On the lowest level of query complexity in testing graph properties

Lidor Avigad

Advisor: Oded Goldreich

Department of Computer Science and Applied Mathematics

Weizmann Institute of Science

December 22, 2009

Abstract

In this thesis, we continue the work of Goldreich and Ron in (ECCC 2008) by presenting an infinite family of natural properties of dense graphs having non-adaptive testers of query complexity of $\tilde{O}(1/\epsilon)$, where ϵ is the proximity parameter. Specifically, for every fixed graph H, we show a non-adaptive tester of query complexity $\tilde{O}(\epsilon^{-1})$ for the property of being a blow-up H. This considerably extends the result of Goldreich and Ron that corresponds to the special case in which H consists of a complete graph.

Our techniques significantly extend those of Goldreich and Ron, while coping with difficulties that emerge from the "non forcing" structure of a blow-up of a non-clique.

Contents

1 Introduction

Property testing of graphs is the task of determining whether a given graph $G = ([N], E)$ has a predetermined property Π (YES instance), or is ϵ-far from any other N-vertex graph having the property (NO instance) where distance between graphs are defined according to their representation. The parameter $\epsilon \in (0,1]$, called the proximity parameter, determines the minimal distance that a "NO" instance has from the property. Therefore, property testing can be considered as a relaxation of the exact (decision) problem, where there the task is to determine whether a graph G has or does not have property Π.

When graphs are represented by functions, the distance between them is measured as the fraction of domain on which the functions differ. Specifically, there are two important representation functions for graphs; the adjacency matrix representation (see [GGR98]) and the bounded degree representation (see [GR97]). In this work we refer to the adjacency matrix representation.

Distinguishing between YES and NO instance graphs is done by a randomized algorithm, called a *tester*, that has an oracle access to a representation function. The tester queries the oracle, that is, the representation function of G. The query complexity of a tester is defined as the number of queries made by it, and usually measured as a function of the size of the graph and ϵ. Usually, the tester is required to err with probability at most $1/3$ (on both YES and NO instances).

If a tester accepts any YES instance with probability 1, and rejects NO instances with constant probability, say $\frac{1}{3}$, then it said to have one-sided error. It is is called non-adaptive if it determines all its queries regardless the result of previous queries; otherwise it is called adaptive.

1.1 General perspective and related work

The notion of property testing was first formulated in [RS96], in the context of "program testing" of algebraic functions. Since then property testing appeared in many contexts such as linear functions, low-degree polynomials, clustering and much more. Property testing in the context of combinational properties of graphs was rediscovered and initiated by Goldreich, Ron and Goldwasser in [GGR98]. For further details see surveys [Ron08, Ron].

There is non-abounded amount of the work devoted to the study of testing graph properties in the adjacency matrix model (see e.g. [GGR98, AFKS99, AFNS06]). In [GGR98] there are many natural graph properties which can be tested within query complexity that is a polynomial in the reciprocal of the proximity parameter, e.g. k-Colorability, for any fixed $k \geq 2$. A characterization of graph properties that can be tested in query complexity that only depends on the the proximity parameter was given in [AFNS06].

Goldreich and Ron in [GR08] initiated a study of graph properties that can be tested in the lowest possible non-adaptive query complexity, that is, linear in the reciprocal of the proximity parameter. They provided non-adaptive testers for classes of graphs that each is a collection of at most constant number of isolated cliques. Indeed, the query complexity was $\tilde{O}(\epsilon^{-1})$.

1.2 Our work

A graph $G = ([N], E)$ is a blow-up of graph $H = ([t], E_H)$, if there is a mapping $\mu : [N] \to [t]$ such that $(u, v) \in E$ if and only if $(\mu(u), \mu(v)) \in E_H$. Pictorially, we can think of G as being constructed in the following way. Start with graph H and replace each vertex by an independent set, called a cloud, of vertices of arbitrary size (possibly empty). Then replace each edge of H by a corresponding complete bipartite graph between the corresponding clouds.

Focusing on the adjacency matrix representation, we demonstrate efficient one-sided error adaptive and non-adaptive testers with query complexity $O(\epsilon^{-1})$ and $\tilde{O}(\epsilon^{-1})$, respectively, for the property of being a blow-up of fixed graph $H = ([t], E_H)$. That is, the testing problem when given a graph $G = ([N], E)$, is to decide whether it is a blow-up of H or ϵ-far from any blow-up of H.

Our testers are suitable for a rich family of natural graph properties. In particular, they generalize the tester for the property of being a constant number of isolated cliques given in [GR08], which in our case is essentially equivalent to testing whether the complement of G is a blow up of $H \stackrel{\text{def}}{=} K_t$ where K_t is the complete graph with t vertices.

Note that as proven in [GR08], the minimal query complexity for any non-trivial tester is $\Omega(\epsilon^{-1})$, see Proposition A.1. Thus, we hit the lower bound up to soft (poly-logarithmic) factor, while demonstrating that there is a rich family of graph properties that can be tested using the (almost) lowest query complexity.

We use asymptotic analysis to describe the behavior of the algorithms. Concretely, we think of t and ϵ as being constants, where the size of graph G is increasing (N increases).

1.3 Organization of this thesis

Our main goal is the non-adaptive tester. However, we do it in the following steps. First we start with algorithm for the exact decision problem for the property of being a blow-up, while highlighting our general approach. Next, we present an adaptive tester, and discuss the differences between these algorithms. Lastly, we construct the non-adaptive algorithm. We stress that the main difficulty is in the analysis of the correctness of the non-adaptive tester.

Specifically, in Section 2 we give the necessary definitions and notations. In Section 3 we give an algorithm for the exact problem. In Section 4 we represent an adaptive tester. In Section 5 we represent a non-adaptive tester.

2 Preliminaries

Throughout this thesis we consider simple (no self-loops), undirected graphs. For $N \in \mathbb{N}$, we denote $[N] \overset{\text{def}}{=} \{1, 2, \ldots, N\}$, and $[[N]] \overset{\text{def}}{=} \{0, 1, 2, \ldots, N\}$ (including zero). Let $H = ([t], E_H)$ and $G = ([N], E)$ be graphs, and let $\mathcal{G} = [g_{u,v}]_{u,v \in [N]}$ be the adjacency matrix of G.

For $S \subseteq [N]$, the subgraph induced by S, denoted $G_{|S} \overset{\text{def}}{=} (S, E(S))$, consists of the vertex set S and the set of edges between pairs of vertices in S; that is, $E(S) \overset{\text{def}}{=} E \cap S^2$.

Signatures For a graph $G' = ([N'], E')$, denote its adjacency matrix as $[g'_{u,v}]_{u,v \in [N]}$ (lowercase). A signature of a vertex $u \in [N']$, denoted $\chi^{G'}(u)$, is the uth row in the adjacency matrix G', that is, $\chi^{G'}(u) = (g'_{u,v})_{v \in [N]}$. For $S \subseteq [N']$, the signature restricted to S of a vertex $v \in [N']$, denoted $\chi^{G'}_S(v)$, is defined as the signature of v restricted to columns specified in S, that is, $\chi^{G'}_S(u) = (g'_{u,v})_{v \in S}$. For shorthand, whenever G' is clear from the context we omit it.

2.1 Distance between graphs

Distance between graphs A generic way of defining distance between arbitrary functions that share the same domain refers to the fraction of domain on which the functions differ. For functions that have different domains, we define the distance to be infinity.

In this work, we represent graphs by the adjacency matrix representation. As a consequence, the distance between graphs $G = ([N], E)$ and $G' = ([N], E')$ is

$$\delta(G, G') \overset{\text{def}}{=} \frac{\left|\{\, (u, v) \in [N]^2 \ : \ g_{u,v} \neq g'_{u,v} \,\}\right|}{N^2}.$$

Note that each pair $\{u, v\}$ is counted twice since it corresponds to two ordered pairs of vertices.

Distance between graph and set of graphs Let $\mathcal{G}$ be a set of graphs. Then, the distance between G and $\mathcal{G}$, denoted $\Delta(G, \mathcal{G})$, is defined to be the minimal distance between G and any graph in $\mathcal{G}$, that is,

$$\Delta(G, \mathcal{G}) \overset{\text{def}}{=} \min_{G' \in \mathcal{G}} \{\delta(G, G')\}$$

When we say that G is ϵ-far from $\mathcal{G}$, we mean that G has distance of at least ϵ from any graph in $\mathcal{G}$.

2.2 Blow-up graph of H

The graph $G = ([N], E)$ is a blow-up of graph $H = ([t], E_H)$, if there is a mapping $\mu : [N] \to [t]$, called a graph blow-up mapping, such that $(u, v) \in E$ if and only if $(\mu(u), \mu(v)) \in E_H$, or equivalently, using our notation, $g_{u,v} = h_{\mu(u),\mu(v)}$ for any $u, v \in [N]$.

Denote by $\mathcal{B}_N(H)$ the set of N-vertex graphs that are blow-ups of H, and by $\mathcal{B}(H)$ the set of all blow-ups of H (with any number of vertices); that is, $\mathcal{B}(H) \overset{\text{def}}{=} \cup_{N\in\mathbb{N}}\mathcal{B}_N(H)$.

We remark that any blow-up of any induced subgraph of H is also a member of $\mathcal{B}(H)$. Clearly, if $G \in \mathcal{B}_N(H)$ then any graph that is isomorphic to G is also a member of $\mathcal{B}_N(H)$. Also note that any graph G is a blow-up of some graph, because G in particular is a blow-up of itself.

The following fact is a simple consequence of a blow-up mapping. It asserts that, in any blow-up mapping, vertices of G that are mapped to the same vertex in H must have identical signatures in G.

Fact 2.1. *Let $H' = ([t'], E_{H'})$ be an arbitrary graph, and let $\pi' : [N] \to [t']$ be a graph blow-up mapping from G to H'. If u and v are vertices of G such that $\pi'(u) = \pi'(v)$, then their signatures in G are identical, that is, $\chi(u) = \chi(v)$.*

Proof. Let $w \in [N]$ be an arbitrary vertex of G. Then,

$$\begin{aligned} g_{u,w} &= h'_{\pi'(u),\pi'(w)} && \pi' \text{ is a blow-up mapping} \\ &= h'_{\pi'(v),\pi'(w)} && \pi'(u) = \pi'(v) \text{ by hypothesis} \\ &= g_{v,w} && \pi' \text{ is a blow-up mapping.} \end{aligned}$$

Hence, $g_{u,w} = g_{v,w}$ for any w, which implies $\chi(u) = \chi(v)$. ∎

Violating pairs Let $\mu : [N] \to [t]$ be a mapping. Then, a violating pair of G with respect to H and μ is a pair that demonstrates that μ is not a blow-up mapping from G to H; that is

Definition 2.2. A violating pair of G with respect to H and μ is a pair $(u, v) \in [N]^2$ such that $g_{u,v} \neq h_{\mu(u),\mu(v)}$.

Observe that being a violating pair is a feature of G, H and a mapping μ. We denote by $Viol^{G,H}_{\mu}$ the set of violating pairs of G with respect to H and μ. Since we think of H as being a fixed graph and G is clear from the context, we use the shorthand, $Viol_{\mu} = Viol_{\mu}^{G,H}$. For $A, B \subseteq [N]$, we denote by $Viol_{\mu}(A,B)$ the set of violating pairs restricted to $A \times B$, that is, $Viol_{\mu}(A,B) \stackrel{\text{def}}{=} Viol_{\mu} \cap (A \times B)$.

Being far from $\mathcal{B}(H)$ Let $\mu : [N] \to [t]$ be an arbitrary mapping, and let $G' = ([N], E')$ be a graph such that for each pair $(u,v) \in [N]^2$ it holds that $g'_{u,v} \stackrel{\text{def}}{=} h_{\mu(u),\mu(v)}$, then it follows that $G' \in \mathcal{B}_N(H)$. By definition of distance, the statement "G is ϵ-far from $\mathcal{B}_N(H)$" implies that $\delta(G,G') \geq \epsilon$, where δ is defined in Section 2.1. Let $(u,v) \in [N]^2$ be a pair that contributes the distance between G and G'. Then,

$$\begin{aligned}
g_{u,v} &\neq g'_{u,v} && \text{by hypothesis} \\
&= h_{\mu(u),\mu(v)} && \mu \text{ is a blow-up mapping} \\
&\Downarrow \\
g_{u,v} &\neq h_{\mu(u),\mu(v)} \\
&\Downarrow \\
&(u,v) \in Viol_{\mu} && \text{by definition of a violating pair}
\end{aligned}$$

Therefore, the pair (u,v) is a violating pair of G with respect to H and μ. Thus, we reach the following conclusion.

Fact 2.3. *Suppose that G is ϵ-far from $\mathcal{B}_N(H)$, and let $\mu : [N] \to [t]$ be an arbitrary mapping. Then, the number of violating pairs of G with respect to H and μ, is at least ϵN^2; that is, $Viol_{\mu}^{G,H} \geq \epsilon N^2$.*

Testing the property of being a blow-up is non-trivial A graph property Π is non-trivial for testing if there exists $\epsilon_0 \in (0,1]$ such that for infinite $N \in \mathbb{N}$ there exists N-vertex graphs G_1 and G_2 such that $G_1 \in \Pi$ and G_2 is ϵ_0-far from Π. Then,

Proposition 2.4. *The property of being a blow-up of $H = ([t], E_H)$ is non-trivial.*

Proof. Fix $\epsilon_0 \in (0,1]$. Let G_1 be the graph constructed by selecting an arbitrary mapping $\pi : [N] \to [t]$, and then define its set of edges such that π will be a blow-up mapping. Clearly, $G_1 \in \mathcal{B}_N(H)$. Next, let $G_2 = ([N], E_2)$ be the complete graph, that is, $E_2 \stackrel{\text{def}}{=} \{\{i,j\} : i \neq j \in [N]\}$. Since there are no self-loops, each vertex has a unique signature in G_2. By Fact 2.1, it follows that for any graph $G' \in \mathcal{B}_N(H)$ the number of unique signature of vertices of G' is at most t. Thus, the number of edges needed to be modified in G_2 it at least $M(N) \overset{\text{def}}{=} t \cdot \frac{N}{t} \cdot (\frac{N}{t} - 1)$. Therefore, for every N such that $M(N) \geq \epsilon_0$, it follows that G_2 is ϵ_0-far from $\mathcal{B}_N(H)$. ■

2.3 Tester for being a blow-up

We are now ready to define formally (one-sided error) testers for the property of being a blow-up of graph $H = ([t], E_H)$.

Definition 2.5. A tester for the graph property of being a blow-up of graph H is a probabilistic oracle machine that, on input parameters N, proximity parameter ϵ, and oracle access to the adjacency matrix of G, output a binary verdict that satisfies the following two conditions.

1. If $G \in \mathcal{B}_N(H)$, then the tester always accepts (that is, accepts with probability 1).
2. If G is ϵ-far from $\mathcal{B}_N(H)$, then the tester accepts with probability at most $1/3$.

A tester is called non-adaptive if it determines all its queries regardless the result of previous queries; otherwise it is called adaptive.

2.4 Knowledge matrix

Loosely speaking, a knowledge matrix, denoted $\mathcal{K}$, is the adjacency matrix of G, denoted $\mathcal{G}$, with small modification; each entry $k_{u,v}$ is allowed to take a special symbol, denoted $*$, that "masks" the actual value of the adjacency matrix (that is, it masks $g_{u,v}$). The knowledge matrix models, in a natural way, the information a tester has on the tested graph.

It is instructive to think about $\mathcal{K}$ in the folowing way. At the beginning of a test, $\mathcal{K}$ has all $*$ entries except the main diagonal that has only 0 entries, because G is simple. Each query $(u,v) \in [N]^2$ sets entry $k_{u,v} \overset{\text{def}}{=} g_{u,v}$; that is, the value $g_{u,v}$ is revealed. Formal definition follows.

Definition 2.6 (Knowledge matrix). A knowledge matrix of G is a matrix $\mathcal{K} \in \{0,1,*\}^{N\times N}$ such that if $k_{u,v} \neq *$, then $k_{u,v} = g_{u,v}$.

Knowledge submatrices Since the number of queries allowed for our testers is much smaller than the number of entries in $\mathcal{G}$, the matrix $\mathcal{K}$ is very sparse with respect to Boolean entries. Thus, for clearer proofs, we focus our attentions on the informative portion of $\mathcal{K}$. The formal way we do it is by using the notion of submatrices.

Notation 2.7. *Let $R, C \subseteq [N]$, where $R = \{r_1, \ldots, r_n\}$. Then, denote by $\mathcal{K}[R,C]$ the submatrix defined by taking the R rows and C columns of the knowledge matrix $\mathcal{K}$. We often let $K \stackrel{def}{=} \mathcal{K}[R,C]$ be an arbitrary submatrix.*

Inconsistency As we show next, it is of our interest to identify vertices in $[N]$ having distinct signatures in G; that is, distinct rows in $\mathcal{G}$. However, all that a tester has is a knowledge matrix $\mathcal{K}$ containing partial information of $\mathcal{G}$. By the interpretation of the $*$ symbol, two rows, $r, r' \in [N]$ in $\mathcal{K}$ have distinct signatures in $\mathcal{G}$ if there is a common column $c \in [N]$ such that $k_{r,c}, k_{r',c} \neq *$, and $k_{r,c} \neq k_{r',c}$. This is the motivation for the following definitions.

Let $a, b \in \{0, 1, *\}$, then a and b are inconsistent, denoted $a \not\approx b$, if $a, b \in \{0, 1\}$ and $a \neq b$. Loosely speaking, inconsistency is a generalization of the equivalence relation taking into considerations of the $*$ symbol; if two elements in $\{0, 1\}$ are inconsistent, then they are distinct. However, we require nothing if they are consistent, spelling it out, they might be distinct or not. The elements a and b are consistent, denoted $a \approx b$, if they are not inconsistent; that is, either $a = *$ or $b = *$, or $a = b$.

We generalize naturally the notion of inconsistent elements into inconsistent vectors. Let $m \in \mathbb{N}$, $\mathbf{a} = (a_1, \ldots, a_m)$, and $\mathbf{b} = (b_1, \ldots, b_m)$ such that $a_i, b_i \in \{0, 1, *\}$ for $i \in [m]$. Then, we say that vectors $\mathbf{a}$ and $\mathbf{b}$ are inconsistent, denoted $\mathbf{a} \not\approx \mathbf{b}$, if there exists a common entry $i^* \in [m]$ on which the vectors are inconsistent, that is, $a_{i^*} \not\approx b_{i^*}$. The vectors $\mathbf{a}$ and $\mathbf{b}$ are consistent, denoted $\mathbf{a} \approx \mathbf{b}$, if for each entry $i \in [m]$ we have that $a_i \approx b_i$.

Since signatures in the knowledge matrix $\mathcal{K}$ are vectors, we say that two rows in K are inconsistent (consistent) if they are inconsistent (consistent) as vectors. We say that two knowledge matrices, having the same sets of rows and columns, are inconsistent (consistent) matrices, if their canonical representations as “long” vectors are inconsistent (consistent).

Next, we define formally the notion of a knowledge matrix K having pairwise inconsistent rows.

Definition 2.8. The matrix $K = \mathcal{K}[R,C]$, has pairwise inconsistent rows if for any two rows $r, r' \in R$ the corresponding vectors are inconsistent, that is, $K[\{r\}, C] \not\approx K[\{r'\}, C]$

Conclusion Let $K = \mathcal{K}[R, C]$ be a knowledge submatrix such that R consists of rows that are pairwise inconsistent in K. Then, R consists of vertices with distinct signatures in $\mathcal{G}$. Having such knowledge submatrix is important, because following Fact 2.1, in any blow-up mapping μ (from G to H), each row in R must be mapped to distinct value; that is, μ restricted to R is an injection into $[t]$.

2.5 Candidate blow-up mapping

In this subsection, unless stated otherwise, the knowledge submatrix $K = \mathcal{K}[R, C]$ has pairwise inconsistent rows. Thus, by Fact 2.1, any blow-up mapping of K to H must be injective.

Consider the submatrix K. If there does not exist a mapping $\rho : R \to [t]$ such that the matrix $K' \in \{0,1\}^{|R|\times|R|}$ where $k'_{r,r'} \overset{\text{def}}{=} h_{\rho(r),\rho(r')}$ is consistent with $K[R, R]$, then clearly $G \notin \mathcal{B}(H)$. But if there is such ρ, we say that K is H-mappable. In such case $\mathcal{K}$ might be extended into an adjacency matrix of a blow-up of H, that is, we can construct a full-fledge candidate blow-up mapping $\mu : [N] \to [t]$ that might be a blow-up mapping. This is useful when G is ϵ-far from $\mathcal{B}(H)$, because by Fact 2.3 in particular $Viol_\mu \geq \epsilon N^2$.

Construction of the candidate blow-up mapping μ is based on two elements. First, we partition $[N]$ according to the knowledge submatrix K such that each row $r \in R$ defines a unique set V_r in the partition. Second, we assume that K is H-mappable; that is, there is $\rho : R \to [t]$ such that $K[R, R]$ is consistent with $H[\rho(R), \rho(R)]$. Then, we define $\mu : [N] \to [t]$ by assigning elements of V_r to $\rho(r)$. Vertices that do not belong to $\cup_{r\in R} V_r$ are mapped to an arbitrary value in $[t]$. Definitions and details follow.

H-mappable Let K be a knowledge matrix with pairwise inconsistent rows. Then, we say that K (or rather its rows) is H-mappable if there exists a mapping $\rho : R \to [t]$, called an H-mapping, such that

- ρ is an injection.
- For every $r, r' \in R$ we have that $k_{r,r'} \approx h_{\rho(r),\rho(r')}$.

The first condition stems from the fact that the rows of K are pairwise inconsistent; thus, each vertex $r \in R$ must be mapped to distinct values in $[t]$. The second condition stems from the fact that we require that ρ will be a blow-up mapping of the graph that is represented by the matrix K. The following fact clearly holds.

Fact 2.9. *If G is a blow-up of H, then (any) knowledge submatrix $K = \mathcal{K}[R, C]$ of G is H-mappable.*

Proof. If G is a blow-up of H, then there exists a blow-up mapping $\mu : [N] \to [t]$ such that for each $u, v \in [N]$, $g_{u,v} = h_{\mu(u),\mu(v)}$. Let $\rho : R \to [t]$ such that $\rho(r) \overset{\text{def}}{=} \mu(r)$. Now, since R contains vertices with distinct signatures in G, by Fact 2.1, we have that μ restricted to R is an injection, thus, ρ is an injection.

Next, let $r, r' \in R$. If $k_{r,r'} = *$, then by definition of $*$ symbol, $k_{r,r'} \approx h_{\rho(r),\rho(r')}$. On other hand, if $k_{r,r'} = g_{r,r'}$ (as must be the case when $k_{r,r'} \neq *$), then

$$\begin{aligned} k_{r,r'} &= g_{r,r'} \\ &= h_{\mu(r),\mu(r')} && \mu \text{ is a blow-up mapping} \\ &= h_{\rho(r),\rho(r')} && \text{definition of } \rho \\ &\Downarrow \\ k_{r,r'} &\approx h_{\rho(r),\rho(r')}. \end{aligned}$$

So the fact follows. ∎

The partition of [N] induced by K For $r \in R$, the set $V_r(K)$ consists of any vertex $v \in [N]$ that has signature consistent with row r in matrix K; that is,

$$V_r(K) \overset{\text{def}}{=} \{\, v \in [N] \ : \ \chi_C(v) \approx K[\{r\}, C] \,\}. \tag{2.1}$$

The set $L(K)$ consists of vertex $v \in [N]$ with signature that is inconsistent with any row in K; that is,

$$L(K) \overset{\text{def}}{=} \{\, v \in [N] \ : \ \forall r \in R, \ \ \chi_C(v) \not\approx K[\{r\}, C] \,\}. \tag{2.2}$$

Equivalently, $L(K) \overset{\text{def}}{=} [N] \backslash (\cup_{i \in [n]} V_{r_i}(K))$.

Consider $(V_{r_1}(K), \ldots, V_{r_n}, L(K))$. Since K has pairwise inconsistent rows, this tuple defines a unique partition of $[N]$. This is true because a vertex $v \in [N]$ belongs to exactly one of these sets. Thus, the partition of $[N]$ induced by K, denoted $P(K)$, is defined as $P(K) \overset{\text{def}}{=} (V_1, \ldots, V_n, L)$ where $V_i \overset{\text{def}}{=} V_{r_i}(K)$ for $i \in [n]$, and $L \overset{\text{def}}{=} L(K)$.

The candidate blow-up mapping μ Next, in case K is H-mappable, we define the candidate blow-up mapping μ.

Definition 2.10. Let $K = \mathcal{K}[R, C]$ be a knowledge submatrix with pairwise inconsistent rows. Let $\rho : R \to [t]$ be an H-mapping, and let $P(K) = (V_1, \ldots, V_n, L)$ be the partition of $[N]$ induced by K. Then, the candidate blow-up mapping $\mu : [N] \to [t]$ is defined as

$$\mu(v) \stackrel{\text{def}}{=} \begin{cases} \rho(r_1) & v \in V_1 \\ \vdots & \vdots \\ \rho(r_n) & v \in V_n \\ \rho(r_1) & v \in L \end{cases}.$$

Observe, vertices in L are mapped to arbitrary value $\rho(r_1)$. We stress that we could have other possible mapping of vertices from L to the V_i's, for example, map each vertex $u \in L$ into $j \in [n]$ such that the number of violating pairs in $Viol_\mu(\{u\}, V)$ is minimum, where $V \stackrel{\text{def}}{=} \cup_{i \in [n]} V_i$ (note that there is no circular reference, as this depends only on the initial mapping of V). However, as we see, our arguments do not relay on the mapping of L but on the facts that the either L is large or the number of violating pairs inside V is large.

3 Efficient algorithm for the exact decision problem

We start by providing an efficient decision algorithm for deciding whether a given graph G is a blow-up of a fixed graph H. We show that this problem decidable in polynomial time (that is, it is in $\mathcal{P}$). Instead of providing the algorithm and then analyze it, we develop it step by step. This way we highlight our main approach used repeatedly.

3.1 The basic idea

A naive approach for this decision algorithm is to check whether there exists a blow-up mapping, by exhaustively trying each mapping from $[N]$ to $[t]$. Clearly, this approach is wasteful and costs $\Omega(t^N)$ time to implement.

The improvement Assume, there is an efficient "compression" operation, denoted $Comp$, that possibly shrinks G into a graph $Comp(G)$ such that $G \in \mathcal{B}(H)$ if and only if $Comp(G) \in \mathcal{B}_t(H)$. In particular whenever $G \in \mathcal{B}(H)$ the graph $Comp(G)$ has at most t vertices. Then, we suggest a better algorithm; first apply the compression operation on G to produce a graph

$Comp(G) = ([m], E_m)$. If $m > t$, then rejects, because $Comp(G) \notin \mathcal{B}_t(H)$, and $G \notin \mathcal{B}(H)$, follows. Otherwise, $m \leq t$ and we proceed as in the naive case; that is, check whether there exists a blow-up mapping, by trying each mapping from $[m]$ to $[t]$. Assuming that the efficient compression has time complexity $C(N)$, the total running time of the algorithm is $C(N) + O(t^t)$. All that is left is to represent such an efficient compression operation.

Compression operation Consider the following operation.

Definition 3.1 (Compression operation)**.** Define the equivalence relation $\sim$ such that for vertices $u, v \in [N]$, $u \sim v$ if and only if $\chi(u) = \chi(v)$. The relation $\sim$ partition $[N]$ into equivalence classes such that each one holds vertices with identical signature. Let $R \subseteq [N]$ be a set of representatives of these equivalence sets. Then, we define the compressed graph of G, denoted $Comp(G)$, as $Comp(G) \overset{\text{def}}{=} G_{|_R}$.

Now, we have to prove that $Comp(G)$ is well defined and that $G \in \mathcal{B}(H)$ if and only if $Comp(G) \in \mathcal{B}_t(H)$. The following proposition states that $Comp(G)$ is defined up to isomorphism; thus, it is well defined.

Proposition 3.2. *Let $R, R' \subseteq [N]$ be sets of representatives of the equivalence classes induced by $\sim$ such that $R' \neq R$. Then, the graph $G_{|_{R'}}$ is isomorphic to $G_{|_R}$.*

Proof. Let $A_1, \ldots, A_n$ be the equivalence classes induced by $\sim$. Assume $R = \{r_1, \ldots, r_n\}$, and $R' = \{r'_1, \ldots, r'_n\}$ such that $r_i, r'_i \in A_i$ for every $i \in [n]$. Let $i, j \in [n]$, then

$$\begin{aligned} g_{r_i,r_j} &= g_{r_i,r'_j} && \chi(r_j) = \chi(r'_j) \\ &= g_{r'_i,r'_j} && \chi(r_i) = \chi(r'_i) \\ &\Downarrow \\ g_{r_i,r_j} &= g_{r'_i,r'_j}. \end{aligned}$$

Thus, the mapping $\rho : R \to R'$ such that $\rho(r_i) = r'_i$ for $r_i \in R$, is an isomorphism between $G_{|_R}$ and $G_{|_{R'}}$. ■

And lastly,

Proposition 3.3. *$G \in \mathcal{B}(H)$ if and only if $Comp(G) \in \mathcal{B}_t(H)$.*

Proof. Let $Comp(G) = G_{|_R}$ for some $R \subseteq [N]$. On the one hand, suppose $G \in \mathcal{B}(H)$. Recall that in this case there exists a blow-up mapping $\mu : [N] \to [t]$. Clearly, there exists a blow-up mapping from any induced subgraph of G to H, because μ restricted to these vertices is a blow-up mapping. Thus, in particular since $Comp(G)$ is an induced subgraph of G, then $\mu' = \mu_{|_R}$ is a blow-up mapping from $Comp(H)$ to H. Moreover, by Fact 2.1 the number of equivalence relation induced by $\sim$ is at most t. Thus, it follows that $Comp(G)$ is a blow-up of $H_{|_{\mu'(R)}}$ which is in $\mathcal{B}_t(H)$.

On the other hand, suppose $Comp(G) \in \mathcal{B}_t(H)$. Since R is a set of representatives of the equivalence classes induced by $\sim$, any vertex $v \in [N]$ has a representative $r \in R$ such that $\chi(v) = \chi(r)$. Let $V_r \stackrel{\text{def}}{=} \{\, v \in [N] \;:\; \chi(v) = \chi(r) \,\}$, then $\{V_r\}_{r \in R}$ is a partition of $[N]$. By hypothesis there exists a blow-up mapping $\pi : R \to [t]$. Next, extend π to the mapping $\mu : [N] \to [t]$ by letting for each vertex $v \in V_r$, $\mu(v) \stackrel{\text{def}}{=} \pi(r)$. Clearly μ is a blow-up mapping, showing that $G \in \mathcal{B}(H)$. ■

Running time The compression operation can be implemented by sorting the vertices by their signatures, identifying distinct signatures, and constructing the induced subgraph. Thus, compression can be implemented in time $O(NlogN + m^2)$, where m is the number of vertices in $Comp(G)$. Assuming the algorithm did not reject, that is $m \leq t$, finding a blow-up mapping from $Comp(G)$ to H can be done as in the naive case by checking each possible mapping from $[m]$ to $[t]$. However, since each vertex in $[m]$ has a distinct signature by Fact 2.1 we only have to check for injections. Implementing this costs time $O\left(\binom{t}{m} \cdot m!\right) = O(t!)$. Thus, the total running time for the suggested algorithm is $O(N \log N + t^2) + O(t!)$, implying that deciding whether $G \in \mathcal{B}(H)$ is in $\mathcal{P}$.

3.2 The algorithm

For the sake of future reference, we present the algorithm explicitly.

Algorithm 3.4. *(Efficient algorithm for deciding whether $G \in \mathcal{B}(H)$) Input: a graph $G = ([N], E)$.*

1. *Construct $Comp(G) = ([m], E_m)$. If $m > t$ then halt with answer "no".*
2. *Otherwise, exhaustively search for a injective blow-up mapping $\mu : [m] \to [t]$. Halt with answer "yes" if and only if such mapping exists.*

4 Adaptive tester

Now we move to the testing model. Recall, in this model, accessing G is by an oracle of the representation function e_G, where each query for e_G is charged. Since our adaptive tester is allowed to have a total number of $O(\epsilon^{-1})$ queries, the number of queries allowed is significantly less than $\binom{N}{2}$. Therefore, $Comp(G)$ cannot be constructed (where $Comp(G)$ is as in Definition 3.1).

Assume that the tester sample the set $S \subseteq [N]$. Then, if it happens that the knowledge matrix $\mathcal{K}$ contains the adjacency matrix $\mathcal{G}[S,S]$, that is, the adjacency matrix of the subgraph $G_{|S}$, then $Comp(G_{|S})$ can be constructed (as done previously but now we consider the graph $G_{|S}$ instead of G). Thus, it is still possible to construct compressed subgraphs of G. But what is the relation between $Comp(G)$ and $Comp(G_{|S})$? This is answered next.

Proposition 4.1. *Let $S \subseteq [N]$, then $Comp(G_{|S})$ is a subgraph of $Comp(G)$.*

Proof. Let R be the set of representatives of vertices in $Comp(G)$, that is, $Comp(G) = (R, E(R))$, and let $\mathcal{A} \overset{\text{def}}{=} \{A_r\}_{r \in R}$ be the equivalence classes induces by the $\sim$ relation on G. Let R' be the set of representatives of vertices in $Comp(G_{|S})$, that is, $Comp(G_{|S}) = G_{|R'} = (R', E(R'))$, and let $\mathcal{A}' \overset{\text{def}}{=} \{A'_{r'}\}_{r' \in R'}$ be the equivalence classes induces by the $\sim$ relation on $G_{|S}$. Since $\mathcal{A}$ is a partition of $[N]$, the function $\theta : R' \to R$ such that $\theta(r')$ is the representative of the equivalence class in $\mathcal{A}$ that r' belongs to is well defined. Observe that for $r' \in R'$ we have that $A_{\theta(r')} \subseteq A'_{r'}$, because the signature of r' in $G_{|S}$ is equivalent to the signature of $\theta(r')$ restricted to S. Note that by definition of the $\sim$ relation, it follows that for any $r'_1, r'_2 \in R'$ we have that $E \cap (A'_{r'_1} \times A'_{r'_2})$ is either $A'_{r'_1} \times A'_{r'_2}$ or the empty set. Putting it all together, we have that for any $r'_1, r'_2 \in R'$, it follows that $(r'_1, r'_2) \in E(R')$ if and only if $(\theta(r'_1), \theta(r'_2)) \in E(R)$, thus, $Comp(G_{|S})$ is a subgraph of $Comp(G)$. ■

4.1 The actual tester

Consider the following algorithm, denoted, *Adap*.

Algorithm 4.2. *(Adaptive tester for being a blow-up of H)*
Inputs: a natural number $N \in \mathbb{N}$, a proximity parameter $\epsilon \in (0,1]$, and oracle access to a graph $G = ([N], E)$.

Initialize $R \overset{def}{=} \{1\}$. *Select uniformly at random* $\Theta(t/\epsilon)$ *pairs of vertices from* $[N]^2$, *denoted* T. *For each pair* $(u,v) \in T$ *do the following:*

1. *Construct* $G_{|_{R\cup\{v,u\}}}$ *by querying the oracle for* (u,v) *as well as all pairs of vertices in* $R \times \{u,v\}$.
2. *Construct the compressed graph* $Comp(G_{|_{R\cup\{v,u\}}}) = G_{|_{R'}}$ *where* $R' \subseteq R\cup\{v,u\}$. *Consider the knowledge submatrix* $K' \overset{def}{=} G_{|_{R'}}$.
   - *(a) If* $|R'| > t$, *then halt with output "*NO*".*
   - *(b) Otherwise, check whether* K' *is* H*-mappable. If the answer is negative, then halt with output "*NO*". Otherwise, let* $R \leftarrow R'$, *and continue to the next pair.*

*If the algorithm has not halted so far, then halt with output "*YES*".*

4.2 Query complexity and running time

Query complexity The algorithm only queries the oracle in Step 1. Note that an invariant that always holds in this step is that $|R| \le t$. Therefore, the total query complexity is $(2t+1)\cdot\Theta(t/\epsilon) = O(t^2/\epsilon)$, which is $O(1/\epsilon)$, because t is a constant.

Running time There are two major time-consuming operations the tester performs for every pair (v,u), constructing the compressed graph $Comp(G_{|_{R\cup\{u,v\}}})$, and checking whether K' is H-mappable. The construction of the compressed graph $Comp(G_{|_{R\cup\{u,v\}}}) = (R', E')$ costs time $O(t\log t)$, because $|R| \le t$. The second operation, namely checking whether K' is H-mappable, can be done as in the decision algorithm, first check that $|R'| \le t$, otherwise reject. Then, try to find an H-mapping by exhaustive search, this takes time $O(t!)$. Therefore, the running time of the tester is $O(t/\epsilon)\cdot\big(O(t!) + O(t^2)\big)$, which is $O(1/\epsilon)$.

4.3 Correctness of algorithm

Next, we prove that the algorithm always accepts graphs that are blow-up of H and rejects (with high probability) graphs that are ϵ-far.

We start with the case that $G \in \mathcal{B}(H)$. Suppose $Comp(G_{|_S}) = (R', E')$ for $S \subseteq [N]$. Then, by Fact 2.9, $K' = \mathcal{K}[R', R']$ $(= \mathcal{G}[R', R'])$ is H-mappable. Therefore, Algorithm 4.2 accepts G with probability 1.

We thus focus on analyzing the behavior of Algorithm 4.2 on graphs that are ϵ-far from $\mathcal{B}(H)$; that is, assume that G is ϵ-far from $\mathcal{B}_N(H)$. Consider the set of representatives R, and let $\mathcal{P}(R)$ be the set of pairs in $[N]^2$ such that if $(u,v) \in \mathcal{P}(R)$, then the number of vertices of $Comp(G_{|_{R\cup\{u,v\}}})$ is strictly bigger than $|R|$. Recall that the definition of an H-mappable knowledge matrix applicable only on matrices having pairwise inconsistent rows. Observe that following the compression operation the matrix $K \overset{\text{def}}{=} \mathcal{G}[R,R] = G_{|_R}$ has pairwise inconsistent rows. Assume, for now, that if K is H-mappable, then the density of $\mathcal{P}(R)$ is at least $\frac{\epsilon}{2}N^2$. We prove this claim latter in Lemma 4.3. Note that Algorithm 5.2, performs Step 1 only if for the current R it holds that K is H-mappable. Thus, in each iteration of Step 1 we choose $(u,v) \in \mathcal{P}(R)$ with probability at least $\epsilon/2$, which causes R' in Step 2 to be larger than R. Thus, with probability at least $2/3$, Algorithm 5.2 rejects G, because in $O(t/\epsilon)$ steps we obtain a set R that is not H-mappable. All that is left to show is the following.

Lemma 4.3. *Let $K = \mathcal{G}[R,R]$ for $R \subseteq [N]$ such that K has pairwise inconsistent rows and is H-mappable. Let $\mathcal{P}(R) \subseteq [N]^2$ such that $\mathcal{P}(R) \overset{def}{=} \{(u,v) : |R| < |R'|\}$ where R' is the set of vertices of $Comp(G_{|_{R\cup\{u,v\}}})$. Then $|\mathcal{P}(R)| \geq \frac{\epsilon}{2}N^2$.*

Proof. By our hypothesis K is H-mappable. Thus, we can construct a candidate blow-up mapping $\mu : [N] \to [t]$. Since G is ϵ-far from $\mathcal{B}(H)$, $Viol_\mu \geq \epsilon N^2$. Assume that $R = \{r_1, \ldots, r_n\}$, and let $P(K) = (V_1, \ldots, V_n, L)$ be the partition induced by K, and let $V \overset{\text{def}}{=} \cup_{i\in[n]} V_i$. Then, there are two cases.

1. $|L| \geq \frac{\epsilon}{2}N$. Define, $\mathcal{D}(R) \overset{\text{def}}{=} L \times [N]$. For any $(u,v) \in \mathcal{D}(R)$ it follows that $|R'| > |R|$, because $\chi_R(v) \neq \chi_R(r_i)$ for $i \in [n]$. Therefore, $\mathcal{D}(R) \subseteq \mathcal{P}(R)$, so $|\mathcal{P}(R)| \geq |\mathcal{D}(R)| \geq \frac{\epsilon}{2}N^2$.
2. $|L| < \frac{\epsilon}{2}N$. Define, $\mathcal{E}(R) \overset{\text{def}}{=} Viol_\mu(V,V)$. Since $|L \times [N]| < \frac{\epsilon}{2}N^2$, it follows that $\mathcal{E}(R) = Viol_\mu(V,V) > \frac{\epsilon}{2}N^2$. Let $(u,v) \in \mathcal{E}(R)$, then $(u,v) \in V_i \times V_j$ for some $i,j \in [n]$. On one hand, $u \in V_i$, thus, $\chi_R(u) = \chi_R(r_i)$, so

$$\begin{aligned}
g_{u,r_j} &= g_{r_i,r_j} && \chi_R(u) = \chi_R(r_i) \\
&= k_{r_i,r_j} && K \text{ is the adjacency matrix of } G_{|_R} \\
&= h_{\mu(r_i),\mu(r_j)} && R \text{ is } H\text{-mappable, } k_{r_i,r_j} \approx h_{\mu(r_i),\mu(r_j)} \\
&= h_{\mu(u),\mu(v)} && \text{By definition of } \mu,\ \mu(u) = \mu(r_j), \text{ and } \mu(v) = \mu(r_j) \\
&\neq g_{u,v} && (u,v) \in \mathcal{E}(R)
\end{aligned}$$

On the other hand, for $i \neq j$, we have that

$$\begin{array}{cc} \chi_R(v) \neq \chi_R(r_i) & v \in V_j \\ \Downarrow & \\ \chi_{R\cup\{u\}}(v) \neq \chi_{R\cup\{u\}}(r_i) & \text{for } i \in [n] \setminus \{j\} \end{array}$$

Also, if $i = j$, then $\chi_{R\cup\{u\}}(v) \neq \chi_{R\cup\{u\}}(r_i)$, because $g_{u,r_j} \neq g_{u,v}$. Therefore, we conclude that $\chi_{R\cup\{u\}}(v) \neq \chi_{R\cup\{u\}}(r_i)$ for every for $i \in [n]$. Hence the signature of v in the adjacency matrix $\mathcal{G}[R \cup \{u\}, R \cup \{u\}] = G_{|_{R\cup\{u\}}}$ is different from all signatures of vertices in R (which are distinct because the rows in K are pairwise inconsistent), and so $R \cup \{v\} \subseteq R'$. It follows, as in the previous case, $|R'| > |R|$, and $\mathcal{E}(R) \subseteq \mathcal{P}(R)$, thus, $|\mathcal{P}(R)| \geq |\mathcal{E}(R)| \geq \frac{\epsilon}{2}N^2$.

This completes the proof of this lemma. ■

5 Non-adaptive tester

In this section we construct a non-adaptive tester for the property of being a blow-up with query complexity $\tilde{O}(\epsilon^{-1})$, and running time is polynomial in ϵ^{-1}. Considering Algorithm 3.4, note that this algorithm inherently uses the adaptiveness; it uses it to construct certain induced subgraphs of G and then check them. However, a non-adaptive tester cannot do so; it must choose the set of quires in advance. Restricting the non-adaptive tester to consider only induced subgraphs is of no use, as such it is equivalent to a canonical tester that samples at most $\tilde{O}(\epsilon^{-1/2})$ vertices. But as Proposition 6.2 in [GR08] states this is not enough (see also Proposition 2.4), because any canonical tester must sample at least $\Omega(\epsilon^{-1})$ vertices. Therefore, during non-adaptive testing, the knowledge matrix that represents the queries made does not necessarily have a rectangular structure, specifically, the boolean entries in it might be very 'scattered'. We next try to find out if the notion of compressed graph can be extended such that it will be applicable to arbitrary knowledge matrices, as we face in this case, as well.

Extending the definition of Comp() operation Recall that in previous cases the graph $Comp(G_{|_S})$, for some $S \subseteq [N]$, was defined as the subgraph induced by an arbitrary selection of representatives in the equivalence classes induced by $\sim$ (see Definition 3.1). Thus, the relation $\sim$ is the heart of the compression operation.

Naive attempt Let $K^T \overset{\text{def}}{=} \mathcal{K}[T,T]$, for some $T \subseteq [N]$, be an arbitrary knowledge matrix. Then, a naive attempt is to extend naturally the relation $\sim$ as follows. For $u, v \in T$, $u \sim v$ if and only if rows u and v in the knowledge submatrix K^T are consistent, that is, $\mathcal{K}[\{u\},T] \approx \mathcal{K}[\{v\},T]$. However, this definition fails, because while $\sim$ is as equivalence relation if K^T is a boolean matrix (i.e., representation of a subgraph of G), it is not necessarily an equivalent relation for arbitrary knowledge matrix, as the following example shows. Consider the knowledge submatrix having the following rows $K[\{1\},\{1,2,3\}] = (0,0,*)$, $K[\{2\},\{1,2,3\}] = (*,0,*)$, and $K[\{3\},\{1,2,3\}] = (1,*,0)$. Clearly, $\sim$ is not an equivalence relation on this matrix, thus, we do not have equivalence classes.

Corrected approach Looking deeper in previous proofs, is it evident that all we need is to find a (new) matrix with pairwise inconsistent rows. This matrix allowed us to reject G in case it was not H-mappable, otherwise we could build a candidate blow-up mapping and to proceed along. Recall that $\mathcal{K}$ is the knowledge matrix during the test. Then following this intuition we suggest the following extension of definition; $Comp(K^T)$ will be the set consisting of all knowledge submatrices of $\mathcal{K}$ that have pairwise inconsistent rows. Formally,

Definition 5.1 (Compression operation). Let $K^T = \mathcal{K}[T,T]$, for some $T \subseteq [N]$, be an arbitrary knowledge submatrix of $\mathcal{K}$. Then,

- Let $I(K^T)$ denote the set of all subsets of pairwise inconsistent rows of K^T; that is $R \subseteq T$ is in $I(K^T)$ if and only if R is a set of pairwise inconsistent rows of $\mathcal{K}[R,T]$ (i.e., for any $r_1, r_2 \in R$ it holds that $\mathcal{K}[\{r_1\},T] \not\approx \mathcal{K}[\{r_2\},T]$).
- Define $Comp(K^T) \overset{\text{def}}{=} \{\, \mathcal{K}[R,T] \;:\; R \in I(K^T) \,\}$.

Note that in this case the result of $Comp(K^T)$ is a set of knowledge matrices. Observe that in Definition 3.1 we considered only the maximal element in the set of all subgraphs having pairwise inconsistent rows.

5.1 The actual tester

We start by presenting the algorithm.

Algorithm 5.2. *(Non-adaptive tester for being a blow-up of H)*

Inputs: a natural number $N \in \mathbb{N}$, a proximity parameter $\epsilon \in (0,1]$, and oracle access to a graph $G = ([N],E)$. The tester proceeds as follows.

1. *Set* $\ell = \log(1/\epsilon) + O(\log\log(1/\epsilon))$*.*
2. *For each* $i \in [[\ell]]$ *select, uniformly at random,* $\text{poly}(\ell)\cdot 2^i$ *vertices from* $[N]$*, denoted* T_i*. Let* $T \overset{def}{=} \cup_{i\in[[\ell]]} T_i$*.*
3. *For each* $i, j \in [[\ell]]$ *such that* $i + j \leq \ell$*, query each pair in* $T_i \times T_j$ *.*
4. *Accept if and only if the knowledge submatrix* $K^T \overset{def}{=} \mathcal{K}[T, T]$ *is* H*-mappable.*

Recall that $k_{u,v} = g_{u,v}$ if the pair (u, v) was queried, otherwise, $k'_{u,v} = *$. Step 4 is implemented by first computing the set $Comp(K^T)$, which in this case is the set consisting of all knowledge submatrices having pairwise inconsistent rows. Then, accepts if and only if every such matrix, i.e., element of $Comp(K^T)$), is H-mappable.

5.2 Query complexity and running time

Query complexity The total amount of queries made is

$$\begin{aligned}\sum_{i+j\leq\ell} |T_i \times T_j| &\leq \sum_{\ell'\leq\ell} \sum_{i+j=\ell'} |T_i \times T_j| \\ &= \sum_{\ell'\leq\ell} \text{poly}(\ell) 2^{\ell'} \\ &= \text{poly}(\ell)\cdot 2^{\ell} \\ &= \tilde{O}(\epsilon^{-1}).\end{aligned}$$

Running time Note that the dominant computation is done in Step 4; that is, constructing $Comp(K^T)$ and checking whether each member of it (knowledge matrix) is H-mappable. The first step, namely construction $Comp(K^T)$, can be done as follows. First try to find knowledge matrices having exactly $t+1$ pairwise inconsistent rows. If such matrix exists then the tester rejects because it is not H-mappable. Otherwise, find all knowledge matrix having at most t pairwise inconsistent row, and for each one check whether it is H-mappable. Note that finding $k \in [|T|]$ pairwise inconsistent rows in K^T can be done in the following way. Construct an auxiliary graph denoted as $Q(K^T) = (T, E_T)$ such that $(u, v) \in T$ if and only if the rows u and v are pairwise inconsistent in K^T. Then, finding all k pairwise inconsistent rows in K^T is equivalent to tracking all the cliques of size k in $Q(K^T)$. Constructing $Q(K^T)$ costs $|T|^2$, while tracking each clique of size k costs $\binom{|T|}{k}$. Therefore the running time of the tester is

$$|T|^2 + \binom{|T|}{t+1} + \sum_{i=1}^{t} \binom{|T|}{i} \cdot t! \leq |T|^{t+1} \cdot (t+1)!,$$

assuming $t \geq 1$. Since $|T| = \tilde{O}(\epsilon^{-1})$ and t is a constant, the tester has running time polynomial in the reciprocal of ϵ.

5.3 Correctness of algorithm

If $G \in \mathcal{B}(H)$, then clearly each knowledge submatrix of G is H-mappable. Therefore, each element in $Comp(K^T)$ is H-mappable as well. Hence, Algorithm 5.2 accepts G with probability 1. The rest of the analysis refers to the case of graphs that are ϵ-far from $\mathcal{B}(H)$. We fix such a graph G for the rest of the analysis.

5.3.1 Preliminaries

Sampling T For the rest of this section we think of sampling the sets $T_0, T_1, \ldots, T_\ell$ as being done in the following way. Let $p(\ell) \stackrel{\text{def}}{=} 2 \cdot (\ell+1)^t + 1$, then each set T_i is sampled in $p(\ell)$ blocks, denoted $T_i^1, \ldots, T_i^{p(\ell)}$, each of size $p'(\ell) \cdot 2^i$ where $p'(\ell)$ is a polynomial in ℓ. Then, define the sets $\mathcal{T}^1, \ldots, \mathcal{T}^{p(\ell)}$ such that $\mathcal{T}^j \stackrel{\text{def}}{=} \cup_{i \in [[\ell]]} T_i^j$ for $j \in [p(\ell)]$. Clearly, sampling $T_0, T_1, \ldots, T_\ell$ is equivalent to sampling the sets $\mathcal{T}^1, \ldots, \mathcal{T}^{p(\ell)}$. Thus, from now on we think of sampling sets $T_1, \ldots, T_\ell$ as sampling $\mathcal{T}^1, \ldots, \mathcal{T}^{p(\ell)}$ instead. For the rest of this section let $j \in [p(\ell)]$, and let $\mathcal{T}^{[j]} \stackrel{\text{def}}{=} \cup_{i \in [j]} \mathcal{T}^i$.

Basic pair Let $C \subseteq \mathcal{T}^{[j]}$, and $R \subseteq C$, then the pair (R, C) is called a j-basic pair. **For the rest of the section assume that (R, C) is a j-basic pair, and $R = \{r_1, \ldots, r_n\}$.**

Remark 5.3. *Whenever we consider a j-basic pair we assume that the sets $\mathcal{T}^1, \ldots, \mathcal{T}^j$ are fixed, that is, they have been sampled already, but the remaining sets $\mathcal{T}^{j+1}, \ldots, \mathcal{T}^{p(\ell)}$ haven't been sampled yet.*

Index function The j-index function, denoted $\mathcal{I}^{[j]}$, is the function from $\mathcal{T}^{[j]}$ to the natural numbers (including zero) in $[[\ell]]$, that is, $\mathcal{I}^{[j]} : \mathcal{T}^{[j]} \rightarrow [[\ell]]$ such that $\mathcal{I}^{[j]}(u)$ is the (subscript) index of the set in $\left\{T_i^j\right\}_{j \in [p(\ell)], i \in [[\ell]]}$ that contains u. We stress that for large N the probability that we hit a vertex u twice or more, is negligible, so we disregard this event. By this assumption it follows that $\mathcal{I}^{[j+1]}(u) = \mathcal{I}^{[j]}(u)$ for every $u \in \mathcal{T}^{[j]}$. For shorthand, we call the value $\mathcal{I}^{[j]}(u)$ the index of u.

We "abuse" this notation and denote by $\mathcal{I}^{[j]}(R)$ the sequence of length $|R|$ defined by ordering in non-increasing order the index value of $r \in R$, that is, $\mathcal{I}^{[j]}(R) \stackrel{\text{def}}{=} (\mathcal{I}^{[j]}(r_{\sigma(1)}), \ldots, \mathcal{I}^{[j]}(r_{\sigma(n)}))$ where $\sigma \in S_n$, and $\mathcal{I}^{[j]}(r_{\sigma(1)}) \geq \cdots \geq \mathcal{I}^{[j]}(r_{\sigma(n)})$.

Remark 5.4. *For the rest of this section let $\succ$ be the lexicographic order on varied length sequences of integers defined as follows. Let $\mathbf{a} = (a_1, \ldots, a_m)$ and $\mathbf{b} = (b_1, \ldots, b_m)$ where $m \in \mathbb{N}$ be sequences of integer numbers. Then, we say that $a \succ b$ if and only if the first element a_i that differs from the corresponding b_i is greater than b_i (for some $i \in [m]$). For two sequences with different length, pad the shorter one with -1's, and then use the previous definition. For example, $(a_1, \ldots, a_{m-1}) \succ (b_1, \ldots, b_m)$ if and only if $(a_1, \ldots, a_{m-1}, -1) \succ (b_1, \ldots, b_m)$.*

Generated knowledge submatrix We define the j-generated knowledge submatrix, denoted $\mathcal{K}^{[j]}[R, C]$ to be the $|R| \times |C|$ knowledge submatrix such that

$$k_{r,c} \stackrel{\text{def}}{=} \begin{cases} g_{r,c} & \mathcal{I}^{[j]}(r) + \mathcal{I}^{[j]}(c) \leq \ell \\ * & \text{otherwise} \end{cases}.$$

Observe that the index value of the vertices determines the structure of the generated knowledge submatrix. From now on, we think about the rows R and columns C of K as being ordered by their index value in increasing order. Therefore, for $r \in R$ we have that $K[\{r\}, C] \in \cup_{k \in [|C|]} \{0,1\}^k \times \{*\}^{|C|-k}$.

Remark 5.5. *Whenever we say that a function $e(\ell)$ is negligible in ℓ we mean that $e(\ell) = O(\exp(-\operatorname{poly}(\ell)))$; that is, $e(\ell)$ decreases exponentially in ℓ.*

It will be convenient to use the logarithm of the reciprocal of the density of sets rather than the density itself.

Definition 5.6 (Logarithm of the reciprocal of density)**.** Define $\varphi : 2^{[N]} \to \mathbb{N}$ such that $\varphi(U) \stackrel{\text{def}}{=} \lceil \log \frac{N}{|U|} \rceil$.

Note that for $W \subseteq U \subseteq [N]$, we have that $\varphi(W) \geq \varphi(U)$. The following proposition is a simple consequence of the definition of φ. The following proposition is a trivial fact.

Proposition 5.7. *Let $U \subseteq [N]$, and let T' be a uniformly selected subset of $[N]$ such that $|T'| = \text{poly}(\ell) \cdot 2^{\varphi(U)}$. Then, with probability $1 - e(\ell)$ such that $e(\ell)$ is negligible in ℓ, there is a vertex $u \in T' \cap U$.*

Proof. First note that,

$$\frac{|U|}{N} = 2^{-\log \frac{N}{|U|}} \geq 2^{-\lceil \log \frac{N}{|U|} \rceil} = 2^{-\varphi(U)}.$$

Thus, the probability that the sample T' of size $\text{poly}(\ell) \cdot 2^{-\varphi(U)}$ misses U is exponentially decreasing in $\text{poly}(\ell)$. ■

Nice pair During this work we will consider a special kind of j-basic pair called j-nice pair.

Definition 5.8 (Nice pair)**.** Let (R, C) be a j-basic pair, and let $K \overset{\text{def}}{=} \mathcal{K}^{[j]}[C, R]$ be the generated matrix. We say that the pair (R, C) is a j-nice pair if the following hold:

1. For $r \in R$, we have that $\mathcal{I}^{[j]}(r) \leq \varphi(V_r(K))$, where V_r as in Eq. (2.1).
2. R are pairwise inconsistent rows in K (see Definition 2.8).

5.3.2 The Increasing Lemma

The analysis of Algorithm 5.2 (when applied to G that is ϵ-far from $\mathcal{B}(H)$) is based on the following "Increasing Lemma".

Lemma 5.9 (Increasing Lemma)**.** *Let (R, C) be a j-nice pair. If $\mathcal{K}[R, C]$ is H-mappable, then with probability at least $1 - e(\ell)$ where $e(\ell)$ is negligible in ℓ, over the choice of $\mathcal{T}^{j+1}$, there is a $(j+1)$-nice pair (R', C') such that $\mathcal{I}^{[j+1]}(R') \succ \mathcal{I}^{[j]}(R)$.*

Recall that $R' \subseteq C' \subseteq \mathcal{T}^{[j+1]}$, which explains the role of $\mathcal{T}^{j+1}$ in the statement. We will first show that the correctness of Algorithm 5.2 follows from Lemma 5.9, and later prove this lemma.

Algorithm 5.2 rejects G with constant probability Suppose that $\mathcal{T}^1$ was sampled. Then, it contains a (trivial) 1-nice pair; specifically, consider any $u \in T_0^1$, and the pair $(R_1, C_1) \overset{\text{def}}{=} (\{u\}, \{u\})$. The 1-by-1 matrix $K = \mathcal{K}[R_1, C_1]$, has pairwise inconsistent rows. Moreover, $\varphi(V_u(K)) \geq 0$ and $\mathcal{I}^{[1]}(u) = 0$, so, $\mathcal{I}^{[1]}(u) \leq \varphi(V_u(K))$. Thus, the pair (R_1, C_1) is

1-nice pair. Assume, for now, that any application of Lemma 5.9 produces a nice pair. Next, for any $j > 1$, let (R_{j-1}, C_{j-1}) be a $(j-1)$-nice pair. If this pair is not H-mappable, then the algorithm rejects and we are done. Otherwise, let (R_j, C_j) be the result of applying the "Increasing Lemma" on (R_{j-1}, C_{j-1}). By this lemma we have that $\mathcal{I}^{[1]}(R_1) \prec \mathcal{I}^{[2]}(R_2) \prec \dots$. Observe that the numbers of sequences containing elements in $[[\ell]]$ of length at most t is $\sum_{k\in[t]}(\ell+1)^k \leq 2\cdot(\ell+1)^t < p(\ell)$. Therefore, after applying Lemma 5.9 for $p(\ell)$ times (assuming the algorithm had not rejected already), it hods that $\mathcal{I}^{[p(\ell)]}(R_{p(\ell)}) \notin \cup_{k\in[t]}[[\ell]]^k$ so $K[R_{p(\ell)}, C_{p(\ell)}]$ is not H-mappable, because $\left|R_{p(\ell)}\right| > t$, so the algorithm rejects. Since we apply Lemma 5.9 at most $p(\ell)$ times, the probability that some application will not result a nice pair is negligible in ℓ (by a union bound).

5.3.3 Proof of Increasing Lemma

We start by defining three operations on basic pairs. Then, we discuss under which conditions each of these operations preserves the niceness property. Then, we prove Lemma 5.11 and Lemma 5.13, which are the main ingredients of proof of the Increasing lemma. Lastly, we prove the Increasing lemma.

Operation on basic pairs For the rest of this section let (R, C) be a j-basic pair, $K \overset{\text{def}}{=} \mathcal{K}^{[j]}[R, C]$ be the corresponding generated matrix, $r \in R$, and $u \in \mathcal{T}^{j+1}$, that is, $\mathcal{I}^{[j+1]}(u)$ is well defined. Then, we define three operations on (R, C). Their names are given based on the effect each one has on the corresponding generated knowledge submatrix.

- **Adding column** This operation simply adds u to the set C. Consider the matrix $K' \overset{\text{def}}{=} \mathcal{K}^{[j+1]}[R, C \cup \{u\}]$. Effectively, K' is K with added column u where entries in column u are determined by the value of $\mathcal{I}^{[j+1]}(u)$ and the index values of vertices in R.
- **Adding row and column** This operation adds u to both sets R and C. Consider the matrix $K' \overset{\text{def}}{=} \mathcal{K}^{[j+1]}[R \cup \{u\}, C \cup \{u\}]$. Effectively K' is K with added row u and column u, where the values in row u and column u are determined by the value of $\mathcal{I}^{[j]}(u)$ and the index values of vertices in R and C.
- **Truncating rows** The truncating rows operator, ϑ_r, takes the elements in R with index value greater or equal to $\mathcal{I}^{[j]}(r)$ and omits the rest. Formally, let $\vartheta_r(R, C) \overset{\text{def}}{=} (R', C')$ such that $R' \overset{\text{def}}{=} \left\{ v \in R \,:\, \mathcal{I}^{[j]}(v) \geq \mathcal{I}^{[j]}(r) \right\}$ and $C' \overset{\text{def}}{=} C$; that is, R' consists

of vertices with index value greater equal to the index value of r. If we think of rows of K (the set R) as being ordered in increasing order by their index value, then, effectively, this operator removes the rows before row r.

Next, we prove that the "adding column" operator preserves the niceness of a pair, that is, if we add column to a j-nice pair, then the result is a $(j+1)$-nice pair.

Proposition 5.10 (adding a column). *$(R, C \cup \{u\})$ is a $(j+1)$-nice pair.*

Proof. Let $(R', C') \overset{\text{def}}{=} (R, C \cup \{u\})$, and $K' \overset{\text{def}}{=} \mathcal{K}^{[j+1]}[R', C']$. Since K' is the matrix K with added column u, the matrix K' has pairwise inconsistent rows (recall $R' = R$). Next we turn to condition 2 in Definition 5.8. Let $r \in R$ $(\subseteq \mathcal{T}^{[j]})$. Then it follows that $V_r(K') \subseteq V_r(K)$, because $C \subseteq C'$. So,

$$\begin{aligned} \mathcal{I}^{[j+1]}(r) &= \mathcal{I}^{[j]}(r) && \text{Definition of } \mathcal{I} \\ &\le \varphi(V_r(K)) && (R,C) \text{ is a } j\text{-nice pair} \\ &\le \varphi(V_r(K')) && V_r(K') \subseteq V_r(K). \end{aligned}$$

Therefore, $\mathcal{I}^{[j+1]}(r) \le \varphi(V_r(K'))$, and (R', C') is a $(j+1)$-nice pair. ■

Clearly, the "truncating rows" operator preserves the niceness property; that is, if we apply it on a j-nice pair, then the result is a j-nice pair as well.

The following lemma states that if it happens that $L(K)$ (see Eq. 2.2) is big enough, then with high probability, over sampling $\mathcal{T}^{j+1}$, there exists a vertex $u \in \mathcal{T}^{j+1} \cup L(K)$. Moreover, if we add u into the rows and columns and then apply the corresponding "truncating row" operator (that is ϑ_u), the result is a $(j+1)$-nice pair.

Lemma 5.11 (truncating a row). *Suppose that $\varphi(L(K)) \le \ell$. Then, with probability $1 - e(\ell)$ where $e(\ell)$ is negligible in ℓ, over sampling $\mathcal{T}^{j+1}$, there exists $u \in \mathcal{T}^{j+1} \cap L(K)$ such that $\vartheta_u(R \cup \{u\}, C \cup \{u\})$ is a $(j+1)$-nice pair.*

Proof. Let $s \overset{\text{def}}{=} \varphi(L(K))$, then $s \le \ell$ and $|L(K)| \ge \epsilon N$. Using Proposition 5.7 it follows that

$$\Pr_{\mathcal{T}^{j+1}}[\mathcal{T}^{j+1} \cap L(K) \neq \emptyset] \ge \Pr_{\mathcal{T}_s^{j+1}}[\mathcal{T}_s^{j+1} \cap L(K) \neq \emptyset] > 1 - e(\ell).$$

Assume that this is indeed that case, and let $u \in \mathcal{T}^{j+1} \cap L(K)$. Let $(R', C') \overset{\text{def}}{=} \vartheta_u(R \cup \{u\}, C \cup \{u\})$ and $K' \overset{\text{def}}{=} \mathcal{K}^{[j+1]}[C', R']$. By Proposition 5.10, $(R, C \cup \{u\})$ is a $(j+1)$-nice pair. Moreover,

$$\begin{aligned}
\mathcal{I}^{[j+1]}(u) &= \varphi(L(K)) && u \in T_s^{j+1}, \text{ and } s = \varphi(L(K)) \\
&\le \varphi(V_u(K')) && V_u(K') \subseteq L(K)
\end{aligned}$$

Therefore, we conclude that for $r \in R \cup \{u\}$, it holds that $\mathcal{I}^{[j+1]}(r) \le \varphi(V_r(K'))$. So, Requirement 1 in Definition 5.8 follows, since $R' \subseteq R \cup \{u\}$. Next, it is left to show that row u in the matrix K' is inconsistent with each row $r \in R \cap R'$; that is, $K'[\{r\}, C'] \not\approx K'[\{u\}, C']$. Let $r \in R \cap R'$

$$\begin{aligned}
&\chi_C(u) \not\approx K[\{r\}, C] && \text{for } r \in R, \text{ because } u \in L(K) \\
&\Downarrow \\
&k_{r,c} \not\approx \chi_{\{c\}}(u) && \text{for some } c \in C \\
&\quad = g_{u,c} && \text{by definition of a signature} \\
&\Downarrow \\
&k_{r,c} \ne * && \text{by definition of } \approx \\
&\Downarrow \\
&\mathcal{I}^{[j]}(c) + \mathcal{I}^{[j]}(r) \le \ell && \text{by definition of } K \\
&\Downarrow \\
&\mathcal{I}^{[j+1]}(c) + \mathcal{I}^{[j+1]}(r) \le \ell && \text{by definition of } \mathcal{I}
\end{aligned}$$

Recall that for every $r \in R'$ it follows that $\mathcal{I}^{[j]}(u) \le \mathcal{I}^{[j]}(r)$, so

$$\begin{aligned}
&\mathcal{I}^{[j+1]}(c) + \mathcal{I}^{[j+1]}(u) \le \ell && \text{for } r \in R \cap R' \\
&\Downarrow \\
&k'_{u,c} = g_{u,c} && \text{by definition of } K' \\
&\quad \not\approx k_{r,c} && \text{see above} \\
&\quad = k'_{r,c} && K \text{ is a submatrix of } K'
\end{aligned}$$

Therefore, $k'_{u,c} \not\approx k'_{r,c}$, so $K'[\{r\}, C'] \not\approx K'[\{u\}, C']$. ■

Proposition 5.12 (truncating a row "increases order"). *For $u \in \mathcal{T}^{j+1} \cap L(K)$, it holds that $(R', C') \stackrel{def}{=} \vartheta_u(R \cup \{u\}, C \cup \{u\})$ satisfies $\mathcal{I}^{[j+1]}(R') \succ \mathcal{I}^{[j+1]}(R)$.*

Proof. Suppose that $m \stackrel{\text{def}}{=} |R|$, and let $\mathcal{I}^{[j+1]}(R) = (n_1, \ldots, n_m)$. Let $n' \stackrel{\text{def}}{=} \mathcal{I}^{[j+1]}(u)$ and $\mathcal{I}^{[j+1]}(R') = (n_1, \ldots, n_k, n')$, for some $k \in [m]$ (recall $n_1 \leq \cdots \leq n_k \leq n'$). Now, if $k = m$ the proposition follows by the definition of $\succ$. Otherwise, $1 \leq k < m$, but in this case (by definition of ϑ_u) we have that $n_{k+1} < n'$, so the proposition follows. ■

The following lemma is the final ingredient in the proof of the Increasing Lemma. Suppose that (R, C) is a j-nice pair and that $K = \mathcal{K}^{[j]}[R, C]$ is H-mappable. Then, the lemma states that if the size of $L(K)$ is small, then with high probability the sample $\mathcal{T}^{j+1}$ contains a vertex w such that $L(K')$ is large, where $K' \stackrel{\text{def}}{=} \mathcal{K}^{[j+1]}[R, C \cup \{w\}]$. Recall that $C \subseteq \mathcal{T}^{[j]}$, $R \subseteq R$ and $R \stackrel{\text{def}}{=} \{r_1, \ldots, r_n\}$. For shorthand let $V_i \stackrel{\text{def}}{=} V_{r_i}(K)$ for $i \in [n]$, and $L \stackrel{\text{def}}{=} L(K)$.

Lemma 5.13. *Suppose that $\varphi(L) > \ell$, and R is H-mappable. Then, there exists index $m \in [[\ell]]$ and with probability $1 - e(\ell)$, where $e(\ell)$ is negligible in ℓ, there is $w \in T^{j+1}_{\ell - m}$ such that $\varphi(L(K^w)) \leq m$, where $K^w \stackrel{def}{=} \mathcal{K}^{[j+1]}[(R, C \cup \{w\})]$.*

Proof. Using the fact that K is H-mappable and that (R, C) is a j-nice pair, there exists a candidate blow-up mapping μ. Since G is ϵ-far from any $\mathcal{B}(H)$, the number of violating pairs in $Viol_\mu$ is at least ϵN^2. Define $V \stackrel{\text{def}}{=} \cup_{i \in [n]} V_i$, and note that (V, L) is a partition of $[N]$. Because $\varphi(L) > \ell$, thus $|L| \leq \frac{\epsilon}{2} N$. Thus, the number of violating pairs in $Viol_\mu([N], L) \leq \frac{\epsilon}{2} N^2$, so the number of violating pairs in $Viol_\mu(V, V) \geq \frac{\epsilon}{2} N^2$. One can readily show that there exist indices $i, j \in [n]$ and $m \in [[\ell]]$, a set $W \subseteq V_j$ with $\varphi(W) \leq \ell - m$ such that for each $w \in W$ there exists a set $U^w \subseteq V_i$ with $\varphi(U^w) \leq m$, and $\{w\} \times U^w \subseteq Viol_\mu$. A proof of the latter assertion appears in appendix B.1.

Let $s \stackrel{\text{def}}{=} \varphi(W)$, then with probability $1 - e(\ell)$ such that $e(\ell)$ is negligible in ℓ, there is $w \in W \cap T^{j+1}_s$, hence, $\mathcal{I}^{[j+1]}(w) \leq \ell - m$. By Proposition 5.10, it follows that $(R, C \cup \{w\})$ is a $(j+1)$-nice pair. For shorthand let $V^w_i \stackrel{\text{def}}{=} V_{r_i}(K^w)$ for $i \in [n]$, and $L^w \stackrel{\text{def}}{=} L(K^w)$ where $K^w \stackrel{\text{def}}{=} \mathcal{K}^{[j+1]}[R, C \cup \{w\}]$. Thus, $(V^w_1, \ldots, V^w_n, L^w)$ is a partition of $[N]$. Our goal is to prove that $\varphi(L^w) \leq m$. We shall achieve this by proving that $U^w \subseteq L^w$. We start by considering

r_i and r_j.

$$\begin{aligned}
\mathcal{I}^{[j]}(r_j) &\le \varphi(V_j) && (R,C) \text{ is a } j\text{-nice pair} \\
&\le \varphi(W) && W \subseteq V_j \\
&\le \ell - m && \text{Proposition B.1} \\
\mathcal{I}^{[j]}(r_i) &\le \varphi(V_i) && (R,C) \text{ is a } j\text{-nice pair} \\
&\le \varphi(U^w) && U^w \subseteq V_i \\
&\le m && \text{Proposition B.1}
\end{aligned}$$

Combining both bounds we get $\mathcal{I}^{[j]}(r_i) + \mathcal{I}^{[j]}(r_j) \le \ell$, which implies, $k_{r_i,r_j} = g_{r_i,r_j}$ (by definition of K). Turning to w itself, we have

$$\begin{aligned}
&\chi_C(w) \approx K[\{r_j\}, C] && w \in V_j \\
&\quad\Downarrow \\
&g_{w,r_i} \approx k_{r_j,r_i} \\
&\quad\Downarrow \\
&g_{w,r_i} = g_{r_j,r_i} && r_i \in R\ (\subseteq C),\ \mathcal{K} \text{ is a symmetric, and } k_{r_i,r_j} = g_{r_i,r_j} \qquad (5.1)
\end{aligned}$$

Therefore, $g_{w,r_i} = g_{r_j,r_i}$. Recalling that μ is a candidate blow-up mapping (based on the fact that K is H-mappable), for each $r, r' \in R$ it holds that $k_{r,r'} \approx h_{\mu(r),\mu(r')}$ (in the adjacency matrix representing H). Note that a key point is that for all $v' \in V$ the value of $\mu(v')$ is not effected by the arbitrary assignment of L. Thus,

$$\begin{aligned}
g_{w,r_i} &= g_{r_j,r_i} && \text{Eq. (5.1)} \\
&= k_{r_j,r_i} && \text{see above} \\
&\approx h_{\mu(r_j),\mu(r_i)} && \mu \text{ restricted to } R \text{ is an } H\text{-mapping} \\
&= h_{\mu(w),\mu(r_i)} && w \in V_j, \text{ so } \mu(w) = \mu(r_j) \text{ (by definition of } \mu) \\
&\Downarrow \\
g_{w,r_i} &= h_{\mu(w),\mu(r_i)} && (5.2)
\end{aligned}$$

Now, consider an arbitrary $u \in U^w$. Then,

$$\begin{aligned}
g_{w,u} &\neq h_{\mu(w),\mu(u)} && \{w\} \times U^w \subseteq Viol_\mu \\
&= h_{\mu(w),\mu(r_i)} && u \in V_i, \text{ so } \mu(u) = \mu(r_i) \text{ (by definition of } \mu) \\
&= g_{w,r_i} && \text{by Eq. (5.2)} \\
&= k^w_{w,r_i} && \mathcal{I}^{[j+1]}(w) + \mathcal{I}^{[j+1]}(r_i) \leq \ell \\
&\Downarrow \\
g_{w,u} &\not\approx k^w_{w,r_i} && \qquad (5.3)
\end{aligned}$$

Next, we prove that $u \in L(K^w)$.

$$\begin{aligned}
\chi_C(u) &\not\approx K[\{r\}, C] && \text{for every } r \in R \backslash \{r_i\}, \text{ becuase } u \in V_i \\
&\Downarrow \\
\chi_{C \cup \{w\}}(u) &\not\approx K[\{r\}, C \cup \{w\}] && \qquad (5.4)
\end{aligned}$$

Combining Eq. (5.3) and Eq. (5.4) we get that $\chi_{C\cup\{w\}}(u) \not\approx K[\{r\}, C \cup \{w\}]$, for every $r \in R$ which means that $u \in L^w$. Thus, $U^w \subseteq L^w$, and the lemma follows. ■

We are now ready to state and proof the Increasing lemma.

Proof of Increasing Lemma. Consider the following possible two cases regarding $K = \mathcal{K}^{[j]}[R, C]$.

- $\varphi(L(K)) \leq \ell$. Applying Lemma 5.11, with probability $1 - e(\ell)$ where $e(\ell)$ is negligible in ℓ, over the choice of $\mathcal{T}^{j+1}$, there is $u \in \mathcal{T}^{j+1} \cap L(K)$ such that $(R', C') \stackrel{\text{def}}{=} \vartheta_u(R \cup \{u\}, C \cup \{u\})$ is a $(j+1)$-nice pair.
- $\varphi(L(K)) > \ell$. Applying Lemma 5.13, there exists index $m \in [[\ell]]$ such that with probability $1 - e(\ell)$ (again $e(\ell)$ is negligible), over the choice of $\mathcal{T}^{j+1}$, there is $w \in \mathcal{T}^{j+1}$ such that $\varphi(L(K^w)) \leq m$ where $K^w \stackrel{\text{def}}{=} \mathcal{K}^{[j+1]}[R, C \cup \{w\}]$. By Proposition 5.10 the pair $(R, C \cup \{w\})$ is a $(j+1)$-nice pair and $\varphi(L(K^w)) \leq \ell$. Now, in a situation similar to the first case with respect to $(R, C \cup \{w\})$. By thinking of $\mathcal{T}^{j+1}$ as being sampled it two blocks $\mathcal{T}_1^{j+1}$ and $\mathcal{T}_2^{j+1}$ each of size of the original $\mathcal{T}^{j+1}$. We conclude that with probability $(1 - e(\ell))^2 \geq 1 - 2 \cdot e(\ell)$, there is $u \in \mathcal{T}^{j+1} \cap L(K^w)$ such that $(R', C') \stackrel{\text{def}}{=} \vartheta_u(R \cup \{u\}, C \cup \{u, w\})$ is a $(j+1)$-nice pair.

Observe, in each case, the last operator is the truncating operator. By Proposition 5.12 it follows that $\mathcal{I}^{[j+1]}(R') \succ \mathcal{I}^{[j]}(R)$. ■

APPENDIX

A Lower bound of queries complexity

We reproduce the proof of [GR08] showing that $\Omega(1/\epsilon)$ queries are required for testing any graph property that is non-trivial for testing regardless the strategy of selecting queries, that is, adaptively and non-adaptively. Recall that a graph property Π is non-trivial for testing if there exists $\epsilon_0 > 0$ such that for infinitely many $N \in \mathbb{N}$ there exist N-vertex graphs G_1 and G_2 such that $G_1 \in \Pi$ and G_2 is ϵ_0-far from Π. Thus,

Proposition A.1. *Let Π be a property that is non-trivial for testing. Then, any tester for Π has query complexity $\Omega(1/\epsilon)$.*

Note that the claim holds also for general properties (i.e., arbitrary sets of functions).

Proof. Let $\epsilon_0 > 0$ be as in the definition, and consider any $N \in \mathbb{N}$ such that Π contains some N-vertex graphs as well as some N-vertex graphs that are ϵ-far from Π. Let G_0 be any N-vertex graph that is ϵ-far from Π, let $G_1 \in \Pi$ be an N-vertex graph closest to G_0, and let $\delta > \epsilon$ denote the relative distance between G_0 and G_1. Let D denote the set of vertex pairs on which G_0 and G_1 differ; indeed, $|D| = \delta \cdot N^2$. Now, for every $\epsilon \leq \epsilon_0$, consider a graph, G, obtained at random from G_0 and G_1 by uniformly selecting a random $R \subseteq D$ of cardinality $\epsilon \cdot N^2$ and letting G agree with G_0 on all pairs in R and agree with G_1 otherwise. Clearly, any tester that makes $o(\epsilon_0/\epsilon)$ queries cannot distinguish G from G_1 (because regardless of is query selection strategy, its next query resides in R with probability at most $|R|/|D| \leq \epsilon/\epsilon_0$). Thus, such a tester cannot decide correctly on both G and G_1 (because G is ϵ-far from Π whereas $G_1 \in \Pi$). Recalling that ϵ_0 is a fixed constant, the proposition follows. ■

We stress that by Proposition 2.4 it follows that being a blow-up of a fixed graph is non-trivial. Hence, the lower bound of the query complexity applicable in our case.

B Trivial combinatorial fact

Proposition B.1. *Assume that the conditions are as in Lemma 5.13, and that $Viol_\mu(V, V) > \frac{\epsilon}{2}N^2$. Then, there exist indices $i, j \in [n]$ and $m \in [[\ell]]$, a set $W \subseteq V_j$ with $\varphi(W) \leq \ell - m$ such that for each $w \in W$ there exists a set $U^w \subseteq V_i$ with $\varphi(U^w) \leq m$, and $\{w\} \times U^w \subseteq Viol_\mu$.*

Proof. By assumption $|Viol_\mu(V,V)| \geq \frac{\epsilon}{2}N^2$. Because $V_1,\ldots,V_n$ partition V, it follows that there exists sets V_i and V_j such that

$$|Viol_\mu(V_i,V_j)| \geq \frac{\epsilon}{2n^2}N^2 \geq \frac{\epsilon}{2t^2}N^2.$$

For each $m \in [[\ell]]$ let W_m be the following set

$$W_m \overset{\text{def}}{=} \left\{ v \in V_j \ : \ |Viol_\mu(\{v\},V_i)| \geq \frac{N}{2^m} \right\}.$$

Since $W_{m-1} \subseteq W_m$ for $1 < m \leq \ell$, each $w \in W_m \backslash W_{m-1}$ satisfies

$$\frac{N}{2^m} \leq |Viol_\mu(\{w\},V_i)| < \frac{N}{2^{m-1}}.$$

Therefore,

$$\sum_{m=1}^{\ell} |W_m \backslash W_{m-1}| \cdot \frac{N}{2^{m-1}} + |V_j| \cdot \frac{N}{2^\ell} \geq \frac{\epsilon}{2t^2}N^2$$

and for $\ell \geq \log 1/\epsilon + \log 4t^2$ we have that,

$$\sum_{m=1}^{\ell} |W_m \backslash W_{m-1}| \frac{N}{2^{m-1}} \geq (\frac{\epsilon}{3t^2} - 1/2^\ell)N^2 \geq \frac{\epsilon}{4t^2}N^2.$$

Thus, there exists $m^* \in [[\ell]]$ such that

$$|W_{m^*} \backslash W_{m^*-1}| \frac{N}{2^{m^*-1}} \geq \frac{\epsilon}{4t^2\ell}N^2,$$

hence,

$$|W_{m^*} \backslash W_{m^*-1}| \geq \frac{\epsilon'}{\ell}2^{m^*}N,$$

where $\epsilon' \overset{\text{def}}{=} \frac{1}{8t^2}\epsilon$. Let $W \overset{\text{def}}{=} W_{m^*}$, then $|W| = |W_{m^*}| \geq \frac{\epsilon'}{\ell}2^{m^*}N$, and $\varphi(W) \leq \ell - m^*$. Moreover, by the definition of W for each $w \in W$, it follows that $|Viol_\mu(\{w\},V_i)| \geq N/2^{m^*}$. Let $U^w \overset{\text{def}}{=} \{v \in V_i \ : \ (w,v) \in Viol_\mu\}$, thus, $|U^w| \geq N/2^{m^*}$, and $\varphi(U) \leq m^*$. Lastly, by definition of U^w, it follows that $\{w\} \times U^w \subseteq Viol_\mu$. This completes the proof. ■

References

[AFKS99] Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. In *Combinatorica*, pages 451–476, 1999.

[AFNS06] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. In *Proc. of STOC 2006*, pages 251–260, 2006.

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:339–348, 1998.

[GR97] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Algorithmica*, pages 406–415, 1997.

[GR08] O. Goldreich and D. Ron. Algorithmic aspects of property testing in the dense graphs model. *ECCC*, TR08-039, 2008.

[Ron] Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in TCS*, to appear.

[Ron08] Dana Ron. Property testing: A learning theory perspective. *Found. Trends Mach. Learn.*, 1(3):307–402, 2008.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials withapplications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.