Tight Bounds for Testing Bipartiteness in General Graphs

Tali Kaufman[1]*, Michael Krivelevich[2]**, and Dana Ron[3]***

[1] School of Computer Science, Tel Aviv University,Tel Aviv 69978 Israel.
[2] Department of Mathematics, Tel Aviv University, Tel Aviv 69978, Israel.
[3] Department of Electrical Engineering-Systems, Tel Aviv University, Tel Aviv 69978, Israel.

Abstract. In this paper we consider the problem of testing bipartiteness of general graphs. The problem has previously been studied in two models, one most suitable for dense graphs, and one most suitable for bounded-degree graphs. Roughly speaking, dense graphs can be tested for bipartiteness with constant complexity, while the complexity of testing bounded-degree graphs is $\tilde{\Theta}(\sqrt{n})$, where n is the number of vertices in the graph. Thus there is a large gap between the complexity of testing in the two cases.
In this work we bridge the gap described above. In particular, we study the problem of testing bipartiteness in a model that is suitable for all densities. We present an algorithm whose complexity is $\tilde{O}(\min(\sqrt{n}, n^2/m))$ where m is the number of edges in the graph, and match it with an almost tight lower bound.

1 Introduction

Property testing algorithms [16, 8] are algorithms that perform *approximate decisions*. Namely, for a predetermined property P they should decide whether a given object O has property P or is *far* from having property P. In order to perform this approximate decision they are given *query access* to the object O. Property testing problems are hence defined by the type of objects in question, the property tested, the type of queries allowed, and the notion of distance to having a property. Much of the focus of property testing has been on testing properties of graphs. In this context several models have been considered. In all models, for a fixed graph property P, the algorithm is required to accept graphs that have P and to reject graphs that are ϵ-far from having P, for a given distance parameter ϵ. In all cases the algorithm is allowed a constant probability of failure. The models differ in the type of queries they allow and in the notion of distance they use (which underlies the definition of being ϵ-far from having the property). The complexity of the algorithm is measured by the number of queries to the object Q it performs.

1.1 Models for Testing Graph Properties

The first model, introduced in [8], is the adjacency-matrix model. In this model the algorithm may perform queries of the form: "Is there an edge between vertices u and v

* E-mail: kaufmant@post.tau.ac.il, This work is part of the author's Ph.D. thesis prepared at Tel Aviv University under the supervision of Prof. Noga Alon, and Prof. Michael Krivelevich.

** E-mail: krivelev@post.tau.ac.il.

*** E-mail: danar@eng.tau.ac.il. Research supported by the Israel Science Foundation (grant number 32/00-1).

in the graph?" That is, the algorithm may probe the adjacency matrix representing the graph. We refer to such queries as *vertex-pair* queries. The notion of distance is also linked to this representation: a graph is said to be ϵ-far from having property P if more than ϵn^2 edge modifications should be performed on the graph so that it obtains the property, where n is the number of vertices in the graph. In other words, ϵ measures the fraction of entries in the adjacency matrix of the graph that should be modified. This model is most suitable for *dense* graphs in which the number of edges m is $\Theta(n^2)$. This model was studied in [8, 3, 2, 1, 4, 11, 7].

The second model, introduced in [9], is the (bounded-degree) incidence-lists model. In this model, the algorithm may perform queries of the form: "Who is the i'th neighbor of vertex v in the graph?" That is, the algorithm may probe the incidence lists of the vertices in the graph, where it is assumed that all vertices have degree at most d for some fixed degree-bound d. We refer to these queries as *neighbor* queries. Here too the notion of distance is linked to the representation: A graph is said to be ϵ-far from having property P if more than ϵdn edge modifications should be performed on the graph so that it obtains the property. In this case ϵ measures the fraction of entries in the incidence lists representation (among all dn entries), that should be modified. This model is most suitable for graphs with $m = \Theta(dn)$ edges; that is, whose maximum degree is of the same order as the average degree. In particular, this is true for *sparse* graphs that have *constant degree*. This model was studied in [10, 9, 6].

In [15] it was suggested to decouple the questions of representation and type of queries allowed from the definition of distance to having a property. Specifically, it was suggested to measure the distance simply with respect to the number of edges, denoted m, in the graph. Namely, a graph is said to be ϵ-far from having a property, if more than ϵm edge modifications should be performed so that it obtains the property. In [15] the algorithm was allowed the same type of queries as in the bounded-degree incidence-lists model, but no fixed upper-bound was assumed on the degrees and the algorithm could query the degree of any vertex. The main advantage of this model over the bounded-degree incidence-lists model is that it is suitable for graphs whose degrees may vary significantly.

The Model Studied in this Paper. In this work we are interested in a model that may be useful for testing all types of graphs: dense, sparse, and graphs that lie in-between the two extremes. As is discussed in more detail in the next subsection, the two extremes sometimes exhibit very different behavior in terms of the complexity of testing the same property. We are interested in understanding the transformation from testing sparse (and in particular bounded-degree) graphs to testing dense graphs.

Recall that a model for testing graph properties is defined by the distance measure used and by the queries allowed. The model of [15] is indeed suitable for all graphs in terms of the distance measure used, since distance is measured with respect to the actual number of edges m in the graph.[1] Thus this notion of distance adapts itself to the density of the graph, and we shall use it in our work.

[1] We assume for simplicity that the number of vertices, n, and the number of edges, m, are both given to the testing algorithm. If they are not known exactly, the algorithm can work using upper bounds on these values. The tightness of these bounds will naturally affect the performance of the algorithm.

The focus in [15] was on testing properties that are of interest in sparse (but not necessarily bounded-degree) graphs, and hence they allowed only neighbor queries. However, consider the case in which the graph is not sparse (but not necessarily dense). In particular suppose that the graph has $\omega(n^{1.5})$ edges, and that we are seeking an algorithm that performs $o(\sqrt{n})$ queries. While in the case of sparse graphs, there is no use in asking vertex-pair queries (i.e., is there an edge between a particular pair of vertices), such queries may become helpful when the number of edges is sufficiently large. Hence, we allow our algorithms to perform both neighbor queries and vertex-pair queries.

1.2 Testing Bipartiteness

One of the properties that has received quite a bit of attention in the context of property testing, is *bipartiteness*. Recall that a graph is bipartite if it is possible to partition its vertices into two parts such that there are no edges with both endpoints in the same part. This property was first studied in [8] where it was shown that bipartiteness can be testing by a simple algorithm using $\tilde{O}(1/\epsilon^3)$ queries. This was improved in [3] to $\tilde{O}(1/\epsilon^2)$ queries. The best lower bound known in this model is $\tilde{\Omega}(1/\epsilon^{1.5})$, due to [7]. Thus the complexity of this problem is independent of the number of vertices n and polynomial in $1/\epsilon$.

The complexity of testing bipartiteness changes significantly when considering the bounded-degree incidence-lists model. In [10] a lower bound of $\Omega(\sqrt{n})$ is established in this model, for constant ϵ and d (the degree bound). An almost matching upper bound of $\tilde{O}(\sqrt{n} \cdot \mathrm{poly}(1/\epsilon))$ is shown in [9]. Thus, in the case of bipartiteness there is a large gap between the results that can be obtained for dense graphs and for constant-degree graphs. Here we venture into the land of graphs that are neither necessarily sparse, nor necessarily dense, and study the complexity of testing bipartiteness. Other graph properties exhibit similar (and sometimes even larger) gaps, and hence we believe that understanding the transformation from sparse to dense graphs is of general interest.

1.3 Our Results

In this work we present two complementary results for n-vertex graphs having m edges:

- We describe and analyze an algorithm for testing bipartiteness in general graphs whose query complexity (and running time) is $O(\min(\sqrt{n}, n^2/m) \cdot \mathrm{poly}(\log n/\epsilon))$. The algorithm has a one-sided error (i.e., it always accepts bipartite graphs). Furthermore, whenever it rejects a graph it provides *evidence* that the graph is not bipartite in the form of an odd-length cycle of length $\mathrm{poly}(\log n/\epsilon)$.
- We present an almost matching lower bound of $\Omega(\min(\sqrt{n}, n^2/m))$ (for a constant ϵ). This bound holds for all testing algorithms (that is, for those which are allowed a two-sided error and are adaptive). Furthermore, the bound holds for regular graphs.

As seen from the above expressions, as long as $m = O(n^{1.5})$, that is, the average degree is $O(\sqrt{n})$, the complexity of testing is $\tilde{\Theta}(\sqrt{n})$. Once the number of edges goes above $n^{1.5}$, we start seeing a decrease in the query complexity which in this case is at most $O((n^2/m) \cdot \mathrm{poly}(\log n/\epsilon))$. In terms of our algorithm, this is exactly the point where our algorithm starts exploiting its access to vertex-pair queries. Our lower bound shows that this behavior of the query complexity is not only an artifact of our algorithm but is inherent in the problem.

Note that even if the graph is sparse then we obtain a new result that does not follow from [9]. Namely, we have an algorithm with complexity $\tilde{O}(\sqrt{n} \cdot \mathrm{poly}(1/\epsilon))$ for sparse graphs with varying degrees.

1.4 Our Techniques

We present our algorithm in two stages. First we describe an algorithm that works for almost-regular graphs, that is, graphs in which the maximum degree is of the same order as the average degree. The algorithm and its analysis closely follow the algorithm and analysis in [9]. Indeed, as long as the degree d of the graph is at most $\sqrt{n}$, we execute the [9] algorithm. The place where we depart from [9] is in the usage of vertex-pair queries once $d > \sqrt{n}$. We refer to our first algorithm as Test-Bipartite-Reg.

In the second stage we show how to reduce the problem of testing bipartiteness of general graphs to bipartiteness of almost-regular graphs. Namely, we show how, for every given graph G, it is possible to define a graph G' such that: (1) G' has roughly the same number of vertices and edges as G, and its maximum degree is of the same order as its average degree (which is roughly the same as the average degree in G); (2) If G is bipartite then so is G', and if G is far from bipartite then so is G'. We then show how to emulate the execution of the algorithm Test-Bipartite-Reg on G' given query access to G, so that we may accept G if it accepts G', and reject G if it rejects G'.

In the course of this emulation we are confronted with the following interesting problem: We would like to sample vertices in G according to their degrees (which aids us in sampling vertices uniformly in G', a basic operation that is required by Test-Bipartite-Reg). The former is equivalent to sampling *edges* uniformly in G. In order not to harm the performance of our testing algorithm, we are required to perform this task in $\tilde{O}(\min(\sqrt{n}, n^2/m))$ queries. If m is sufficiently large (once again, if $m \geq n^{1.5}$), this can be performed simply by sampling sufficiently many pairs of vertices in G. However, we do not know how to perform this task exactly (in an efficient manner) when the number of edges is significantly smaller than $n^{1.5}$. Nonetheless, we provide a sampling procedure that selects edges according to a distribution that approximates the desired uniform distribution on edges, and is sufficient for our purposes. The approximation is such that for all but a small fraction of the m edges, the probability of selecting an edge is $\Omega(1/m)$. This procedure may be of independent interest.

We also conjecture that variants of our construction of G' (and in particular a probabilistic construction we suggest in the long version of this paper [12]), may be useful in transforming other results that hold for graphs whose maximum degree is similar to their average degree, to results that hold for graphs with varying degrees.

We establish our lower bound by describing, for every pair n, d (n even, $d \geq 64$), two distributions over d-regular graphs. In one distribution all graphs are bipartite by construction. For the other distribution we prove that almost all graphs are far from bipartite. We then show that every testing algorithm that can distinguish between a graph chosen randomly from the first distribution (which it should accept with probability at least $2/3$), and a graph chosen randomly from the second distribution (which it should reject with probability at least $2/3$), must perform $\Omega(\min(\sqrt{n}, n/d)) = \Omega(\min(\sqrt{n}, n^2/m)$ queries. In the lower bound proof we show the necessity of both *neigbhor queries* and *vertex-pair queries*. Specifically by using only one type of queries the lower bound increases.

1.5 Further Research

As noted previously, there are other problems that exhibit a significant gap between the query complexity of testing dense graphs (in the adjacency-matrix model) and the complexity of testing sparse, bounded-degree graphs (in the bounded-degree incidence-lists model). In particular this is true for testing k-colorability. It is possible to test dense graphs for k-colorability using $\mathrm{poly}(k/\epsilon)$ queries [8, 3], while testing sparse graphs requires $\Omega(n)$ queries [6]. We stress that these bounds are for query complexity, where we put time complexity aside. We would like to understand this transformation from essentially constant complexity (for constant k and ϵ) to linear complexity, and we would like to know whether any intermediate results can be obtained for graphs that are neither sparse nor dense. Other problems of interest are testing whether a graph has a relatively large clique [8], testing acyclicity of directed graphs [5], and testing that a graph does not contain a certain subgraph [1].

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with n vertices labeled $1, ..., n$, and let $m = m(G) = |E(G)|$ be the total number of edges in G. Unless stated otherwise, we assume that G contains no multiple edges. For each vertex $v \in V$ let $\Gamma(v)$ denote its set of neighbors, and let $\deg(v) = |\Gamma(v)|$ denote its degree. The edges incident to v (and their end-points, the neighbors of v), are labelled from 1 to $\deg(v)$. Note that each edge has two, possibly different, labels, one with respect to each of its end-points. We hence view edges as quadruples. That is, if there is an edge between v and u, and it is the i-th edge incident to v and the j-th edge incident to u, then this edge is denoted by (u, v, i, j). When we want to distinguish between the quadruple (u, v, i, j) and the pair (u, v) then we refer to the latter as an *edge-pair*. We let $d_{\max} = d_{\max}(G)$ denote the maximum degree in the graph G and $d_{\mathrm{avg}} = d_{\mathrm{avg}}(G)$ denote the average degree in the graph (that is, $d_{\mathrm{avg}}(G) = 2m(G)/n$).

Distance to having a property. Consider a fixed graph property $\mathcal{P}$. For a given graph G, let $e_{\mathcal{P}}(G)$ be the minimum number of edges that should be added to G or removed from G so that it obtain property $\mathcal{P}$. The distance of G to having property $\mathcal{P}$ is defined as $e_{\mathcal{P}}(G)/m(G)$. In particular, we say that graph G is *ϵ-far* from having the property $\mathcal{P}$ for a given distance parameter $0 \leq \epsilon < 1$, if $e_{\mathcal{P}}(G) > \epsilon \cdot m(G)$. Otherwise, it is *$\epsilon$-close* to having property $\mathcal{P}$. In some cases we may define the distance to having a property with respect to an upper bound $m_{\max} \geq m(G)$ on the number of edges in the graph (that is, the distance to having property $\mathcal{P}$ is defined as $e_{\mathcal{P}}(G)/m_{\max}$). For example, if the graph is dense, so that $m(G) = \Omega(n^2)$ then we set $m_{\max} = n^2$, and alternatively, if the graph has some bounded degree d, then we set $m_{\max} = d \cdot n$. (In the latter case we could set $m_{\max} = (d \cdot n)/2$, but for simplicity we set the slightly higher upper bound.) If $e_{\mathcal{P}}(G)/m_{\max} > \epsilon$ then we shall say that the graph is ϵ-far from property $\mathcal{P}$ *with respect to* $m_{\max}$.

Testing algorithms. A testing algorithm for a graph property $\mathcal{P}$ is required to accept with probability at least $2/3$ every graph that has property $\mathcal{P}$ and to reject with probability at least $2/3$ every graph that is ϵ-far from having property $\mathcal{P}$, where ϵ is a given distance parameter. If the algorithm always accepts graphs that have the property then it is a *one-sided error* algorithm. The testing algorithm is given the number of vertices in the graph, the number of edges in the graph, or an upper bound on this number, and it is provided with *query access* to the graph. Specifically we allow the algorithm the following types of queries.

- The first type of queries are *degree* queries. That is, for any vertex u of its choice, the algorithm can obtain $\deg(u)$. We assume that a degree query has cost one. In fact it can be easily implemented using neighbor queries with cost $O(\log d_{\max}) = O(\log n)$.
- The second type of queries are *neighbor* queries. Namely, for every vertex u and index $1 \leq i \leq \deg(u)$, the algorithm may obtain the i-th neighbor of vertex u.
- The third type of queries are *vertex-pair* queries. Namely, for any pair of vertices (u, v), the algorithm can query whether there is an edge between u and v in G.

Bipartiteness. In this work we focus on the property of *bipartiteness*. Let (V_1, V_2) be a partition of V. We say that an edge $(u, v) \in E$ is a *violating* edge with respect to (V_1, V_2), if u and v belong to the same subset V_b, (for some $b \in \{1, 2\}$). A graph is *bipartite* if there exists a partition of its vertices with respect to which there are no violating edges. By definition, a graph is ϵ-far from bipartite if for every partition of its vertices, the number of violating edges with respect to the partition is greater than $\epsilon \cdot m$. Recall that a graph is bipartite if and only if it contains no odd-length cycles.

3 The Algorithm for the Almost-Regular Case

In this section we describe an algorithm that accepts every bipartite graph and that rejects with probability at least $2/3$ every graph that is ϵ-far from bipartite with respect to an upper bound $m_{\max} = d_{\max}n$ on the number of edges. Namely, this algorithm rejects (with probability at least $2/3$) graphs for which the number of edges that need to be removed so that they become bipartite is greater than $\epsilon \cdot m_{\max} = \epsilon \cdot d_{\max}n$. The query complexity (and running time) of this algorithm is $O(\min(\sqrt{n}, n/d_{\max}) \cdot \text{poly}(\log n/\epsilon))$.

In the case where the graph is almost-regular, that is, the maximum degree of the graph $d_{\max}$ is of the same order as the average degree, d_{avg}, then we essentially obtain a tester as desired (since in such a case $\epsilon d_{\max}n = O(\epsilon m)$). However, in general, $d_{\max}$ may be much larger d_{avg} (for example, it is possible that $d_{\max} = \Theta(n)$ while $d_{\text{avg}} = \Theta(1)$). To deal with the general case we show in the next section (Section 4) how to reduce the problem in the general case to the special case of $d_{\max} = O(d_{\text{avg}})$.

A High Level Description of the Algorithm. Throughout this section let $d = d_{\max}$. Our algorithm builds on the testing algorithm for bipartiteness described in [9] whose query complexity is $O(\sqrt{n} \cdot \text{poly}(\log n/\epsilon))$ (and which works with respect to $m_{\max} = dn$ as well). In fact, as long as $d \leq \sqrt{n}$ our algorithm is equivalent to the algorithm in [9]. In particular, as in [9], our algorithm selects $\Theta(1/\epsilon)$ *starting vertices* and from each it performs several random walks (using neighbor queries), each walk of length $\text{poly}(\log n/\epsilon)$. If $d \leq \sqrt{n}$ then the number of these walks is $O(\sqrt{n} \cdot \text{poly}(\log n/\epsilon))$, and the algorithm simply checks whether an odd-length cycle was detected in the course of these random walks (possibly relying on information from more than one random walk to find an odd cycle).

If $d > \sqrt{n}$ then there are two important modifications: (1) The number of random walks performed from each vertex is reduced to $O(\sqrt{n/d} \cdot \mathrm{poly}(\log n/\epsilon))$; (2) For each pair of end vertices reached in these walks with the same parity, the algorithm performs a vertex-pair query. Similarly to the $d \leq \sqrt{n}$ case, the graph is rejected if an odd-length cycle is found in the subgraph induced by all queries performed. Pseudo-code for the algorithm is shown in Figure 1.

Random Walks and Paths in the Graph. The random walks performed are defined as follows: At each step, if the degree of the current vertex v is $d' \leq d$, then the walk *remains* at v with probability $1 - \frac{d'}{2d} \geq \frac{1}{2}$, and for each $u \in \Gamma(v)$, the walk *traverses* to u with probability $\frac{1}{2d}$. The important property of the random walk is that the stationary distribution it induces over the vertices is uniform.

For every walk (or, more generally, for any sequence of steps), there corresponds a *path* in the graph. The path is determined by those steps in which an edge is traversed (while ignoring all steps in which the walk stays at the same vertex). Such a path is not necessarily simple, but does not contain self loops. Note that when referring to the length of a walk, we mean the total number of steps taken, including steps in which the walk remains at the current vertex, while the length of the corresponding path does not include these steps.

```
**Test-Bipartite-Reg**($n, d_{\max}, \epsilon$)
• Repeat $T = \Theta(\frac{1}{\epsilon})$ times:
    1. Uniformly select $s$ in $V$.
    2. If Odd-Cycle($s$) returns found then output reject.
• In case no call to Odd-Cycle returned found then output accept.
```

```
**Odd-Cycle(s)**
  1. If $d = d_{\max} \leq \sqrt{n}$ then let $K \stackrel{\text{def}}{=} \Theta(\frac{\log^{1/2}(n/\epsilon)}{\epsilon^3})$ and $L \stackrel{\text{def}}{=} \Theta(\frac{\log(n/\epsilon)^3}{\epsilon^5})$. Otherwise
     ($d > \sqrt{n}$), let $K \stackrel{\text{def}}{=} \Theta\left(\frac{\log^{1/2}(n/\epsilon)\cdot\sqrt{n/d}}{\epsilon^8}\right)$, and $L \stackrel{\text{def}}{=} \Theta\left(\frac{\log^6(n/\epsilon)}{\epsilon^8}\right)$.
  2. Perform $K$ random walks starting from $s$, each of length $L$.
  3. Let $A_0$ ($A_1$) be the set of vertices that appear on the ends of the $K$ walks whose paths are
     of even (odd) length.
  4. If $d \leq \sqrt{n}$ then check whether $A_0 \cap A_1 \neq \emptyset$. If the intersection is non-empty then return
     found, otherwise return not-found.
  5. Else ($d > \sqrt{n}$), perform vertex-pair queries between every pair of vertices $u, v \in A_0$
     ($u, v \in A_1$). If an edge is detected then return found, otherwise return not-found.
```

Fig. 1. Algorithm Test-Bipartite-Reg for testing bipartiteness with respect to the upper bound $m_{\max} = d_{\max} \cdot n$ on the number of edges, and the procedure Odd-Cycle for detecting odd-length cycles in the graph G.

Theorem 1 *The algorithm Test-Bipartite-Reg accepts every graph that is bipartite, and rejects with probability at least* $2/3$ *every graph that is* ϵ*-far from bipartite with respect to* $m_{\max} = d_{\max}n$*. Furthermore, whenever the algorithm rejects a graph it outputs a* certificate *to the non-bipartiteness of the graph in form of an odd-length cycle of length* $\mathrm{poly}(\log n/\epsilon)$*. The query complexity and running time of the algorithm are* $O(\min(\sqrt{n}, n/d_{\max}) \cdot \mathrm{poly}(\log n/\epsilon))$.

Note that the algorithm can work when G contains self-loops and multiple-edges. The latter will be of importance in the next section. The corollary below will become useful in the next section as well.

Corollary 2 *If G is ϵ-far from bipartite with respect to $m_{\max} = d_{\max} n$, then $\Omega(\epsilon)$-fraction of its vertices s are such that* Odd-Cycle(s) *returns* found *with probability at least $\frac{2}{3}$.*

Since the proof of Theorem 1 has similar structure to the proof given in [9], we omit it from this extended abstract. All details of this proof, as well as other proofs, can be found in the full version of this paper [12].

4 The Algorithm for the General Case

In this section we build on the testing algorithm presented in the previous section and show a one-sided error bipartite testing algorithm that works with respect to the actual number of edges $m = m(G)$. Hence this algorithm is suitable for general graphs (for which $d_{\max}$ may vary significantly from d_{avg}). The query complexity and running time of the algorithm are of the same order of magnitude as for Test-Bipartite-Reg, that is, $O(\min(\sqrt{n}, n^2/m) \cdot \text{poly}(\log n/\epsilon))$. We note that once the graph becomes very dense, that is $m = \Omega(n^2/\log^c n)$ (where c is approximately 4), it is preferable to use the adjacency-matrix model algorithm [8, 3] with distance parameter $\epsilon/(n^2/m)$.

A High Level Description of the Algorithm. The basic idea is to reduce the problem of testing with respect to the actual number of edges m to the problem of testing with respect to the upper bound $m_{\max} = d_{\max} \cdot n$. Specifically, for any graph G we show how to define a graph G' over $\Theta(n)$ vertices that has the following useful properties. First, the maximum degree in G' is roughly the same as the average degree, and furthermore, this degree is roughly the same as the average degree in G. In particular this implies that the two graphs have roughly the same number of edges. Second, G' approximately preserves the distance of G to bipartiteness. More precisely, if G is bipartite then so is G', but if G is far from bipartite with respect to $m(G)$, then G' is far from bipartite with respect to $m_{\max} = d_{\max}(G')n'$. Thus G' can be viewed as a kind of "regularized-degree version" of G.

If we had direct access to G', then by the above we would be done: by running the algorithm Test-Bipartite-Reg on G' we could decide whether G is bipartite or far from bipartite. However, we only have access to G. Nonetheless, given query access to G we can efficiently "emulate" queries in G'. This would almost suffice for running Test-Bipartite-Reg on G'. One more issue is the uniform selection of starting vertices in G', required by Test-Bipartite-Reg. As we shall see, selecting a vertex uniformly from G' is (roughly) equivalent to uniformly selecting an edge in G. We shall approximate the latter process.

In what follows we assume that $m \geq n'$ and that there are no multiple edges (where we can actually deal with the case in which there are multiple edges but they do not constitute more than a constant fraction of the total number of edges).

The main theorem of this subsection follows.

Theorem 3 *For every graph G having n vertices and $m \geq n$ edges, we can define a graph G' having n' vertices and m' edges for which the following holds:*

1. *$n \leq n' \leq 3n$, $m \leq m' \leq 6m$, and $d_{\max}(G') \leq 2d_{\text{avg}}(G)$.*
2. *If G is bipartite then G' is bipartite, and if G is ϵ-far from bipartite with respect to m, then G' is ϵ'-far from bipartite with respect to $m_{\max}(G') = d_{\max}(G')n'$ for $\epsilon' = \Theta(\epsilon)$.*
3. *Given a starting vertices s in G', it is possible to emulate random walks in G' starting from s, by performing queries to G. The amortized cost of each random walk step is $O(\log^2 n)$ (degree and neighbor) queries in G. By emulating these random walks it is possible to execute a slight variant of* Odd-Cycle(s) *in G' which we denote* Odd-Cycle'(s). *This variant is such that* $\Pr[\text{Odd-Cycle'}(s)\text{=found}] \geq \Pr[\text{Odd-Cycle}(s)\text{=found}]$, *where if* Odd-Cycle'($s$) *returns* found, *then we can obtain an odd-length cycle of length* $\text{poly}(\log n/\epsilon)$ in the original graph G.
4. *There exists a procedure* Sample-Vertices-Almost-Uniformly-in-G' *that for any given parameter $0 < \delta \leq 1$, performs $\tilde{O}(\min(\sqrt{n/\delta}, n^2/m))$ queries in G and returns a vertex in G' such that the following holds: For all but at most $\delta n'$ of the vertices x in G', the probability that x is selected by the procedure is $\Omega(1/n')$.*

We note that for every graph G there is actually a *family* of graphs G' with the above properties (all defined over the same set of vertices). When we run algorithm Test-Bipartite-Gen, we construct one such (arbitrary) graph G' in the family as we go along. As a corollary to Theorem 3 and Corollary 2 we obtain:

Corollary 4 *Algorithm Test-Bipartite-Gen (see Figure 2) accepts every graph G that is bipartite, and rejects with probability at least $2/3$ every graph G that is ϵ-far from bipartite (with respect to $m(G)$). Furthermore, whenever the algorithm rejects a graph it outputs a* certificate *to the non-bipartiteness of the graph G in form of an odd-length cycle of length* $\text{poly}(\log n/\epsilon)$.
The query complexity and running time of the algorithm are $O\left(\min(\sqrt{n}, n^2/m) \cdot \text{poly}(\log n/\epsilon)\right)$.

```
Test-Bipartite-Gen($n, d_{\text{avg}}, \epsilon$)
• Repeat $T = \Theta(\frac{1}{\epsilon})$ times:
    1. Set $\epsilon' = \epsilon/108$.
    2. Select a vertex $s$ in $G'$ by calling the procedure Sample-Vertices-Almost-Uniformly-in-
       G' with $\delta = \epsilon'/c$ (where $c$ is a sufficiently large constant).
    3. Apply Odd-Cycle'($s$).
    4. If Odd-Cycle'($s$) returns found then output reject.
• In case no call to Odd-Cycle' returned found then output accept.
```

Fig. 2. Algorithm Test-Bipartite-Gen for testing bipartiteness with respect to the actual number of edges $m = m(G)$ in the graph G.

4.1 Defining G' and proving the first item in Theorem 3

In all that follows, let $d = d_{\text{avg}}(G)$, and let $d' = d_{\max}(G')$. We shall assume that d is a sufficiently large constant. If $d_{\text{avg}}(G)$ is not sufficiently large then we still set d in the construction below to be sufficiently large, and run the algorithm with ϵ set to $\epsilon/(d/d_{\text{avg}}(G))$.

The Construction of G'. For each vertex v in G such that $\deg(v) \leq d$, we have a single vertex in G'. For each vertex v in G such that $\deg(v) > d$ we have in G' a subgraph, denoted $H(v)$. It is a bipartite graph over two subsets of vertices, one denoted $X(v)$, the *external* part, and one denoted $I(v)$, the *internal* part. Both parts consist of $\lceil \deg(v)/d \rceil$ vertices. Every vertex in $X(v)$ represents up to d specific neighbors of v according to some fixed, *but arbitrary* partition of the neighbors of v. We refer to the vertices in the two subsets by $\{X_i(v)\}_{i=1}^{\lceil \deg(v)/d \rceil}$ and $\{I_i(v)\}_{i=1}^{\lceil \deg(v)/d \rceil}$, respectively. The edges in $H(v)$ are determined as follows. In case $\deg(v)/d < d$ then we have $\lceil d^2/\deg(v) \rceil$-multiple edges between every internal vertex and every external vertex in $H(v)$. In case $\deg(v)/d \geq d$, denote $s = \lceil \deg(v)/d \rceil$ and let $H(v)$ be a bipartite expander where each of its sides has s vertices ($s \geq d$). Each vertex in $H(v)$ has degree d. All eigenvalues of the adjacency matrix of H, but the largest one and the smallest one (which are equal to d and $-d$, respectively), are at most $d/4$ in their absolute values. Explicit constructions of such expanders can be found, e.g., in [14, 13]. Furthermore, these constructions allow the determination of the i-th neighbor of any given vertex in constant time.

We have described how vertices of G are transformed into vertices of G'. It remains to describe the relevant transformation to the edges of G. Consider an edge $(u, v) \in E(G)$ where v is the i-th neighbor of u and u is the j-th neighbor of v. Let $X_k(u)$ and $X_\ell(v)$ be the external vertices representing the i-th neighbor of u, and the j-th neighbor of v, respectively. Then, there is an edge $(X_k(u), X_\ell(v))$ in G'. It directly follows that every vertex in G' has degree at most $2d$ and that $n' = |V(G')| \leq \sum_{v \in G} 2\lceil \deg(v)/d \rceil \leq 3n$, and $m' = m(G') \leq 3dn = 6m$.

In the long version of this paper [12] we suggest the following alternative probabilistic construction of G' that establishes Theorem 3. Every vertex of G is transformed into $\lceil \deg(v)/d \rceil$ vertices. Denote by $X(v)$ the vertices in G' related to a vertex $v \in V(G)$. The vertices in $X(v)$ are denoted by $X_i(v), 1 \leq i \leq \lceil \deg(v)/d \rceil$. Thus, $n' = |V(G')| \leq \sum_{v \in G} \lceil \frac{\deg(v)}{d} \rceil \leq 2n$. The edges of G' are determined as follows: an edge $(u, v) \in E(G)$ chooses independently uniformly at random a vertex from $X(v)$ and a vertex from $X(u)$. In G' there will be an edge between these two randomly chosen vertices. Clearly, $m' = |E(G')| = |E(G)| = (nd)/2$.

The probabilistic construction is simpler and more robust than the deterministic one, and it may be applicable to other problems as well. However in this construction we need that $d = \Omega(1/\epsilon)$.

4.2 Establishing Items 2 and 3 in Theorem 3

The proofs of these two items are ommitted from this extended abstract, and can be found in [12]. We note that Item 2 builds on the expander graphs defined in the construction of G'.

4.3 Establishing Item 4 in Theorem 3

In this subsection we provide a sketch for the proof of the last item in Theorem 3. Consider the construction of G'. Sampling a vertex uniformly at random from G' is equivalent to sampling a vertex v from G with probability proportional to its degree (and then taking randomly and uniformly one of the vertices belong to $H(v)$). The latter is equivalent to sampling randomly uniformly an edge from G, and taking one of its end-points at random. Thus, the proof of this item is based on a presentation of a procedure for sampling edges almost uniformly from G.

We consider two cases: $d > \sqrt{\delta n}$ and $d \leq \sqrt{\delta n}$ (recall that $d = d_{\text{avg}}(G)$ is the average degree in G and that our goal is to use $\tilde{O}(\min(\sqrt{n/\delta}, n/d))$ queries to G). The first case is easy since if G contains sufficiently many edges then we simply sample $\Theta(n/d) = \Theta(n^2/m)$ pairs of vertices in order to obtain an edge.

In the second case, where G contains fewer edges ($d \leq \sqrt{\delta n}$), we do not have an algorithm that selects an edge uniformly from G (using relatively few queries). However, we can show the following lemma, from which Item 4 in Theorem 3 can be derived. The proof of this lemma can be found in [12].

Lemma 1 *There exists a procedure* Sample-Edges-almost-Uniformly-in-G *that uses* $\tilde{O}(\sqrt{n/\delta})$ *degree and neighbor queries in G and for which the following holds: For all but* $(\delta/4)m$ *of the edges* e *in G, the probability that the procedure outputs* e *is at least* $1/(64m)$. *Furthermore, there exists a subset* $U_0 \subset V(G)$, $|U_0| \leq (\delta n/2)$, *such that for all edges* $e = (u, v)$ *that are output with probability less than* $1/(64m)$, *we have* $u, v \in U_0$.

5 A Lower Bound

In this section we present a lower bound on the number of queries necessary for testing bipartiteness. Similarly to the lower bound presented in [9], this lower bound holds for testing algorithms that are allowed a two-sided error, and the graphs used for the lower bound construction are regular graphs. However, the lower bound of $\Omega(\sqrt{n})$ (for constant ϵ) established in [9], holds for graphs having constant degree (e.g., degree 3), and when the algorithm is allowed only neighbor queries. Our lower bound is more general in that it allows the algorithm to perform both neighbor queries and vertex-pair queries, and it is applicable to all graphs.

Theorem 5 *Every algorithm for testing bipartiteness with distance parameter* $\epsilon \leq 2^{-4}$ *must perform* $\Omega(\min(\sqrt{n}, n^2/m))$ *queries.*

The high-level structure of our proof is similar to other lower-bound proofs for testing, which can be traced back to [17]. We present two distributions over graphs, where all graphs generated by one distribution are bipratite (and hence should be accepted), while with very high probability a graph generated according to the other distribution is far from bipartite. We then show that any algorithm with query complexity below the lower bound, cannot distinguish between the two distributions (and hence must have a large failure probability).

Specifically, both distributions, denoted $\mathcal{G}(n, d)$, and $\mathcal{G}(n/2, n/2, d)$, are over d-regular graphs having n vertices, where we assume for simplicity that n is even. A graph generated according to $\mathcal{G}(n, d)$ is obtained by selecting, uniformly and independently, d perfect matchings between the n vertices. A graph generated according to $\mathcal{G}(n/2, n/2, d)$ is obtained by first randomly partitioning the n vertices into two equal parts, and then selecting, uniformly and independently, d perfect matchings between the two parts. By definition, all graphs in the support of $\mathcal{G}(n/2, n/2, d)$ are bipartite, and we prove that graphs generated according to $\mathcal{G}(n, d)$ are ϵ-far from bipartite with high probability, for $\epsilon \leq 1/16$ and $d \geq 64$

We then show that the following two claims hold when a graph is generated either according to $\mathcal{G}(n,d)$ or according to $\mathcal{G}(n/2,n/2,d)$: (1) Any algorithm that asks $o(n^2/m) = o(n/d)$ queries, will not detect an edge by any vertex-pair query with very high probability. (2) Any algorithm that asks $o(\sqrt{n})$ queries will not receive as an answer to any neighbor query, a vertex it has already observed in a previous query (with very high probability as well). From this we can conclude that any algorithm that asks $o(\min(\sqrt{n}, n^2/m))$ queries cannot distinguish between the two distributions, as desired. In the lower bound proof we show the necessity of both *neigbhor queries* and *vertex-pair queries*. Specifically by using only one type of queries the lower bound increases.

References

1. N. Alon. Testing subgraphs of large graphs. *Random Structures and Algorithms*, 21:359–370, 2002.
2. N. Alon, E. Fischer, M. Krivelevich, and M Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.
3. N. Alon and M. Krivelevich. Testing k-colorability. *SIAM Journal on Discrete Math*, 15(2):211–227, 2002.
4. N. Alon and A. Shapira. Testing subgraph in directed graphs. Submitted, 2002.
5. M. Bender and D. Ron. Testing properties of directed graphs: Acyclicity and connectivity. *Random Structures and Algorithms*, pages 184–205, 2002.
6. A. Bogdanov, Kenji Obata, and L. Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science*, pages 93–102, 2002.
7. A. Bogdanov and L. Trevisan. Lower bounds for testing bipartiteness in dense graphs. Submitted, 2002.
8. O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. In *Proceedings of the Thirty-Seventh Annual Symposium on Foundations of Computer Science*, pages 339–348, 1996.
9. O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.
10. O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002.
11. O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. In *Proceedings of the Forty-Second Annual Symposium on Foundations of Computer Science*, pages 460–469, 2001.
12. T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. 2003.
13. A. Lubotzky, R. Phillips, and P. Sarnak. Explicit expanders and the ramanujan conjectures. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 240 –246, 1986.
14. Gregory A. Margulis. Explicit constructions of expanders. *Problemy Peredachi Informatsii*, 9(4):71–80, 1973. expanders construction.
15. M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, 20(2):165–183, 2002.
16. R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

17. A.C. Yao. Probabilistic computation, towards a unified measure of complexity. In *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.