Testing Acyclicity of Directed Graphs in Sublinear Time*

Michael A. Bender† SUNY Stony Brook

Dana Ron‡ Tel Aviv University

Abstract

This paper initiates the study of testing properties of directed graphs. In particular, the paper considers the most basic property of directed graphs – acyclicity. Because the choice of representation affects the choice of algorithm, the two main representations of graphs are studied. For the adjacency-matrix representation, most appropriate for dense graphs, a testing algorithm is developed that requires query and time complexity of $\tilde{O}(1/\epsilon^2)$, where ϵ is a distance parameter *independent* of the size of the graph. The algorithm, which can probe the adjacency matrix of the graph, accepts every graph that is acyclic, and rejects, with probability at least $2/3$, every graph whose adjacency matrix should be modified in at least ϵ fraction of its entries so that it becomes acyclic. For the incidence list representation, most appropriate for sparse graphs, an $\Omega(|\mathrm{V}|^{1/3})$ lower bound is proved on the number of queries and the time required for testing, where V is the set of vertices in the graph.

Along with acyclicity, this paper considers the property of *strong connectivity*. Contrasting upper and lower bounds are proved for the incidence list representation. In particular, if the testing algorithm can query on both incoming and outgoing edges at each vertex, then it is possible to test strong connectivity in $\tilde{O}(1/\epsilon)$ time and query complexity. On the other hand, if the testing algorithm only has access to outgoing edges, then $\Omega(\sqrt{N})$ queries are required to test for strong connectivity.

Keywords: Property Testing, Sublinear Time, Random Sampling, Directed Graphs, Directed Acyclic Graphs, Strongly Connected Components, Adjacency-Matrix Representation, Incidence-List Representation, Algorithms.

1 Introduction

The Problem. Deciding whether a graph is *acyclic* is one of the basic algorithmic questions on directed graphs. It is well known that this problem can be solved by depth first search in

*An earlier version of this paper was presented at the 27th International Colloquium on Automata, Languages, and Programming (ICALP 2000).

†Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA. Email: bender@cs.sunysb.edu. Supported in part by the ISX Corporation and Hughes Research Laboratories.

‡Department of Electrical Engineering – Systems, Tel Aviv University, Ramat Aviv, Israel. Email: danar@eng.tau.ac.il.

time linear in the size of the graph. A natural generalization of this problem is asking how *close to acyclic* is a given graph. That is, what is the minimum number of edges (or vertices) that should be removed from the graph so that there are no remaining directed cycles. This problem is known as the *minimum feedback arc (or vertex) set* problem. Unfortunately, this problem is NP-hard [26] and even APX-hard [25]. Consequently researchers have developed approximation algorithms in various settings (including studying the complementary problem of the *maximum acyclic subgraph*) [12, 17, 32, 28, 8, 22, 3, 14].

TESTING GRAPH PROPERTIES. The field of *Testing Graph Properties* [19] suggests an alternate framework with which to study the above problem, and this is the approach that we take in this paper. A *property tester* determines whether a graph $G = (V, E)$ has a given property or is *far from* having the property. More formally, Testing Graph Properties is the study of the following family of tasks. Let P be a predetermined graph property (such as acyclicity, connectivity, or 3-colorability). A testing algorithm for property P is given *access* to a graph G so it can *query* the incidence relationships between vertices. If G has property P then the algorithms should accept with probability at least 2/3. If many edge modifications should be performed so that G has the property, then the algorithm should reject with probability at least 2/3. The success probability of the algorithm can clearly be amplified by repetitions to be arbitrarily close to 1.

We thus relax the task of deciding *exactly* whether the graph has the property, but expect of the algorithm to perform its task by observing as few vertices and edges in the graph as possible. Specifically, we are only willing to spend time that is *sublinear in* or even *independent of* the size of the graph. Thus, in contrast to the standard notion of graph algorithms, property testing algorithms are not provided the whole graph and required to run in time polynomial in the size of the graph. Rather, they are provided access to the graph and are expected to run in sublinear time.

More concretely, in this paper we study the question of whether a graph G is acyclic or far from acyclic. If the graph is far from acyclic (that is, many edges should be removed so that no cycle remains), then the tester should reject; if the graph actually *is* acyclic, the tester should accept; if the graph is *nearly* acyclic, then the tester may answer either way. Thus, we excuse the tester from answering the most difficult instances correctly, but we require the tester to execute much more quickly then any exact decision algorithm.

ALTERNATE NOTION OF APPROXIMATION. In view of the above, property testing suggests an alternative notion of approximation that is related to the notion of *dual approximation* [23, 24]. An approximation algorithm is a mechanism that trades *accuracy* for *speed*. Given an optimization problem that associates costs with solutions, the more standard notion of approximation is to find a solution that is *close* to the cost of the optimal solution. By "close," we mean that the value found is within some multiplicative factor of the optimal cost.

A property tester also trades accuracy for speed, but may use a different notion of distance. Specifically, distance is measured in terms of the number of *edge insertions* and *deletions* necessary to obtain a particular property (which, in particular, may be having a solution with a given cost).

The following example illustrates the two notions of distance. A graph G might be *nearly* 3-colorable in the sense that there is a 3-colorable graph G′ at small edit distance to G, but *far from* 3-colorable in the sense that many colors are required to color G. Alternatively, a graph G might be nearly 3-colorable in the sense that it is 4-colorable, but far from 3-colorable in the sense that no graphs having small edit distance to G are 3-colorable. Both notions are natural and the preferred choice depends on the context. In some cases the two notions coincide (e.g., Max-Cut [19]).[1]

APPLICATIONS. A graph-property tester may be employed in several contexts. (1) A fast property tester can be used to speed up the slow exact decision procedure as follows. Before running the slow decision procedure, run the tester. If the fast inexact tester rejects, then we know with high confidence that the property does not hold and it is unnecessary to run the slow tester. In fact, it is often the case that when the testing algorithm rejects, it provides a *witness* that the graph does not have the property (in our case, a cycle). If the fast inexact tester accepts, then a slow exact decision procedure will determine whether the property is close to holding or actually holds. (2) There are circumstances in which knowing that a property nearly holds is good enough and consequently exact decision is unnecessary. (3) It may even be NP-hard to answer the question exactly, and so some form of approximation is inevitable.

IMPACT OF GRAPH REPRESENTATION. We now define precisely the notion of distance and how the tester actually probes the graph. In fact, there are two traditional representations for graphs, *adjacency matrices* and *incidence lists*. The choice of representation strongly affects these issues, as well as the applicable algorithmic techniques. We summarize the properties of each representation here.

- *Adjacency-Matrix Model.* Goldreich, Goldwasser, and Ron [19] consider the adjacency-matrix representation of graphs, where the testing algorithm is allowed to probe into the matrix. That is, the algorithm can query whether there is an edge between any two vertices of its choice. In the undirected case the matrix is symmetric, whereas in the directed case it may not be. In this representation the distance between graphs is the *fraction of entries* in the adjacency matrix on which the two graphs differ. By this definition, for a given distance parameter ϵ, the algorithm should reject every graph that requires more than $\epsilon \cdot |V|^2$ edge modifications in order to acquire the tested property. This representation is most appropriate for dense graphs, and the results for testing in this model are most meaningful for such graphs.
- *(Bounded-Length) Incidence-Lists Model.* Goldreich and Ron [20] consider the incidence-lists representation of graphs. In this model, graphs are represented by lists of *length d*, where d is a bound on the degree of the graph. Here the testing algorithm can query, for every vertex v and index $i \in \{1, \ldots, d\}$, which is the i'th neighbor of v. If no such neighbor exists then the answer

[1]In the case of acyclicity, the notion of distance in the context of property testing and the cost approximated in the minimum feedback arc set problem are in fact the same. However, the two problems do differ mainly because the former is a "promise" problem and so nothing is required of the algorithm in case the graph is close to acyclic.

is '0'. In the case of directed graphs each such list corresponds to the outgoing edges from a vertex.[2] Analogously to the adjacency matrix model, the distance between graphs is defined to be the fraction of entries on which the graphs differ according to this representation. Since the total number of incidence-list entries is $d \cdot |V|$, a graph should be rejected if the number of edges modifications required in order to obtain the property is greater than $\epsilon \cdot d|V|$. [3]

TECHNIQUES. The applicable techniques depend on the choice of representation. A central technique that is used for the *adjacency matrix* representation is *random sampling*. Specifically, as in this paper, the algorithm randomly selects a small set U of vertices from the graph G, finds the edges interconnecting the vertices, and determines whether the property holds for the small subgraph induced by U. If so, then the algorithm accepts. If not, the algorithm rejects. Often the crux of the proof is in showing that if a graph is far from having the property, then it contains many (very small) subgraphs that do not have the property.

The incidence-lists representation requires a different set of techniques, more applicable to sparse graphs. Specifically, because the graphs have few edges, a small random sample of vertices typically has no edges internal to the sample, that is, it is just the empty graph. Thus, past algorithms for this setting use additional techniques besides pure random sampling. In particular, some algorithms apply various forms of exhaustive local search (such as performing a breadth-first-search until a particular number of vertices are observed) [20, 29]. Other algorithms use random walks starting from randomly selected vertices [21].

TESTING DIRECTED GRAPHS. This paper studies property testing for *directed graphs*. Typically, a given problem on a directed graph is more difficult than the same problem on an undirected graph. In particular, testing acyclicity of undirected graphs in the adjacency-matrix representation is straightforward: Assume $\epsilon \geq \frac{2}{|V|}$ (or otherwise it is possible to make an exact decision in time polynomial in $1/\epsilon$ by looking at the whole graph). The basic observation is that *any* graph having at most $\epsilon|V|^2$ edges is nearly acyclic (because it is ϵ-close to the empty graph), while *only* very sparse graphs (having at most $|V| - 1 < \frac{\epsilon}{2}|V|^2$ edges) may be acyclic. Hence the algorithm can estimate the number of edges in the graph by sampling, and accept or reject based on this estimate. Testing acyclicity of undirected graphs in the incidence-list (bounded-degree) representation, is more interesting and is studied in [20]. However, this result does not extend to testing directed graphs.

OUR RESULTS. We first consider the problem of testing acyclicity in the adjacency matrix representation. We describe a tester whose query complexity and running time are *independent* of the size of the graph and *polynomial* in the given distance parameter ϵ. Specifically, the query complexity

[2]Actually, the lower bound we prove for this model holds also when the algorithm can query about the *incoming* edges to each vertex (where the number of incoming edges is bounded as well). We note that allowing to query about incoming edges can make testing strictly easier.

[3]A variant of the above model allows the incidence lists to be of different lengths [29]. In such a case, the distance is defined with respect to the total number of edges in the graph.

and running time are both $O\left(\frac{\log^2(1/\epsilon)}{\epsilon^2}\right)$. As mentioned above, the algorithm works by randomly and uniformly selecting a set of $\tilde{O}(1/\epsilon)$ vertices, and verifying whether the small subgraph induced by these vertices is acyclic. Thus, an acyclic graph is always accepted, and for all rejected graphs, the algorithms provides a "witness" that the graph is not acyclic in the form of a short cycle. A key (combinatorial) lemma used in proving the correctness of the algorithms shows that a graph that is far from acyclic contains a relatively large subset of vertices for which every vertex in the subset has many outgoing edges extending to other vertices in the subset. We then show that a sample of vertices from within this subset likely induces a subgraph that contains a cycle.

We next turn to the problem of acyclicity testing in the incidence-lists representation. We demonstrate that the problem is significantly harder in this setting. Specifically, we show an $\Omega(|\mathrm{V}|^{1/3})$ lower bound on the number of queries required for testing in this setting. To prove the bound we define two classes of directed graphs – one containing only acyclic graphs and one containing mostly graphs that are far from acyclic. We show that $\Omega(|\mathrm{V}|^{1/3})$ queries are required in order to determine from which class a randomly selected graph was chosen.

It appears that the techniques used in testing undirected graphs in the incidence-lists representation, cannot be applied directly to obtain an efficient acyclicity testing algorithm for directed graphs. Consider a graph that contains a relatively large subgraph that is far from acyclic, but such that many edges connect this subgraph to acyclic regions of the graph. By our lower bound, any testing algorithm should perform many queries concerning edges within this subgraph (to distinguish it from the case in which the subgraph is acyclic). However, both exhaustive local searches and random walks will "carry the algorithm away" to the acyclic regions of the graph. It would be interesting to develop an acyclicity tester that uses $O(|\mathrm{V}|^{1-\alpha})$ queries, for *any* $\alpha > 0$.

TESTING OTHER PROPERTIES OF DIRECTED GRAPHS. As noted in [19, Subsection 10.1.2], some of the properties studied in that paper (in the adjacency matrix model) have analogies in directed graphs. Furthermore, the algorithms for testing these properties can be extended to directed graphs. In particular, these properties are defined by partitions of the vertices in the graph with certain constraints on the sizes of the sets in the partition as well as on the density of edges between these sets. The techniques of [1] (for testing properties of undirected graphs in the adjacency-matrix representation), can also be extended to testing properties of directed graphs (Private communications with Noga Alon).

Another basic property of directed graphs, is (*strong*) *connectivity*. Namely, a directed graph is strongly connected if there is a directed path in the graph from any vertex to any other vertex. Testing this property is most meaningful in the incidence-lists model, as every graph can be made strongly connected by adding at most $2N$ directed edges. The undirected version of this problem is studied in [20], where an algorithm having query and times complexities $\tilde{O}(1/\epsilon)$ is presented and analyzed. As we show in Section 5, this algorithm can be extended to the directed case *if the algorithm can also perform queries about the incoming edges to each vertex*. Otherwise, (the algorithm can only perform queries about outgoing edges), a lower bound of $\Omega(\sqrt{N})$ on the number of queries can be obtained.

Related Work. Property testing of functions was first explicitly defined in [31] and extended in [19]. Testing algebraic properties (e.g., linearity or being a polynomial of low-degree) plays an important role in the settings of Program Testing (e.g., [9, 31, 30]) and Probabilistically-Checkable Proof systems (e.g., [7, 6, 13, 5, 4]). As mentioned previously, the study of testing graph properties was initiated in [19], where, in particular, the adjacency-matrix model was considered. Some of the properties studied in that work are bipartitness, k-colorability, having a clique of a certain size and more. In [20], testing properties of graphs represented by their incidence lists was considered. Some of the the properties studied in that work are k-connectivity and acyclicity.

Ergun *et. al.* [11] give a poly$(1/\epsilon)$-time algorithm for testing whether a relation is a total order. This can viewed as a special case of testing acyclicity in the adjacency-matrix model, where it is assumed that a directed edge exists between *every* two vertices.

Other papers concerning testing of graph properties and other combinatorial properties include [21, 11, 27, 18, 10]. Recently, Alon *et. al.* [1] presented a general family of graph properties that can be tested using a sample that is independent of the size of the graph (though the dependence on the distance parameter ϵ is somewhat high). In [2] it is shown that all properties defined by regular languages can be tested using a sample of size almost linear in the distance parameter.

As mentioned previously, the related minimum feedback set problem is APX-Hard [25], and its complementary, maximum acyclic subgraph is APX-Complete [28]. The former can be approximated to within a factor of $O(\log |\mathrm{V}| \log\log |\mathrm{V}|)$ [12], and the latter to within $2/(1+\Omega(1/\sqrt{\Delta}))$ where Δ is the maximum degree [8, 22]. Variants of these problems are studied in the following papers [32, 17, 3]. Perhaps the result most closely related to our work is that of Frieze and Kannan [14]. They show how to approximate the size of the maximum acyclic subgraph to within an additive factor of $\epsilon|\mathrm{V}|^2$, in time exponential in $1/\epsilon$. In comparison to their result, we solve a more restricted problem in time polynomial in $1/\epsilon$ as opposed to exponential. In addition, since our analysis is tailored to the particular problem (as opposed to theirs which follows from a general paradigm that gives rise a family of approximation algorithms), it may give more insight into the problem in question.

With the current trend of increasing memory and storage sizes, the problem of examining large structures in sublinear time has been studied in other contexts. For example, Gibbons and Matias [15, 16] develop a variety of data structures that glean information from large databases so they can be examined in sublinear time.

Organization This paper is organized as follows: In section 2 we present definitions and terminology that will be used in the rest of the paper. In Section 3 we preset a property tester for the adjacency-matrix representation. In Section 4 we present a lower bound on the number of queries required for property testing in the adjacency-matrix representation. In Section 5 we consider the problem of testing the property of stong connectivity.

2 Definitions

Let $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ be a directed graph, where $|\mathrm{V}| = N$, and $\mathrm{E} \subseteq \mathrm{V} \times \mathrm{V}$ consists of *ordered pairs* of vertices. For a given set of vertices $\mathrm{U} \subseteq \mathrm{V}$, let $\mathrm{G}(\mathrm{U})$ denote the subgraph of G induced by U, and for any two sets of vertices U_1 and U_2, let $\mathrm{E}(\mathrm{U}_1, \mathrm{U}_2)$ denote the set of edges going from vertices in U_1 to vertices in U_2. That is, $\mathrm{E}(\mathrm{U}_1, \mathrm{U}_2) \stackrel{\mathrm{def}}{=} \{(u_1, u_2) \in \mathrm{E} : \ u_1 \in \mathrm{U}_1, \ u_2 \in \mathrm{U}_2\}$.

We say that a graph G is *acyclic* if it contains no directed cycles. In other words, G is acyclic if and only if there exists a (one-to-one) ordering function $\phi : \mathrm{V} \mapsto \{1, \ldots, N\}$, such that for every $(v, u) \in \mathrm{E}$, $\phi(v) < \phi(u)$. We say that an edge $(v, u) \in \mathrm{E}$ is a *violating edge* with respect to an ordering $\phi(\cdot)$, if $\phi(v) > \phi(u)$.

We consider two representations of (directed) graphs. In the *adjacency-matrix* representation, a graph G is represented by a 0/1 valued $N \times N$ matrix M_G, where for every pair of vertices $u, v \in \mathrm{V}$, $\mathrm{M}_\mathrm{G}[u, v] = 1$ if and only if $(u, v) \in \mathrm{E}$. This representation is more appropriate for dense graphs than sparse graphs, because with sparse graphs the representation entails a large space wastage. In the *incidence-lists* representation, a graph G is represented by an $N \times d$ matrix L_G, (which can be viewed as N *lists*), where d is a bound on the outdegree of each vertex in G. For $v \in \mathrm{V}$ and $i \in [d]$, $\mathrm{L}_\mathrm{G}[v, i] = u$, if an only if the i'th edge going out of v is directed to u. If such an edge does not exist then the value of the entry is '0'.

For any $0 \leq \epsilon \leq 1$, a graph G in either of the two representations, is said to be *ϵ-close to acyclic*, if at most an ϵ-fraction of entries in G's representation need to be modified to make G acyclic. If *more* than an ϵ fraction of entries must be modified, than it is *ϵ-far from acyclic*. Because the adjacency-matrix representation has size N^2, this means that a graph G in the adjacency-matrix representation is ϵ-close to being acyclic if at most $\epsilon \cdot N^2$ edges can be removed to make G acyclic. Because the incidence-lists representation has size $d \cdot N$, the number of edges that should be removed in this representation is at most $\epsilon \cdot dN$. Note that a graph is ϵ-close to acyclic if and only if there exists an order function $\phi(\cdot)$, with respect to which there are at most ϵN^2 (similarly, $\epsilon \cdot dN$), violating edges. We say in this case that $\phi(\cdot)$ is *ϵ-good*.

A testing algorithm for acyclicity is given a distance parameter ϵ, and oracle access to an unknown graph G. In the adjacency-matrix representation this means that the algorithm can query for any two vertices u and v whether $(u, v) \in \mathrm{E}$. In the incidence-lists representation this means that the algorithm can query, for any vertex v and index $i \in [d]$, what vertex does the i'th edge going out of v point to. If the graph G is acyclic then the algorithm should accept with probability at least 2/3, and if it is ϵ-far from acyclic then the algorithm should reject with probability at least 2/3.

3 Testing Acyclicity in the Adjacency-Matrix Representation

We next give our algorithm for testing acyclicity when the graph is represented by its adjacency matrix. Similarly to several previous testing algorithms in the (un-directed) adjacency-matrix model, the algorithm is the "natural" one. Namely, it selects a random subgraph of G (having only $\tilde{O}(1/\epsilon)$ vertices), and checks whether this subgraph is acyclic (in which case it accepts) or not (in which case it rejects). Observe that the sample size is independent of the size of G.

Acyclicity Testing Algorithm

1. Uniformly and independently select a set of $\Theta(\log(1/\epsilon)/\epsilon)$ vertices and denote the set by U.
2. For every pair of vertices $v_1, v_2 \in \mathrm{U}$, query whether either $(v_1, v_2) \in \mathrm{E}$ or $(v_2, v_1) \in \mathrm{E}$, thus obtaining the subgraph G(U) induced by U.
3. If G(U) contains a cycle, then *reject*, otherwise *accept*.

Theorem 1 *The algorithm described above is a testing algorithm for acyclicity having query and time complexity $\tilde{O}(1/\epsilon^2)$. Furthermore, if the graph* G *is acyclic it is always accepted, and whenever the algorithm rejects a graph it provides a certificate of the graph's cyclicity (in the form of a short cycle).*

The bound on the query and time complexity of the algorithm follows directly from the description of the algorithm. In particular, there are $O(\log^2(1/\epsilon)/\epsilon^2)$ pairs of vertices in U, which limits the number of queries made as well as the number of edges in G(U). To verify whether G(U) is acyclic or not, a Breadth-First-Search (BFS) can be performed starting from any vertex in G(U). The time complexity of this search is bounded by the number of edges in G(U), as desired. The second statement in the theorem is immediate as well. It remains to be shown that every graph that is ϵ-far from being acyclic is rejected with probability at least 2/3.

Proof Idea. We prove Theorem 1 using two lemmas. Lemma 2 shows that if a graph G is far from acyclic, then G contains a relatively large set W such that each vertex in W has many outgoing edges to other vertices in W. (In fact, as we showed in an earlier version of this paper, every vertex in W also has many incoming edges from other vertices in W. However, we no longer need this stronger property.) Lemma 3 shows that if we uniformly select a sufficient number of vertices from W, then with probability at least 9/10 the underlying graph induced by these vertices contains a cycle. To prove Theorem 1, we show that with sufficiently high probability, a large enough constant-size sample of vertices in G contains enough vertices in W to find a cycle with the desired probability.

Definitions. To formalize the above ideas, we use the following definitions. For any vertex $v \in \mathrm{V}$, let $\mathrm{O}(v) \stackrel{\mathrm{def}}{=} \{u : (v,u) \in \mathrm{E}\}$ be the set of v's outgoing edges. Given a set $\mathrm{W} \subseteq \mathrm{V}$, we say that v has *low outdegree with respect to* W, if $|\mathrm{O}(v) \cap \mathrm{W}| \leq \frac{\epsilon}{2}N$; otherwise it has *high outdegree with respect to* W.

Lemma 2 *If* G *is ϵ-far from acyclic, then there exists a set $\mathrm{W} \subseteq \mathrm{V}$, such that $|\mathrm{W}| \geq \sqrt{\frac{\epsilon}{2}}N$, and every $v \in \mathrm{W}$ has high outdegree with respect to* W.

Lemma 3 *Let* $\mathrm{W} \subseteq \mathrm{V}$ *be a set of vertices such that for every* $v \in \mathrm{W}$, $|\mathrm{O}(v) \cap \mathrm{W}| \geq \epsilon'|\mathrm{W}|$ *for some* $\epsilon' > 0$. *Suppose we uniformly and independently select* $\Theta(\log(1/\epsilon')/\epsilon')$ *vertices in* W. *Then with probability at least* 9/10 *(over this selection) the subgraph induced by these vertices contains a cycle.*

We prove the two lemmas momentarily, but first we show how Theorem 1 follows from the two lemmas.

Proof of Theorem 1: If G is acyclic, then clearly it always passes the test. Thus, consider the case in which G is ϵ-far from acyclic. By Lemma 2, there exists a set $\mathrm{W} \subseteq \mathrm{V}$, such that $|\mathrm{W}| \geq \sqrt{\frac{\epsilon}{2}}N$, and every $v \in \mathrm{W}$ has high outdegree with respect to W. Let $\alpha \stackrel{\mathrm{def}}{=} |\mathrm{W}|/N$ be the *fraction of graph vertices that belong to* W, so that $\alpha \geq \sqrt{\frac{\epsilon}{2}}$. By applying a (multiplicative) Chernoff bound we have that for every integer $m > 12$, with probability at least 9/10, a uniformly and independently selected sample of $2m/\alpha$ vertices contains at least m (not necessarily distinct) vertices in W (where these vertices are uniformly distributed in W). Assume this is in fact the case (where we account for the probability of error and set m below).

Let $\epsilon' \stackrel{\mathrm{def}}{=} \frac{\epsilon}{2\alpha}$ so that by definition of α, for every $v \in \mathrm{W}$, $|\mathrm{O}(v) \cap \mathrm{W}| \geq \epsilon'|\mathrm{W}|$. By setting the quantity m to be $\Theta(\log(1/\epsilon')/\epsilon')$, and applying Lemma 3, we obtain that conditioned on there being m elements in the sample that belong to W, a cycle is observed with probability at least 9/10. Adding the two error probabilities, and noting that the total size of the sample is $O(m/\alpha) = O(\log(1/\epsilon)/\epsilon)$, the theorem follows. ■

Now that we have demonstrated how Theorem 1 follows from Lemmas 2 and 3, we prove the lemmas themselves.

Proof of Lemma 2: We prove the *contrapositive* of the statement in the lemma: If such a set W does not exist, then the graph is ϵ-close to being acyclic. In other words, we show that if for every subset $\mathrm{Z} \subseteq \mathrm{V}$ having size at least $\sqrt{\frac{\epsilon}{2}}N$, there exists at least one vertex in Z having small outdegree with respect to Z, then the following holds. There exists an order $\phi : \mathrm{V} \mapsto [N]$, and a set of edges T of size at most ϵN^2, such that the edges of T are the only violating edges with respect to ϕ.

We define ϕ and construct T in N steps. At each step we select a vertex v for which ϕ is not yet determined, and set the value of $\phi(v)$. We maintain an index ℓ (*last*), where initially $\ell = N$. At the start of a given step, let $\mathrm{Z} \subseteq \mathrm{V}$ denote the set of vertices for which ϕ is yet undefined (where initially, $\mathrm{Z} = \mathrm{V}$). As long as $|\mathrm{Z}| \geq \sqrt{\frac{\epsilon}{2}}N$, we do the following. Consider any vertex v that has low outdegree with respect to Z (where the existence of such a vertex is ensured by our (counter) assumption). Then we set $\phi(v) = \ell$, decrease ℓ by 1, and let $\mathrm{T} \leftarrow \mathrm{T} \cup \{(v,u) \in \mathrm{E} : \ u \in \mathrm{Z}\}$. Hence, at each step, the size of T increases by at most $\frac{\epsilon}{2}N$.

Finally, when $|\mathrm{Z}| < \sqrt{\frac{\epsilon}{2}}N$, so that the vertices in Z may all have high outdegree with respect to Z, we order the vertices in Z arbitrarily between 1 and ℓ, and add to T all (at most $|\mathrm{Z}|^2 < \frac{\epsilon}{2}N^2$) edges between vertices in Z. Thus, the total number of edges in T is bounded by ϵN^2, as desired.

It remains to show that there are no other violating edges with respect to ϕ. Namely, for every $(v, u) \in \mathrm{E} \setminus \mathrm{T}$, it holds that $\phi(v) < \phi(u)$. Consider any such edge $(v, u) \in \mathrm{E} \setminus \mathrm{T}$. We claim that necessarily the value of ϕ was first defined for u, implying that in fact $\phi(v) < \phi(u)$ (as the value ℓ given by ϕ) decreases as the above process progresses). This must be the case since otherwise, if the value of ϕ was first defined for v then the edge (v, u) would have been added to T, contradicting our assumption that $(v, u) \in \mathrm{E} \setminus \mathrm{T}$. ■

Proof of Lemma 3: Let $m = \frac{c \cdot \ln(1/\epsilon')}{\epsilon'} + 1$ be the number of vertices selected (uniformly and independently) from W, where c is a constant that is set below. We shall show that with probability at least 9/10 over the choice of such a sample U, for every vertex $v \in \mathrm{U}$, there is another vertex $u \in \mathrm{U}$ such that $(v, u) \in \mathrm{E}$. This implies that the subgraph induced by U has no sink vertex, and hence contains a cycle.

Let the m vertices selected in the sample U be denoted $v_1, \ldots, v_m$. For each index $1 \leq i \leq m$, let $\mathcal{E}_i$ be the event that there exists a vertex v_j such that $(v_i, v_j) \in \mathrm{E}$. In other words, $\mathcal{E}_i$ is the (desirable) event that v_i has non-zero outdegree in the subgraph induced by U. We are interested in upper bounding the probability that for some i the event $\mathcal{E}_i$ does not hold. That is, $\Pr\left[\bigcup_{i=1}^{m} \neg\mathcal{E}_i\right]$. Since the vertices in the sample are chosen uniformly and independently, and every vertex in W has outdegree at least $\epsilon' \cdot |\mathrm{W}|$, for each fixed i,

$$\Pr[\neg\mathcal{E}_i] \;\leq\; (1 - \epsilon')^{m-1} \;<\; \exp(-(m-1)\epsilon') \;=\; \exp(-c \ln(1/\epsilon')) \;=\; (\epsilon')^{-c} \tag{1}$$

By applying a probability union bound,

$$\Pr\left[\bigcup_{i=1}^{m} \neg\mathcal{E}_i\right] \;\leq\; \sum_{i=1}^{m} \Pr[\neg\mathcal{E}_i] \;<\; \left(\frac{c \cdot \ln(1/\epsilon')}{\epsilon'} + 1\right) \cdot (\epsilon')^{-c} \tag{2}$$

Setting c to be a sufficiently large constant (say, $c \geq 10$), for any $\epsilon' \leq 1/2$ the above probability is at most 1/10 as required. ■

The Proof of Lemma 3 concludes the analysis of our algorithm for testing acyclicity in the adjacency-matrix representation. In the next section we consider the same problem in the incidence-lists representation, and we show that the problem has a very different complexity.

4 Testing Acyclicity in the Incidence-Lists Representation

In this section we prove a lower bound of $\Omega(N^{1/3})$ for testing acyclicity in the incidence-lists representation, when d and ϵ are constant. This lower bounds holds even when the algorithm may query about the incoming edges to each vertex (where the indegree of each vertex is also at most d). We conjecture that the problem is even harder than this lower bound suggests and discuss the hardness further at the end of this section. The question of how to obtain a formal argument for a higher lower bound remains as an open problem.

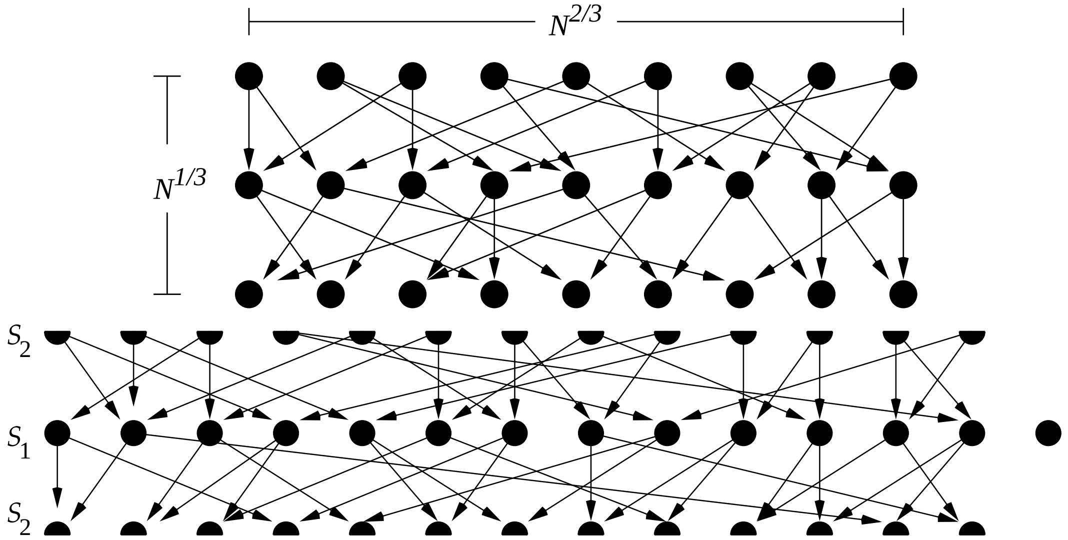

Figure 1: Top: The structure of a representative graph in the family $\mathcal{G}_1$. All graphs in $\mathcal{G}_1$ are acyclic. Bottom: The structure of a representative graph in the family $\mathcal{G}_2$. Almost all graphs in $\mathcal{G}_2$ are far from acyclic. (Note that for ease of representation, the vertices in S_1 are doubly represented in the top row and in the bottom row.

Theorem 4 *Testing Acyclicity in the incidence-lists representation with distance parameter $\epsilon \leq \frac{1}{16}$, requires more than $\frac{1}{4} \cdot N^{1/3}$ queries.*

To prove the bound we define two classes of (directed) graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, each over N vertices, with degree bound d. All graphs in $\mathcal{G}_1$ are acyclic, and almost all graphs in $\mathcal{G}_2$ are ϵ-far from acyclic (for $\epsilon = \frac{1}{16}$). We then show that no algorithm can distinguish between a graph chosen randomly in $\mathcal{G}_1$ and a graph chosen randomly in $\mathcal{G}_2$ in less than $\alpha \cdot N^{1/3}$ queries, for $0 < \alpha \leq \frac{1}{4}$. The classes are defined as follows (see Figure 1).

- Each graph in $\mathcal{G}_1$ consists of $K = N^{1/3}$ layers, $\mathrm{L}_1, \ldots, \mathrm{L}_K$, each having $M = N^{2/3}$ vertices. From each layer L_i there are $d \cdot |\mathrm{L}_i| = d \cdot M$ edges going to layer L_{i+1}, where these edges are determined by d matchings between the the vertices in the two layers.
- Each graph in $\mathcal{G}_2$ consists of two equal-size subsets of vertices, S_1 and S_2. There are $d \cdot \frac{N}{2}$ edges going from S_2 to S_1, and $d \cdot \frac{N}{2}$ edges going from S_1 to S_2. The two sets of edges are each defined by d matching between S_1 and S_2.

In both cases, every edge has the same label at both its ends (determined by the matching).

Clearly, all graphs in $\mathcal{G}_1$ are acyclic. We next show that almost all graphs in $\mathcal{G}_2$ are far from acyclic.

Lemma 5 *For every $N > d \geq 128$, with probability at least $1 - 2^{-N}$, a randomly selected graph in $\mathcal{G}_2$ is ϵ-far from acyclic, for $\epsilon = 1/16$.*

Proof: Consider all orderings $\phi : \mathrm{V} \mapsto N$, and let us partition them into classes as follows. Each class is defined by an equal partition $(\mathrm{V}^f, \mathrm{V}^\ell)$ of V, where each ordering ϕ in the class maps the vertices in V^f to $\{1, \ldots, N/2\}$ and the vertices in V^ℓ to $\{N/2+1, \ldots, N\}$. For a given fixed partition $(\mathrm{V}^f, \mathrm{V}^\ell)$ we show that the probability that a random graph in $\mathcal{G}_2$ has less than ϵdN edges going from vertices in V^ℓ to vertices in V^f (i.e., in violation of each ordering in the class), is at most 2^{-2N}. It follows that the probability that some ordering, belonging to any one if the classes, has less than ϵdN violating edges is bounded by 2^{-N} as desired. Details follow.

A random graph in $\mathcal{G}_2$ is chosen by selecting d random matchings corresponding to edges going from S_1 to S_2, and d random matching corresponding to edges going form S_2 to S_1. Let $\mathrm{V}_1^f \stackrel{\text{def}}{=} \mathrm{V}^f \cap \mathrm{S}_1$, and let V_2^f, V_1^ℓ, and V_2^ℓ be defined analogously. Without loss of generality, assume $|\mathrm{V}_1^f| \geq |\mathrm{V}_1^\ell|$, so that $|\mathrm{V}_1^f| \geq N/4$ (since $|\mathrm{V}_1^f| + |\mathrm{V}_1^\ell| = |\mathrm{S}_1| = N/2$). Note that since $|\mathrm{V}_1^f| + |\mathrm{V}_2^f| = N/2$, it holds that $|\mathrm{V}_2^\ell| = |\mathrm{V}_1^f| \geq N/4$ as well. We show that with exponentially high probability there are at least ϵdN edges extending from vertices in V_2^ℓ to vertices in V_1^f (and for simplicity we ignore the contribution of edges going from V_1^ℓ to V_2^f).

Consider a single random matching. Such a matching can be determined by a choice of a random subset $\mathrm{T}_1 \subseteq \mathrm{S}_1$ of size $|\mathrm{V}_2^\ell|$, and two random matchings: one between the vertices of V_2^ℓ and the vertices T_1, and one between V_2^f and $\mathrm{S}_1 \setminus \mathrm{T}_1$. We next bound the probability that $|\mathrm{T}_1 \cap \mathrm{V}_1^f| < N/8$, that is, that there are less then $N/8$ edges in this matching that go from V_2^ℓ to V_1^f. Since $|\mathrm{V}_2^\ell| = |\mathrm{V}_1^f| \geq N/4$, which is half the size of S_1, the probability that $|\mathrm{T}_1 \cap \mathrm{V}_1^f| < N/8$ decreases as $|\mathrm{V}_2^\ell|$ increases. Hence it suffices to analyze the case $|\mathrm{V}_2^\ell| = |\mathrm{V}_1^f| = N/4$. Furthermore, for any integer $i < N/8$, the probability that in choosing $N/4$ elements *without repetitions* from S_1, the number of elements selected from V_1^f is i, is an upper bound on the probability that such a number of elements are selected when *repetitions are allowed*. (The former probability is $\frac{\binom{N/2}{i} \cdot \binom{N/2}{N/2-i}}{\binom{N}{N/2}}$, and the latter is $\frac{\binom{N/2}{i}}{2^{N/2}}$.) Hence, the probability that $|\mathrm{T}_1 \cap \mathrm{V}_1^f| < N/8$ is bounded by the probability that in $N/2$ Bernoulli trials (where the probability that the outcome is 1 is 1/2), the number of trials with outcome 1 is less than half the expected value. This probability is at most $\exp(-(1/2) \cdot (1/2)^2 \cdot (N/4)) = \exp(-N/32)$.

If among the d random matchings, at least $d/2$ have at least $N/8$ edges going from V_2^ℓ to V_1^f, then the number of violating edges is at least $\frac{1}{16} \cdot d \cdot N$. Since the d random matchings are selected independently, the probability that in more than half of them there are less then $N/8$ edges that go from V_2^ℓ and V_1^f, is less than

$$2^d \cdot \exp(-(N/32) \cdot (d/2)) < \exp(-d \cdot (N/64 - 1))$$

which is less than 2^{-2N} for $d \geq 128$. ■

Let A be an algorithm for testing acyclicity using $m = m(N)$ queries (where ϵ is constant). Namely, A is a (possibly probabilistic) mapping from *query-answer histories* $[(q_1, a_1), \ldots, (q_t, a_t)]$ to q_{t+1}, for every $t < m$, and to $\{\mathsf{accept}, \mathsf{reject}\}$, for $t = m$. A query q_t is a triplet (v_t, i_t, b_t), where $v_t \in \mathrm{V}$, $i_t \in [d]$, and $b_t \in \{in, out\}$. An answer a_t is simply a vertex $u_t \in \mathrm{V}$, where, in case $b = out$

this means that there is an edge going from v_t to u_t labeled by i_t, and in case $b = in$ this means that there is an edge coming into v_t from u_t labeled i_t. We assume that the mapping is defined only on histories that are consistent with some graph. Any query-answer history of length $t-1$ can be used to define a *knowledge* graph, G^{kn}, at time $t-1$ (i.e., before the t^{th} query). The vertex set of G^{kn} contains all vertices that appear in the history (either in queries or as answers), and its edge set contains the edges between $v_{t'}$ and $a_{t'}$ for all $t' < t$ (with the appropriate labelings and directions). Thus, G^{kn} is a labeled subgraph of the labeled graph tested by A.

In what follows we describe two random processes, P_1 and P_2, which interact with an arbitrary algorithm A, so that for $j \in \{1,2\}$, P_j answers A's queries while constructing a random graph from $\mathcal{G}_j$. For a fixed A that uses m queries, and for $j \in \{1,2\}$, let D_j^{A} denote the distribution on query-answer histories (of length m) induced by the interaction of A and P_j. We shall show that for any given A that uses $m \leq \alpha N^{1/3}$ queries, the statistical difference between D_1^{A} and D_2^{A} is at most $\alpha(1+\alpha)$. We then combine this with Lemma 5 to obtain the lower bound of Theorem 4.

DEFINITION OF P_1. The process has two stages. The first stage goes on as long as the algorithm performs queries. In this stage the process answers the algorithm's queries and updates the knowledge graph. In the second stage the process completes the knowledge graph into a graph in $\mathcal{G}_1$. In the first stage, whenever a new vertex is added to the knowledge graph, the process assigns it to a particular layer. Note that this information is *not* included in the knowledge graph, and is only implicit, whenever a vertex is know to be a source or a sink. Starting from $t = 1$, for each query $q_t = (v_t, i_t, b)$ of A, process P_2 proceeds as follows:

1. If v_t does not belong to the knowledge graph G^{kn}, then P_1 first assigns v_t to one of the layers L_j in the following manner. Let n_j be the number of vertices in G^{kn} that are already assigned to layer L_j. Then v_t is assigned to L_j with probability $\frac{M-n_j}{N-\sum n_j}$. (Recall that $M = N^{2/3}$ is the size of each layer.)

   Let L_j be the layer that vertex v_t is assigned to and assume $b_t = out$ (the case $b = in$ is treated analogously). If $j = K$ then the answer is '0' (as all vertices in the last layer are sinks). Otherwise, the answer u_t is chosen among the vertices in L_{j+1} and the vertices not yet in the knowledge graph (whose set we denote by R) as follows. Let $n_j^{i_t}$ be the number of vertices in L_j that already have an incoming edge labeled i_t. Then with probability $\frac{n_j - n_j^{i_t}}{M - n_j^{i_t}}$ a vertex is chosen uniformly among the vertices assigned to L_j that do not have an incoming edge labeled i_t, and with probability $1 - \frac{n_j - n_j^{i_t}}{M - n_j^{i_t}}$, a vertex is chosen uniformly in R (and assigned to layer L_{j+1}.
2. If v_t belongs to G^{kn}, and the edge queried is in G^{kn} as well, then P_1 answers consistently with G^{kn}.
3. If v_t belongs to G^{kn} but the edge queried does not belong to the knowledge graph then the answer u_t is chosen as described in the second part of Item 1 above.

In the second stage of P_1 the process uniformly selects a graph in $\mathcal{G}_1$ among all those consistent with the final knowledge graph G^{kn}. More precisely, for each $j \in [K]$ let n_j be the number of vertices assigned to L_j. Then the algorithm first randomly partitions the vertices not in G^{kn} into K subsets, where the j's subset has size $M - n_j$, and assigns them to the respective layers. The process then randomly completes the d matchings from each L_j to L_{j+1}.

DEFINITION OF P_2. Similarly to P_2 it consists of two stages. In the first stage, for each query $q_t = (v_t, i_t, b)$ of A, process P_1 proceeds as follows:

1. If v_t does not belong to G^{kn}, then P_2 first assigns v_t either to S_1 or to S_2 in the following (random) manner. Let n_1 be the number of vertices in the current knowledge graph that are already in S_1, and let n_2 be the number of vertices in S_2. Then, for $j \in \{1,2\}$, v_t is assigned to S_j with probability $\frac{N/2-n_j}{N-(n_1+n_2)}$.

   Suppose v_t is assigned to S_1, and assume $b_t = out$ (the other cases — v_t assigned to S_2 and/or $b_t = in$ — are treated analogously). The the answer u_t is chosen among the vertices in S_2 and the vertices not yet in the knowledge graph (whose set we denote by R) as follows. Let $n_2^{i_t}$ be the number of vertices in S_2 that already have an incoming edge labeled i_t. Then with probability $\frac{n_2-n_2^{i_t}}{N/2-n_2^{i_t}}$ a vertex in this set is chosen uniformly, and with probability $1-\frac{n_2-n_2^{i_t}}{N/2-n_2^{i_t}}$, a vertex in R is chosen uniformly.
2. If v_t belongs to G^{kn}, and the edge queried is in G^{kn} as well, then P_2 answers consistently with G^{kn}.
3. If v_t belongs to G^{kn} but the edge queried does not belong to the knowledge graph then the answer u_t is chosen as describe in Item 1 above.

In the second stage of P_2 the process uniformly selects a graph in $\mathcal{G}_2$ among all those consistent with the final knowledge graph G^{kn}. More precisely, if n_1 and n_2 are the number of vertices assigned to S_1 and S_2, respectively, then the algorithm first randomly selects a subset of size $N/2-n_1$ among the vertices not in G^{kn} and assigns them to S_1. The rest of the vertices no in G^{kn} are assigned to S_2. The process then randomly completes the d matchings from S_1 to S_2 and the d matchings from S_2 to S_1.

Lemma 6 *For every algorithm* A, *the process* P_1 *(P_2), when interacting with* A, *uniformly generates graphs in* $\mathcal{G}_1$ *($\mathcal{G}_2$).*

The lemma easily follows by induction on the maximum number of queries performed by A, where one may consider only deterministic algorithms, as every probabilistic algorithm can be viewed as a distribution over deterministic ones. The base case is clear, since if no query is made then the distribution on the resulting graph is clearly uniform. The induction step follows directly from the definition of the processes. In particular, the distribution on answers for any given query is such that the following holds. The distribution on graphs resulting from the process switching to the second stage *after* it answers the query is exactly the same as the distribution resulting from the process performing the second stage *without* answering the query.

Lemma 7 *Let $\alpha < 1$ and $m \leq \alpha N^{1/3}$. Then, for every algorithm* A *that asks m queries, the statistical distance between D_1^{A} and D_2^{A} is at most $\alpha(1+\alpha)$.*

Recall that D_i^{A} denotes the distribution on query-answer histories (of length m) induced by the interaction of A and P_i.

Proof: We assume without loss of generality that A does not ask queries whose answer can be derived from its knowledge graph, since those give it no new information. Under this assumption, we show that both in D_1^{A} and in D_2^{A}, the total weight of query-answer histories in which for some $t \leq m$ either a vertex in G^{kn} is returned as an answer to the t^{th} query (i.e., there exist $t' < t$ such that $a_t = v_{t'}$ or $a_t = a_{t'}$), or, the answer is '0', is at most $\alpha(1+\alpha)$. Note that by the definition of P_1 and P_2, in both distributions, for every history prefix, conditioned on the event that the new answer is not '0' and does not equal some previous query or answer, the new answer is uniformly distributed among all vertices not appearing in the history. Since A's queries only depend on the preceding query-answer history, the two distributions differ only in the probability assigned to sequences either containing the answer '0', or a vertex that already belongs to the induced knowledge graph. Since we show that the total weight of such sequences is at most $\alpha(1+\alpha)$, the bound on the statistical difference between the two distributions follows.

For any $t \leq m$, consider first the event that a vertex in G^{kn} is returned as an answer to the t^{th} query. In other words, v_t is matched (either via its outgoing edge, i_t, or via its incoming edge, i_t), to some vertex in G^{kn}. In the worst case, for both processes, there are at most $t-1$ vertices in the knowledge graph. In the case of P_1, when $b = in$ they may all belong to the level following that of v_t, and not have an incoming edge labeled i_t, and when $b = out$ they may all belong to the level preceding v_t (and not have an outgoing edge labeled i_t). In either case, the probability that v_t is matched to any of them is at most $\frac{(t-1)}{N^{2/3}-(t-1)}$, which for $t < N^{1/3}$ is less than $\frac{2(t-1)}{N^{2/3}}$. In the case of P_2 the probability that such an event occurs is only at most $\frac{(t-1)}{N/2-(t-1)} < \frac{2(t-1)}{N/2}$. Thus, the probability that such an event occurs for either process in a sequence of $\alpha N^{1/3}$ queries is at most $\sum_{t=1}^{m} \frac{2(t-1)}{N^{2/3}} < \alpha^2$.

We next bound the probability that when the algorithm interacts with P_1, it answers '0' (i.e., v_t is a sink and $b_t = out$ or v_t is a source and $b_t = in$). We bound this probability conditioned on no answer being selected from the knowledge graph (since we have bounded the probability that such an event occurs above). We view the algorithm as a strategy that tries to maximize the probability of reaching either a source or a sink. Each such strategy, together with the answers it receives (where given our conditioning, each answer is a uniformly distributed vertex), induces a distribution on a number s of *starting vertices* and $2s$ *lengths*. Each starting vertex corresponds to a query (v_t, i_t, b_t), where v_t is not in the current knowledge graph. For each such starting vertex v_t, there are two lengths – one determining the length of the path, starting from v_t, using incoming edges, and one using outgoing edges. (As the algorithm is trying to maximize the probability of reaching a source or a sink, branching from such paths only wastes queries.) As the algorithm makes $\alpha N^{1/3}$ queries, the sum of all lengths is $\alpha N^{1/3}$. It thus suffices to bound the probability of reaching either a source or a sink, for any fixed choice of s, and the lengths $\ell_1^1, \ell_1^2, \ldots, \ell_s^1, \ell_s^2$ such that $\sum_{j=1}^{s}(\ell_j^1 + \ell_j^2) = \alpha N^{1/3}$. Let h_j be a random variable whose value is the level the j'th starting vertex belongs to. Then the above probability equals

$$\begin{aligned}\sum_{j=1}^{s} \Pr\left[h_j \leq \ell_j^1 \text{ or } h_j \geq K - \ell_j^2\right] &= \sum_{j=1}^{s}\left(\frac{\ell_j^1}{K} + \frac{\ell_j^2}{K}\right) &(3)\\ &= \frac{1}{K}\cdot\sum_{j=1}^{s}(\ell_j^1 + \ell_j^2) &(4)\\ &= \alpha &(5)\end{aligned}$$

■

Proof of Theorem 4: Assume, contrary to the claim that there exists a testing algorithm A that uses at most $\frac{1}{4}N^{1/3}$ queries. Recall that by Lemma 6, the process P_1, when interacting with A, uniformly generates graphs in $\mathcal{G}_1$, and the process P_2, when interacting with A, uniformly generates graphs in $\mathcal{G}_2$. Since all graphs in $\mathcal{G}_1$ are acyclic, algorithm A, when interacting with P_1, should accept with probability at least 2/3. By Lemma 7, the statistical difference between D_1^{A} and D_2^{A} is at most $\frac{1}{4}(1+\frac{1}{4}) = \frac{5}{16}$. It follows that the probability that algorithm A, when interacting with P_2, accepts, is at least $\frac{2}{3} - \frac{5}{16} = \frac{17}{48}$. On the other hand, by Lemma 5 (and the lower bound on the size of N), much fewer than $\frac{1}{48}$ of the graphs in $\mathcal{G}_2$ are $\frac{1}{16}$-close to acyclic. Since A should accept each graph that is $\frac{1}{16}$-far from acyclic with probability smaller than 1/3, we get that the probability it accepts when interacting with P_2 is less than $\frac{1}{3} + \frac{1}{48} = \frac{17}{48}$, contradicting the lower bound we obtained above on this probability. ■

4.1 Difficulty of This Setting

We close this section by discussing the apparent difficulties that a testing algorithm must overcome, which leads us to believe that the problem may be harder than stated above.

Consider a directed graph G that is ϵ-far from acyclic. Let $\mathrm{C}_1, \ldots, \mathrm{C}_k$ be its *strongly connected components* (where some of these components may include only a single vertex). For $1 \leq i \leq k$, let e_i be the minimum number of edges that need to be removed from C_i in order to make it acyclic. Then, by definition, $\sum_i e_i > \epsilon dN$. In particular, this implies that there is a relatively large fraction of vertices that belong to strongly connected components for which e_i is relatively large. Thus, with non-negligible probability, a uniformly selected vertex will belong to such a component. However, in order to detect that the component contains cycles, the algorithm must "stay inside the component" (that is, perform many queries concerning edges between vertices in the component). The difficulty is how to distinguish between edges inside the components and edges between components (which "lead the algorithm out of the component").

In view of the above, we conjecture that some construction along the following lines could bring about a lower bound that improves on Theorem 4. Similarly to the proof of Theorem 4, we define two classes of (directed) graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, each over N vertices, with degree bound d. All graphs in $\mathcal{G}_1$ are acyclic, and almost all graphs in $\mathcal{G}_2$ are ϵ-far from acyclic (for some constant ϵ). In both classes, the vertices are partitioned into three subsets: V_ℓ, V_m and V_r (where ℓ stands for 'left', m stands for 'middle' and r stands for 'right'). The sizes of V_ℓ and V_r are twice the size of V_m. The edges between the sets only go from V_ℓ to V_m and from V_m to V_r.

In $\mathcal{G}_1$, the edges within the three sets are similar to those in the class of acyclic graphs constructed in the proof of Theorem 4, except that we allow for some variability in the number of incoming/outgoing edges each vertex has from/to the previous/next layer. In addition, vertices in the last layer of V_m have edges going to the first layer of V_m, and vertices in the last layer of V_m have edges going to the first layer of V_r. Finally, every vertex in V_m has one incoming edge from some vertex in the last half layers of V_ℓ and an edge going to some vertex in the first half layers of V_r. (The reason for variability in degrees is so that vertices in V_ℓ won't necessarily have a higher outdegree than those in V_m, and vertices in V_r won't all have higher indegrees.)

In $\mathcal{G}_2$, the edges within the sets V_ℓ and V_r are the same as in $\mathcal{G}_1$. The edges within V_m are similar to those defined for the class of graphs that are far from acyclic in the proof of Theorem 4, where again we allow for variability of degrees. Here too every vertex in V_m has one incoming edge from some vertex in the last half layers of V_ℓ and an edge going to some vertex in the first half layers of V_r. Finally, there are edges going from the last layer of V_ℓ to a subset of vertices of the same size in V_m, and similarly edges coming from another such subset to the first layer in V_ℓ (the first set may have only outgoing edges to other vertices in V_m, and the second may have only incoming edges from other vertices in V_m). In both classes the labels of the edges going between the sets vary (so there is no particular edge label that always "brings" from one set to the other).

The idea is that the only way to distinguish between the two classes is to ask a sufficient number of queries that correspond to edges within V_m. However, it is not clear how the algorithm can determine when it has "exited" V_m via an incoming edge to V_ℓ or an outgoing edge to V_r, so as to be able to "stay inside" V_m. On the other hand, it is an open question how to formalize the above intuition into a rigorous lower bound.

5 Testing Strong Connectivity

We say that a directed graph is strongly connected if there is a directed path from every vertex in the graph to every other vertex. As noted in the introduction, the natural model for testing this property is the incidence-lists model. We first show that if the testing algorithm can query not only on outgoing edges from each vertex, but also on incoming edges, then it is possible to test strong connectivity in $\tilde{O}(1/\epsilon)$ time. We next show that if the testing algorithm only has access to outgoing edges, then the same test requires $\Omega(\sqrt{N})$ queries. These contrasting results demonstrate how sensitive property testing algorithms can be to the choice of representation.

5.1 Testing Using Queries on both Outgoing and Incoming Edges

We first introduce some definitions. The *strongly connected components* of a graph $G = (V, E)$ are maximal subsets $C \subseteq V$ such that there is a directed path from each vertex in C to every other vertex in C. In particular, a strongly connected graph has a single strongly connected component, and for all graphs, the strongly connected components are disjoint. We define an auxiliary directed graph, $H(G)$, whose vertices correspond to the components of G. There is an edge from the vertex representing component C, to the vertex representing component C', if and only if, there is at least one edge in G from some vertex in C to some vertex in C'. This auxiliary graph is clearly acyclic, as a cycle in $H(G)$ would imply that all vertices belonging to components on the cycle are actually connected to each other (contradicting the maximality of the components). We say that a vertex in $H(G)$ is a *source* vertex, if it has only outgoing edges, and that it is a *sink* vertex, if it has only incoming edges. We refer to the corresponding components as source components and sink components, respectively.

The following lemma generalizes what is shown in [20] for undirected graphs. The lemma, and the discussion following refer to the bounded-degree incidence-lists model, but can be generalized to the case in which there is only a bound on the number of edges in the graph (as done for the undirected case in [29]). For sake of symmetry, we assume that both the outdegree and the indegree of the vertices are bounded by d.

Lemma 8 *If a directed graph* G *is* ϵ*-far from the class of strongly connected graphs on* N *vertices, and the maximum indegree and outdegree is* d*, then the number of vertices in* $H(G)$ *that are either source vertices or sink vertices is greater than* $\frac{\epsilon}{2} \cdot dN$*.*

Proof: Assume, contrary to what is claimed in the lemma, that there are at most $\frac{\epsilon}{2} \cdot dN$ source and sink vertices in $H(G)$. We next show that by adding (and possibly removing) at most $\epsilon \cdot dN$ edges to G we can make it strongly connected. But this contradicts the premise of the lemma by which G is ϵ-far from being strongly connected, and the lemma follows.

Assume first that in each component of G that corresponds to a source or sink vertex in $H(G)$ there is at least one vertex that has indegree at most $d-1$, and one vertex that has outdegree at most $d-1$. Consider any ordering $C_{\mathrm{so}1}, \ldots, C_{\mathrm{so}t}$ on the source components, and any ordering $C_{\mathrm{si}1}, \ldots, C_{\mathrm{si}r}$ on the sink components. The we can connect the components in a cycle $C_{\mathrm{so}1}, \ldots, C_{\mathrm{so}t}, C_{\mathrm{si}1}, \ldots, C_{\mathrm{si}r}$ (so that there is an edge going from each component to the next and from $C_{\mathrm{si}r}$ to $C_{\mathrm{so}1}$, where the edges are between the vertices having degree less than d). It is easy to verify that for any two vertices v and u, there is a directed path from v to u in the modified graph. The number of edges added is equal to the number of source and sink components, and so is bounded by $\frac{\epsilon}{2} \cdot dN$.

If in some component there is no vertex with either outdegree at most $d-1$ or indegree at most $d-1$, we first remove one edge between some pair of vertices v and u in the component, and then add edges as described above. If there is an additional path from v to u (going through other vertices in the component), then we did not affect the connectivity of the component. Otherwise, there is now a new path from v to u in the modified graph using the added edges. We thus obtain a strongly connected graph at the cost of at most $\epsilon \cdot dN$ edge modifications. ■

By using a simple counting argument, we obtain the following corollary:

Corollary 9 *If a graph* G *is ϵ-far from the class of N-vertex strongly connected graphs with indegree and outdegree bounded by d, then* G *has at least $\frac{\epsilon dN}{4}$ source and sink components each containing less than $\frac{4}{\epsilon d}$ vertices.*

This corollary suggests the following algorithm: Uniformly and independently select $m = \Theta(1/(\epsilon d))$ vertices in G. From each vertex selected, perform one Breadth First Search (BFS) using *outgoing* edges and one using *incoming* edges (that is, going against the direction of the edges). Stop each of these searches when $\frac{4}{\epsilon d}$ vertices have been reached or when no new vertex can be reached. (We assume that the total number of vertices in the graph is at least $\frac{4}{\epsilon d}$ or else in $O(1/\epsilon)$ time it is possible to exactly decide whether the graph is strongly connected by looking at the whole graph). In the latter case the algorithm has discovered a small source or sink component, implying that the graph is not strongly connected, and it rejects. If no search causes the algorithm to reject, it accepts.

Clearly, the algorithm always accepts a strongly connected graph. If the graph is ϵ-far from strongly connected then by Corollary 9 it has at least $\frac{\epsilon dN}{4}$ source and sink components with less than $\frac{4}{\epsilon d}$ vertices. The probability that none of the uniformly selected vertices belongs to such a component is at most $\left(1 - \frac{\epsilon d}{4}\right)^m$, which is less than $1/3$ for the appropriate constant in the $\Theta(\cdot)$ notation of m. Given that at least one such vertex is selected, the algorithm rejects the graph as required. The running time of the above algorithm is $O(1/(\epsilon^2 \cdot d))$ (as there are $O(1/(\epsilon \cdot d))$ starting vertices and from each the algorithm performs two searches each at a cost of at most $d \cdot \frac{4}{\epsilon d} = O(1/\epsilon)$. Using techniques from [20], it is possible to reduce the complexity of the algorithm by a factor of $\Omega(\log(1/(\epsilon d))/(\epsilon d))$ by slightly modifying the algorithm.

5.2 Testing Using Queries on Outgoing Edges Only

Unfortunately, if the algorithm can only perform queries about outgoing edges (as the basic model allows), then testing strong connectivity requires $\Omega(\sqrt{N})$ queries for constant d and ϵ. Below we sketch the details.

Consider the following two families of directed graphs (see Figure 2):

1. The first family, $\mathcal{G}_1$, consists of all graphs whose N vertices lie on one single directed cycle.
2. The second family, $\mathcal{G}_2$ consists of all graphs such that $N/2$ of the vertices lie on a cycle, and each of the remaining vertices has a single outgoing edge to a unique vertex on the cycle.

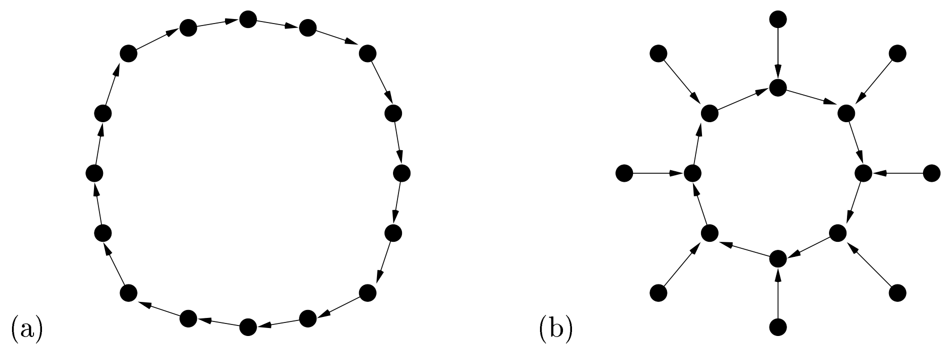

Figure 2: In (a), the structure of the graphs in $\mathcal{G}_1$ is shown. In (b), the structure of the graphs in $\mathcal{G}_2$ is shown. The graphs in a family differ only in the labels assigned to the vertices.

In both families, the outdegree of every vertex is 1, and the indegree is at most 2. All graphs in the first family are strongly connected, since there is a path along the cycle between every two vertices. On the other hand all graphs in the second family are at least $\frac{1}{2}$-far from the class of strongly connected graphs with degree bound 1, since each of the $N/2$ vertices with indegree 0 must have indegree 1 in order for the graph to be strongly connected.

However, we claim that for some constant $\alpha < 1$, any algorithm that performs less than $\alpha \cdot \sqrt{N}$ queries, cannot distinguish between a randomly selected graph in $\mathcal{G}_1$ and a randomly selected graph in $\mathcal{G}_2$ (with sufficient success probability). The structure of the proof is similar to that of Theorem 4 and is hence only sketched briefly.

Here too we define two processes that answer queries of any testing algorithm while constructing a uniformly chosen graph in one of the two families. The first process (which constructs graphs in $\mathcal{G}_1$) is very simple. Upon each new query (recall that there is only a single edge going out of each vertex), it either uniformly selects a new vertex that is not yet in the knowledge graph, or it selects (uniformly) a vertex that already belongs to the knowledge graph and has no incoming edge from another vertex in the knowledge graph. The decision between selecting a new vertex or a known vertex, is done randomly based on the current size of the knowledge graph. When the testing algorithm terminates, the process uniformly selects a graph in $\mathcal{G}_1$ that is consistent with the knowledge graph. The second process is similar, except that whenever a query about a new vertex is made, the algorithm decides whether the new vertex is a "cycle vertex" or an "outside vertex" (having indegree 0). This decision again is made according to the proportion of such vertices in the knowledge graph.

It is easy to verify that the two processes in fact construct uniformly selected graphs in the respective families of graphs. Furthermore, for both processes we can bound the probability that the process answers a query with a vertex already in the knowledge graph in any sequence of at most $\alpha\sqrt{N}$ queries. Given that such an event does not occur, then the answers to the queries in both cases are uniformly selected vertices, and no algorithm can distinguish between the two processes.

References

[1] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. In *Proceedings of the Fortieth Annual Symposium on Foundations of Computer Science*, pages 656–666, 1999.

[2] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. In *Proceedings of the Fortieth Annual Symposium on Foundations of Computer Science*, pages 645–655, 1999.

[3] S. Arora, A. Frieze, and H. Kaplan. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. In *37th Annual Symposium on Foundations of Computer Science*, pages 21–30, 1996.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *Journal of the Association for Computing Machinery*, 45(3):501–555, 1998.

[5] S. Arora and S. Safra. Probabilistic checkable proofs: A new characterization of NP. *Journal of the Association for Computing Machinery*, 45(1):70–122, 1998.

[6] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[7] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[8] B. Berger and P. W. Shor. Tight bounds for the maximum acyclic subgraph problem. *Journal of Algorithms*, 25(1):1–18, Oct. 1997.

[9] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of the Association for Computing Machinery*, 47:549–595, 1993.

[10] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonocity. In *Proceedings of RANDOM99*, 1999.

[11] F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. In *Proceedings of the Thirty-Second Annual ACM Symposium on the Theory of Computing*, pages 259–268, 1998.

[12] G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20, 1998.

[13] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *Journal of the Association for Computing Machinery*, 43(2):268–292, 1996.

[14] A. Frieze and R. Kanan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[15] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. *SIGMOD Record: Proc. ACM SIGMOD Int. Conf. Management of Data*, 27(2):331–342, 2–4 June 1998.

[16] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization*, A, 1999. to appear.

[17] M. Goemans and D. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Combinatorica*, 18, 1998.

[18] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samordinsky. Testing monotonicity. Manuscript. A preliminary (and weaker) version of this work appeared in FOCS98, 1999.

[19] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the Association for Computing Machinery*, 45(4):653–750, 1998. An extended abstract appeared in FOCS96.

[20] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 406–415, 1997. To appear in *Algorithmica*.

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

[22] R. Hassin and S. Rubinstein. Approximations for the maximum acyclic subgraph problem. *Information Processing Letters*, 51(3):133–140, Aug. 1994.

[23] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the Association for Computing Machinery*, 34(1):144–162, Jan. 1987.

[24] D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for machine scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[25] V. Kann. *On the Approximability of NP-Complete Optimization Problems.* PhD thesis, Department of Numberical Analysis and Computer Science, Royal Institute of Technology, Stockholm, 1992.

[26] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, New York, 1972. Plenum Press.

[27] M. Kearns and D. Ron. Testing problems with sub-learning sample complexity. In *Proceedings of the Eleventh Annual ACM Conference on Computational Learning Theory*, pages 268–279, 1998.

[28] C. Papadimitriou and M. Yannakakis. Optimization, approximization and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[29] M. Parnas and D. Ron. Testing the diameter of graphs. In *Proceedings of Random99*, pages 85–96, 1999.

[30] R. Rubinfeld. Robust functional equations and their applications to program testing. In *Proceedings of the Thirty-Fifth Annual Symposium on Foundations of Computer Science*, pages 288–299, 1994.

[31] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[32] P. D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15, 1995.