of Algorithm 2

Algorithm 8 is the local implementation of Algorithm 2 and is listed next.

◼ **Algorithm 8** Local algorithm for approximated-MST in unbounded-degree minor-free graphs.

Input: $\{u, v\} \in E$.
Output: YES if $\{u, v\}$ belongs to the approximated-MST and NO otherwise.
1. If both u and v are in H return YES.

2. If both u and v are light:
   a. Query the partition oracle on u and v and return YES if they belong to different parts.
   b. Find the sub-parts of u and v by running Algorithm 4.
   c. If u and v are in the same sub-part:
      i. Return YES if $\{u, v\}$ is in the set of edges returned by Algorithm 4 (when running on u).
      ii. Otherwise, return NO.
   d. Otherwise, set C_u to be the center of u and C_v to be the center of v.
   e. If $C_u = C_v$ return NO.

3. Otherwise, if u is light and v is heavy (and analogously if v is light and u is heavy) then:
   a. Find the sub-part of u and set C_u to be the center of this sub-part
   b. Set $C_v = v$

4. Run Algorithm 3 on C_u and C_v and return YES if the edge $\{u, v\}$ is lighter than the edge returned by the algorithm. Otherwise, return NO.