The Subgraph Testing Model

Oded Goldreich
Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel
oded.goldreich@weizmann.ac.il

Dana Ron
School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel
danaron@tau.ac.il

Abstract

We initiate a study of testing properties of graphs that are presented as subgraphs of a fixed (or an explicitly given) graph. The tester is given free access to a base graph $G = ([n], E)$, and oracle access to a function $f : E \to \{0, 1\}$ that represents a subgraph of G. The tester is required to distinguish between subgraphs that posses a predetermined property and subgraphs that are far from possessing this property.

We focus on bounded-degree base graphs and on the relation between testing graph properties in the subgraph model and testing the same properties in the bounded-degree graph model. We identify cases in which testing is significantly easier in one model than in the other as well as cases in which testing has approximately the same complexity in both models. Our proofs are based on the design and analysis of efficient testers and on the establishment of query-complexity lower bounds.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Property Testing, Graph Properties

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.37

Funding This research was partially supported by the Israel Science Foundation (grants No. 671/13 and 1146/18).

1 Introduction

Property testing refers to probabilistic algorithms with sub-linear complexity for deciding whether a given object has a predetermined property or is far from any object having this property. Such algorithms, called testers, obtain local views of the object by *performing queries* and their performance guarantees are stated with respect to a distance measure that (combined with a distance parameter) determines what objects are considered far from the property.

In the last couple of decades, the area of property testing has attracted significant attention (see, e.g., [13, 31, 32, 14]). Much of this attention was devoted to testing graph properties in a variety of models ranging from the dense graph model [15], to the bounded-degree graph model [17], and to the sparse and general graph models [30, 24].[1] These models differ in two main parameters: the *types of queries* that potential testers can make, and the *distance measure* against which their performance is measured.

[1] These models are surveyed in Chapters 8, 9, and 10 of the textbook [14].

In all aforementioned models, the input graph is arbitrary, except for its size (and possibly its degree, in the case of the bounded-degree graph model). The same holds with respect to the graphs that are used to determine the distance of the input from the property. While some prior works (see, e.g., [3, 22, 6, 20, 9, 7, 29, 5]) restrict the input graph in certain natural ways, the restrictions considered so far were expressed in terms of general ("uniform") graph properties (such as degree bound, hyperfiniteness, planarity, etc). See further discussion in Section 1.5.1.

In contrast, we envision circumstances in which the input is restricted to be a subgraph of some fixed graph that is known beforehand. For example, the fixed graph may represent an existing (or planned) network, and the subgraph represents the links that are actually in operation (or actually constructed). Alternatively, the graph may represent connections between data items that may exist under some known constraints, and the edges of the subgraph represent connections that actually exist. Either way, the input is a subgraph of some fixed graph, and the distance to having the property is measured with respect to subgraphs of the same fixed graph.

1.1 The model

In accordance with the foregoing discussion, in the subgraph testing model, there is a fixed base graph, denoted $G = ([n], E)$, and the tester is given oracle access to a function $f : E \to \{0,1\}$ that represents a subgraph of G in the natural manner (i.e., f represents the subgraph $([n], \{e \in E : f(e) = 1\})$). Alternatively, the base graph G is not fixed, but the tester is given free access to G.

▶ **Definition 1.1** (subgraph tester)**.** Fixing $G = ([n], E)$ and $\Pi_G \subseteq \mathcal{F}_G \stackrel{\text{def}}{=} \{f : E \to \{0,1\}\}$, a subgraph tester for Π_G is a probabilistic oracle machine, denoted T, that, on input a (proximity) parameter ϵ, and oracle access to a function $f : E \to \{0,1\}$, outputs a binary verdict that satisfies the following two conditions.

1. *T accepts inputs in Π_G:* For every $\epsilon > 0$, and for every $f \in \Pi_G$, it holds that $\Pr[T^f(\epsilon) = 1] \geq 2/3$.
2. *T rejects inputs that are ϵ-far from Π:* For every $\epsilon > 0$, and for every $f : E \to \{0,1\}$ that is ϵ-far from Π_G it holds that $\Pr[T^f(\epsilon) = 0] \geq 2/3$, where f is ϵ-far from Π_G if for every $h \in \Pi_G$ it holds that $|\{e \in E : f(e) \neq h(e)\}| > \epsilon \cdot |E|$.

If the first condition holds with probability 1 (i.e., $\Pr[T^f(\epsilon) = 1] = 1$ for $f \in \Pi_G$), then we say that T has one-sided error; otherwise, we say that T has two-sided error.

In the alternative formulation, the subgraph tester is given G as an explicit input (along with ϵ). In this case, the random variable being considered is $T^f(G, \epsilon)$.

Definition 1.1 falls within the framework of massively parameterized property testing (cf. [28]). The massive parameter is the base graph $G = ([n], E)$, and the actual input is a function $f : E \to \{0,1\}$ (which represents a subgraph of G).

(The subgraph testing model is syntactically identical to the *orientation model* [21], but semantically these models are fundamentally different; see further discussion in Section 1.5.2.)

As usual in the area, our primary focus is on the query complexity of such testers, and our secondary focus is on their time complexity. Both complexities are stated as a function of the proximity parameter ϵ and the base graph G. Indeed, the dependency of these complexities on G, or rather on some parameters of G, will be of major interest.

As an illustration, consider the problem of testing whether the subgraph is bipartite. If the base graph is bipartite, then this problem is trivial (since every subgraph is bipartite).

Figure 1 For $n \geq 6$, the n-vertex path is oriented by the additional edge $\{n-3, n-1\}$.

If the base graph is $\mathcal{M}$-minor free[2], for any fixed family of graphs $\mathcal{M}$, then testing (with distance parameter ϵ) can be done in $\mathrm{poly}(1/\epsilon)$-time (see Proposition 2.2). Lastly, if the base graph is of bounded-degree, then testing can be done in $\mathrm{poly}(1/\epsilon) \cdot \widetilde{O}(\sqrt{n})$-time (see Theorem 1.2), and this result is optimal in general (i.e., for arbitrary bounded-degree base graphs, see Part 1 of Theorem 1.4).

Our main focus will be on the case that the base graph is sparse (e.g., of bounded-degree). Furthermore, we shall be interested in cases in which the subgraph testing model is different from other testing models. Still, let us make a couple of comments regarding cases in which the subgraph testing model coincides with other testing models.

The dense graph model is a special case of subgraph testing. For the base graph $G = K_n$ (i.e., the n-vertex clique), the subgraph testing model coincides with the dense graph model. This is the case since adjacency queries (as in the dense graph model) correspond to edges of the base graph G, and the distance measure used in both models is the same.

General property testing as a special case of subgraph testing. If the base graph G is sparse and asymmetric (i.e., its automorphism group consists solely of the identity permutation), then the subgraph testing model captures property testing (for Boolean functions) at large. This is shown as follows.

For $n \geq 6$, consider an n-vertex graph G' consisting of an n-vertex long path augmented with the edge $\{n-3, n-1\}$ (see Figure 1). Observe that the only automorphism of this graph is the identity permutation, and augment G' with self-loops on each of the n vertices, deriving a base graph G with $2n$ edges. (We note that the construction can be modified so that self-loops are avoided, by replacing them with disjoint cycles of length 3.) Lastly, associate any function $f : [n] \to \{0,1\}$ with a subgraph of G that contains G' as well as the self-loop on vertices in $f^{-1}(1)$. Note that, by the asymmetry of G', there is a bijection between the set of Boolean functions over $[n]$ and the subgraphs of G that contain G', and that distances between the two models are preserved up to a factor of 2.[3]

On our terminology: Testing graph properties in the subgraph model. Unless the base graph $G = ([n], E)$ is closed under all possible relabelings of $[n]$ (which happens if and only if G is either the complete graph or the empty graph),[4] we cannot expect a (non-empty) set of

[2] Recall that a graph M is a minor of graph G if M can be obtained from G by vertex deletions, edge deletions and edge contractions; a graph G is $\mathcal{M}$-minor free for a family of graphs $\mathcal{M}$, if no graph in $\mathcal{M}$ is a minor of G.

[3] The argument extends to any sparse graph G' that is asymmetric. Recall that almost all (bounded degree) graphs are asymmetric (cf. [10, 25]). On the other hand, an asymmetric graph cannot contain two isolated vertices, and thus it must contain at least a linear number of edges.

[4] Indeed, the n-vertex complete graph and the empty (n-vertex) graph are closed under all possible relabelings of $[n]$. On the other hand, if G is neither the complete graph nor the empty graph, then G is not closed under all possible relabelings of $[n]$. To see this observe that G must contain a vertex w that has degree in $[n-2]$; that is, its neighbor set, denoted $\Gamma_G(w)$, is neither empty nor contains all other vertices in G. Picking $u \in \Gamma_G(w)$ and $v \notin \Gamma_G(w)$, observe that the permutation π that switches u

its subgraphs, Π_G, to constitute a graph property (i.e., to be closed under all relabelings of $[n]$). That is, in the general case, the property $\Pi_G \subseteq \mathcal{F}_G$ is not a graph property, since it is not closed under isomorphism (because $\mathcal{F}_G$ is not closed under isomorphism). Nevertheless, for any base graph G and every graph property Π, we shall refer to $\Pi_G = \mathcal{F}_G \cap \Pi$ as a graph property.

Throughout this paper, we assume that G contains no isolated vertices. This can be assumed without loss of generality, because, for every graph G' that is obtained from the graph $G = ([n], E)$ by adding isolated vertices, it holds that $\mathcal{F}_G = \mathcal{F}_{G'}$, since in both cases the subgraphs are represented by Boolean functions on the same edge-set (i.e., E).

1.2 Results

Throughout this paper, the base graph G is viewed as a varying parameter, which may grow when other parameters (e.g., the degree bound d) are fixed. We focus on bounded-degree base graphs and on the relation between testing graph properties in the subgraph model and testing the same properties in the bounded-degree graph (BDG) model.

Recall that in the BDG model [17], the tester is explicitly given three parameters: n, d, and ϵ. Its goal is to distinguish with probability at least $2/3$ between the case that a graph $G = ([n], E)$ (of maximum degree bounded by d) has a prespecified property Π, and the case that G is ϵ-far from having the property Π, where a graph is said to be ϵ-far from having Π if more than $\epsilon \cdot d \cdot n$ edge modifications (additions or removals) are required in order to obtain a graph (of maximum degree bounded by d) that has Π. To this end the tester can perform queries of the form "who is the i^{th} neighbor of vertex v?", for $v \in [n]$ and $i \in [d]$.[5] Unless stated explicitly otherwise, the degree bound d is a constant.

Obviously, the relationship between the subgraph model and the BDG model depends on the property being tested as well as on the base graph used in the subgraph model. We identify cases in which testing is significantly easier in one model than in the other as well as cases in which testing has approximately the same complexity in both models.

More specifically, we distinguish *downward-monotone* graph properties from other graph properties, where a graph property is called downward-monotone if it is preserved under omission of edges (i.e., if $G = ([n], E)$ has the property, then so does $G' = ([n], E')$ for every $E' \subset E$).

For each of the theorems stated in this section, we provide either a proof outline or a proof idea in Section 1.3. Full proofs of Theorems 1.2, 1.3, and 1.7 (as well as Propositions 1.6 and 2.1) appear in Section 2. The proofs of our other results can be found in our technical report [19].

1.2.1 Downward-monotone properties

We first observe that, for every bounded-degree graph $G = ([n], E)$ and any downward-monotone graph property Π, testing $\Pi \cap \mathcal{F}_G$ in the subgraph model (w.r.t. the base graph G) reduces to testing Π in the BDG model.

▶ **Theorem 1.2** (a general upper bound on the complexity of testing downward-monotone properties (see Section 2.1))**.** *Let Π be a downward-monotone graph property that is testable with query complexity $Q_d(\cdot,\cdot)$ in the bounded-degree graph model, where $d \geq 2$ denotes the*

---

and v, while keeping all other vertices intact, does not preserve the graph G (i.e., $\pi(G) \neq G$).

[5] If v has less than i neighbors, then a special symbol is returned. It is sometimes assumed that the algorithm can also query the degree of any vertex of its choice, but this assumption saves at most a multiplicative factor of $\log d$ in the complexity of the algorithm.

degree bound, and Q_d is a function of the proximity parameter and (possibly) *the size of the graph. Then, for every base graph $G = ([n], E)$ of degree d, testing whether a subgraph of G satisfies* Π (with proximity parameter ϵ) *can be done with query complexity $d \cdot Q_d(\epsilon', n)$, where $\epsilon' = \epsilon/d$. The same holds with respect to the time complexity. Furthermore, one-sided error is preserved.*

(Note that, for constant d, it holds that $\epsilon' = \Omega(\epsilon)$.) Properties covered by Theorem 1.2 include bipartitness, cycle-freeness, and all subgraph-freeness and minor-freeness properties. Hence, testers known for these properties in the BDG model (see [14, Chap. 9]) get translated to testers of similar complexity for the subgraph testing model.

While Theorem 1.2 asserts that testing downward-monotone graph properties in the subgraph model is not harder than testing these properties in the BDG model, it raises the question of whether the former task may be easier.

Testing in the subgraph model may be trivial. A trivial positive answer holds in case the base graph itself has the property (i.e, $G \in \Pi$). In this case, by the downward-monotonicity of Π, every subgraph of G has the property Π, which means that testing $\Pi \cap \mathcal{F}_G$ in the subgraph model (w.r.t. the base graph G) is trivial.

Testing in the subgraph model may be easier (than in the BDG model). A more interesting case in which testing in the subgraph model may be easier than in the BDG model occurs when the base graph is not in Π, but has some suitable property Π' that is not related to Π. In particular, if the base graph is $\mathcal{M}$-minor free, for some fixed set of graphs $\mathcal{M}$, then, for any downward-monotone graph property Π, testing $\Pi \cap \mathcal{F}_G$ in the subgraph model has complexity that is independent of the size of the tested graph, whereas testing Π in the BDG model may require query complexity that depends on the size of the tested graph. More generally, we consider *hyperfinite* families of graphs [8], where an n-vertex graph G is t-hyperfinite for $t : (0, 1) \to \mathbb{N}$ if, for every $\epsilon > 0$, removing at most ϵn edges from G results in a graph with no connected component of size exceeding $t(\epsilon)$. We mention that minor-free (bounded-degree) graphs are hyperfinite (with $t(\epsilon) = O(1/\epsilon^2)$).

▶ **Theorem 1.3** (on the complexity of testing downward-monotone properties of subgraphs of hyperfinite base graphs (see Section 2.2))**.** *Let Π be a downward-monotone graph property and $\mathcal{G}$ be a family of t-hyperfinite graphs. Then, for every bounded-degree base graph $G = ([n], E)$ in $\mathcal{G}$, testing whether a subgraph of G satisfies Π can be done in time that depends only on the proximity parameter ϵ. Furthermore, if Π is additive* (i.e., a graph is in Π iff all its connected components are in Π)*, then its query complexity is $O(t(\epsilon/4)/\epsilon)$ and the tester has one-sided error.*[6]

Note that by Theorem 1.3, testing bipartiteness of subgraphs of any (bounded-degree) planar graph G has complexity $\mathrm{poly}(1/\epsilon)$, whereas (by [17]) testing bipartiteness of n-vertex graphs in the BDG model has complexity $\Omega(\sqrt{n})$.[7]

Testing in the subgraph model may not be easier (than in the BDG model). On the other hand, there are cases in which the testers provided by Theorem 1.2 are essentially the best possible. Indeed, these cases correspond to base graphs that are not hyperfinite.

[6] In general, the tester has two-sided error and the query complexity is at most exponential in $O(t(\epsilon/4)^2)$.
[7] As discussed in Section 1.5.1, weaker results may be obtained by using testers for the BDG model that work under the corresponding promise.

▶ **Theorem 1.4** (testing c-colorability of subgraphs of general bounded-degree base graphs (see [19, Sec. 3]))**.**

1. *There exist explicit bounded-degree graphs $G = ([n], E)$ such that testing whether a subgraph of G is bipartite, with proximity parameter $1/\mathrm{poly}(\log |E|)$, requires $\widetilde{\Omega}(\sqrt{|E|})$ queries.*
2. *There exist bounded-degree graphs $G = ([n], E)$ such that testing whether a subgraph of G is 3-colorable, with constant proximity parameter, requires $\Omega(|E|)$ queries.*

Item 2 asserts that the complexity of testing 3-Colorability in the subgraph model may be linear, just as in the BDG model. Item 1 should be contrasted with the tester obtained by applying Theorem 1.2 to the known tester for the BDG model [16]. The derived tester has complexity $\mathrm{poly}(1/\epsilon) \cdot \widetilde{O}(\sqrt{|E|})$, where ϵ denotes the proximity parameter, whereas Item 1 implies that one cannot obtain better complexity in terms of ϵ and $|E|$ (e.g., complexity $\mathrm{poly}(1/\epsilon) \cdot Q(|E|)$ is possible only for $Q(n) = \widetilde{\Omega}(\sqrt{n})$).

1.2.2 Other properties (i.e., non downward-monotone properties)

When turning to graph properties that are not downward-monotone, the statement of Theorem 1.2 no longer holds. There exist graph properties that are significantly harder to test in the subgraph model than in the BDG model. Specifically:

▶ **Theorem 1.5** (testing in the subgraph model may be harder than in the BDG model (see [19, Sec. 4]))**.** *There exists a property of graphs Π for which the following holds. On one hand, Π is testable in $\mathrm{poly}(1/\epsilon)$-time in the bounded-degree graph model. On the other hand, there exist explicit graphs $G = ([n], E)$ of constant degree such that testing whether a subgraph of G satisfies Π requires $\Omega(\log \log n)$ queries. Furthermore, the property Π is* (upwards) *monotone, and the family of base graphs is hyperfinite.*[8]

The first part of the furthermore-clause implies that a result analogous to Theorem 1.2 does not hold for monotone (rather than downward-monotone) graph properties. The second part of the furthermore-clause implies that also a result analogous to Theorem 1.3 does not hold for monotone graph properties.

We comment that the property Π used in Theorem 1.5 is related to being Eulerian, and the base graphs are related to a cyclic grid. Hence, it is interesting to note that testing whether subgraphs of a cyclic grid are Eulerian can be done in complexity that only depends on the proximity parameters (see [19, Prop. 4.2]).

Turning back to monotone graph properties, we first note the trivial case in which the base graph G does not have the property (which implies that all its subgraphs lack this property as well). A non-trivial case is that of testing minimum degree (see Proposition 2.1). A more interesting case is that of connectivity.

▶ **Proposition 1.6** (testing connectivity in the subgraph model – see Section 2.1)**.** *For every base graph $G = ([n], E)$, testing whether a subgraph of G is connected can be done in $\mathrm{poly}(1/\epsilon)$-time.*

We mention that even the case of 2-edge connectivity, which has an efficient tester in the BDG model, seems challenging in the subgraph testing model (see Problem 1.9).

[8] See the definition of hyperfinite graphs preceding Theorem 1.3.

A relatively general positive result. We next state a result for a class of properties that are not downward-monotone (and not necessarily monotone either). This result is of the flavor of Theorem 1.2, but introduces an overhead that is logarithmic in the number of vertices. Specifically, we refer to the class of all graph properties that have proximity-oblivious testers of constant query complexity (in the BDG model) [18, Sec. 5]. We mention that such properties are "local" in the sense that satisfying them can be expressed as the conjunction of conditions that refer to constant-distance neighborhood in the graph (see Definition 2.4).

▶ **Theorem 1.7** (testing local properties in the subgraph model (see Section 2.3))**.** *Let* Π *be a local property and suppose that the base graph* G *is outerplanar and of bounded degree. Then, testing whether a subgraph of* $G = ([n], E)$ *has property* Π *can be done using* $O(\epsilon^{-1} \log n)$ *queries.*

The result stated in Theorem 1.7 extends to every graph having constant-size separating sets (the dependence on the size of the separating sets is given explicitly in Theorem 2.5).

Testing in the subgraph model may be easier than in the BDG model. Lastly we note that moving from the BDG model to the subgraph testing model makes the testing task *potentially* easier, since the subgraph tester knows *a priori* the possible locations of edges. This is reflected by the following result, which refers to any (bounded-degree) base graph.

▶ **Theorem 1.8** (a property that is extremely easier in the subgraph model)**.** *For every constant* d*, there exists a graph property* Π_d *that requires linear query complexity in the bounded-degree model but can be tested using* $O(1/\epsilon)$ *queries in the subgraph model w.r.t. every base graph of maximum degree* d*.*

Since the proof of Theorem 1.8 is short and simple, we provide it next.

Proof. Fixing d, let Π_d be a set of d-regular graphs such that testing Π_d in the BDG model (with degree bound d) requires a linear number of queries (e.g., Π_d is the set of 3-colorable d-regular graphs [4]). To establish the upper bound in the subgraph model, observe that for any base graph G that has maximum degree d, the only subgraph of G that may be d-regular is G itself. Therefore, if the base graph G is not in Π_d, then the subgraph-tester can reject without performing any queries. If $G \in \Pi_d$, then the subgraph-tester simply tests whether the subgraph of G is G itself (by performing $O(1/\epsilon)$ queries). ◀

The proof of Theorem 1.8 begs the question of whether the theorem holds also for downward-monotone properties, and more generally, which properties Π_d can be tested using $O(1/\epsilon)$ queries in the subgraph model *w.r.t. every base graph of maximum degree* d? Alternatively, one may reverse the order of quantifiers and ask whether there exists a graph property Π that satisfies the conclusion of Theorem 1.8 *for any constant* d.

1.3 Techniques

Some of the testers (algorithms) presented in this paper (e.g., Theorems 1.3 and 1.7) are based on structural properties of the base graph. In some cases (e.g., Theorem 1.3) these structural properties, which are inherited by the subgraphs, make the testing task (in the subgraph model) easier than in the BDG model. The proofs of the lower bounds constitute the more technically challenging part of the paper. Typically, the challenge is emulating lower bounds obtained for other testing models on the subgraph testing model. The brief overviews, especially those referring to the lower bounds, are merely intended to give a flavor of the techniques (and are not supposed to convince the reader of the validity of the claims).

1.3.1 Algorithms

The tester used in proving Theorem 1.2 is a simple emulation of the BDG-model tester by the subgraph tester, and its analysis is based on the observation that the distance between a graph G' and a downward-monotone graph property Π equals the number of edges that should be omitted from G' in order to place the resulting graph in Π. Proposition 1.6 is also proved by a simple emulation of the BDG-model tester, but the analysis of the resulting tester relies on special features of connectivity (and does not extend to 2-connectivity; see Problem 1.9).

The proofs of Theorems 1.3 and 1.7 are more interesting. In both cases we reduce testing subgraphs of the base graph G to testing subgraphs of a fixed subgraph G' of G such that G' is close of G and testing subgraphs of G' is (or seems) relatively easier. Such a reduction is valid since the property that we test is downward-monotone, and the subgraph G' is found without making any queries.

In the proof of Theorem 1.3 the fixed subgraph G' consists of small connected components. Hence, in the special case of Theorem 1.3 (i.e., properties that are determined by their connected components), it suffices to test that the subgraphs induced by the connected components of the base graph have the relevant property. In the general case, we follow Newman and Sohler [29] in estimating the frequency of the various graphs that are seen in these induced subgraphs. We stress that, unlike in [29], we do not use a *partition oracle of the tested graph* (which may be implemented based on standard queries (following Hassidim et al. [22])), but rather determine such a *partition of the base graph* (without making any queries).

Theorem 1.7 is proved by applying a recursive decomposition of the base graph using constant-size separating sets. Essentially, in addition to checking the local neighborhood of random vertices, we also check the local neighborhoods of the vertices in the separating sets that correspond to the path in the recursion tree (i.e., the tree of recursive decomposition) that isolates the chosen vertices. Actually, we check that all these local neighborhoods are consistent with some subgraph that has the property. These additional checks are used in the analysis in order to establish the consistency of the various local neighborhoods (i.e., not only those examined in the same execution).

We highlight the fact that the foregoing testers are non-adaptive. This is remarkable, because the corresponding testers for the BDG model (which in some cases are actually emulated by our testers) are inherently adaptive. However, these "BDG model testers" utilize their adaptivity only for conducting local searches in the input graph, whereas in the subgraph testing model the input is a subgraph of a fixed (or *a priori* known) graph, and so the queries that support these local searches can be determined non-adaptively.

1.3.2 Lower bounds

The lower bound on testing 3-colorability of a subgraph (asserted in Part 2 of Theorem 1.4) is established by combining the query complexity lower bound of [2] with a variant of the standard reduction of `3SAT` to `3COL` (cf. [12, Prop. 2.27]). Recall that Ben-Sasson et al. [2] prove the existence of (sparse) 3CNF formulae for which testing satisfiability of a given assignment requires linear query complexity. We reduce this testing problem, which refers to an explicitly given 3CNF formula (viewed as a massive parameter), to testing 3-colorability of a subgraph of a base graph. That is, we view the NP-reduction of `3SAT` to `3COL` as a mapping of a parameter (i.e., 3CNF formula) of one testing problem to a parameter (i.e., base graph) of another testing problem. In addition, we establish a one-to-one correspondence between

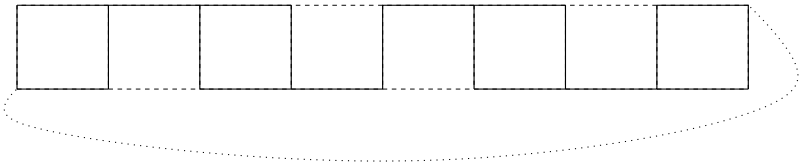

Figure 2 A subgraph of the 2-by-8 grid that misses 4 edges. The subgraph is marked by solid lines, the missing edges by dashed lines, and an external edge that makes this subgraph 2-connected is dotted.

the bits of the assignment and part of the edges in the base graph, while considering only subgraphs that contain all other edges of the base graph (i.e., those not in correspondence to the bits of the assignment).

The proof of Item 1 of Theorem 1.4 (which refers to testing 2-colorability of a subgraph) is more complicated. The basic idea is to emulate the lower bound on bipartite testing established in the BDG model [17]. The obvious question is what should be the base graph. It is natural to pick a base graph that allows an embedding of any bounded-degree graph in it such that edges of the embedded graph are mapped to short vertex-disjoint paths. Furthermore, the mapping of edges to paths should be determined in a local manner. We use a base graph that is related to a routing network of logarithmic depth, and employ (randomized) oblivious routing on it. This allows us to map bipartite graphs (of the BDG model) to bipartite subgraphs of the base graph, while mapping graphs that are far from bipartite (in the BDG model) to subgraphs that are far from bipartite (in the subgraph testing model). The actual analysis of this construction is quite complicated (as evident from the length of [19, Sec. 3.1]), because we have to *locally emulate* a subgraph of the base graph (by making few queries to the input graph in the BDG model).

The proof of Theorem 1.5 uses a reduction from testing Eulerian orientations of cyclic grids in the orientation model. As discussed in Section 1.5.2, the orientation model (presented by Halevy et al. [21]) is related but different from the subgraph testing model. Our reduction maps the (cyclic) grid, used in the lower bound of Fischer et al. [11], to a base graph that looks like such a grid, except that edges are replaced by small gadgets. The orientations of edges in the orientation model are mapped to choices of subgraphs of the corresponding gadgets. In this case, it is easy to locally emulate a subgraph of the base graph by making queries to the orientation oracle, and the claimed $\Omega(\log \log n)$ lower bound follows (from the analogous lower bound of [11]). On the other hand, the property at the image of the reduction is local, and so it is testable within $\text{poly}(1/\epsilon)$ queries in the BDG model.

1.4 Open problems

Moving from the BDG model to the subgraph testing model makes the testing task potentially easier, since the subgraph tester knows *a priori* the possible locations of edges. But, when dealing with properties that are not downward-monotone, there is an opposite effect that arises from the fact that the distance to the set of subgraphs (of G) that have graph property Π may be much bigger than the distance to the set of (bounded-degree) graphs that have property Π. This may require the subgraph tester to reject the input (since its distance to $\Pi \cap \mathcal{F}_G$ is large), whereas the BDG model tester may be allowed to accept the input (since its distance Π is small). This difficulty is reflected in the following open problems.

▶ **Problem 1.9** (testing 2-connectivity of subgraphs). *Is the query complexity of testing 2-edge-connectivity in the subgraph testing model independent of the size of the graph? What about c-edge-connectivity for any constant $c \in \mathbb{N}$?*

Recall that c-connectivity is testable in the BDG model within complexity that depends only on the proximity parameter [17]. We note that a straightforward emulation of the BDG-model tester (for 2-connectivity) calls for trying to find a small 2-connected component that has a cut of size at most 1 to the rest of the graph. But this approach fails when considering a base graph that is a 2-by-n grid (since, as illustrated in Figure 2, any subgraph that misses at most one horizontal edge of each vertex (of degree 4) is $O(1/n)$-close to being 2-connected but may be far from any 2-connected subgraph of the 2-by-n grid).

The straightforward emulation of the BDG-model tester also fails for testing whether a subgraph of the n-cycle is a perfect matching (i.e., is 1-regular), but a tester that considers the locations of edges in the subgraph does work (we discuss this shortly at the very end of [19, Sec. 4]). Testers of complexity that does not depend on the graph size do exist for this property when the base graph is a tree (since each tree has at most one perfect matching)[9], but we do not know if they exist when the graph is outerplaner.

▶ **Problem 1.10** (testing whether the subgraph is a perfect matching)**.** *What is the complexity of testing 1-regularity when the base graph is outerplanar? What about the case that the base graph is planar* (e.g., a grid)*? And what about testing degree-regularity?*

Note that c-connectivity, degree-regularity, and Eulerianity are the only properties covered in [14, Chap. 9] that are not downward-monotone. Also note that [19, Prop. 4.2] refers to the complexity of testing the Eulerian property for a base graph that is a grid, and it is clear that the underlying ideas apply to base graphs of "similar structure" (as arising in the proof of [19, Prop. 4.2]). But what about going beyond that?

▶ **Problem 1.11** (testing whether the subgraph is Eulerian)**.** *What is the complexity of testing the Eulerian property in any base graph of bounded degree?*

The foregoing problems are all rooted in the difficulties that are introduced by the fact that distances under the subgraph model may be significantly larger than under the BDG model, which makes the task of the tester potentially harder. On the other hand, the fact that the base graph is known to the tester makes its task potentially easier. Recalling that only the latter effect is relevant in the case of downward-monotone properties, begs the following question.

▶ **Problem 1.12** (a property that is always easier in the subgraph model)**.** *Does there exist a downward-monotone graph property* Π *such that testing* Π *in the bounded-degree model has higher query complexity than testing* Π *in the subgraph model w.r.t. every base graph of bounded-degree?*

Recall that Theorem 1.8 refers to an upward-monotone property (which depends on the degree bound).

The foregoing problems are aimed at concretizing the abstract challenge of making better use of the knowledge of the base graph that is available to the tester. Although Theorem 1.4 indicates that this extra knowledge is not always helpful, other results point out to cases in which it is helpful. We would like to see more such cases and more substantial use of the knowledge of the base graph.

---

[9] This perfect matching is determined by a pruning process (started at the leaves), and the tester just checks that the subgraph equals this perfect matching (if it exists). Note that, also in this case, the tester does not emulate the BDG-model tester (which just samples vertices and checks their degree); such an emulation will fail poorly (even when the base graph is a path).

1.5 Related models

1.5.1 Testing under a promise

As mentioned earlier, testing graph properties under the promise that the tested graph has some (other) property was considered before (see discussion in [14, Sec. 12.2]). In fact, the bounded-degree graph model itself may be viewed as postulating such a promise. More conspicuous cases include the study of testing under the promise that the graph is hyperfinite [29] or more specifically planar [3], or with bounded tree-width [7]. In continuation to Theorem 1.2, we observe that testing downward-monotone graph properties in the subgraph model can be reduced to testing the same property under a promise that contains the base graph.

▶ **Theorem 1.13** (a generalization of Theorem 1.2)**.** *Let $\mathcal{G}$ and Π be downward-monotone graph properties such that $\mathcal{G}$ contains graphs of degree at most d. Suppose that, when promised that the tested graph is in $\mathcal{G}$, the property Π is testable* (in the bounded-degree graph model) *with query complexity $Q_{\mathcal{G}}(\cdot,\cdot)$, where $Q_{\mathcal{G}}$ is a function of the proximity parameter and* (possibly) *the size of the graph. Then, for every base graph $G=([n],E)$ in $\mathcal{G}$, testing whether a subgraph of G satisfies Π* (with proximity parameter ϵ) *can be done with query complexity $d\cdot Q_{\mathcal{G}}(\epsilon',n)$, where $\epsilon'=\epsilon/d$.*

Hence, results weaker than Theorem 1.3 may be obtained by combining Theorem 1.13 with the tester provided in [29] (see discussion in Section 2.2). Indeed, the improved results are due to the fact that in the subgraph model the tester is given the base graph for free. In the current case (of hyperfinite graphs), the tester does not need to query the tested graph in order to obtain a partition oracle of the tested graph; it may just use an adequate partition of the base graph. In general, a main challenge in the study of the subgraph model is in how to utilize the knowledge of the base graph in order to improve the complexity of testing.

1.5.2 The orientation model

A property testing model that is related to the subgraph model is the *orientation model*, which was introduced by Halevy et al. [21]. Similarly to the subgraph model, in the orientation model there is a fixed (undirected) base graph $G=([n],E)$. However, the goal in the latter model is to test properties of *directed graphs* (digraphs) that are defined by *orientations* of the edges of G. That is, for each edge $\{u,v\}\in E$, either the edge is directed from u to v, or from v to u, and the algorithm may perform queries in order to obtain the orientation of edges of its choice. For a property Π of digraphs, the algorithm should distinguish (with probability at least $2/3$) between the case that the tested orientation $\vec{G}$ has the property Π and the case in which the orientation of more than $\epsilon\cdot|E|$ edges should be flipped in order to obtain the property.

While the subgraph model and the orientation model are syntactically identical, the semantics are very different, as we explain next. Similarly to the subgraph model, an orientation $\vec{G}=([n],\vec{E})$ of G is defined by a function $f:E\to\{0,1\}$. Here, $f(e)=1$ indicates that in $\vec{G}$ the edge e is directed from its smaller (id) endpoint to its larger endpoint. Querying the orientation of an edge hence corresponds to querying f, and distance between two functions f and f' (representing two different digraphs) is simply the Hamming distance between the functions.

The fundamental difference in the semantic between the two models is reflected in the fact that natural properties of digraphs in the orientation model do not correspond to nature properties of graphs in the subgraph testing model, and vice versa. For example, the functions

f that define Eulerian orientations of an undirected graph $G = ([n], E)$ (as described above) do not necessarily define subgraphs of G (i.e., in which $f(e) = 1$ indicates that e belongs to the subgraph) that are Eulerian. Hence, natural properties in one model do not necessarily translate to natural properties in the other model. Still, it may be possible to emulate or reduce testing properties in one model to testing properties in the other model, as we do in the proof of Theorem 1.5.

2 Algorithms

In this section we prove Theorems 1.2, 1.3, and 1.7 (as well as Propositions 1.6 and 2.1). These results refer to different types of base graphs and different classes of properties. We have organized them according to the type of the base graph. Recall that G is assumed to have no isolated vertices, so that $|E| \geq n/2$.

2.1 General bounded-degree base graphs

In this section $d \geq 2$ is a fixed constant, and the base graph G is an arbitrary graph in which each vertex has degree at most d (and at least 1).

2.1.1 Testing downward-monotone properties

We first consider any graph property Π that is preserved under edge omission. Such properties are called downward-monotone or downwards monotone. We prove Theorem 1.2, which asserts that *for every graph $G = ([n], E)$ of degree at most d and any downward-monotone graph property Π, testing $\Pi \cap \mathcal{F}_G$ in the subgraph model* (w.r.t. the base graph G) *is not harder than testing Π in the bounded-degree graph* (BDG) *model.*

Proof of Theorem 1.2. Given oracle access to $f : E \to \{0, 1\}$, the subgraph tester invokes the tester of the BDG model, and emulates an incidence oracle for the subgraph of G represented by f in the natural manner. That is, the query $(v, i) \in [n] \times [d]$ is answered with the i^{th} vertex in the set $\Gamma_f(v) = \{u : \{u, v\} \in E \ \& \ f(u, v) = 1\}$, where vertices are ordered arbitrarily (e.g., by lexicographic order), and the answer is $\perp$ if $|\Gamma_f(v)| < i$. This means that each query (v, i) of the BDG model tester, denoted T, in answered by first retrieving $\Gamma_f(v)$, which in turn amounts to making at most d queries to f (i.e., querying all edges incident to v in G). Hence, the subgraph tester emulates the execution of T on the graph $G^f = ([n], \{e \in E : f(e) = 1\})$.

In the analysis, downward monotonicity is used to associate distance from Π in each of the two models with the number of edges that should be omitted from the subgraph. Specifically, in both cases, the distance from the property reflects the number of edges that should be omitted in order to make the graph satisfy the property (because adding edges never decreases that distance). Specifically, if $f \in \Pi \cap \mathcal{F}_G$, then $G^f \in \Pi$, and T accepts (with probability at least 2/3 in general, and with probability 1 if T has one-sided error). On the other hand, if $f : E \to \{0, 1\}$ is ϵ-far from $\Pi \cap \mathcal{F}_G$, then (by downward-monotonicity of Π) any graph in Π that is closest to G^f must be a subgraph of G^f (i.e., is in $\Pi \cap \mathcal{F}_G$ and so differs from G^f on more than $\epsilon \cdot |E|$ edges). It follows that G^f is ϵ'-far from Π for $\epsilon' = \frac{\epsilon \cdot |E|}{dn/2} \geq \frac{\epsilon}{d}$. ◀

Proof of Theorem 1.13. The proof is identical to the proof of Theorem 1.2, except that here we rely on the hypothesis that $\mathcal{G}$ is downward-monotone. ◀

2.1.2 Testing monotone properties

Theorem 1.2 does not apply to monotone properties. Still, several such properties are quite easy to test in the subgraph testing model. One simple example is the property of having a specified minimal degree.

▶ **Proposition 2.1** (testing minimal degree in the subgraph model). *For $d' \geq 1$, testing whether all vertices in the subgraph have degree at least d' can be done in time $O(d/\epsilon)$.*

Proof. If d' is bigger than the minimum degree of the base graph $G = ([n], E)$, then the tester rejects without performing any queries. Otherwise, the tester selects $\Theta(1/\epsilon)$ vertices, uniformly at random, and computes their degrees in the tested subgraph G^f, by querying all their incident edges in G. The tester accepts if and only if all sampled vertices have degree at least d'.

Hence, the tester makes $O(d/\epsilon)$ queries, and always accepts subgraphs that have the property. To prove that it rejects subgraphs that are ϵ-far from having the property with probability at least $2/3$, we establish the contrapositive statement. Consider a graph G^f that is accepted with probability at least $1/3$. This implies that the number of vertices in G^f whose degree is smaller than d' is at most $(\epsilon/2) \cdot n$. Since in G every vertex has degree at least d', it is possible to add edges to G^f in order to obtain a subgraph that has the property, whereas the number of required added edges is at most $(\epsilon n/2) \cdot d' \leq \epsilon \cdot |E|$. ◀

Proof of Proposition 1.6. We now turn to the proof of Proposition 1.6, which asserts *a* poly(d/ϵ)*-time tester for connectivity in the subgraph model.* If the base graph $G = ([n], E)$ is not connected, then testing is trivial (since all subgraphs of G are disconnected). Otherwise (i.e., the base graph G is connected), connectivity of the input $f \in \mathcal{F}_G$ can be tested by emulating the tester used for the BDG model [17]. This tester samples vertices and explores their local neighborhood in search of small connected components.

The analysis is even simpler than the original (bounded-degree) one since we can add edges without worrying about the degree bound (similarly to the analysis of testing connectivity in the sparse (unbounded-degree) model [30]). Specifically, we use the fact that if f represents a subgraph with t connected components, then by modifying f at one entry we can obtain a function that represents a subgraph with $t - 1$ connected components. (This relies on the fact that G must contain edges between the connected components of G^f.) ◀

As noted in the introduction (see Section 1.4), the argument does not extend to 2-connectivity. The reason is that in that case the known tester for the BDG model [17] does not search for arbitrary 2-connected components but rather for 2-connected components that are connected to the rest of the graph by at most one edge. The problem with that tester is that its analysis requires the ability to add edges between any given pair of such 2-connected components, whereas we can only add edges that exist in the base graph.

2.2 Hyperfinite base graphs

A graph $G = ([n], E)$ is said to have an (ϵ, t)-partition if its vertex set can be partitioned into connected components of size at most t such that the number of edges between these components is at most ϵn.

Recall that a graph M is called a minor of a graph G if M is isomorphic to a graph that can be obtained by (zero or more) edge contractions on a subgraph of G. A graph G is M-minor free if M is not a minor of G. If G has degree at most d and is minor-free (i.e., G is M-minor free for some fixed subgraph M), then it has an $(\epsilon, O((d/\epsilon)^2))$-partition, for every $\epsilon > 0$ (the size of M is "hidden" in the $O(\cdot)$ notation – see [1, 22]).

More generally, Theorem 1.3 refers to any family of hyperfinite graphs, where a family of graph $\mathcal{G}$ is hyperfinite if there exists a function $t : (0,1) \to \mathbb{N}$ such that, for every $\epsilon > 0$, every graph in the family has an $(\epsilon, t(\epsilon))$-partition. We shall first prove the second clause in the theorem, which refers to downward-monotone properties that are additive (i.e., determined by the connected components of the graph).

A special case of interest. We say that a graph property Π is additive if it holds that a graph is in Π if and only if all its connected components are in Π. We comment that not every downward-monotone graph property is additive. For example, consider the graph property Π that consists of all graphs that either constitute of a single (Hamiltonian) cycle or consist of a collection of isolated paths and vertices. Note that Π is closed under omission of edges and vertices, but a graph that consists of several isolated cycles is not in Π (i.e., Π is not additive).[10]

▶ **Proposition 2.2** (testing downward-monotone properties that are additive). *Let $\Pi \neq \emptyset$ be a downward-monotone graph property that is additive. Let $G = ([n], E)$ be a graph of maximum degree d, and $t : (0,1) \to \mathbb{N}$ be such that, for every $\epsilon > 0$, the graph G has an $(\epsilon, t(\epsilon))$-partition. Then, testing whether a subgraph of G is in Π can be done by performing $O(d^2 \cdot t(\epsilon/4)/\epsilon)$ queries. Furthermore, the tester is non-adaptive and has one-sided error.*

In particular, Proposition 2.2 implies that, for every fixed graph M, *testing bipartiteness of a subgraph of G, when G is M-minor free, can be done in* $\mathrm{poly}(d/\epsilon)$*-time, when given an* $(\epsilon/4, O((d/\epsilon)^2))$*-partition of G.*[11] This is much more efficient than testing bipartitness in the bounded-degree model, for which the query complexity is $\Omega(\sqrt{n})$ [17]. It is also more efficient than testing bipartiteness of bounded-degree graphs under the promise that the graph is minor-free, let alone under the weaker promise that the graph is t-hyperfinite. Indeed, under these promises, the tester may implement an $(\epsilon/4, t(\epsilon/4))$-partition oracle of the tested subgraph, but such an implementation requires more than $\mathrm{poly}(t(\epsilon/4))$ queries. Specifically, in the special case of minor-free graphs the best implementation known uses $O(d/\epsilon)^{O(\log(1/\epsilon))}$ queries [26], whereas in the general (t-hyperfinite) case the best implementation known uses $\exp(d^{O(t(\mathrm{poly}(1/\epsilon)))})$ queries [22],

Proof Sketch. Let $(C_1, \ldots, C_r)$ be an $(\epsilon/4, t(\epsilon/4))$-partition of G. Given query access to $f : E \to \{0,1\}$, which represents the subgraph $G^f = ([n], \{e \in E : f(e) = 1\})$, we select at random $\Theta(d/\epsilon)$ vertices, and for each selected vertex v we inspect all edges in the subgraph of $G = ([n], E)$ induced by the part C_i that contains v (i.e., we query all pairs $(u, w) \in E \cap (C_i \times C_i)$). We accept if and only if all the retrieved subgraphs are in Π; that is, we accept if and only if for each inspected C_i it holds that the subgraph of G^f induced by C_i is in Π.

Using the fact that Π is preserved by omission of edges (and omission of connected components), we observe that if G^f is in Π, then so are the subgraphs of G^f induced by the C_i's. Hence, our tester accepts $G^f \in \Pi$ with probability 1. On the other hand, if G^f is ϵ-far from Π, then the subgraph of G^f obtained by omitting all edges between the C_i's is $(\epsilon/2)$-far from Π (since $(\epsilon/4)n \leq (\epsilon/2)|E|$). Denoting the latter subgraph by $\hat{G}^f$, we claim that at least $(\epsilon/4)n/d$ of its vertices reside in connected components that are not in Π, and it follows that the tester rejects G^f with high probability (since each connected component is contained in one of the C_i's).

[10] Indeed, if a graph is in (this) Π, then all its connected are in Π, but the converse does not hold.
[11] Such a partition can be found in polynomial-time [1].

The foregoing claim is proved by relying on the hypothesis that Π is additive. Specifically, if less than $(\epsilon/4)n/d$ vertices reside in connected components that are not in Π, then by omitting less that $(\epsilon/4)n < (\epsilon/2)|E|$ edges we can make all connected components reside in Π (since Π contains the graph consisting of a single vertex). This implies that the graph consisting of these modified connected components is in Π, which in turn contradicts the hypothesis that $\hat{G}^f$ is $(\epsilon/2)$-far from Π. ◀

Greater generality at larger cost. A more general result refers to graph properties Π that are only preserved under edge omission (and to hyperfinite base graphs G). The cost of this generalization is an increase in the query complexity of the tester, as asserted next.

▶ **Proposition 2.3** (testing general downward-monotone properties)**.** *Suppose that Π is a downward-monotone graph property and that, for some $t : [0,1] \to \mathbb{N}$ and every $\epsilon > 0$, the graph $G = ([n], E)$ has an $(\epsilon, t(\epsilon))$-partition. Then, we can test whether a subgraph of G is in Π with query complexity $O(d^2 \cdot \exp(t(\epsilon/4)^2)/\epsilon^2)$.*

We mention that the exponential dependence on t of query complexity of the foregoing tester is unavoidable (in the general case of downward-monotone graph properties). Consider, for example, the case that the base graph is an $\sqrt{n}$-by-$\sqrt{n}$ grid augmented by diagonal edges in each small square, and the following downward-monotone property Π: A graph is in Π if there exists a k such that the graph consists of connected component that are each a k-by-k grid augmented by some of the foregoing diagonal edges such that at most half of the possible patterns occur in these small grids. Now, on proximity parameter $\epsilon > 0$, consider the task of distinguishing the case that the subgraph consists of $0.1/\epsilon$-by-$0.1/\epsilon$ grids in which half of the possible patterns occur from the case in which all patterns occur. A lower bound that is a square root of the nubmber of patterns follows from a birthday paradox argument (and a lower bound that is almost linear follows from [34, 33]).

Proof Sketch. By the premise of the proposition, for every $\epsilon > 0$, the base graph G has an $(\epsilon/4, t(\epsilon/4))$-partition. Let $g \in \mathcal{F}_G$ denote the all-ones function, and let g' be $\epsilon/2$-close to g and describe a subgraph of G in which each connected component has size at most $t(\epsilon/4)$. Hence, $G^{g'}$ is a subgraph of G that is obtained from G by removing the at most $(\epsilon/4)n \leq (\epsilon/2)|E|$ edges between parts in the $(\epsilon/4, t(\epsilon/4))$-partition.

By the closure of Π to edge omissions, every function $f \in \mathcal{F}_G \cap \Pi$ is 0.5ϵ-close to the function $f' \in \mathcal{F}_G \cap \Pi$ such that $f'(e) = f(e) \wedge g'(e)$. Let Π'_G denote the set of graphs obtained in this way; that is, $\Pi'_G = \{f \wedge g' : f \in \mathcal{F}_G \cap \Pi\}$. Since Π is a graph property, it follows that $\Pi'_G = \mathcal{F}_G \cap \Pi'$, where Π' is the set of all graphs that are isomorphic to graphs that appear in Π'_G. Hence, the set Π'_G is closed under all automorphisms of the graph G.

Recalling that Π'_G and likewise Π' contain only graphs that consist of connected components of size at most $t = t(\epsilon/4)$, it follows Π' is characterized by the frequencies in which the various graphs of size at most t appear as connected components. Hence, $f \in \mathcal{F}_G$ describes a graph in Π if and only if $f' = f \wedge g'$ is in $\mathcal{F}_G \cap \Pi'$, where Π' is characterized in terms of the number of connected component that are isomorphic to each of the graphs with at most $t(\epsilon/4)$ vertices (and contain no smaller connected components). It follows that testing with proximity parameter ϵ whether subgraphs of G satisfy Π can be performed by estimating these numbers in the subgraph described by $f \wedge g'$, where f is the tested function. Lastly, we note that estimating the frequencies in which the various $t(\epsilon/4)$-vertex graphs appear as connected components can be done using $O(d^2 \cdot \exp(t(\epsilon/4)^2)/\epsilon^2)$ queries, where $\exp(t(\epsilon/4)^2)$ account for the number of $t(\epsilon/4)$-vertex graphs. ◀

2.3 Local properties and base graphs with small separators

Loosely speaking, a graph property is called *local* if satisfying it can be expressed as the conjunction of local conditions, where each local condition refers to a constant-distance neighborhood of one of the graph's vertices. A precise definition is given next.

▶ **Definition 2.4.** For a constant $\ell \in \mathbb{N}$, the ℓ-neighborhood of a vertex v in a graph G is the subgraph of G induced by all vertices that are at distance at most ℓ from v. A property Π of n-vertex graphs is called ℓ-local if there exists a graph property Π' such that G is in Π if and only if the ℓ-neighborhood of each vertex is G is in Π'. (Actually, Π' is a set of rooted graphs, where the root corresponds to the "center" of the ℓ-neighborhood.)[12] A graph property $\Pi = \bigcup_n \Pi_n$ is local if there exists a constant ℓ such that Π_n is an ℓ-local property of n-vertex graphs.

We mention that this definition coincides with [18, Def. 5.2], and that (in the bounded degree graph model) every graph property that has a proximity-oblivious tester of constant query complexity is local [18, Sec. 5].

For $s : \mathbb{N} \to \mathbb{N}$ we say that a graph $G = ([n], E)$ has separating sets of size s if for every set of vertices $U \subseteq [n]$ there exists a subset $S \subseteq U$ of at most $s(|U|)$ vertices such that the subgraph of G induced by $U \setminus S$ has no connected component of size greater than $\frac{2}{3} \cdot |U|$. For example, every tree has separating sets of size 1, every outerplanar graph has separating sets of size 2 [23, Lem. 3], and n-vertex planar graphs have separating sets of size $O(\sqrt{n})$ [27].

▶ **Theorem 2.5** (Theorem 1.7, generalized)**.** *Let Π be an ℓ-local property and let $s : \mathbb{N} \to \mathbb{N}$. Suppose that the base graph G is of bounded degree d and has separators of size s. Then, testing whether a subgraph of $G = ([n], E)$ has property Π can be done by performing $O(\epsilon^{-1} s(n) \log n \cdot d^{\ell+1})$ queries. Furthermore, the tester is non-adaptive and has one-sided error.*

Proof. We consider a recursive decomposition of the graph G, obtained by applying the guaranteed separators, and a tree that corresponds to these applications. Specifically, the root of the tree corresponds to the separating set, denoted S_λ, that disconnects the graph $G_\lambda \stackrel{\text{def}}{=} G$. Collecting the resulting connected components into two subgraphs, each containing at most two-thirds of G's vertices, we proceed to obtain separating sets, denoted S_0 and S_1, for each of these two subgraphs, denoted G_0 and G_1, respectively. In general, an internal node in the tree is labeled by a string α and corresponds to the subgraph G_α as well as to a separating set S_α for G_α. The children of this node correspond to subgraphs $G_{\alpha 0}$ and $G_{\alpha 1}$ that result from removing S_α from G_α (where the number of vertices in each of these subgraphs is at most two-thirds of the number of vertices in G_α). When the subgraph reaches some constant size, the process stops. Hence, the leaves of the tree correspond to subgraphs of constant size. For a leaf labeled by α, we let S_α be the set of vertices of the subgraph G_α.

For the sake of clarity, we reserve the term 'node' for nodes in the tree (describing the recursive decomposition), and the term 'vertex' for the vertices of G. We shall never talk of edges of the (rooted) tree, but only of the descendance and ancestry relations induced by it. Recall that each node in the tree is associated with a set of vertices of G, and note that these sets form a partition of the vertex set of G. We say that vertex v resides in a node labeled by α if $v \in S_\alpha$. Observe that edges of the graph G can connect vertices that reside in

[12] Marking the root is important only in case that the center of the graph of radius ℓ cannot be uniquely determined.

the same node and vertices that reside in nodes that are in an ancestry relation, but *cannot* connect vertices that reside in nodes that are not in an ancestry relation (equiv., reside in nodes $\alpha'0\alpha''$ and $\alpha'1\alpha'''$ for any $\alpha', \alpha'', \alpha'' \in \{0,1\}^*$).

We are now ready to describe the tester for Π, which is an ℓ-local property for some constant $\ell \in \mathbb{N}$. Given a fixed based graph $G = ([n], E)$ and oracle access to a subgraph represented by $f : E \to \{0,1\}$, the tester repeats the following procedure $\Theta(d/\epsilon)$ times, where if no invocation of the procedure causes rejection, then it accepts.

1. Uniformly select a vertex that resides in one of the leaves of the decomposition tree. (Note that a constant fraction of the vertices of G resides in leaves of the tree.)
2. For each vertex v of G that resides in a node on the path from the selected leaf to the root (including both the leaf and the root), explore the ℓ-neighborhood of v in G (i.e., query f on each of the edges in that neighborhood).
3. If the subgraph discovered in the previous step is not consistent with any n-vertex subgraph of G that has property Π, then reject.

   Note that the aforementioned discovered subgraph includes not only the explored edges but also indication that certain edges do not exist in the subgraph (i.e., the latter include all non-edges of G as well as some edges of G that were queried by the procedure and answered by the value 0).

The query complexity of this procedure is $O(s(n) \log n \cdot d^\ell)$, where d is the degree-bound of G. Clearly, the tester always accepts subgraphs of G that have the property Π. It remains to show that if the subgraph is ϵ-far from Π, then the probability that a single invocation of the procedure causes rejection is $\Omega(\epsilon/d)$.

We establish the contrapositive statement. Suppose that the foregoing procedure rejects with probability $\rho < 1$. We show that it suffices to modify an $O(\rho \cdot d)$ fraction of the edges in G in order to obtain a graph that satisfies Π. We say that a leaf of the tree is good if the procedure does not reject when it selects a vertex that resides in this leaf. We say that an internal node of the tree is good if it appears on the path from some good leaf to the root. Note that $\rho < 1$ implies that there exist good leaves, and hence the root of the tree is good. More generally, if a node is good, then all its ancestors are good. Also note that each vertex that resides in a good node has an ℓ-neighborhood in G^f that satisfies the local condition (i.e., the ℓ-neighborhood is in Π'), where recall that G^f denotes the subgraph of G defined by f.

Hence, we only need to modify the neighborhoods of vertices residing in bad nodes, and we should do so without harming the neighborhoods of vertices that reside in good nodes. But before explaining how this is done, we note that the number of vertices that reside in internal nodes belonging to the subtree rooted in node α is only a constant factor larger than the number of vertices that reside in the leaves of this subtree. On the other hand, considering the set of bad nodes that have good parents, we note that ρ equals the fraction of vertices that reside in leaves of the subtrees rooted at these bad nodes.

Consider an arbitrary bad node, denoted $\alpha\sigma$, that has a good parent, denoted α. Then, the ℓ-neighborhoods of the vertices residing in node α satisfy the local condition (in the subgraph G^f). We claim that the ℓ-neighborhoods of vertices in $G_{\alpha\sigma}$ can be modified so that they satisfy the local conditions as well without modifying the ℓ-neighborhoods of any vertex that resides in a good node. To verify this claim observe the intersection of the ℓ-neighborhoods of vertices in $G_{\alpha\sigma}$ and the ℓ-neighborhoods of vertices that reside in good nodes is contained in the intersection of the ℓ-neighborhoods of vertices in $G_{\alpha\sigma}$ and the ℓ-neighborhoods of vertices that reside either in node α or in one of its ancestors. The reasoning is that if vertex v in $G_{\alpha\sigma}$ is adjacent in G to a vertex u, then either u is in $G_{\alpha\sigma}$ or u is in $S_{\alpha'}$ such that α' is a (not necessarily proper) prefix of α.

Recall that by Item 3 of the procedure (based on which the notion of good node is defined) the fact that node α is good, implies that the ℓ-neighborhoods of vertices in $G_{\alpha\sigma}$ can be modified to satisfy Π' in a manner that is consistent with the ℓ-neighborhoods of all vertices that reside in node α and its ancestors, and so with the ℓ-neighborhoods of all vertices that reside in good nodes. It follows that by modifying f on $G_{\alpha\sigma}$, while maintaining the ℓ-neighborhoods of vertices in S_α (as well as $S_{\alpha'}$ for each α' that is an ancestor of α) intact, we can "fix" the ℓ-local neighborhood of all vertices in $G_{\alpha\sigma}$.

The foregoing process modifies f into a function that describes a subgraph of G that is in Π, while modifying $O(\rho \cdot d \cdot n) = O(\rho \cdot d \cdot |E|)$ edges. The theorem follows. ◀

References

1 N. Alon, P.D. Seymour, and R. Thomas. A separator theorem for graphs with an excluded minor and its applications. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing (STOC)*, pages 293–299, 1990.

2 E. Ben-Sasson, P. Harsha, and S. Raskhodnikova. 3CNF properties are hard to test. *SIAM Journal on Computing*, 35(1):1–21, 2005.

3 I. Benjamini, O. Schramm, and A. Shapira. Every minor-closed property of sparse graphs is testable. *Advances in mathematics*, 223:2200–2218, 2010.

4 A. Bogdanov, K. Obata, and L. Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science (FOCS)*, pages 93–102, 2002.

5 A. Czumaj, M. Monemizadeh, K. Onak, and C. Sohler. Planar Graphs: Random Walks and Bipartiteness Testing. In *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science (FOCS)*, pages 423–432, 2011.

6 A. Czumaj, A. Shapira, and C. Sohler. Testing Hereditary Properties of Nonexpanding Bounded-Degree Graphs. *SIAM Journal on Computing*, 38(6):2499–2510, 2009.

7 A. Edelman, A. Hassidim, H. N. Nguyen, and K. Onak. An Efficient Partitioning Oracle for Bounded-Treewidth Graphs. In *Proceedings of the Fifteenth International Workshop on Randomization and Computation (RANDOM)*, pages 530–541, 2011.

8 G. Elek. The combinatorial cost. *ArXiv Mathematics e-prints*, 2006. `arXiv:math/0608474`.

9 G. Elek. Parameter testing with bounded degree graphs of subexponential growth. *Random Structures and Algorithms*, 37:248–270, 2010.

10 P. Erdos and A. Renyi. Asymmetric graphs. *Acta Mathematica Hungarica*, 14(3):295–315, 1963.

11 E. Fischer, O. Lachish, A. Matsliah, I. Newman, and O. Yahalom. On the query complexity of testing orientations for being Eulerian. *ACM Transactions on Algorithms*, 8(2):15:1–15:41, 2012.

12 O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

13 O. Goldreich, editor. *Property Testing: Current Research and Surveys*. Springer, 2010. LNCS 6390.

14 O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

15 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.

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

17 O. Goldreich and D. Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, 32(2):302–343, 2002.

18 O. Goldreich and D. Ron. On Proximity Oblivious Testing. *SIAM Journal on Computing*, 40(2):534–566, 2011.

19 O. Goldreich and D. Ron. The Subgraph Testing Model. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:45, 2018.

20 M. Gonen and D. Ron. On the Benefit of Adaptivity in Property Testing of Dense Graphs. *Algorithmica*, 58(4):811–830, 2010. Special issue for APPROX-RANDOM 2007.

21 S. Halevy, O. Lachish, I. Newman, and D. Tsur. Testing Orientation Properties. *Electronic Colloquium on Computational Complexity (ECCC)*, 153, 2005.

22 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)*, pages 22–31, 2009.

23 L.S. Heath. Embedding outerplanar graphs in small books. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):198–218, 1987.

24 T. Kaufman, M. Krivelevich, and D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004.

25 J.H. Kim, B. Sudakov, and V.H. Vu. On the asymmetry of random regular graphs and random graphs. *Random Structures and Algorithms*, 21(3-4):216–224, 2002.

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

27 R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Discrete Math*, 1979.

28 I. Newman. Property Testing of Massively Parametrized Problems – A Survey. In O. Goldreich, editor, *Property Testing - Current Research and Surveys*, pages 142–157. Springer, 2010. LNCS 6390.

29 I. Newman and C. Sohler. Every Property of Hyperfinite Graphs Is Testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013.

30 M. Parnas and D. Ron. Testing the Diameter of Graphs. *Random Structures and Algorithms*, 20(2):165–183, 2002.

31 D. Ron. Property Testing: A Learning Theory Perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008.

32 D. Ron. Algorithmic and Analysis Techniques in Property Testing. *Foundations and Trends in Theoretical Computer Science*, 5(2):73–205, 2010.

33 G. Valiant. *Algorithmic Approaches to Statistical Questions*. PhD thesis, University of California at Berkeley, 2012.

34 G. Valiant and P. Valiant. Estimating the Unseen: Improved Estimators for Entropy and Other Properties. *Journal of the ACM*, 64(6):37:1–37:41, 2017.