Tight Bounds for Testing Bipartiteness in General Graphs

Tali Kaufman[*]
Dept. of Computer Science
Tel Aviv University
Tel Aviv, Israel
kaufmant@post.tau.ac.il

Michael Krivelevich[†]
Dept. of Mathematics
Tel Aviv University
Tel Aviv, Israel
krivelev@post.tau.ac.il

Dana Ron[‡]
Dept. of EE – Systems
Tel Aviv University
Tel Aviv, Israel
danar@eng.tau.ac.il

March 1, 2004

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 (and $\tilde{\Theta}(f(n))$ means $\Theta(f(n) \cdot \text{polylog}(f(n)))$). 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.

[*] 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.

[†] Research supported in part by a USA Israel BSF grant and by a grant from the Israel Science Foundation.

[‡] Research supported by the Israel Science Foundation (grant number 32/00-1).

1 Introduction

Property testing algorithms [21, 12] 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 that the algorithm performs.

1.1 Models for Testing Graph Properties

The first model, introduced in [12], 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 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, denoted m, is $\Theta(n^2)$. This model was studied in [12, 3, 2, 1, 4, 16, 8].

The second model, introduced in [13], 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 [14, 13, 7].

In [20] it was suggested to decouple the issues 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 [20] 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. To illustrate this, consider a (sparse) graph having $m = O(n)$ edges, where the maximum degree d of vertices in the graph is $\Omega(n)$. Suppose we want to determine whether the graph is bipartite or ϵ-far from being bipartite for some constant ϵ. If we worked in the bounded-degree incidence-lists model then we could trivially accept all graphs, since for $d = \Omega(n)$ and constant ϵ, every graph having $O(n) = o(dn)$ edges is ϵ-close to being bipartite. However, this is no longer true when distance is measured with respect to the actual number of edges m.

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 [20] 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. Thus this notion of distance adapts itself to the density of the graph, and we shall use it in our work.

The focus in [20] 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 [12] where it was shown that bipartiteness can be tested in the adjacency-matrix model 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 [8]. Thus the complexity of this problem in the adjacency-matrix model is independent of the number of vertices n and is polynomial in $1/\epsilon$. It is interesting to note that testing bipartiteness is considered implicitly already in [9]. The result in [9] can be used to obtain a testing algorithm in the adjacency-matrix model whose query complexity does not depend on the size of the graph, but whose dependence on ϵ is a tower of height polynomial in $1/\epsilon$.

The complexity of testing bipartiteness is significantly different when considering the bounded-degree incidence-lists model. In [14] a lower bound of $\Omega(\sqrt{n})$ was established in this model, for constant ϵ and constant d (where d is the degree bound). An almost matching upper bound of $\tilde{O}(\sqrt{n} \cdot \mathrm{poly}(1/\epsilon))$ was shown in [13]. 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. As we discuss briefly in Subsection 1.5, 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 are $O(\min(\sqrt{n}, n^2/m) \cdot \mathrm{poly}(\log n/\epsilon))$. 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 cycle[1] 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 that 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 (in terms of the dependence on n) 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.

Notes:

1. Observe that even if the graph is sparse then we obtain a new result that does not follow from [13]. Namely, we have an algorithm with complexity $\tilde{O}(\sqrt{n} \cdot \mathrm{poly}(1/\epsilon))$ for sparse graphs with varying degrees.

2. We note that the algorithm does not actually require to be given the number of edges, m, in the graph, but can instead compute an estimate of this number. Such an estimate can be obtained without increasing the query complexity of the algorithm [11, 15], and we discuss this issue shortly in Section 4.

3. We assume that $m = \Omega(n)$. Since our distance measure is with respect to the number of edges in the graph, without such an assumption it would be impossible to distinguish between the following two (families of) graphs: a graph that consists of many isolated edges and a single very small subgraph that is far from bipartite (e.g., a clique), a graph that consists of many isolated vertices and a very small bipartite subgraph. We discuss this issue shortly in Section 4 as well.

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 [13]. Indeed, as long as the degree d of the graph is at most $\sqrt{n}$, we execute the algorithm described in [13]. The place where we depart from [13] 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 testing bipartiteness of almost-regular graphs. Namely, we show how, for every given

[1] We use the term "odd cycle" as a shorthand for "odd-length cycle".

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 being 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 using $\tilde{O}(\min(\sqrt{n}, n^2/m))$ queries. If m is sufficiently large (once again, if $m \geq n^{1.5}$), this can be performed without increasing the complexity of our algorithm 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 simple probabilistic construction we suggest), 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 being 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.

Our lower bound proof implies the necessity of both neighbor queries and vertex-pair queries in obtaining an upper bound whose dependence on n and m is $\tilde{O}(\min(\sqrt{n}, n^2/m))$. Specifically, if only neighbor queries are allowed then our analysis implies a lower bound of $\Omega(\sqrt{n})$ for every m, which is higher than $\tilde{O}(n^2/m))$ when $m = \omega(n^{1.5})$. If only vertex-pair queries are allowed then our analysis implies a lower bound of $\Omega(n^2/m)$, which is above the upper bound of $\tilde{O}(\sqrt{n})$ when $m = o(n^{1.5})$.

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 $\text{poly}(k/\epsilon)$ queries [12, 3, 10], while testing sparse graphs requires $\Omega(n)$ queries [7]. 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 [12], testing acyclicity of directed graphs [6], and testing that a graph does not contain a certain subgraph [1, 4].

1.6 Organization of the Paper

In Section 2 we give some basic definitions and notation. In Sections 3 and 4 we describe and analyze our testing algorithms. In Section 5 we present our lower bound.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with n vertices labeled $1, \cdots, n$, 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 are labeled from 1 to $\deg(v)$. We make no assumption on the corresponding order of the neighbors of a vertex. 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 $d_{\text{avg}} = d_{\text{avg}}(G)$ denote the average degree in the graph (that is, $d_{\text{avg}}(G) = 2m(G)/n$).

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 obtains 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 having 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[2], 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.

[2] As noted in the introduction, we can remove this assumption and have the algorithm compute an estimate of the number of edges.

- 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.

Note that degree queries can be easily implemented using neighbor queries with cost $O(\log d_{\max}) = O(\log n)$.

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 being 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 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 being 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 are $O(\min(\sqrt{n}, n/d_{\max}) \cdot \text{poly}(\log n/\epsilon))$.

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, whose pseudo-code appears in Figure 1, builds on the testing algorithm for bipartiteness described in [13]. The query complexity of that algorithm is $O(\sqrt{n} \cdot \text{poly}(\log n/\epsilon))$ and it works with respect to $m_{\max} = dn$ as well. In fact, as long as $d \leq \sqrt{n}$ our algorithm is the same as the algorithm in [13].

In particular, as in [13], our algorithm selects $\Theta(1/\epsilon)$ *starting vertices* and from each it performs several random walks (using neighbor queries), each walk of length $\text{poly}(\log n/\epsilon)$. The exact form of these random walks is described momentarily. If $d \leq \sqrt{n}$ then the number of these random walks from each starting vertex s is $O(\sqrt{n} \cdot \text{poly}(\log n/\epsilon))$, and the algorithm simply checks whether an odd cycle was detected in the course of these random walks. Specifically, the algorithm checks whether there exists some vertex v that is reached at the end of two different walks from s, where one walk corresponds to a path in the graph with even length, and one walk corresponds to a path with odd length. The existence of such a vertex v implies an odd cycle that contains s and v, and the algorithm rejects in such a case.

If $d > \sqrt{n}$ then there are two important modifications as compared to the case $d \leq \sqrt{n}$ (which, as noted above, follows [13]). These modifications reduce the number of queries performed as the degree increases.

1. The number of random walks performed from each starting vertex is reduced to $O(\sqrt{n/d} \cdot \mathrm{poly}(\log n/\epsilon))$ (as compared to $O(\sqrt{n} \cdot \mathrm{poly}(\log n/\epsilon))$ walks for the case $d \leq \sqrt{n}$).
2. The algorithm still considers the vertices reached at the end of these walks, but now it performs an additional step. It partitions these vertices into two subsets, denoted A_0 and A_1, according to the parity of the lengths of the paths corresponding to the walks. The algorithm then performs vertex-pair queries on each pair of vertices that belong to the same subset. If any edge (u, v) is detected for $u, v \in A_0$ or $u, v \in A_1$, then the algorithm has evidence to an odd cycle that included s (the starting vertex), u and v, and it rejects. The total number of vertex-pair queries is $O((n/d) \cdot \mathrm{poly}(\log n/\epsilon))$.

On a very intuitive level, if a graph is far from being bipartite then many edges (and vertices) belong to many odd cycles. The difference between the two cases described above is that if $d \leq \sqrt{n}$ then the algorithm tries to reach the same vertex twice, once via an even-length path and once via an odd-length path. To this end it performs about $\sqrt{n}$ random walks (so as to "hit" the same vertex twice). In the case $d > \sqrt{n}$, the algorithm performs much fewer walks and so we cannot expect to reach the same vertex twice. However, since the edge-density is higher, we do expect to have an edge in the subgraph induced by the ending points of the walk. In particular, as our analysis shows, we expect to see such an edge between vertices that are reached via paths whose lengths have the same parity.

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.

To every walk (or, more generally, to 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.

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 being 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 cycle of length* $\mathrm{poly}(\log n/\epsilon)$. *The query complexity and running time of the algorithm are* $O\left(\min(\sqrt{n}, n/d_{\max}) \cdot \mathrm{poly}(\log n/\epsilon)\right)$.

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

As a direct corollary of Theorem 1 (using $m(G) = (n d_{\mathrm{avg}}(G))/2$) we get:

```
Test-Bipartite-Reg($n, d_{\max}, \epsilon$)
  • Repeat $T = \Theta(\frac{1}{\epsilon})$ times:
      1. Uniformly select a vertex $s$ 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 $K \overset{\text{def}}{=} \Theta\left(\frac{\sqrt{n}\cdot\log^{1/2}(n/\epsilon)}{\epsilon^3}\right)$ and $L \overset{\text{def}}{=} \Theta\left(\frac{\log^3(n/\epsilon)}{\epsilon^5}\right)$. Otherwise
     ($d > \sqrt{n}$), let $K \overset{\text{def}}{=} \Theta\left(\frac{\sqrt{n/d}\cdot\log^{1/2}(n/\epsilon)}{\epsilon^8}\right)$, and $L \overset{\text{def}}{=} \Theta\left(\frac{\log^6(n/\epsilon)}{\epsilon^8}\right)$.
  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 at the ends of the walks performed in the
     previous step 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.
```

Figure 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 cycles in the graph G.

Corollary 2 *For a given graph G, let $\gamma(G) \overset{\text{def}}{=} d_{\max}(G)/d_{\text{avg}}(G)$. Then Test-Bipartite-Reg($n, d_{\max}(G), \epsilon/(2\gamma(G))$) accepts every graph that is bipartite, and rejects with probability at least 2/3 every graph that is ϵ-far from being bipartite (with respect to $m(G)$).*

The corollary below will become useful in the next section.

Corollary 3 *If G is ϵ-far from being 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 $\frac{2}{3}$.*

The completeness part of Theorem 1 (i.e., showing that the algorithm accepts bipartite graphs) is straightforward. We focus on proving the soundness of the algorithm (i.e., that graphs that are ϵ-far from being bipartite are rejected with probability $\frac{2}{3}$). What we eventually show (in Subsection 3.6) is the contrapositive statement. Namely, that if the test accepts G with probability greater than $\frac{1}{3}$ then there exists an ϵ-good partition of G.

Our analysis follows the analysis presented in [13] quite closely. In particular, whenever possible we refer the reader to proofs given in [13]. Here we present what is necessary to establish the correctness of our algorithm and in particular those proofs in which we diverge from [13]. Since the algorithm for the case $d_{\max} \leq \sqrt{n}$ is fully analyzed in [13], from this point on we assume $d_{\max} > \sqrt{n}$ and analyze the algorithm for this case.

3.1 Gaining Intuition: The Rapidly–Mixing Case

To gain some intuition, consider first the following "ideal" case: From each starting vertex s in G, and for every $v \in V$, the probability that a random walk of length $L = \text{poly}((\log n)/\epsilon)$ ends at v is at least $\frac{1}{2n}$ and at most $\frac{2}{n}$ – i.e., approximately the probability assigned by the stationary distribution. (Note that this ideal case occurs when G is an expander). Let us fix a particular starting vertex s. For each vertex v, let p_v^0 be the probability that a random walk (of length L) starting from s, ends at v and corresponds to an even-length path. Define p_v^1 analogously for odd paths. Then, by our assumption on G, for every v, $p_v^0 + p_v^1 \geq \frac{1}{2n}$. We consider two cases regarding the sum $\sigma(G) \stackrel{\rm def}{=} \sum_{\substack{v,u \in V \\ (v,u)\in E}} (p_v^0 p_u^0 + p_v^1 p_u^1)$.

In case $\sigma(G)$ is (relatively) "small", we show that there exists a partition (V_0, V_1) of V that is ϵ-good, and so G is ϵ-close to being bipartite. Otherwise (i.e., when the sum is not "small"), we show that the rejection probability is bounded away from zero. This implies that in case G is accepted with probability at least $\frac{1}{3}$ then G is ϵ-close to being bipartite.

Consider first the case in which $\sigma(G) < c \cdot \frac{\epsilon d}{n}$ for some suitable constant $c < 1$. Let the partition (V_0, V_1) be defined as follows: $V_0 = \{v : p_v^0 \geq p_v^1\}$ and $V_1 = \{v : p_v^1 > p_v^0\}$. Consider a particular vertex $v \in V_0$. By definition of V_0 and our rapid-mixing assumption, $p_v^0 \geq \frac{1}{4n}$.

$$\begin{aligned}
\sigma(G) &= \sum_{\substack{v,u\in V \\ (v,u)\in E}} (p_v^0 p_u^0 + p_v^1 p_u^1) \\
&\geq \sum_{\substack{v,u\in V_0 \\ (v,u)\in E}} p_v^0 p_u^0 \quad + \quad \sum_{\substack{v,u\in V_1 \\ (v,u)\in E}} p_v^1 p_u^1 \\
&\geq \sum_{\substack{v,u\in V_0 \\ (v,u)\in E}} \frac{1}{16n^2} + \sum_{\substack{v,u\in V_1 \\ (v,u)\in E}} \frac{1}{16n^2} \\
&\geq \frac{1}{16n^2} \cdot \text{(The number of violating edges w.r.t. } (V_0, V_1)) \ . \qquad (1)
\end{aligned}$$

Thus, if there are more than ϵdn violating edges with respect to (V_0, V_1), then $\sigma(G) > \frac{1}{16} \cdot \frac{\epsilon d}{n}$ which contradicts our case hypothesis concerning $\sigma(G)$ assuming $c \leq 1/16$.

We now turn to the second case, $\sigma(G) \geq c \cdot \frac{\epsilon d}{n}$. For every fixed pair $i, j \in \{1, \cdots, K\}$, (recall that $K = \Theta(\sqrt{n/d} \cdot \text{poly}(\log n/\epsilon))$ is the number of walks taken from s), consider the 0/1 random variable $\eta_{i,j}$ that is 1 if and only if both the i-th and the j-th walks have path length with the same parity, and if the end-points of the paths are vertices u, v such that $(u, v) \in E$. Then for every pair i, j,

$$\text{Exp}[\eta_{i,j}] \ = \sum_{u,v\in V,(u,v)\in E} (p_v^0 p_u^0 + p_v^1 p_u^1) \ = \ \sigma(G) \ . \qquad (2)$$

Since there are $K^2 = \Theta(n/d \cdot \text{poly}(\log n/\epsilon))$ such pairs i, j, the expected value of the sum over all $\eta_{i,j}$'s is greater than some constant $c' > c$. These random variables are not pairwise independent, nonetheless we can obtain a constant bound on the probability that the sum is 0 using Chebyshev's inequality (cf., [5, Sec. 4.3]).

Unfortunately, we may not assume in general that for every (or even some) starting vertex, all (or even almost all) vertices are reached with probability $\Theta(1/n)$. However, roughly speaking, we are able to show that every graph can be partitioned into parts such that within each part we can perform an analysis that builds on the ideas presented above. Furthermore, the different parts are separated by small cuts so that if each part is close to being bipartite, then so is the whole graph. An important component in the analysis is the definition of the Markov Chain $\mathrm{M}_{\ell_1}^{\ell_2}(H)$, and we turn to this definition in the next subsection.

3.2 The Markov Chain $\mathrm{M}_{\ell_1}^{\ell_2}(H)$

Let H be an induced subgraph of G. For any given pair of lengths, ℓ_1 and ℓ_2, we define a Markov Chain $\mathrm{M}_{\ell_1}^{\ell_2}(H)$. Roughly speaking, $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ captures random walks of length at most $\ell_1 \cdot \ell_2$ in G that do not exit H for (sub)walks of length ℓ_2 or more. The states of the chain consist of the vertices of H and some additional auxiliary states. For vertices that do not have neighbors outside of H, the transition probabilities in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ are exactly as in walks on G. However, for vertices v that have neighbors outside of H there are two modifications: (1) For each vertex u, the transition probability from v to u, denoted $q_{v,u}$, is the probability of a walk (in G) starting from v and ending at u after less than ℓ_2 steps (without passing through any other vertex in H). Thus, walks of length less than ℓ_2 out of H (and in particular the walk $v - u$ in case $(v, u) \in E$), are contracted into single transitions. Note that for every u and v in H we have $q_{u,v} = q_{v,u}$. (2) There is an auxiliary path of length ℓ_1 emitting from v. The transition probability from v to the first auxiliary vertex on the path equals the probability that a walk starting from v exits H and does not return in less than ℓ_2 steps. From the last vertex on the auxiliary path there are transitions to vertices in H with the corresponding conditional probabilities of reaching them after such a walk.
A more formal definition of $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ appears in the appendix, together with an illustration (see Figure 8). The following definition and lemma will be instrumental in our analysis.

Definition 3.1 *We say that a vertex s is* useful *with respect to $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ if the probability that a walk in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ starting from s enters an auxiliary path after at most ℓ_1 steps, is at most $\frac{2\ell_1}{\ell_2} \cdot \frac{n}{|H|}$.*

Lemma 1 *Let H be a subgraph of G, and let ℓ_1 and ℓ_2 be integers. Then at least half of the vertices s in H are useful with respect to $\mathrm{M}_{\ell_1}^{\ell_2}(H)$.*

The proof of the lemma appears in [13].

3.3 Useful Vertices and Small Cuts

The following lemma can be viewed as presenting a “contrapositive statement” of the work of Mihail [19]. While Mihail showed that high expansion leads to fast convergence of random walks to the stationary distribution, the lemma below shows that too slow of a convergence implies small cuts that have certain additional properties. In particular, the vertices on one side of the cut can be reached with roughly the same, relatively high probability from some vertex s (where s need not necessarily be on the same side of the cut). In the special case where $H = G$ and G is rapidly mixing, the set S will be all of V, but in the general case it will be a subset of those vertices that are reached from s with probability that is not much smaller than that assigned by the stationary distribution (of $\mathrm{M}_{\ell_1}^{\ell_2}(H)$).

For states x and y in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ and an integer t, let $q_{x,y}(t)$ denote the probability that a random walk in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ that starts at x, ends at y after t steps.

Lemma 2 *Let H be a subgraph of G with at least $\frac{\epsilon}{4}n$ vertices, and let $\ell_1 = \Theta\left(\left(\frac{\log(n/\epsilon)}{\epsilon}\right)^3\right)$, $\ell_2 = \Theta\left(\frac{\ell_1}{\epsilon^2}\right)$, and $F = O\left(\frac{1}{\epsilon}\right)$. Then for every vertex s that is* useful *with respect to $\mathrm{M}_{\ell_1}^{\ell_2}(H)$, there exists a subset of vertices S in H, an integer t, $\ell_1/2 \le t \le \ell_1$, and a value $\beta = \Omega\left(\frac{\epsilon^2}{\log(n/\epsilon)}\right)$, such that:*

1. *The number of edges between S and the rest of H is at most $\frac{\epsilon}{2} \cdot d \cdot |S|$.*

2. *For every $v \in S$,*

$$\sqrt{\frac{1}{|S|} \cdot \frac{\beta}{|H|}} \leq q_{s,v}(t) \leq F \cdot \sqrt{\frac{1}{|S|} \cdot \frac{\beta}{|H|}} \; .$$

The proof of the lemma appears in [13].

3.4 Sufficient Conditions for Good Partitions

In the next lemma we give sufficient conditions under which subsets of vertices can be partitioned without having many violating edges. For each $b \in \{0,1\}$ let $q^b_{s,v}(t)$ denote the probability in $\mathrm{M}^{\ell_2}_{\ell_1}(H)$ of a walk of length t starting from s, ending at v, and corresponding to a path whose length has parity b. What the lemma essentially requires is that for some fixed vertex s and subset of vertices S in H, there is a lower bound on the probability that each vertex in S is reached from s (in t steps), and there aren't too many vertices v in the subset such that both $q^0_{s,v}(t)$ and $q^1_{s,v}(t)$ are large (with respect to this lower bound).

Lemma 3 *Let H be a subgraph of G, s a vertex in H, S a subset of vertices in H and ℓ_1 and ℓ_2 integers. Assume that for some $\alpha > 0$, $t < \ell_1$, the following holds in $\mathrm{M}^{\ell_2}_{\ell_1}(H)$:*

1. *For every $v \in S$, $q_{s,v}(t) \geq \alpha$;*

2. *$\sum_{v,u \in S,(v,u)\in E} (q^0_{s,v}(t) q^0_{s,u}(t) + q^1_{s,v}(t) q^1_{s,u}(t)) < \frac{\epsilon}{c} \cdot d \cdot |S| \cdot \alpha^2$ for some constant c.*

Let (S_0, S_1) be a partition of S, where $S_0 = \{v : q^0_{s,v}(t) \geq q^1_{s,v}(t)\}$, and $S_1 = \{v : q^1_{s,v}(t) > q^0_{s,v}(t)\}$. Then the number of violating edges in G with respect to (S_0, S_1) is at most $\frac{\epsilon}{c} \cdot d \cdot |S|$.

Proof: Consider a vertex v and let $v \in S_b$, for $b \in \{0,1\}$. By definition of the partition (S_0, S_1), $q^b_{s,v}(t) \geq \frac{1}{2} q_{s,v}(t) \geq \frac{\alpha}{2}$.

Assume, contrary to what is claimed in the lemma, that the number of violating edges with respect to (S_0, S_1) is more than $\frac{\epsilon}{c} \cdot d \cdot |S|$. Then

$$\begin{aligned}
&\sum_{v,u \in S,(v,u)\in E} (q^0_{s,v}(t) q^0_{s,u}(t) + q^1_{s,v}(t) q^1_{s,u}(t)) \\
&\qquad \geq \sum_{v,u\in S,(v,u)\in E,u,v\in S_0} (q^0_{s,v}(t) q^0_{s,u}(t)) + \sum_{v,u\in S,(v,u)\in E,u,v\in S_1} (q^1_{s,v}(t) q^1_{s,u}(t)) && (3) \\
&\qquad \geq \sum_{v,u\in S,(v,u)\in E,u,v\in S_0} \frac{\alpha^2}{4} + \sum_{v,u\in S,(v,u)\in E,u,v\in S_1} \frac{\alpha^2}{4} && (4) \\
&\qquad \geq \frac{\alpha^2}{4} \cdot \frac{\epsilon}{c} \cdot d \cdot |S| \; . && (5)
\end{aligned}$$

But this contradicts the second hypothesis of the lemma. ■

3.5 Sufficient Conditions for Detecting Odd Cycles

In the next lemma we describe sufficient conditions for "detecting" odd cycles when performing walks in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ starting from some vertex s. What the lemma essentially requires is that there exists a subset S of vertices such that there are both lower and upper bounds on the probability that each vertex in S is reached from s (in $t < \ell_1$ steps), and there are many vertices v in S such that both $q^0_{s,v}(t)$ and $q^1_{s,v}(t)$ are large (with respect to the lower bound). As stated later in Corollary 4, these conditions are sufficient for detecting odd cycles when performing random walks in G of length $\ell_1 \cdot \ell_2$.

Lemma 4 *Let H be a subgraph of G, s a vertex in H, S a subset of vertices in H and ℓ_1 and ℓ_2 integers. Assume that for some $\alpha > 0$ and $F = \Theta(\frac{1}{\epsilon})$ and $t < \ell_1$, the following holds in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$:*

1. *For every $v \in S$, $\alpha \leq q_{s,v}(t) \leq F \cdot \alpha$;*
2. *$\sum_{v,u\in S,(v,u)\in E}(q^0_{s,v}(t)q^0_{s,u}(t) + q^1_{s,v}(t)q^1_{s,u}(t)) \geq \frac{\epsilon}{c}\cdot d \cdot |S| \cdot \alpha^2$ for some constant c.*

Suppose we perform $O\left(\frac{F^5}{\epsilon\cdot\alpha\sqrt{d}\sqrt{|S|}}\right)$ random walks of length t starting from s in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$. Let A_0 (A_1) be the set of vertices that appear at the end of the walks whose corresponding paths have even (odd) length, and let G_0 (G_1) be the subgraph induced by A_0 (A_1). Then with probability at least 0.99 *(taken over the random walks), either G_0 contains an edge or G_1 contains an edge (i.e., the algorithm detects an odd cycle).*

We note that when we apply Lemma 4, we set $\alpha = \mathrm{poly}(\epsilon/(\log n))/\sqrt{|S|\cdot|H|}$, and $F = O(1/\epsilon)$, so that the number of random walks that should be performed is $O(\sqrt{n/d}\cdot\mathrm{poly}((\log n)/\epsilon))$.

Proof: Let $\gamma \stackrel{\rm def}{=} \sum_{v,u\in S,(v,u)\in E}(q^0_{s,v}(t)q^0_{s,u}(t) + q^1_{s,v}(t)q^1_{s,u}(t))$ so that by the second hypothesis of the lemma $\gamma \geq \frac{\epsilon}{c}\cdot d\cdot|S|\cdot\alpha^2$. Consider $m = O\left(\frac{F^5}{\epsilon\cdot\alpha\cdot\sqrt{d}\sqrt{|S|}}\right)$ random walks of length t starting from s. For $1 \leq i \neq j \leq m$, let $\eta_{i,j}$ be a 0/1 random variable that is 1 if and only if both the i-th and the j-th walk have path length with the same parity, and if the end-points of the paths are the vertices $u, v \in S$ such that $(u,v) \in E$.

Thus, we would like to bound the probability that $\sum_{i<j}\eta_{i,j} = 0$. The difficulty is that the $\eta_{i,j}$'s are not pairwise independent. Yet, since the sum of the covariances of the dependent $\eta_{i,j}$'s is quite small, Chebyshev's Inequality is still very useful (cf., [5, Sec. 4.3]). Details follow. For every $i \neq j$,

$$\mathrm{Exp}[\eta_{i,j}] = \sum_{v,u\in S,(v,u)\in E}(q^0_{s,v}(t)q^0_{s,u}(t) + q^1_{s,v}(t)q^1_{s,u}(t)) = \gamma .$$

By Chebyshev's inequality,

$$\Pr\left[\sum_{i<j}\eta_{i,j} = 0\right] \leq \frac{\mathrm{Var}\left[\sum_{i<j}\eta_{i,j}\right]}{\left(\mathrm{Exp}\left[\sum_{i<j}\eta_{i,j}\right]\right)^2} < \frac{\mathrm{Var}\left[\sum_{i<j}\eta_{i,j}\right]}{\left(\binom{m}{2}\cdot\gamma\right)^2} . \tag{6}$$

We now bound $\mathrm{Var}[\sum_{i<j}\eta_{i,j}]$. Since the $\eta_{i,j}$'s are not pairwise independent, some care is needed: Let $\bar{\eta}_{i,j} \stackrel{\rm def}{=} \eta_{i,j} - \mathrm{Exp}[\eta_{i,j}]$.

$$\mathrm{Var}\left[\sum_{i<j}\eta_{i,j}\right] = \mathrm{Exp}\left[\left(\sum_{i<j}\bar{\eta}_{i,j}\right)^2\right]$$

$$\begin{aligned}
&= \sum_{i<j}\sum_{k<\ell}\mathrm{Exp}\left[\bar{\eta}_{i,j}\cdot\bar{\eta}_{k,\ell}\right] \\
&= \sum_{i<j}\mathrm{Exp}\left[\bar{\eta}^2_{i,j}\right] + 4\sum_{i<j<k}\mathrm{Exp}\left[\bar{\eta}_{i,j}\cdot\bar{\eta}_{j,k}\right] + 0 \\
&= \binom{m}{2}\cdot\mathrm{Exp}[\bar{\eta}^2_{1,2}] + 4\cdot\binom{m}{3}\cdot\mathrm{Exp}\left[\bar{\eta}_{1,2}\cdot\bar{\eta}_{2,3}\right]\ . \qquad (7)
\end{aligned}$$

The factor of 4 in the third equality is the number of possibilities that among the four elements i, j, k, ℓ (where $i < j$ and $k < \ell$) exactly two are equal (namely: $i = k < j < \ell$; $i < j = k < \ell$; $i < k < j = \ell$; and $k < i = \ell < j$). The 0 term is due to the fact that when i, j, k, l are all distinct,

$$\begin{aligned}
\mathrm{Exp}\left[\bar{\eta}_{i,j}\cdot\bar{\eta}_{k,\ell}\right] &= \mathrm{Exp}\left[\eta_{i,j}\cdot\eta_{k,\ell}\right] - \gamma^2 \\
&= \sum_{i,j,k,\ell\in S,(i,j),(k,\ell)\in E} q^0_{s,i}(t)q^0_{s,j}(t)q^0_{s,k}(t)q^0_{s,\ell}(t) \\
&+ \sum_{i,j,k,\ell\in S,(i,j),(k,\ell)\in E} q^0_{s,i}(t)q^0_{s,j}(t)q^1_{s,k}(t)q^1_{s,\ell}(t) \\
&+ \sum_{i,j,k,\ell\in S,(i,j),(k,\ell)\in E} q^1_{s,i}(t)q^1_{s.j}(t)q^0_{s,k}(t)q^0_{s,\ell}(t) \\
&+ \sum_{i,j,k,\ell\in S,(i,j),(k,\ell)\in E} q^1_{s,i}(t)q^1_{s,j}(t)q^1_{s,k}(t)q^1_{s,\ell}(t) - \gamma^2 \\
&= \left(\sum_{(i,j)\in E(S)} q^0_{s,i}(t)q^0_{s,j}(t) + q^1_{s,i}(t)q^1_{s,j}(t)\right)^2 - \gamma^2 = \gamma^2 - \gamma^2 = 0\ . \qquad (8)
\end{aligned}$$

We next bound each of the two terms in Equation (7).

$$\mathrm{Exp}[\bar{\eta}^2_{1,2}] \ \leq\ \mathrm{Exp}[\eta^2_{1,2}] \ =\ \mathrm{Exp}[\eta_{1,2}] \ =\ \gamma\ . \qquad (9)$$

Let v_i be a random variable that represents the vertex that the i-th walk ends at.

$$\begin{aligned}
\mathrm{Exp}[\bar{\eta}_{1,2}\cdot\bar{\eta}_{2,3}] &\leq \mathrm{Exp}[\eta_{1,2}\cdot\eta_{2,3}] \\
&\leq \sum_{v_1,v_2,v_3\in S,(v_1,v_2),(v_3,v_2)\in E} q^0_{s,v_1}(t)q^0_{s,v_2}(t)q^0_{s,v_3}(t) + q^1_{s,v_1}(t)q^1_{s,v_2}(t)q^1_{s,v_3}(t) \\
&\leq \text{(number of pairs of edges in } S \text{ with a common vertex in S)} \cdot 2(\max_v\{q_{s,v}(t)\})^3 \\
&\leq 2\cdot\min(|S|^2 d, |S|d^2)\cdot F^3\cdot\alpha^3 \qquad (10)
\end{aligned}$$

Since by the lemma's second hypothesis $\gamma \geq \frac{\epsilon}{c}\cdot d\cdot|S|\cdot\alpha^2$, we can replace α in Equation (10) with $\sqrt{\frac{c\cdot\gamma}{\epsilon\cdot d\cdot|S|}}$ and get

$$\mathrm{Exp}[\bar{\eta}_{1,2}\cdot\bar{\eta}_{2,3}] \ \leq\ 2\cdot\min(|S|^2 d, |S|d^2)\cdot F^3\cdot\left(\frac{c\gamma}{\epsilon d|S|}\right)^{\frac{3}{2}} \qquad (11)$$

Combining Equations (6)–(11) we get

$$\Pr\left[\sum_{i<j}\eta_{i,j} = 0\right] \ =\ O\left(\frac{m^2\cdot\gamma + m^3\cdot\min(|S|^2 d, |S|d^2)\cdot F^3\cdot(\frac{\gamma}{\epsilon d|S|})^{\frac{3}{2}}}{m^4\cdot\gamma^2}\right)$$

$$= O\left(\frac{1}{\gamma \cdot m^2} + \frac{F^3 \min(\sqrt{\frac{|S|}{d}}, \sqrt{\frac{d}{|S|}})}{m \cdot \epsilon^{\frac{3}{2}} \cdot \sqrt{\gamma}}\right)$$

$$= O\left(\frac{\epsilon}{F^{10}} + \frac{1}{F^2 \cdot \epsilon}\right) = O(\epsilon) \tag{12}$$

As observed above, by the lemma's hypothesis concerning γ, it holds that $\alpha = O(\sqrt{\gamma/(\epsilon d|S|)})$. Since $m = \Omega\left(\frac{F^5}{\epsilon \cdot \alpha \cdot \sqrt{d}\sqrt{|S|}}\right)$, we have that $m = \Omega\left(F^5\sqrt{\frac{1}{\epsilon \cdot \gamma}}\right)$, and the lemma follows. ■

Based on the construction of $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ we can map walks of length $\ell_1 \cdot \ell_2$ in G to walks of length ℓ_1 in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$, and obtain as a corollary to Lemma 4:

Corollary 4 *Let H be a subgraph of G and let S, s, ℓ_1, ℓ_2, t, α and F be as in Lemma 4. Suppose we perform $\Theta\left(\frac{F^5}{\epsilon \cdot \alpha \cdot \sqrt{d}\sqrt{|S|}}\right)$ random walks of length $\ell_1 \cdot \ell_2$ starting from s in G. Let A_0 (A_1) be the set of vertices that appear at the end of the walks whose corresponding paths have even (odd) length, and let G_0 (G_1) be the subgraph induced by A_0 (A_1). Then with probability at least* 0.99, *either G_0 contains an edge or G_1 contains an edge (i.e., the algorithm detects an odd cycle).*

The proof of the corollary is similar to that of an analogous corollary that appears in [13].

3.6 Proof of Theorem 1

Recall that we need to show that if the test accepts G with probability greater than $\frac{1}{3}$ then G is ϵ-close to being bipartite.

We say that a vertex s in G is *good* (for defining a partition) if the following holds. Suppose we take K random walks of length L in G starting from s. Then the probability that we reach two vertices u and v such that $(u, v) \in E$ and both u and v appear at the ends of walks whose corresponding paths have lengths with the same parity, is at most 0.1. If a vertex is not good then it is *bad*. Here K and L are are set in the algorithm.

Since the test rejects G with probability less than $\frac{2}{3}$, and $T = \Theta(1/\epsilon)$, we know that, for an appropriate constant in the $\Theta(\cdot)$ notation above, the fraction of bad vertices in G is at most $\frac{\epsilon}{16}$. We now show that in such a case we can find a partition of the graph vertices that has at most ϵdn violating edges. We shall do so in steps, where in each step we partition a new set of vertices, denoted S, until we are left with at most $\frac{\epsilon}{4}n$ vertices. For each partitioned set S we show that: (1) there are few (at most $\frac{\epsilon}{4}d|S|$) violating edges with respect to the partition of S; and (2) there are few (at most $\frac{\epsilon}{2}d|S|$) edges between S and the yet "unpartitioned" vertices R so that no matter how the vertices in R are partitioned, the number of violating edges between S and R is small.

At each step, let D be the set of vertices we have already partitioned, and let H be the subgraph induced by $V \setminus D$. Initially, $D = \emptyset$, and $H = G$. Let ℓ_1 and ℓ_2 be as required by Lemma 2, and let the length L of the walks we perform on G be $\ell_1 \cdot \ell_2$. Since $\ell_1 = O\left(\left(\frac{\log(n/\epsilon)}{\epsilon}\right)^3\right)$, and $\ell_2 = O\left(\frac{\ell_1}{\epsilon^2}\right)$, we get that $L = O\left(\frac{\log^6(n/\epsilon)}{\epsilon^8}\right)$. Let $\mathrm{M} \stackrel{\mathrm{def}}{=} \mathrm{M}_{\ell_1}^{\ell_2}(H)$. While $|H| \geq \frac{\epsilon}{4}n$ we do the following. We select any vertex s in H that is both *good* and *useful* with respect to M (see Definition 3.1). By Lemma 1, at least half of the vertices in H are *useful*. Since $|H| \geq \frac{\epsilon}{4}n$ and the total number of *bad* vertices is $\frac{\epsilon}{16}n < \frac{\epsilon}{8}n$, there exist at least $\frac{\epsilon}{16}n$ vertices which are *good* and *useful*.

We next apply Lemma 2 to determine a set S, and an integer t, $\ell_1/2 \le t \le \ell_1$, with the properties stated in the lemma. In particular, the number of edges between S and the rest of H is at most $\frac{\epsilon}{2}d|S|$, and for every $v \in S$, $\sqrt{\frac{\beta}{|S|\cdot|H|}} \le q_{s,v}(t) \le F \cdot \sqrt{\frac{\beta}{|S|\cdot|H|}}$, where $F = O\left(\frac{1}{\epsilon}\right)$, and $\beta = \Omega\left(\frac{\epsilon^2}{\log(n/\epsilon)}\right)$. We claim that it must be the case that $\sum_{v,u\in V,(v,u)\in E}(p_v^0(t)p_u^0(t) + p_v^1(t)p_u^1(t)) \le \frac{\epsilon\cdot\beta\cdot d}{|H|}$. This claim, (which we establish momentarily) implies that we can apply Lemma 3 (with $\alpha = \sqrt{\frac{\beta}{|S|\cdot|H|}}$) to show that S can be partitioned so that there are at most $\frac{\epsilon}{4}d|S|$ violating edges with respect to this partition. The claim holds since otherwise we could apply Corollary 4 and reach a contradiction. Specifically, by letting the number of walks performed from each starting vertex be

$$O\left(\frac{F^5}{\epsilon\cdot\alpha\cdot\sqrt{d}\cdot\sqrt{|S|}}\right) = O\left(\frac{\sqrt{|H|}}{\epsilon^6\cdot\sqrt{d}\cdot\sqrt{\beta}}\right) = O\left(\frac{\log^{1/2}(n/\epsilon)\cdot\sqrt{n/d}}{\epsilon^8}\right) = K$$

(where F, α and β are as set above), we would obtain a contradiction to our assumption that s is *good*.

Thus, as long as $|H| \ge \frac{\epsilon}{4}n$, each set S contributed at most $\frac{\epsilon}{4}\cdot|S|\cdot d + \frac{\epsilon}{2}\cdot|S|\cdot d$ violating edges to the partition. Since these sets are disjoint, all these violating edges sum up to $\frac{3\epsilon}{4}\cdot d\cdot n$. The final H contributes at most $\frac{\epsilon}{4}\cdot n\cdot d$, and so G is ϵ-close to being bipartite.

Verifying that indeed $T = O(1/\epsilon)$, $K = \Theta(\sqrt{n/d}\cdot\text{poly}(\log n/\epsilon))$, and $L = \text{poly}((\log n)/\epsilon))$, and that the algorithm can be implemented using $O(K\cdot L + K^2) = O(n/d\cdot\text{poly}(\log n/\epsilon))$ queries, the theorem follows. (Recall that if $d < \sqrt{n}$ then we obtain the bound of $O(\sqrt{n}\cdot\text{poly}(\log n/\epsilon))$.

4 The Algorithm for the General Case

In this section we build on the testing algorithm presented in the previous section and describe a one-sided error testing algorithm for bipartiteness 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 [12, 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 being bipartite with respect to $m(G)$, then G' is far from being 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 being 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.

While we do not know how to efficiently select a vertex in G' uniformly, we describe a different selection procedure that suffices for our purposes. Specifically, the selection procedure is such that for all but a small fraction of the n' vertices in G', the probability of selecting a vertex v is $\Omega(1/n')$. With slight abuse of terminology we shall refer to this procedure as Sample-Vertices-Almost-Uniformly-in-G'.[3] By Corollary 3, this suffices for our purposes.

Multiple Edges and the Relation Between m and n. The analysis of the algorithm Test-Bipartite-Reg did not require any assumptions on the actual number of edges m in the graph, and it did not preclude the existence of multiple edges. Here we consider graphs that do not contain any multiple edges and we assume that the number of edges m in G is $\Omega(n)$. To justify the assumption on the number of edges, consider a graph that consists of a clique over $k = o(\sqrt{n})$ vertices, where all remaining vertices are isolated. This graph has $m = \Theta(k^2)$ edges, and is clearly far from being bipartite. However, in order to distinguish it from a graph that consists of a complete bipartite graph over $2k$ vertices (where all remaining vertices are isolated and is clearly bipartite), we need $\Omega(n/k) = \omega(\sqrt{N})$ queries. (Taking this to an extreme, if $k = \Theta(1)$ then we will need $\Omega(n)$ queries.) We note that we could replace this assumption by introducing to the complexity of the algorithm a dependence on n/m. This would however make the analysis more cumbersome, without much benefit.

Another alternative assumption would be that the algorithm has the ability to "ignore" isolated vertices (that is, vertices that have no incident edges and are hence immaterial to the question of bipartiteness), and sample uniformly from the *non-isolated* vertices. This would effectively imply that the algorithm is executed on a subgraph induced by the $n' \leq n$ non-isolated vertices, where within this subgraph, the number of edges $m' = m$, is at least $n'/2$.

For simplicity we assume from this point on that $m \geq n$.

We also note that we can actually deal with the case where there are multiple edges, but they do not constitute more than a constant fraction of the total number of edges.[4] However, in order to deal with this case efficiently, we need to assume that there is a concise way to represent the sets of labels of multiple edges that are incident to each vertex. (In particular this holds if the labels of multiple edges incident to each vertex are consecutive). For simplicity we assume there are no multiple edges.

The main theorem of this subsection follows.

Theorem 5 *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. $n \leq n' \leq 4n$, $m \leq m' \leq 8m$, *and* $d_{\max}(G') \leq 2d_{\text{avg}}(G)$.

[3] The reason we say that we abuse terminology is that the distribution on vertices in G' induced by this procedure may be very far from uniform according to any standard distance measure (e.g. statistical difference). However, it approximates the uniform distribution in the sense of assigning relatively large weight to every sufficiently large subset.

[4] If the number of multiple edges is more than a constant fraction then it is possible to obtain a lower bound on the number of queries that depends on the ratio between the number of multiple edges and the total number of edges. Specifically, consider a graph that contains a small clique with many multiple edges, which is far from bipartite, but cannot be distinguished from a bipartite graph that contains a small complete bipartite graph with many multiple edges.

2. *If G is bipartite then G' is bipartite, and if G is ϵ-far from being bipartite with respect to m, then G' is ϵ'-far being 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[Odd-Cycle'(s)=found] $\geq$ Pr[Odd-Cycle(s)=found], *where if* Odd-Cycle'(s) *returns* found, *then we can obtain an odd cycle of length* 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. One difficulty that arises is that when the algorithm asks a neighbor query of the form: "Who is the i-th neighbor of v"?, it gets a vertex name u as an answer. However, the algorithm lacks the information that v is (say) the j-th neighbor of u. This lack of knowledge makes the emulation of random walks and Odd-Cycle(s) in G' more complicated. Due to that, Item 3 of Theorem 5 is somewhat more involved.
As a corollary to Theorem 5 and Corollary 3 we obtain:

Corollary 6 *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 being 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 cycle of length* 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 $\epsilon' = \epsilon/144$.
    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.
```

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

Note that d_{avg}, the average degree of the graph, is given as a parameter to the algorithm. Since the algorithm does not actually need the exact value of d_{avg} it can instead estimate it. Specifically, Feige [11] shows how to obtain an estimate that is within a factor of roughly 2 of the true value by performing at most $O(\sqrt{n/d_0})$ degree queries where d_0 is an apriori lower bound on d_{avg}.

Inspired by our procedure Sample-Edges-Almost-Uniformly-in-G', Goldreich and Ron [15] suggest an alternative procedure that improves on the quality of the estimate. More importantly in our context, they observe that both in the case of their procedure and in the case of Feige's procedure, it is possible to eliminate the need of the lower bound d_0. That is, given's Feige's algorithm, it is possible to obtain a constant factor estimate by performing $O(\sqrt{n/d_{avg}}) \leq O(\min(\sqrt{n}, n^2/m))$ queries (but without any knowledge about d_{avg}).

We now turn to proving Theorem 5. We actually provide two proofs: one is based on a deterministic construction of G' and one on a probabilistic construction. We start by presenting the deterministic construction.

4.1 Defining G' and Proving the First Two Items in Theorem 5

In all that follows, let $d = d_{\rm avg}(G)$, and let $d' = d_{\max}(G')$. We shall assume that d is a sufficiently large constant. If $d_{\rm avg}(G)$ 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 $\epsilon/(d/d_{\rm avg}(G))$.

The Idea. Recall that part of our goal is to have G' be a "regularized" version of G in the sense that in G' all vertices have degree at most $2d$ (while in G there may be vertices with degree much higher than the average degree d). To this end, every vertex of G with degree higher than d is represented in G' by a subset of vertices. Each such subset is partitioned into two equal parts: an *external* subset (consisting of *external* vertices), and an *internal* subset (consisting of *internal* vertices). The edges between external vertices in G' are determined by the edges of G. Namely, if (u, v) is an edge in G, then in G' there is an edge between one of the vertices in the external subset of u to one of the vertices in the external subset of v. In addition, for every vertex v (with degree greater than d) there is a bipartite subgraph between its internal and external vertices. All vertices in the subgraph have degree d, and the subgraph has good expansion properties.

The role of these subgraphs between external and internal vertices is to ensure that if G is far from being bipartite then so is G'. To gain some intuition observe that for every partition (V_1, V_2) of G, there exists a corresponding partition (V'_1, V'_2) of G' that has the same number of violating edges. Specifically, for every vertex $v \in V_1$ ($v \in V_2$) we put in V'_1 (V'_2) all external vertices that correspond to v, and we put in V'_2 (V'_1) all internal vertices that correspond to v. By this construction, there is a one-to-one mapping between the edges in G that are violating with respect to (V_1, V_2) and the edges in G' that are violating with respect to (V'_1, V'_2) (where all edges in the subgraphs between external and internal vertices are non-violating). Thus, if G is far from being bipartite, so that all partitions (V_1, V_2) of G have many violating edges, then this is also true of all partitions (V'_1, V'_2) as defined above. However, there are other partitions of G' that may split the external vertices that correspond to the same vertex in G into different parts and possibly have fewer violating edges. What we show is that such splits must introduce violating edges in the subgraphs between external and internal edges, where this is due to the expansion properties of the subgraphs. Roughly speaking, if a partition "avoids violations" between external vertices by splitting sets of external vertices, then it "pays" by introducing violations between external and internal vertices.

4.1.1 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 $\deg(v) > d$ the graph G' contains 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 $X(v)$ is assigned up to d specific neighbors of v according to some fixed, *but arbitrary* partition of the neighbors of v. As we shall see below, this assignment determines the edges between pairs of external vertices in G' that correspond to different vertices in G. 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 $\frac{\lfloor d^2/\deg(v) \rfloor}{2}$-multiple edges between every internal vertex and every external vertex in $H(v)$. It follows that the degree of every vertex within $H(v)$ is

$$\frac{\lfloor d^2/\deg(v) \rfloor}{2} \cdot \lceil \deg(v)/d \rceil \;\leq\; \frac{d^2}{2 \cdot \deg(v)} \cdot \frac{\deg(v)+d}{d} \;\leq\; \frac{d^2}{2 \cdot \deg(v)} \cdot \frac{2\deg(v)}{d} \;=\; d$$

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 [18, 17]. Furthermore, these constructions allow the determination of the i-th neighbor of any given vertex in constant time.

For sake of the presentation, when $\deg(v) \leq d$, so that v is represented by a single vertex, we let $H(v)$ be the subgraph that consists of this single vertex. This vertex is considered an external vertex, denoted $X_1(v)$, and it is assigned all neighbors of v.

We have described how vertices of G are transformed into vertices of G' (some of which are connected by edges). 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 that are assigned 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 $n' = |V(G')| \leq \sum_{v \in G} 2\lceil \deg(v)/d \rceil \leq 4n$, and $m' = m(G') \leq 4dn = 8m$.

For an illustration of the construction of G', see Figure 3.

We have thus established the first item in Theorem 5, and we turn to the second item. From the construction of G' it is obvious that if G is bipartite, then so is G'. It remains to prove the following lemma.

Lemma 5 *If G is ϵ-far from being bipartite (with respect to $m = (dn)/2$) then G' is ϵ'-far from being bipartite with respect to $d'n'$, for $\epsilon' = \frac{\epsilon}{144}$.*

In order to prove Lemma 5, we first prove the following proposition concerning bipartite expander graphs. For any two (disjoint) subsets of vertices, A and B, we let $e(A, B)$ denote the number of edges with one end-point in A and another in B.

Proposition 7 *Let $G = (A \cup B, E)$ be a d-regular bipartite graph with sides A and B of size s. Assume that all eigenvalues of the adjacency matrix of G, but the largest one and the smallest one, are at most λ in their absolute values. Assume further that $\lambda \leq d/4$. Then for every two partitions $A = A_1 \cup A_2$, $B = B_1 \cup B_2$, satisfying $|A_1| \geq s/2$,*

$$e(A_1, B_1) + e(A_2, B_2) \geq \frac{d|A_2|}{8} \; .$$

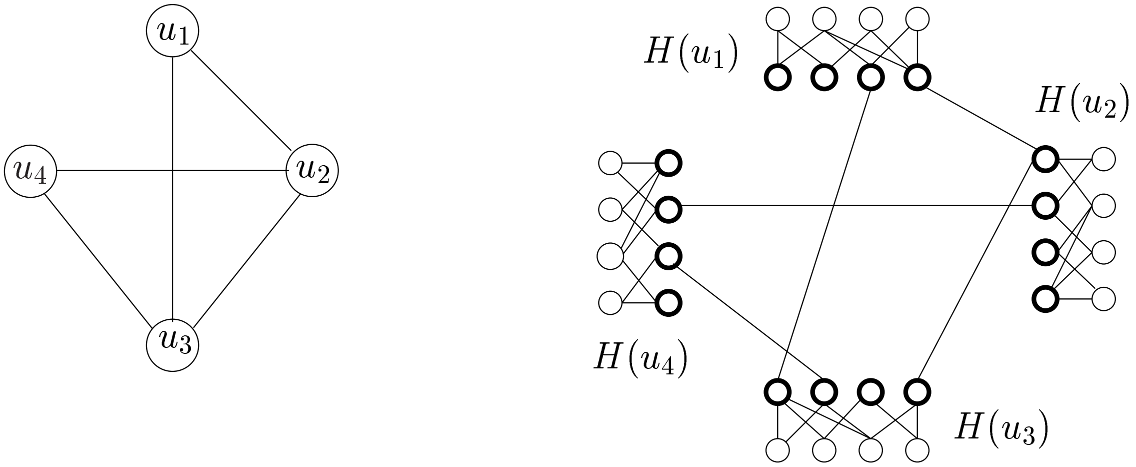

Figure 3: An illustration for the construction of G'. On the left are 4 vertices in G and their induced subgraph. On the right are the 4 corresponding subgraphs in G' and the edges between the external vertices in these subgraphs. (The external vertices are marked in bold, and there are additional edges that do not appear in the figure, between the external vertices in the figure and external vertices of other subgraphs.)

Proof: It is well known that the larger is the "spectral gap" (i.e., the difference between d and λ), the closer the edge distribution in G approaches that of a truly random bipartite graph with sides of size s and edge probability d/s. Specifically, for every $A_0 \subseteq A$, $B_0 \subseteq B$ of sizes $|A_0| = a_0$, $|B_0| = b_0$,

$$\left| e(A_0, B_0) - \frac{d a_0 b_0}{s} \right| \leq \lambda \sqrt{a_0 b_0} \tag{13}$$

(see, e.g., Chapter 9 of [5]).

Let $|A_1| = a_1$, $|A_2| = a_2 = s - a_1$, $|B_1| = b_1$, $|B_2| = b_2 = s - b_1$. It is given that $a_1 \geq s/2$. We may obviously assume that $b_1 \geq a_2/2$, as otherwise at least half of the edges incident to A_2 have their other endpoint outside B_1, implying $e(A_2, B_2) \geq d a_2/2$.

Applying the bound in Equation (13) twice we get:

$$\begin{aligned} e(A_1, B_1) &= d|B_1| - e(A_2, B_1) \geq d b_1 - \frac{d a_2 b_1}{s} - \lambda\sqrt{a_2 b_1} = \frac{d a_1 b_1}{s} - \lambda \sqrt{a_2 b_1}\,, \qquad (14) \\ e(A_2, B_2) &= d|A_2| - e(A_2, B_1) \geq d a_2 - \frac{d a_2 b_1}{s} - \lambda\sqrt{a_2 b_1} = \frac{d a_2 b_2}{s} - \lambda \sqrt{a_2 b_1}\ . \qquad (15) \end{aligned}$$

Consider first the case $b_2 \leq s/2$. In this case it follows from Equation (14) that

$$\begin{aligned} e(A_1, B_1) &\geq \frac{d a_1 b_1}{s} - \lambda\sqrt{a_2 b_1} \geq \frac{d(s/2)(s/2)}{s} - \lambda\sqrt{(s/2)s} \\ &= \frac{ds}{4} - \frac{\lambda s}{\sqrt{2}} \geq \frac{ds}{14} \geq \frac{d a_2}{7}\ . \end{aligned}$$

We thus assume that $b_2 > s/2$. If $d a_2 b_2/s \geq 2\lambda\sqrt{a_2 b_1}$, we obtain from Equation (15) that

$$e(A_2, B_2) \geq \frac{d a_2 b_2}{2s} \geq \frac{d a_2 (s/2)}{2s} = \frac{d a_2}{4}\ .$$

Hence we may assume that $da_2b_2/s \leq 2\lambda\sqrt{a_2b_1}$. If $da_1b_1/s \geq 2\lambda\sqrt{a_2b_1}$, then it follows from Equation (14) that

$$e(A_1, B_1) \geq \frac{da_1b_1}{2s} \geq \frac{d(s/2)(a_2/2)}{2s} = \frac{da_2}{8} ,$$

as required. Hence we may assume that $da_1b_1/s \leq 2\lambda\sqrt{a_2b_1}$. It remains to check that the latter assumption together with $da_2b_2/s \leq 2\lambda\sqrt{a_2b_1}$ bring to a contradiction. Indeed, multiplying these inequalities we get: $d^2a_1a_2b_1b_2/s^2 \leq 4\lambda^2a_2b_1$, or $d^2a_1b_2 \leq 4\lambda^2s^2$. Recalling that $a_1 \geq s/2$, $b_2 > s/2$, it follows that $d < 4\lambda$ – a contradiction to our assumption on λ. ■

Proof of Lemma 5: We shall show that the number of edges that should be removed from G' so as to make it bipartite, is at most a constant factor smaller than the number of edges that should be removed from G so as to make G bipartite. Since the total number of edges in G and in G' is of the same order, this suffices to prove the lemma. To this end we prove the contrapositive statement. Specifically, suppose G' is ϵ'-close to being bipartite with respect to $m_{\max} = d'n'$. Namely, there exists a partition $P' = (V_0', V_1')$ of the vertices in G' with respect to which there are at most $\epsilon' \cdot d'n'$ violating edges in G'. We shall show how to construct, based on P', a partition $P = (V_0, V_1)$ of the vertices in G with respect to which there are at most $\epsilon dn/2 = \epsilon m$ violating edges in G, thus proving the lemma.

Consider a particular vertex v in G, and the subset of external vertices $X(v)$ in G' that correspond to v. Let $X^0(v) = X(v) \cap V_0'$, and let $X^1(v) = X(v) \cap V_1'$. We refer to the larger subset as the *majority subset* of v, and to the smaller subset as the *minority subset* of v. We define $P = (V_0, V_1)$ by assigning each vertex v in G according to its majority subset. Namely, if $|X^0(v)| \geq |X^1(v)|$ then v is assigned to V_0, otherwise it is assigned to V_1. In what follows it will be convenient to refer to 0 and 1 as *colors*.

Also, when we refer to edges in G' as violating edges, we mean with respect to P', and when we refer to edges in G as violating edges, we mean with respect to P. Note that the partition P is defined only according to the coloring of the external vertices in G', ignoring the coloring of the internal vertices. Also recall that there is a one-to-one mapping between edges in G and edges in G' whose end-points are both external vertices.

Since each vertex v in G is assigned the color of its majority subset, the violating edges in G' between pairs of vertices that both belong to majority subsets, or between pairs of vertices that both belong to minority subsets, be violating edges in G. Similarly, non-violating edges in G' between vertices in majority subsets, or between vertices in minority subsets, become non-violating edges in G. It remains to deal with edges between minority and majority subsets in G'. These edges can be non-violating in G', but may become violating in G.

We next show that the total number of vertices in G' that belong to minority subsets can be bounded as a function of the number of violating edges in G'. To this end we show that if there were many minority vertices, then there would be many violating edges in G' *between internal and external vertices*.

For each vertex v in G, consider the majority and minority subsets of (the external vertices of) v. Let the majority subset of $X(v)$ be $X^\alpha(v)$ and let the minority subset be $X^\beta(v)$ (where $\alpha, \beta \in \{0, 1\}$).

Claim 5.1 *For every vertex v in G, the number of violating edges in G' between vertices in $X(v)$ and vertices in $I(v)$ is at least $\left|X^\beta(v)\right| \cdot (d/8)$.*

Proof: Similarly to our notation for external vertices, for the internal vertices of v let $I^0(v) \stackrel{\text{def}}{=} I(v) \cap V_0'$ and $I^1(v) \stackrel{\text{def}}{=} I(v) \cap V_1'$. Consider first the case $\frac{\deg(v)}{d} < d$. By construction of G',

$|X(v)| = |I(v)| = \lceil \deg(v)/d \rceil$, and there are $\frac{\lfloor d^2/\deg(v) \rfloor}{2}$ multiple edges between every pair of vertices (x, y) such that $x \in X(v)$ and $y \in I(v)$. Hence the number of edges between $X(v)$ and $I(v)$ that are violating (with respect to P') is

$$\begin{aligned} & \Big(|X^{\alpha}(v)||I^{\alpha}(v)| + |X^{\beta}(v)||I^{\beta}(v)|\Big) \cdot \frac{\lfloor d^2/\deg(v) \rfloor}{2} \\ \geq \quad & \Big(|X^{\beta}(v)||I^{\alpha}(v)| + |X^{\beta}(v)||I^{\beta}(v)|\Big) \cdot \frac{d^2}{4\deg(v)} = |X^{\beta}(v)| \cdot \frac{d}{4}. \end{aligned}$$

Next consider the more interesting case where $\frac{\deg(v)}{d} \geq d$. In this case Claim 5.1 directly follows from Proposition 7. $\diamond$ (Proof of Claim 5.1.)

Thus we can conclude that if there are w external vertices that belong to minority subsets then they contribute at least $w \cdot d/8$ violating edges in G'. Since the number of violating edges in G' is at most $\epsilon' n' d' \leq 8\epsilon' nd$, we have that $w \leq 64\epsilon' n$. As noted previously, the total number of violating edges in G is upper bounded by the number of violating edges in G' plus the number of edges between minority and majority (external) subsets. By the above discussion and the fact that every external vertex has at most d neighbors that are external vertices, the number of violating edges in G is at most $8\epsilon' nd + 64\epsilon' nd = 72\epsilon' nd$. Since $\epsilon' = \epsilon/144$, and $m = (nd)/2$, the lemma follows. ■

We have completed proving the first two items of Theorem 5, and we now turn to the third item.

4.2 Establishing Item 3 in Theorem 5

Here we stress that if the neighbor and the vertex-pair queries would have returned more information, then the proof of the current item would be significantly simpler. In particular, suppose that a neighbor query (u, i) is answered with a pair (v, j) (instead of only v), which means that v is the i-th neighbor of u, and u is the j-th neighbor of v. Suppose also that when performing a vertex-pair query (u, v), the algorithm is not only told whether (u, v) is an edge or not, but rather, in the former case it is also provided with pair (i, j), which means that v is the i-th neighbor of u, and u is the j-th neighbor of v. With this additional information, the structure of G' is implicitly given and, thus, the emulation of random walks in the execution of the procedure Odd-Cycle on G', is straightforward. Here we need to work harder to overcome the lack of information.

Random-Walk Steps. We first shortly discuss the emulation of random walks. If the walk stays at the current vertex, then clearly there is no need for any emulation. Hence, we only need to consider the case in which we have to select a random neighbor. Recall that vertices in G' are either of the form $X_i(v)$ (the i-th external vertex corresponding to vertex v in G), or $I_i(v)$ (the i-th internal vertex), where $1 \leq i \leq \lceil \deg(v)/d \rceil$. Recall that we can obtain $\deg(v)$ for any v by a single degree query, and in particular use this to find the degree of vertices in G'. For simplicity of the presentation, we assume from this point on that for every vertex v in G, the degree of every vertex $I_i(v)$ in G' is d (instead of being at most d), and the degree of every $X_i(v)$ is $2d$ (instead of being at most $2d$).

Performing a random-walk step in G' from an internal vertex $I_i(v)$ can be easily done by using the explicit structure of the graph $H(v)$ (which is either a complete bipartite graph with multiple edges, or an explicitly constructible expander). In order to perform a random-walk step from an external vertex, $X_i(v)$, we first determine whether to take one of the d edges within the graph $H(v)$, or whether to take one of the d edges going from $X_i(v)$ to another external vertex. In the first case we then select an internal neighbor given the explicit structure of $H(v)$. It remains to deal with selecting an external neighbor. Note that in the special, but easy, case in which $\deg(v) \le d$ and so $H(v)$ is a single vertex, there is only the latter option.

As noted just following the statement of Theorem 5, we actually construct G' as we go along. The important thing to note is that the definition of G' allows us to assign the vertices in $X(v)$ edges of v in an *arbitrary* manner (as long as each $X_i(v)$ is assigned (at most d) different edges). Hence, all we need to take care of is to be consistent with previous choices, and to ensure the correct distribution in the choice of the random walk step. To this end we may think of each external vertex as having d "ports", labeled $1, \cdots, d$, which are initially unassigned. As the algorithm proceeds, it puts a "link" between, say, the t'th port of $X_i(v)$ and the ℓ'th port of $X_j(u)$ (where $(v,u) \in E(G)$). When performing a random-walk step from $X_i(v)$ (to a vertex outside of $H(v)$), we uniformly select a port. Let us denote the index of the port selected by t. If port t of $X_i(v)$ is already linked to another port, then we simply take this link to the port (and vertex) at the other end. Otherwise, we first set the link, and then take it. In order to set the link properly, we need to uniformly select a neighbor u of v among the neighbors of v *that were not yet assigned* (to any port of one of the external vertices of v). After doing so, and selecting a neighbor u, we need to select a yet unassigned port of one of the external vertices of u. The above can be done, with the aid of sampling, at an amortized cost of $O(\log n)$ queries in G. Details follow.

For each external vertex $X_i(v)$, the algorithm keeps a vector of length d, $\Gamma_i(v)$. The k'th entry of $\Gamma_i(v)$ contains the k'th neighbor of $X_i(v)$, if it was determined, and is empty otherwise. Denote by $f_i(v)$ the number of free entries in the vector $\Gamma_i(v)$. Also denote by $A(v)$ the set of neighbors of v that were already assigned to external vertices of v, and by $NA(v)$ the vertices that were not assigned. When setting a link to a yet-unassigned port (filling in a new entry in $\Gamma_i(v)$), we distinguish between two cases.

Case 1: If less than half of the neighbors of v in G are in $A(v)$, the algorithm repeats at most $(\log^2 n)$ times the following procedure: It chooses uniformly at random a neighbor of v in G. If that neighbor belongs to $NA(v)$, then a desired neighbor is found. By repeating the above procedure $O(\log^2 n)$ times, the probability that the algorithm didn't find a desired neighbor is at most $o(1/n)$. In this case we say that the algorithm fails. Since the total number of queries that the algorithm performs is at most $o(n)$, the total failure probability of the algorithm is $o(1)$. Suppose that a desired neighbor of v, with the name u is found. In that case the algorithm should move to one of the external vertices of u. The selected vertex $X_k(u)$ is chosen with probability: $\frac{f_k(u)}{\sum_{1 \le j \le \lceil \deg(u)/d \rceil} f_j(u)}$. According to the chosen $X_j(u)$, the algorithm sets $\Gamma_i(v)[t] \leftarrow X_j(u)$. In addition, the algorithm chooses uniformly at random one of the $f_j(u)$ free entries in the vector $\Gamma_j(u)$. Assume that the chosen index is t', $1 \le t' \le d$, then, the algorithm sets $\Gamma_j(u)[t'] \leftarrow X_i(v)$.

Case 2: If more than half of the neighbors of v in G are in $A(v)$, the algorithm reads all the neighbors of v in G that belong to $NA(v)$, and attaches them arbitrarily to the unoccupied entries in $\Gamma_j(v)$, $1 \le j \le \lceil \deg(v)/d \rceil$. By doing this, the algorithm (at most) doubles the number of neighbor queries performed on vertex v of G. Now, suppose that in $\Gamma_i(v)[t]$ there is a name of a vertex of G, say u. In that case, the algorithm should move to one of the external vertices $X_k(u)$, $1 \le k \le \lceil \deg(u)/d \rceil$, and this is done as in the first case.

Modifying the procedure Odd-Cycle. The procedure Odd-Cycle' is the same as Odd-Cycle in terms of the performance of random walks, which are emulated as described above. The only modification is in the last stage, where the procedure performs vertex-pair queries. Let (x,y) be the pair of vertices queried in G'. We answer the query as follows. If $(x,y) = (X_i(v), I_j(v))$ for some vertex v in G, then we answer according to the explicit construction of the subgraph $H(v)$.

If $(x, y) = (X_i(v), I_j(u))$ for $u \neq v$, then the answer is always negative. If $(x, y) = (X_i(u), X_j(v))$ then we query the pair (u, v) in G. If there is no edge between (u, v) in G, we answer that there is no edge between $X_i(u)$ and $X_j(v)$. Otherwise, we give a positive answer. While this answer may be inconsistent with the construction of G', (since it would correspond to having a complete bipartite subgraph in G' between the external vertices of u and the external vertices of v), it always provides evidence to an odd cycle in the input graph G. An explanation follows.

Consider two paths in G', where both paths start at the same vertex $x \in H(s)$, end at a pair of external vertices $X_i(v)$ and $X_j(u)$, respectively, and whose lengths have the same parity (so that $X_i(v)$ and $X_j(u)$ both belong to the same A_b, $b \in \{0,1\}$). By construction of G', such a pair of paths in G' corresponds to a pair of paths in G, which start at s, end at v and u respectively, and have the same parity b as well. But if there is an edge in G between v and u, then there is an odd cycle in G.

4.3 Establishing Item 4 in Theorem 5

In this subsection we prove the last item in Theorem 5. Recall that we are interested in a procedure for selecting a vertex in G' so that there is a sufficiently high probability of hitting any fixed sufficiently large subset of vertices in G'. In particular, if G' is far from being bipartite then we are interested in hitting the subset of vertices s for which Odd-Cycle(s) returns found with probability at least $2/3$.

Let $V_\ell(G) = \{v \in V(G) : \deg(v) \leq d\}$ and let $V_h(G) = \{v \in V(G) : \deg(v) > d\}$, where '$\ell$' stands for *low*, and 'h' for *high*, and as before, $d = d_{\text{avg}}(G)$ denotes the average degree of vertices in G. We also define the corresponding sets in G': $V_\ell(G') = \{x \in V(H(v)) : v \in V_\ell(G)\}$ and $V_h(G') = \{x \in V(H(v)) : v \in V_h(G)\}$. (Recall that $H(v)$ is the subgraph in G' that corresponds to v, and $V(H(v))$ is its set of vertices.) Since $V(G') = V_\ell(G') \cup V_h(G')$, it follows that selecting a vertex uniformly in $V(G')$ can be done by first deciding whether to pick a vertex from $V_\ell(G')$ or from $V_h(G')$ with probability proportional to the size of each set (relative to $n' = |V(G')|$), and then picking a vertex uniformly from the selected set.

Recall that for every $v \in V_\ell(G)$ we have $|V(H(v))| = 1$ while for every $v \in V_h(G)$, $|V(H(v))| = 2\lceil \deg(v)/d \rceil$. Therefore, picking a vertex uniformly in $V_\ell(G')$ corresponds to picking a vertex uniformly in $V_\ell(G)$, while picking a vertex uniformly in $V_h(G')$ corresponds to picking a vertex in $V_h(G)$ with probability proportional to its degree.

Since we are not required to actually select every vertex in $V(G')$ with exactly equal probability, but rather we are required to be able to select all but $\delta n'$ of the vertices in $V(G')$ with probability at least $\Omega(1/n')$, we may perform the above steps in an approximate manner. In particular, by taking a sample of $\Theta(1/\delta^2)$ vertices in G and querying their degrees, we may obtain an estimate, denoted $\hat{\mu}(G')$, of $|V_\ell(G')|/n' = |V_\ell(G)|/n'$ such that if $|V_\ell(G')|/n' \geq \delta/2$ then $(1/8)|V_\ell(G')|/n' \leq \hat{\mu}(G') \leq 2(|V_\ell(G')|/n')$ (recall that $n \leq n' \leq 4n$). In order to uniformly select a vertex in $V_\ell(G)$ (so as to obtain a uniformly selected vertex in $V_\ell(G')$), we can simply take a sample of vertices from $V(G)$, query their degrees, and pick the first vertex in the sample that belongs to $V_\ell(G)$, if such exists. If $|V_\ell(G')|/n' \geq \delta/2$, so that $|V_\ell(G)|/n \geq \delta/2$, then a sample of size $O(1/\delta)$ suffices to ensure that with high constant probability, the sample will indeed contain a vertex in $V_\ell(G)$.

The only step that is more involved is that of selecting a vertex in $V_h(G)$ with probability proportional to its degree. Observe that selecting a vertex from all of $V(G)$ with probability proportional to its degree can be performed by uniformly selecting an *edge* in $E(G)$ and then selecting one of its two end-points with equal probability. In the next subsection we describe and analyze a procedure that performs a certain approximation to the uniform selection of an edge in

$E(G)$. In Subsection 4.3.2 we return to the problem of selecting a vertex in G' almost uniformly.

4.3.1 Sampling Edges Almost Uniformly in G

We consider two cases: $d > \sqrt{\delta n}$ and $d \leq \sqrt{\delta n}$. Recall 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.

```
Sample-Edges-Uniformly-in-G
    Repeat $\Theta(n/d)$ times:
    • Select two vertices in $G$ uniformly and at random.
    • Check if there is an edge between these two vertices (by performing a single vertex-pair
      query). If the answer is positive then output this edge (and exit the repeat loop).
```

Figure 4: A procedure for sampling edges uniformly in G.

The straightforward procedure for this case is given in Figure 4. Clearly every edge in G has equal probability of being selected by the procedure, and the query complexity of this procedure is $\Theta(n/d)$, which for $d > \sqrt{\delta n}$ is $\Theta(\min(\sqrt{n/\delta}, n/d)$ as required. (To be more precise, there is some probability that this procedure fails to output an edge. However, the probability that this occurs can be made sufficiently small so as to have a negligible effect on the success probability of our algorithm.)

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 6 *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* $u, v \in U_0$*.*

```
Sample-Edges-Almost-Uniformly-in-G($\delta$)
    1. Let $t = 2\sqrt{n/\delta} \cdot \log m$. Uniformly select a subset of vertices $S \subset V(G)$, where $|S| = t$.
    2. Partition the sampled vertices into subsets according to their degree: $S_i = \{v \in S :
       \deg(v) \in (2^{i-1}, 2^i]\}$.
    3. Choose an index $i$, $1 \leq i \leq \log m$ with probability $\frac{|S_i| 2^i}{\sum_i |S_i| 2^i}$.
    4. Uniformly select a vertex $v \in S_i$.
    5. Uniformly select an edge incident to $v$.
```

Figure 5: A procedure for selecting an edge in G so that all but at most a $(\delta/4)$-fraction of the edges are selected with probability $\Omega(1/m)$.

The procedure referred to in Lemma 6 is described in Figure 5. Before proving Lemma 6 we provide some intuition concerning this procedure. We define the following $\log m$ "buckets": for $1 \leq i \leq \log m$,

$$B_i = \left\{ v \in V(G) \ : \ \deg(v) \in (2^{i-1}, 2^i] \right\} \ . \tag{16}$$

Thus, in each bucket, all vertices have approximately the same degree. Suppose we had a way to pick an index i with probability proportional to $|B_i| \cdot 2^i$, which is approximately the same as picking i with probability proportional to the number of edges incident to vertices in B_i. Further assume that for each i we could uniformly select a vertex in B_i. Then we could select an edge almost uniformly by selected an index i as described above, uniformly selecting a vertex $v \in B_i$, and then uniformly selected an edge incident to v.

The procedure Sample-Edges-Almost-Uniformly-in-G can be viewed as approximating this "ideal" procedure. Assume first that all buckets are relatively large (i.e., $|B_i| = \Omega(\sqrt{n})$ for every $1 \leq i \leq \log m$). Then, by taking a uniformly selected sample of $\tilde{\Theta}(\sqrt{n})$ vertices in G, we can obtain a good estimate of $|B_i|$ (and by that, of $|B_i| \cdot 2^i$), for every i, and we can uniformly select a vertex $v \in B_i$ for any given i. Unfortunately, if some buckets are small, then we might not sample from them at all.

To illustrate the seeming difficulty with such a case, suppose the graph is a star, so that there is one vertex, denoted v^* with degree $n-1$, and all other vertices have degree 1. In terms of our buckets we have $|B_0| = n-1$ so that $|B_0| \cdot 2^0 = n-1$, and $|B_{\log n}| = 1$, so that $|B_{\log n}| \cdot 2^{\log n} = n$. The "ideal" procedure would select each of the two buckets roughly with equal probability, and if it selects the bucket $B_{\log n}$ then it picks v^*. In other words, it picks v^* with probability roughly 1/2. But if we take a sample of $\tilde{\Theta}(\sqrt{n})$ vertices then the probability v^* falls in the sample is extremely small. However, this turns out to be almost immaterial to the analysis since we are interested in the end result of the distribution on *edges*. In the case of the star graph, every edge incident to v^* is also incident to one of the degree-1 vertices, and hence will be selected with equal probability.

In general, as we shall see in detail in the proof of Lemma 6, we can lower bound the probability of selecting each edge that has both end-points in sufficiently large buckets. On the other hand, we can upper bound the total number of edges that have both end-points is small buckets. Details follow.

Proof of Lemma 6: Let the subsets ("buckets") B_i be as defined in Equation (16). Note that in the procedure Sample-Edges-Almost-Uniformly-in-G (Figure 5), the subsamples S_i are simply $S_i = S \cap B_i$. Next we define a set of indices I_0, that includes indices of all buckets B_i that have "few" elements. More precisely,

$$I_0 = \left\{ i \ : \ 1 \leq i \leq \log m \text{ and } |B_i| \leq \frac{\sqrt{\delta n}}{2 \log m} \right\} \ . \tag{17}$$

Let U_0 be the set of vertices that belong to buckets with "few" elements: $U_0 = \bigcup_{i \in I_0} B_i$.

Consider any fixed vertex $v \notin U_0$, and let $i(v)$ be the index of the bucket that v belongs to, that is, $v \in B_{i(v)}$. We denote by C_v the event that v is selected in the Step 4 of the sampling algorithm. Then, for a vector $(s_1, \cdots, s_{\log m})$ we can estimate:

$$\Pr[C_v \ : \ |S_1| = s_1, \cdots, |S_{\log m}| = s_{\log m}] = \frac{s_{i(v)}}{|B_{i(v)}|} \cdot \frac{s_{i(v)} 2^{i(v)}}{\sum_i s_i 2^i} \cdot \frac{1}{s_{i(v)}} \tag{18}$$

(first require that v falls in $S_{i(v)}$, then choose the bucket $S_{i(v)}$, and then choose v inside $S_{i(v)}$). Thus the above conditional probability is:

$$\frac{s_{i(v)}}{|B_{i(v)}|} \cdot \frac{2^{i(v)}}{\sum_i s_i 2^i} \geq \frac{s_{i(v)}\deg(v)}{|B_{i(v)}|\sum_i s_i 2^i} \ . \tag{19}$$

The random variable $s_{i(v)}$ is hypergeometrically distributed with parameters n, $|B_{i(v)}|$ and t. It thus has mean $t|B_{i(v)}|/n$, and using known bounds on the tails of the hypergeometric distribution and our assumption on v ($v \notin U_0$ and therefore $B_{i(v)}$ is large), we can get:

$$\Pr\left[\frac{s_{i(v)}}{|B_{i(v)}|} \geq \frac{t}{2n}\right] \geq \frac{3}{4} \ .$$

Consider the sum $\sum_{i=1}^{t} s_i 2^i$. We have:

$$\begin{aligned}
\text{Exp}\left[\sum_i |S_i| 2^i\right] &= \sum_i \frac{|B_i|}{n} \cdot t \cdot 2^i \\
&= \frac{t}{n} \cdot \sum_i |B_i| 2^i \\
&\leq \frac{t}{n} \cdot \sum_i \sum_{v \in B_i} (2\deg(v)) \quad \leq \quad \frac{4 \cdot t \cdot m}{n} \ .
\end{aligned}$$

By Markov's inequality, the probability that $\sum_i |S_i| 2^i > \frac{16tm}{n}$ is less than 1/4. It thus follows that

$$\Pr\left[\left(\frac{s_{i(v)}}{|B_{i(v)}|} \geq \frac{t}{2n}\right) \& \left(\sum_i s_i 2^i \leq \frac{16tm}{n}\right)\right] \geq \frac{1}{2} \ . \tag{20}$$

Consider only such vectors $(s_1, \cdots, s_t)$. From Equation (19) we obtain:

$$\Pr[C_v] \geq \frac{1}{2} \cdot \frac{t}{2n} \cdot \frac{\deg(v)}{\frac{16tm}{n}} = \frac{\deg(v)}{64m} \ . \tag{21}$$

Finally, for an edge $e \in E(G)$, let us denote by C_e the event that e is selected in the fourth step of the procedure Sample-Edges-Almost-Uniformly-in-G. Let $E(U_0)$ denote the set of edges between pairs of vertices in U_0, that is, $E(U_0) = \{(u,v) \in E(G) \ : \ u, v \in U_0\}$. By definition of U_0, $|U_0| \leq \sqrt{\delta n}/2$, and hence $|E(U_0)| \leq (\delta n)/4$. Therefore, for all but at most $(\delta n)/4 \leq \delta m/4$ of the edges in $E(G)$, at least one of their end-points is *not* in U_0. (Note that here we have used the assumption that $m \geq n$.) For each such edge (u,v) where $u \notin U_0$ (or $v \notin U_0$), we have

$$\Pr[C_e] \geq \Pr[C_u] \cdot \frac{1}{\deg(u)} \geq \frac{\deg(u)}{64m} \cdot \frac{1}{\deg(u)} = \frac{1}{64m} \ . \tag{22}$$

■

4.3.2 Sampling Vertices in G'

We now return to selecting vertices in G'. As discussed earlier, we can easily obtain an estimate of $|V_\ell(G')|/n' = |V_\ell(G)|/n'$, denoted $\hat{\mu} = \hat{\mu}(G')$, such that if $|V_\ell(G')|/n' \geq \delta/2$ then $(1/8)|V_\ell(G')|/n' \leq \hat{\mu}(G') \leq 2(|V_\ell(G')|/n')$, and hence we assume that we indeed have such an estimate.

```
Sample-Vertices-Almost-Uniformly-in-G'($d, \delta, \hat{\mu}$)

    1. Flip a coin with bias $\hat{\mu}$.
    2. If the outcome is "heads" then do (select a vertex in $V_\ell(G')$):
        (a) Uniformly select $\Theta(1/\delta)$ vertices in $G$ and query their degrees.
        (b) If some vertex in the sample belongs to $V_\ell(G)$ then let $v$ be the first such vertex and
            output the single vertex $x \in V(H(v))$. Otherwise, pick an arbitrary vertex $v$ in the
            sample and output an arbitrary vertex $x \in V(H(v))$.
    3. Else (the outcome is "tails") do (select a vertex in $V_h(G')$):
        (a) If $d > \sqrt{\delta n}$ then sample an edge $e \in E(G)$ by running the procedure Sample-Edges-
            Uniformly-in-G. In case the procedure fails, pick an arbitrary edge $e$ in $E(G)$.
        (b) Else ($d \leq \sqrt{\delta n}$), sample an edge $e \in E(G)$ by running the procedure Sample-Edges-
            Almost-Uniformly-in-G($\delta/2$).
        (c) Choose with equal probability one of the end-points $v$ of the edge $e$.
        (d) Choose uniformly at random one of the vertices $x$ in $V(H(v))$.
```

Figure 6: A procedure for selecting a vertex in G' so that all but at most a δ-fraction of the vertices are selected with probability $\Omega(1/n')$.

Proof of Item (4) in Theorem 5. We now show that the procedure Sample-Vertices-Almost-Uniformly-in-G' (see Figure 6), is as required in Item (4) of Theorem 5. Consider first the vertices in $V_\ell(G')$. If $|V_\ell(G')|/n' \geq \delta/2$, then for each vertex $x \in V_\ell(G')$, the probability that we select x is $\hat{\mu}(G')$, times the probability that the sample contains a vertex in $V_\ell(G')$, times $1/|V_\ell(G')|$. Since $\hat{\gamma} = \Omega(|V_\ell(G')/n')$ and the sample contains a vertex from $V_\ell(G)$ with constant probability, the probability that we select x is $\Omega(1/n')$ as required. If $|V_\ell(G')|/n' < \delta/2$ then the probability that we obtain any vertex in $V_\ell(G')$ may be very small, but we are allowed to have $\delta n'$ such vertices and we shall accoount for these at most $(\delta/2)n'$ vertices.

We now turn to the vertices in $V_h(G')$. For an edge $e \in E(G)$, let C_e denote the event that e is selected by Sample-Edges-Uniformly-in-G in case $d > \sqrt{\delta n}$, or by Sample-Edges-Almost-Uniformly-in-G in case $d \leq \sqrt{\delta n}$. For $v \in V(G)$ let C_v denote the event that v is selected in Step 3c of procedure Sample-Vertices-Almost-Uniformly-in-G'. For $x \in V(G')$ let C_x denote the event that x is selected in Step 3d of the procedure and let $v(x)$ be such that $x \in H(v(x))$. Recall that for every $v \in V_h(G)$, we have $|V(H(v))| = 2\lceil \deg(v)/d \rceil$. Therefore for every $x \in V_h(G')$,

$$\Pr[C_x] \;\geq\; \Pr[C_{v(x)}] \cdot \frac{1}{2\lceil \deg(v(x))/d \rceil} \;=\; \sum_{e=(u,v(x))} \frac{1}{2} \Pr[C_e] \cdot \frac{1}{2\lceil \deg(v(x))/d \rceil} \tag{23}$$

If $d > \sqrt{\delta n}$ then $\Pr[C_e]$ is only slightly smaller than $1/m$ (since there is a probability that the procedure Sample-Edges-Uniformly-in-G fails to output an edge), implying that $\Pr[C_x] = \Omega(1/n')$. If $d \leq \sqrt{\delta n}$, then $\Pr[C_e]$ is determined by the procedure Sample-Edges-Almost-Uniformly-in-G. Let U_0 be as defined in Lemma 6, and consider first the case where $v(x) \notin U_0$. Then for every edge e that is incident to $v(x)$, we have that $\Pr[C_e] \geq 1/(64m) = 1/(32dn)$. By Equation (23), for each such vertex x we have that

$$\Pr[C_x] \geq \frac{1}{2}\deg(v(x)) \cdot \frac{1}{32dn} \cdot \frac{1}{2\lceil \deg(v(x))/d \rceil} \geq \frac{1}{256n} \geq \frac{1}{256n'} \tag{24}$$

as required. Next consider the case that $v(x) \in U_0$ but $\deg(v(x)) \geq 2|U_0|$. In such a case for at least half of the edges e incident to $v(x)$ we have that $\Pr[C_e] \geq 1/(32dn)$, and we can deduce that $\Pr[C_x] \geq \frac{1}{512n'}$. It remains to show that the total number of vertices x such that $v(x) \in U_0$ and $\deg(v) \leq 2|U_0|$, is at most $(\delta/2)n'$. Using the fact that $|U_0| \leq \sqrt{(\delta/2)n}/2$ (recall that the procedure Sample-Edges-Almost-Uniformly-in-G is called with its input parameter set to $\delta/2$), and for every vertex $v \in V_h(G)$, $|V(H(v))| = 2\lceil \deg(v)/d \rceil \leq 2\deg(v)$, we get:

$$\sum_{v \in U_0 \colon \deg(v) \leq 2|U_0|} |V(H(v))| \;\leq\; 4|U_0|^2 \;\leq\; (\delta/2)n' \; . \tag{25}$$

The lemma follows.

4.4 A Probabilistic Construction of G'

In this subsection we describe an alternative, probabilistic construction of a graph G' that establishes Theorem 5. More precisely, we describe such a construction that, under certain conditions on d and ϵ, results, with very high probability, in a graph G' as required in the theorem. Here too the graph is constructed as the algorithm proceeds. In all that follows, let $d = d_{\rm avg}(G)$, and let $d' = d_{\max}(G')$. Let n (n') be the number of vertices in G (G'), and m (m') be the number of edges in G (G'). The probabilistic construction is simpler and it is possible that it may be applicable to other problems as well. However in this construction we need that $d = \Omega(\log n + 1/\epsilon)$.

4.4.1 The Construction

As in the deterministic construction, 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, $m' = |E(G')| = |E(G)| = (nd)/2$.

Lemma 7 *For $d = \Omega(\log n)$, the maximum degree d' of G' constructed above is $2d$ with probability $1 - o(1)$.*

Proof: Let $W_v^{i,j}$ be an indicator random variable which is 1, if the j-th induced edge of $v \in V(G)$ $(1 \leq j \leq \deg(v))$, chooses vertex $X_i(v) \in V(G')$. Let $W_v^i \stackrel{\rm def}{=} \sum_{1 \leq j \leq \deg(v)} W_v^{i,j}$. W_v^i is a sum of j independent indicator variables. $\mathrm{Exp}[W_v^i] = \deg(v) \cdot 1/(\deg(v)/d) = d$. Let X_v^i be an indicator variable which is 1, if $W_v^i > 2\mathrm{Exp}[W_v^i] = 2d$, and 0 otherwise.

Using standard bounds on tails of sums of bounded random variables (see, e.g., [5]), for a specific $v \in V(G)$ and $1 \leq i \leq \lceil \deg(v)/d \rceil$, it follows that $\Pr[X_v^i = 1] < e^{-cd}$. Using a union bound over all X_v^i's $v \in V(G)$ and $1 \leq i \leq \lceil \deg(v)/d \rceil$, we get that the probability that there exists a vertex v and an index i for which $X_v^i = 1$ is at most $n' \cdot e^{-cd} = o(1)$, thus with probability $1 - o(1)$, the maximum degree of G' constructed above is $2d$. ■

Lemma 8 *For a graph G with $d > 512/\epsilon$ the following holds: If G is ϵ-far from being bipartite (with respect to $m = (dn)/2$), then with probability $1 - o(1)$, G' is ϵ'-far from being bipartite with respect to $d'n'$, for $\epsilon' = \frac{\epsilon}{128}$.*

Proof: Consider a fixed partition $P' = (V'_0, V'_1)$ of the vertices in G'. The partition P' induces a partition of $X(v)$ for every v. Let us denote by $X^\alpha(v)$ the majority subset of $X(v)$ induced by P'. Consider a partition $P = (V_0, V_1)$ of the vertices of G induced by P' in the following way. For $v \in V(G)$, if $X^\alpha(v) \subset V'_0$, then $v \in V_0$, otherwise $v \in V_1$. Since G is ϵ-far from being bipartite, at least one of the subsets V_0, V_1 contains $\frac{1}{4}\epsilon nd$ edges. W.l.o.g. assume that $|E(V_0)| \geq \frac{1}{4}\epsilon nd$. Let H' be a subgraph of G', defined as follows. The vertices of H' are:

$$\bigcup_{v \in V_0} X^\alpha(v)$$

The edges of H' are the edges of G', induced by $V(H')$. Thus, $V(H') \subset V'_0$ and $E(H') \subset E(V'_0)$.

Claim 8.1 *$\Pr[|E(H')| \leq (\epsilon/32)nd] < 2^{-c \cdot n}$, where $c > 1$.*

Once Claim 8.1 is proved, by taking a union bound over all possible partitions P' of the vertices of G' we get that for *every* partition $P' = (V'_0, V'_1)$ of the vertices of G', the number of violating edges in G' is at least $\epsilon/32 \cdot n \cdot d$ with probability $1 - o(1)$. Recall that $n' \leq 2n$ and $d' \leq 2d$ with probability $1 - o(1)$. Thus, for *every* partition of the vertices of G', the number of violating edges is at least $\epsilon/128 \cdot n' \cdot d'$ with probability $1 - o(1)$, as required.

Proof of Claim 8.1 Consider an edge $e = (u, v) \in E(G)$, and let $\phi(e) = (X_i(v), X_j(u))$ be its corresponding edge in $E(G')$. For $e \in E(V_0)$, $\Pr[\phi(e) \in E(H')] \geq 1/2 \cdot 1/2 = 1/4$. Thus,

$$\text{Exp}[|E(H')|] \geq 1/4|E(V_0)|.$$

As $|E(H')|$ is a sum of $|E(V_0)|$ independent Bernoulli random variables, each with expectation at least 1/4, it follows from standard bounds on the tails of binomial random variables (see, e.g., [5]) that

$$\Pr[X_{|E(V_0)|} < (1/32)\epsilon nd] < e^{-\epsilon nd/512}$$

Thus, for $d \geq 512/\epsilon$ we have

$$\Pr[|E(H')| \leq \epsilon/32nd] < 2^{-cn}$$

for $c > 1$, and the lemma is proved. ■(Proof of Claim 8.1 and Lemma 8.)

We have completed proving the first two items of Theorem 5, with regard to the probabilistic construction.

4.4.2 Proving Item (3) in Theorem 5

In order to prove this item we need to explain how to emulate queries in G' by performing queries in G.

Degree Queries. Here we assume that in G' the maximum degree is $2d$. It was proved before to be true with probability $1 - o(1)$. Consider the following procedure Assign-Edges(v) where v is a vertex of G: Find $\deg(v)$ by one query on G; for each edge-index $r \in \{1, \cdots, \deg(v)\}$, choose one of the vertices $X_i(v) \in X(v)$ $(1 \leq i \leq \lceil \deg(v)/d \rceil)$, uniformly at random. Denote by $Q_i(v)$ a vector of $2d$ cells. This vector contains the indices of edges that are assigned to $X_i(v)$. Finally, choose an empty cell t in the vector $Q_i(v)$, and set $Q_i(v)[t] := r$. This procedure corresponds to the random construction of G'.

Neighbor Queries. For each vertex $X_i(v)$, the algorithm keeps a vector (of length $2d$), $W_i(v)$ that contains an ordered list of all the vertices of G' s.t. the algorithm is committed to the existence of an edge between them and $X_i(v)$. That is, if $W_i(v)[t] = X_j(u)$, it means that there is an edge $(X_i(v), X_j(u))$ in G'. In this case $W_j(u)[t'] = X_i(v)$ for some t', $1 \leq t' \leq 2d$. Note that the procedure Assign-Edges(v) assigns each of the edges of v independently to one of the vertices $X_i(v)$, $1 \leq i \leq \lceil \deg(v)/d \rceil$.

Suppose that the algorithm is located in $X_i(v)$ and it wishes to perform a neighbor query. Choose uniformly at random, a number k, $1 \leq k \leq 2d$. Take an empty vector $Q_i(v)$ of length $2d$ and copy into it the vector $W_i(v)$. By that the algorithm keeps its commitment about the edges which have already been exposed. Then, apply the Assign-Edges(v) procedure only for edges of v, that do not appear in one of the $W_i(v)$, $1 \leq i \leq \lceil \deg(v)/d \rceil$ (i.e. to edges which were not exposed yet by the algorithm). If $Q_i(v)[k]$ is empty then the walk remains at $X_i(v)$. If $Q_i(v)[k]$ contains a name of a vertex in G', the walk moves to that vertex, otherwise, $Q_i(v)[k]$ contains an index of a neighbor of v in G. By performing a single neighbor query on G, the algorithm determines the name of that neighbor (say) u in G. Then the walk needs to move to one of the vertices $X_j(u)$, $1 \leq j \leq \deg(u)/d$. It chooses one of the vertices $X_j(u)$ uniformly at random. For the chosen j set $W_i(v)[k] := X_j(u)$. Now choose a free cell k', $1 \leq k' \leq 2d$, in $W_j(v)$, and set $W_j(v)[k'] := X_i(v)$. The walk is now at $X_j(u)$.

Modifying the procedure Odd-Cycle. The procedure is modified in a similar manner to that described for the deterministic construction. In particular, random-walks are emulated as described above. In the last stage, when vertex-pair queries are performed we do the following. In order to answer a vertex-pair query $(X_i(v), X_j(u))$ in G', perform a vertex pair query (u, v) in G. If there is no edge between (u, v) in G, answer that there is no edge between $X_i(u)$ and $X_j(v)$. Otherwise, as explained for the determinstic case, an odd cycle is detected in G.

4.4.3 Establishing Item 4 in Theorem 5

In this subsection we prove the last item in Theorem 5 (for the probabilistic construction). The proof is essentially a simplified version of the proof for the deterministic case, but for sake of completeness we include the details.

Recall that we are interested in a procedure for selecting a vertex in G' such that there is sufficiently high probability of hitting any fixed sufficiently large subset of vertices in G' (and in particular of hitting the subset of vertices s for which Odd-Cycle(s) returns found with probability at least 2/3 (in the case G' is far from being bipartite)).

We now show that the procedure Sample-Vertices-Almost-Uniformly-in-G' (see Figure 7) is as required in Item (4) of Theorem 5.

For an edge $e \in E(G)$, let C_e denote the event that e is selected by Sample-Edges-Uniformly-in-G in case $d > \sqrt{\delta n}$, or by Sample-Edges-Almost-Uniformly-in-G in case $d \leq \sqrt{\delta n}$. For $v \in V(G)$ let C_v denote the event that v is selected in Step 3 of procedure Sample-Vertices-Almost-Uniformly-in-G'. For $x \in V(G')$ let C_x denote the event that x is selected in Step 4 of the procedure and let $v(x)$ be such that $x \in X(x(v))$. Recall that $|X(v)| = \lceil \deg(v)/d \rceil$. Therefore for every $x \in V(G')$,

$$\Pr[C_x] \;\geq\; \Pr[C_{v(x)}] \cdot \frac{1}{\lceil \deg(v(x))/d \rceil} \;=\; \sum_{e=(u,v(x))} \frac{1}{2} \Pr[C_e] \cdot \frac{1}{\lceil \deg(v(x))/d \rceil}. \tag{26}$$

If $d > \sqrt{\delta n}$ then $\Pr[C_e]$ is only slightly smaller than $1/m$ (since there is a probability that the procedure Sample-Edges-Uniformly-in-G fails to output an edge), implying that $\Pr[C_x] = \Omega(1/n')$.

```
**Sample-Vertices-Almost-Uniformly-in-G'**$(d, \delta)$

1. If $d > \sqrt{\delta n}$ then sample an edge $e \in E(G)$ by running the procedure Sample-Edges-Uniformly-in-G. In case the procedure fails, pick an arbitrary edge $e$ in $E(G)$.
2. Else ($d \leq \sqrt{\delta n}$), sample an edge $e \in E(G)$ by running the procedure Sample-Edges-Almost-Uniformly-in-G($\delta/2$).
3. Choose with equal probability one of the end-points $v$ of the edge $e$.
4. Choose uniformly at random one of the vertices $x$ in $X(v)$.
```

Figure 7: A procedure for selecting a vertex in G' so that all but at most a δ-fraction of the vertices are selected with probability $\Omega(1/n')$.

If $d \leq \sqrt{\delta n}$, then $\Pr[C_e]$ is determined by the procedure Sample-Edges-Almost-Uniformly-in-G. Let U_0 be as defined in Lemma 6, and consider first the case where $v(x) \notin U_0$. Then for every edge e that is incident to $v(x)$, we have that $\Pr[C_e] \geq 1/(64m) = 1/(32dn)$. By Equation (26), for each such vertex x we have that

$$\Pr[C_x] \geq \frac{1}{2}\deg(v(x)) \cdot \frac{1}{32dn} \cdot \frac{1}{\lceil \deg(v(x))/d \rceil} \geq \frac{1}{128n} \geq \frac{1}{128n'}. \tag{27}$$

as required. Next consider the case that $v(x) \in U_0$ but $\deg(v(x)) \geq 2|U_0|$. In such a case for at least half of the edges e incident to $v(x)$ we have that $\Pr[C_e] \geq 1/(32dn)$, and we can deduce that $\Pr[C_x] \geq \frac{1}{256n'}$. It remains to show that the total number of vertices x such that $v(x) \in U_0$ and $\deg(v) \leq 2|U_0|$, is at most $(\delta/2)n'$. Using the fact that $|U_0| \leq \sqrt{(\delta/2)n}/2$ (recall that the procedure Sample-Edges-Almost-Uniformly-in-G is called with its input parameter set to $\delta/2$), and for every vertex $v \in V(G)$, $|X(v)| = \lceil \deg(v)/d \rceil \leq \deg(v)$, we get:

$$\sum_{v \in U_0 \colon \deg(v) \leq 2|U_0|} |X(v)| \;\leq\; 2|U_0|^2 \;\leq\; (\delta/2)n' \;. \tag{28}$$

Thus, Item 4 of Theorem 5 follows.

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 [14], 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 [14], 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 degrees. Indeed, the two families of graphs that we define below in our lower bound construction, can be viewed as generalizing the two families presented in [14]. However, since we have to deal with any given degree d and not only with $d = 3$, and since we have to deal with both types of queries, the analysis itself does not follow as a straightforward generalization of the analysis in [14].

Theorem 8 *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 [22]. Specifically, we present two distributions over graphs (which are defined in detail below): $\mathcal{G}(n,d)$, and $\mathcal{G}(n/2, n/2, d)$, where d is the degree of the vertices in the graphs. (For simplicity we assume that n is even.) We prove that graphs created randomly according to $\mathcal{G}(n,d)$ are ϵ-far from being bipartite with high probability, while all graphs in the support of $\mathcal{G}(n/2, n/2, d)$ are bipartite. We then show that a bipartite tester that asks $o(\min(\sqrt{n}, n^2/m))$ queries, cannot distinguish with sufficiently high probability whether a graph is created according to the distribution $\mathcal{G}(n,d)$ or $\mathcal{G}(n/2, n/2, d)$. Note that by definition, in both cases $m = (nd)/2$, and so $n^2/m = 2n/d$.

The Graph Distribution $\mathcal{G}(n,d)$. A graph G in the support of the distribution $\mathcal{G}(n,d)$ is composed of n vertices, and it is a d-regular graph. The edges of G are determined according to the following random process. Consider a two-dimensional table of size $n \times d$, which we refer to as the *matching table*. Each cell in the matching table is denoted by $c_{u,i}$, where u denotes the row in which the cell is located, (and corresponds to a vertex in the graph) and i denotes the column in which the cell is located (and corresponds to a label of an edge incident to v).

Consider a perfect matching over the cells of the table, randomly chosen over all possible perfect matching. This randomly chosen matching defines the edges of the graph G. That is, if the cells $c_{u,i}$ and $c_{v,j}$ are matched then this means that there is an edge (u, v, i, j) in the graph.

The Graph Distribution $\mathcal{G}(n/2, n/2, d)$. A graph G in the support of the distribution $\mathcal{G}(n/2, n/2, d)$ is a bipartite d-regular graph composed of n vertices, where each side of the partition is of size $n/2$. The edges of G are determined according to the following random process. First we select a random partition of the graph vertices $\{1, \cdots, n\}$ into two parts (sides), each of size $n/2$. Second, consider two tables, each of size $n/2 \times d$, where each is attached to one of the partition sides. The rows in each table refer to the vertices that belong to the relevant side of the partition. Let us refer to the table attached to the first side of the partition as the *even table*, and to the table attached to the second side of the partition as the *odd table*.

A cell in these tables is denoted by $c_{b,u,i}$, where $b \in \{1, 2\}$ denotes the relevant table, u denotes the row in which the cell is located, and i denotes the column in which the cell is located.

Now, consider a perfect matching over the cells of the two tables, randomly chosen over all possible perfect matchings that are restricted to matching cells from the odd table with cells from the even table. This randomly chosen matching defines the edges of the graph G. That is, if the cell $c_{1,u,i}$ is matched to the cell $c_{2,v,j}$, this means that there is an edge (u, v, i, j) in the graph (where u belongs to one side of the bipartite graph and v to the other side).

By definition, all graphs in the support of $\mathcal{G}(n/2, n/2, d)$ are bipartite. We can show that almost all graphs in the support of $\mathcal{G}(n,d)$ are far from being bipartite.

Lemma 9 *With probability $1 - o(1)$, a graph chosen uniformly according to the distribution $\mathcal{G}(n,d)$ is ϵ-far from being bipartite for every $\epsilon \leq 1/16$ and $d \geq 64$.*

Proof: We shall show that for every fixed partition (V_0, V_1) of V, the probability, taken over the selection of a graph in $\mathcal{G}(n,d)$, that there are more than $\epsilon \cdot n \cdot d$ violating edges with respect to (V_0, V_1), is very close to one. The lemma will then follow by a union bound (over all partitions).

Let $P = (V_0, V_1)$ be an arbitrary fixed partition of V. Without loss of generality let $|V_0| \geq n/2$. Consider the following process, denoted by C_P, for choosing a random d-regular graph having n vertices (i.e., a perfect matching over a matching table of size $n \times d$). Starting from $b = 0$, choose an arbitrary cell $c_{u,i}$ such that $u \in V_b$, and match $c_{u,i}$ to a randomly chosen unmatched cell $c_{v,j}$. If the number of unmatched cells that belong to vertices in V_b is smaller than the number of unmatched cells that belong to the other side of the partition, then switch sides (i.e. let $b \leftarrow 1 - b$). Finish when all the cells are matched. Thus, the number of steps in this process is $\frac{nd}{2} - 1$. (In the last step there are two unmatched cells, which are matched together). We denote by G_P the graph chosen according to process C_P.

Claim 9.1 *For every fixed partition P, the distribution on graphs induced by the process C_P is identical to $\mathcal{G}(n,d)$*

Proof: For each step t in the process C_P, $1 \leq t \leq \frac{nd}{2} - 1$, let e_t be the edge selected in that step. Let R_0 be the set of all graphs in the support of $\mathcal{G}(n,d)$ and denote by $R_t \subset R_0$ the set of graphs in R_0 that contain the edges $e_1, \cdots, e_t$. Using this notation, at the start of process C_P, before any edge was selected (any two cells were matched), the process can potentially select any graph in R_0. After performing t steps it is restricted to selecting graphs from R_t.

The probability that a particular graph G in the support of $\mathcal{G}(n,d)$ is selected is

$$\Pr[G_P = G] \;=\; \Pr[G \in R_1] \cdot \Pr[G \in R_2 | G \in R_1] \cdot \;\cdots\; \cdot \Pr[G \in R_{\frac{nd}{2}-1} | G \in R_{\frac{nd}{2}-2}] \,.$$

Consider any particular term $\Pr[G \in R_t | G \in R_{t-1}]$ in the above expression. Conditioned on G belonging to R_{t-1} (that is, the edges $e_1, \cdots, e_{t-1}$ selected by the process C_P are all edges in G), we would like to know what is the probability that G belongs to R_t as well. Let $c_{u,i}$ be the arbitrary unmatched cell selected by C_P in step t. Let v be the i-th neighbor of u in G, where u is the j-th neighbor of v. Since $G \in R_{t-1}$, necessarily $c_{v,j}$ is an unmatched cell in the matching table. Hence, the probability that G belongs to R_t (conditioned on it belonging to R_{t-1}) equals the probability that $c_{u,i}$ is matched to $c_{v,j}$ which is $1/(dn - 2(t-1) - 1)$. Therefore,

$$\begin{aligned} \Pr[G_P = G] &= \frac{1}{dn-1} \cdot \frac{1}{dn-3} \cdot \;\cdots\; \cdot \frac{1}{3} \;=\; \frac{dn \cdot (dn-2) \cdot \;\cdots\; \cdot 2}{(dn)!} \\ &= \frac{(2(dn/2)) \cdot (2((dn/2)-1)) \cdot \;\cdots\; \cdot 2}{(dn)!} \;=\; \frac{2^{dn/2} \cdot (dn/2)!}{(dn)!} \,. \end{aligned} \tag{29}$$

It remains to show that the above expression equals $1/|R_0|$. But this is easy to verify: By definition, $|R_0|$ is the total number of possible perfect matchings between the dn cells in the matching table. Each such matching can be obtained by ordering all the cells and letting the matching be defined by successive pairs. The number of such orderings is $(dn)!$. However, different orderings may produce the same matching. In particular we need to take into account the over-counting due to orderings within the pairs (i.e., the pairs $(c_{v,i}, c_{u,j})$ and $(c_{u,j}, c_{v,i})$ are the same), and the over-counting due to orderings between the different pairs. In other words, we need to divide $(dn)!$ by $2^{dn/2} \cdot (dn/2)!$. We have thus obtained the inverse of the expression in Equation (29), as required. ⋄ (Proof of Claim 9.1.)

We have thus completed proving the claim and may view a graph G selected uniformly from $\mathcal{G}(n,d)$, as if it is created according to the process C_P, for any fixed partition P. It remains to show that with very high probability over the choice of such a graph, it has more than $\epsilon \cdot nd$ violating edges with respect to the partition P.

By definition, during the process C_P, we always try to match a cell that belongs to a side of the partition having more unmatched cells. Thus, at each step we create a violating edge with probability at least $\frac{1}{3}$ (except the last stage), independent of the past events. It follows that the expected number of violating edges in the final resulting graph G, with respect to a particular partition P, is $\mu \geq (nd/2 - 1) \cdot \frac{1}{3} > \frac{nd}{8}$. Let Y_P denote the number of violating edges with respect to P. By applying the Chernoff bound we have that for any constant $0 < \alpha < 1$,

$$\Pr[Y_P < (1-\alpha)\mu] < \exp(-(1/2)\alpha^2\mu) \ .$$

In particular, for $\alpha = 1 - 8\epsilon$ (recall that $\epsilon \leq 1/16$) we obtain:

$$\begin{aligned} \Pr[Y_P < \epsilon nd] &\leq \Pr[Y_P < 8\epsilon\mu] \\ &< \exp(-(1/2)(1-8\epsilon)^2\mu) \\ &\leq \exp(-(1-8\epsilon)^2 nd/16) \end{aligned}$$

Since the total number of partitions is 2^n, if we take a union bound over all possible partitions then we can deduce that the probability that G has less then ϵnd violating edges with respect to *any* partition P is bounded from above by $2^n \cdot \exp(-(1-8\epsilon)^2 nd/16)$. Taking $\epsilon \leq 1/16$ and $d \geq 64$ we get that a graph constructed according to a distribution $\mathcal{G}(n,d)$ is ϵ-far from being bipartite with probability $1 - o(1)$ as required. ■

Let ALG be an algorithm for testing bipartiteness using $Q = Q(n)$ queries. Namely, ALG is a (possibly probabilistic) mapping from *query-answer histories* $\langle (q_1, a_1), \cdot, (q_t, a_t) \rangle$ to q_{t+1}, for every $t < Q$, and to $\{\mathsf{accept}, \mathsf{reject}\}$ for $t = Q$. Recall that a query q_t can be of two types: a neighbor query and a vertex-pair query. A neighbor query q_t is a pair (u_t, i_t), where $u_t \in V$, and $i_t \in \{1, \cdots, \deg(u_t)\}$. Here the corresponding answer a_t is a pair (v_t, j_t) where $v_t \in V$ and $j_t \in \{1, \cdots, \deg(v_t)\}$ such that v_t is the i_t-th neighbor of u_t and u_t is the j_t-th neighbor of v_t. A vertex-pair query q_t is a pair (u_t, v_t), where $u_t, v_t \in V$, and the corresponding answer is of the form $a_t = (y_t, i_t, j_t)$. If there is an edge between u_t and v_t then $y_t = 1$, and i_t and j_t are the corresponding labels of the edge. If no such edge exists then $y_t = 0$, and i_t and j_t are set arbitrarily, say to 1. In the former case we shall say that ALG has *detected an edge* (by a vertex-pair query). In the latter case we'll say that (u_t, v_t) is a *non-edge* (and that ALG has detected a non-edge)[5]. Note, that the answers to the queries of ALG, contain additional indexing information, that does not appear in our model. We show that even by using the extra information, ALG cannot do better than the stated lower bound.

We assume that the mapping determined by the algorithm is defined only on histories that are consistent with some n-vertex graph. Any query-answer history of length t can be used to define a *knowledge graph* G_t, at time t. The vertex set of G_t contains all vertices that appear in the history (either in queries or in answers). For every neighbor query $(u_{t'}, i_{t'})$ answered by $(v_{t'}, j_{t'})$ $(t' \leq t)$, the graph G_t contains the edge $(u_{t'}, v_{t'}, i_{t'}, j_{t'})$, and similarly for every vertex-pair query $(v_{t'}, u_{t'})$ that is answered by $(1, i_{t'}, j_{t'})$. In addition, for every vertex-pair query $(v_{t'}, u_{t'})$ that is answered by $(0,1,1)$, the knowledge graph maintains the information that $(v_{t'}, u_{t'})$ is a non-edge. Thus G_t is a subgraph of the graph tested by ALG.

In what follows we describe two random processes, P^1 and P^2, which interact with an arbitrary algorithm ALG. The process P^1 answers ALG's queries while constructing a random graph from $\mathcal{G}(n,d)$, and the process P^2 answers ALG's queries while constructing a random graph from

[5] In case multiple edges are allowed then the answer is of the form (y_t, I_t, J_t) where I_t and J_t are sets of labels.

$\mathcal{G}(n/2, n/2, d)$. We assume without loss of generality that ALG does not ask queries whose answer can be derived from its knowledge graph, since such queries give it no new information. (For example ALG does not ask a vertex-pair query about a pair of vertices that are already known to be connected by an edge due to a neighbor query).

For a fixed algorithm ALG that uses Q queries, and for $b \in \{1, 2\}$, let D^b_{ALG} denote the distribution on query-answers histories (of length Q) induced by the interaction of ALG and P^b. We show that for any given ALG that uses $o(\min(\sqrt{n}, n/d))$ queries, the statistical distance between D^1_{ALG} and D^2_{ALG} is $o(1)$. Combining this with Lemma 9 (and the fact that $m = (nd)/2$), Theorem 8 follows.

Definition of P^b, $b \in 1, 2$.

- Let R^1 be the set of all graphs in the support of $\mathcal{G}(n, d)$, and let R^2 be the set of all graphs in the support of $\mathcal{G}(n/2, n/2, d)$.

  For an edge $e_k = (u, v, i, j)$, denote by $R^b_{e_k} \subset R^b$ the subset of graphs in R^b that contain e_k. We may refer to $R^b_{e_k}$ as $R^b_{(u,v,i,j)}$.

  For an edge-pair $f_k = (u, v)$, denote by $R^b_{f_k} \subset R^b$ the subset of graphs in R^b that contain an edge between the vertices u, v. Denote by $R^b_{\overline{f_k}} \subset R^b$ the subset of graphs in R^b that do not contain an edge between the vertices u, v.
- The process P^b answers queries as follows. Initialize $R^b_0 = R^b$ (in general, as is explained in more detail below, for any $t \geq 0$, the set R^b_t consists of all graphs in R^b that are consistent with the first t queries and answers).
  1. If the t-th query is a vertex-pair query $q_t = (u_t, v_t)$ then the answer $a_t = (y_t, i_t, j_t)$ is selected as follows. Let $y_t = 1$ with probability $|R^b_{(u_t,v_t)} \cap R^b_{t-1}|/|R^b_{t-1}|$ and let $y_t = 0$ with probability $1 - |R^b_{(u_t,v_t)} \cap R^b_{t-1}|/|R^b_{t-1}|$. In the former case, for any pair $i, j \in \{1, \cdots, d\}$, the probability that $i_t = i$ and $j_t = j$ is $|R^b_{(u_t,v_t,i,j)} \cap R^b_{t-1}|/|R^b_{t-1}|$ and we set $R^b_t = R^b_{(u_t,v_t,i,j)} \cap R^b_{t-1}$. In the latter case, the values of i_t and j_t can be selected arbitrarily (say to 1), and we set $R^b_t = R^b_{\overline{(v_t,u_t)}} \cap R^b_{t-1}$.
  2. If the t-th query is a neighbor query $q_t = (u_t, i_t)$ then for each $v \in V$, $j \in \{1, \cdots, d\}$, answer $a_t = (v, j)$ with probability $|R^b_{(u_t,v,i_t,j)} \cap R^b_{t-1}|/|R^b_{t-1}|$, and set $R^b_t = R^b_{(u_t,v_t,i,j)} \cap R^b_{t-1}$
- After all queries are answered (that is, after Q queries), uniformly choose a random graph G from R^b_Q. This is the graph constructed by $P^b, b \in 1, 2$.

Lemma 10 *For every algorithm* ALG*, the process P^1, when interacting with* ALG*, uniformly generates graphs in $\mathcal{G}(n, d)$, and the process P^2, when interacting with* ALG*, uniformly generates graphs in $\mathcal{G}(n/2, n/2, d)$.*

Proof: Consider a specific graph $G \in R^b_0$. Recall that $R^1_0 = \mathcal{G}(n, d)$, and $R^2_0 = \mathcal{G}(n/2, n/2, d)$.

The probability that G is generated by $P^b, b \in 1, 2$ is:

$$\Pr[G \in R^b_1] \cdot \Pr[G \in R^b_2 | G \in R^b_1] \cdot \ \cdots \ \cdot \Pr[G \in R^b_Q | G \in R^b_{Q-1}] \cdot \frac{1}{|R^b_Q|}$$

$$= \frac{|R^b_1|}{|R^b_0|} \cdot \frac{|R^b_2|}{|R^b_1|} \cdot \ \cdots \ \cdot \frac{|R^b_Q|}{|R^b_{Q-1}|} \cdot \frac{1}{|R^b_Q|} \quad = \quad \frac{1}{|R^b_0|}$$

and the lemma follows. ■

We next want to upper bound the probability that ALG, after performing $o(n/d)$ queries, gets a positive answer (that is, of the form $(1,*,*)$) when it performs a new vertex-pair query and interacts with either one of the two processes. We first introduce some more notation. For $b \in \{1,2\}$ and any set of edges $B = \{e_1, \cdots, e_\ell\}$, let $R^b_B \stackrel{\rm def}{=} R^b_{e_1} \cap \ldots \cap R^b_{e_\ell}$ (where $R^b_{e_j}$ is as defined above in the description of the processes $\{P^b\}_{b\in\{1,2\}}$). Similarly, for any set of edge-pairs $D = \{f_1, \cdots, f_h\}$, let $R^b_{\overline{D}} \stackrel{\rm def}{=} R_{\overline{f_1}} \cap \ldots \cap R_{\overline{f_h}}$. Finally, let $R^b_{B,\overline{D}} = R^b_B \cap R^b_{\overline{D}}$. That is, $R^b_{B,\overline{D}}$ is the subset of all graphs in R^b that are consistent with a particular set of edges and a particular set of non-edges. In particular, if B and D correspond to the set of edges and non-edges, respectively, that were observed by ALG in the course of its first t queries, then $R^b_{B,\overline{D}} = R^b_t$.

Lemma 11 *Let $B = \{e_1, \cdots, e_\ell\}$ be any set of edges, and let $D = \{f_1, \cdots, f_h\}$ be any set of edge-pairs such that no edge-pair in D appears in B and such that $|B|, |D| = o(n/d)$. Then for each $b \in \{1,2\}$, and for every $(u,v,i,j) \notin B$ we have*

$$\frac{|R^b_{(u,v,i,j)} \cap R^b_{B,\overline{D}}|}{|R^b_{B,\overline{D}}|} \le \frac{4}{dn} .$$

Proof: Consider first the process P^1 and recall that R^1 is the set of graphs in the support of $\mathcal{G}(n,d)$. For sake of brevity we remove the superscript 1 for the remaining discussion (until we turn to the process P^2). For each $e_k \in B$, let $e_k = (u_k, v_k, i_k, j_k)$. For any $w \le \ell = |B|$, the total number of possible perfect matchings in the matching table that match c_{u_k,i_k} to c_{v_k,j_k} for $1 \le k \le w$ (that is, the number of matchings that are consistent with the edges $e_1, \cdots, e_w$), is

$$\frac{(dn-2w)!}{2^{(\frac{dn}{2}-w)}(\frac{dn}{2}-w)!} .$$

The argument required for establishing this expression is a slight extension of the one used in the proof of Lemma 9 (when determining the size of R_0). It follows that for any (u,v,i,j) such that neither $c_{u,i}$ nor $c_{v,j}$ is yet matched,

$$\frac{|R_{(u,v,i,j)} \cap R_B|}{|R_B|} = \frac{\frac{(dn-2(k+1))!}{2^{(\frac{dn}{2}-(k+1))}(\frac{dn}{2}-(k+1))!}}{\frac{(dn-2k)!}{2^{(\frac{dn}{2}-k)}(\frac{dn}{2}-k)!}} = \frac{1}{dn-2k-1} .$$

For an edge-pair (u,v) we can thus deduce that

$$\begin{aligned} \frac{|R_{(u,v)} \cap R_B|}{|R_B|} &= \frac{\left|\left(\bigcup_{1\le i,j\le d} R_{(u,v,i,j)}\right) \cap R_B\right|}{|R_B|} \\ &\le \frac{\sum_{1\le i,j\le d} |R_{(u,v,i,j)} \cap R_B|}{|R_B|} \\ &\le \frac{d^2}{dn-2k-1} \;=\; O\left(\frac{d}{n}\right) , \end{aligned} \tag{30}$$

Hence we obtain that

$$\begin{aligned}
\frac{|R_{B,\overline{D}}|}{|R_B|} &= \frac{\left|R_B \cap \left(\bigcap_{1\le w\le h} R_{\overline{f_w}}\right)\right|}{|R_B|} \\
&= \frac{\left|R_B \setminus \left(\bigcup_{1\le w\le h} R_{f_w}\right)\right|}{|R_B|} \\
&\ge 1 - \sum_{1\le w\le h} \frac{|R_{f_w} \cap R_B|}{|R_B|} \\
&= 1 - o\left(\frac{n}{d}\right)\cdot O\left(\frac{d}{n}\right) \quad = \quad 1 - o(1)\ , \qquad (31)
\end{aligned}$$

and so $\frac{|R_B|}{|R_{B,\overline{D}}|} = 1 + o(1)$.

Putting the above together we get that for every $(u, v, i, j) \notin B$:

$$\begin{aligned}
\frac{|R_{(u,v,i,j)} \cap R_{B,\overline{D}}|}{|R_{B,\overline{D}}|} &= \frac{|R_{(u,v,i,j)} \cap R_B|}{|R_B|} \cdot \frac{|R_{(u,v,i,j)} \cap R_{B,\overline{D}}|}{|R_{(u,v,i,j)} \cap R_B|} \cdot \frac{|R_B|}{|R_{B,\overline{D}}|} \\
&\le \frac{1}{dn - 2k - 1} \cdot 1 \cdot (1 + o(1)) \quad \le \quad \frac{4}{dn}\ . \qquad (32)
\end{aligned}$$

We now turn to $P^b = P^2$ and recall that R^2 is the support of $\mathcal{G}(n/2, n/2, d)$. Also recall that in order to construct a random graph from $\mathcal{G}(n/2, n/2, d)$ we do the following: we partition the n vertices into two equal parts at random and we use two matching tables (one named the odd table and one the even table). We then randomly select a perfect matching between cells in the odd table and cells in the even table. Here too we remove the superscript $b = 2$ for sake of brevity.

Let $w = o(n/d)$, and let M_w be the number of possibilities to partition n vertices into two parts of equal size so that the end-points of the edges $e_1, \cdots, e_w$ belong to different sides of the partition. The total number of graphs in $R = R^2$ that contain the edges $e_1, .., e_w \in B$ is $M_w \cdot (dn/2 - w)!$. To verify this, consider any fixed equal-partition amongst the M_w possible ones (such that the end-points of the edges $e_1, .., e_w$ belong to different sides of the partition). For each such partition, consider the $dn/2 - w$ cells in the odd/even tables which are not part of the edges $e_1, .., e_w$. A matching between them can be uniquely defined by an arbitrary order on the un-matched cells in the odd table, and a permutation on the un-matched cells in the even table.

Note that by the definition of M_w we have that $\frac{M_{w+1}}{M_w} \le 1$. Therefore,

$$\frac{|R_{(u,v,i,j)} \cap R_B|}{|R_B|} = \frac{M_{k+1} \cdot (dn/2 - (k+1))!}{M_k \cdot (dn/2 - k)!} \le \frac{1}{dn/2 - k}\ . \qquad (33)$$

As in the case of $P^b = P^1$ it follows that

$$\frac{|R_{(u,v)} \cap R_B|}{|R_B|} \le \frac{d^2}{dn/2 - k} = O\left(\frac{d}{n}\right)\ , \qquad (34)$$

from which we can obtain that $\frac{|R_B|}{|R_{B,\overline{D}}|} = 1 + o(1)$.

By combining the above we get that

$$\frac{|R_{(u,v,i,j)} \cap R_{B,\overline{D}}|}{|R_{B,\overline{D}}|} = \frac{|R_{(u,v,i,j)} \cap R_B|}{|R_B|} \cdot \frac{|R_{(u,v,i,j)} \cap R_{B,\overline{D}}|}{|R_{(u,v,i,j)} \cap R_B|} \cdot \frac{|R_B|}{|R_{B,\overline{D}}|}$$
$$\leq \frac{1}{dn/2 - k} \cdot 1 \cdot (1 + o(1)) \leq \frac{4}{dn} . \quad (35)$$

■

The next two lemmas follow as corollaries of Lemma 11.

Lemma 12 *Let* ALG *be an algorithm that interacts with a process* P^b*,* $b \in 1, 2$ *and performs* $o(n/d)$ *queries. The probability that* ALG *detects an edge by a vertex-pair query at any step, is* $o(1)$*.*

Proof: Consider the interaction of ALG with P^b. For each $1 \leq t \leq Q$, $Q = o(n/d)$, let B_{t-1} be the set of edges in the induced knowledge graph G_{t-1}, and let D_{t-1} be the set of edge-pairs that are non-edges in the graph. By definition of the process P^b, we know that $R^b_{t-1} = R^b_{B_{t-1},\overline{D_{t-1}}}$. Suppose that the t-th query of ALG is a vertex-pair query (u_t, v_t) (where by our assumption, the pair (u_t, v_t) is neither an edge(-pair) nor a non-edge in G_{t-1}, but other than that it may be any vertex-pair). Then by Lemma 11 (and using the fact that $R^b_{t-1} = R^b_{B_{t-1},\overline{D_{t-1}}}$), for every $i, j \in \{1, \cdots, d\}$, $|R^b_{(u_t,v_t,i,j)} \cap R^b_{t-1}|/|R^b_{t-1}| \leq 4/(dn)$. Therefore,

$$\frac{\left|R^b_{(u_t,v_t)} \cap R^b_{t-1}\right|}{|R^b_{t-1}|} = \frac{\left|\left(\bigcup_{i,j\in\{1,\cdots,d\}} R^b_{u_t,v_t,i,j}\right) \cap R^b_{t-1}\right|}{|R^b_{t-1}|} \leq \frac{\sum_{i,j=1}^d \left|\left(R^b_{u_t,v_t,i,j} \cap R^b_{t-1}\right)\right|}{|R^b_{t-1}|} \leq \frac{4d}{n} . \quad (36)$$

It follows by definition of P^b that the answer $a_t = (y_t, i_t, j_t)$ satisfies $y_t = 1$ with probability at most $4d/n$. Note that the above is true for every step t, for every pair of sets B_{t-1} and D_{t-1} and for every vertex pair (u_t, v_t) (that does not already appear in B_{t-1} or in D_{t-1}). Hence, the probability that $y_t = 1$ in any one of the $Q = o(n/d)$ queries performed by the algorithm is $o(1)$ as required. ■

Next we turn to neighbor queries and prove a similar claim.

Lemma 13 *Let* ALG *be an algorithm that interacts with process* P^b*,* $b \in 1, 2$ *and performs* $o(\sqrt{n})$ *queries. Then the probability that* ALG*, while performing a neighbor query* $q_t = (u_t, i_t)$ *at any step* t *of the interaction, receives as an answer* $a_t = (v_t, j_t)$ *where* v_t *is a vertex that belongs to the knowledge graph* G_{t-1}*, is* $o(1)$*.*

Proof: Let $t = o(\sqrt{n})$ and let $q_t = (u_t, i_t)$ be a neighbor query. Denote by p_t the probability that following this query ALG reaches a vertex that was previously visited (i.e., p_t is the probability that in the answer $a_t = (v_t, j_t)$ we get a vertex v_t that already belongs to the knowledge graph G_{t-1}). Then by definition of P^b (for both $b = 1$ and $b = 2$),

$$p_t = \frac{\left|\bigcup_{v\in G_{t-1},\, 1\leq j\leq d} \left(R^b_{(u_t,v,i_t,j)} \cap R^b_{t-1}\right)\right|}{|R^b_{t-1}|} . \quad (37)$$

By Lemma 11 (using the same argument as the one applied in the proof of Lemma 12), we can deduce that for any specific v and j, $v \in G_{t-1}$ and $1 \leq j \leq d$,

$$\left|R^b_{(u_t,v,i_t,j)} \cap R^b_{t-1}\right| \leq \frac{4}{dn} \cdot \left|R^b_{t-1}\right| . \quad (38)$$

Therefore,

$$\left|\bigcup_{v\in G_{t-1},\,1\leq j\leq d}\left(R^b_{(u_t,v,i_t,j)}\cap R^b_{t-1}\right)\right| \leq \sum_{v\in G_{t-1},\,1\leq j\leq d,}\left|R^b_{(u_t,v,i_t,j)}\cap R^b_{t-1}\right|$$
$$\leq d\cdot 2t\cdot\frac{4}{dn}\cdot\left|R^b_{t-1}\right| = \frac{8t}{n}\cdot\left|R^b_{t-1}\right| . \qquad (39)$$

Since $t = o(\sqrt{n})$ we get that $p_t = o(1/\sqrt{n})$. Since this holds for every $1 \leq t \leq Q = o(\sqrt{n})$, the probability that for some t (in which a neighbor query is performed), the algorithm reaches a vertex in the knowledge graph is $o(1)$. ■

In particular we can obtain the following corollary:

Corollary 9 *Let* ALG *be an algorithm that interacts with process* P^b *and performs* $Q = o(\min(\sqrt{n}, n/d))$ *queries. Then* ALG *does not detect a cycle with probability* $1 - o(1)$.

Recall that D^b_{ALG}, $b \in \{1,2\}$, denotes the distribution on query-answer histories (of length Q), induced by the interaction of ALG and P^b. We are now ready to show that the two distributions are indistinguishable if Q is sufficiently small.

Lemma 14 *For every algorithm* ALG *that asks* $Q = o(\min(\sqrt{n}, n/d))$ *queries, the statistical distance between* D^1_{ALG} *and* D^2_{ALG} *is at most* $o(1)$. *Furthermore, with probability at least* $1 - o(1)$ *the knowledge graph at time of termination of* ALG *contains no cycles.*

Proof: Recall that G_Q, the knowledge graph after Q queries, contains all the vertices that appeared in one of the queries or answers, all the edges found by queries, and all the non-edges found by queries.

Recall that we assume without loss of generality that ALG does not ask queries whose answer can be derived from the knowledge graph, since those give it no new information. Under this assumption the following holds:

1. According to Lemma 12, both in D^1_{ALG} and in D^2_{ALG} the total weight of query answer histories in which an edge is detected by a vertex-pair query is $o(1)$. In all other histories (whose weight is $1-o(1)$ under both distributions), all answers to vertex-pair queries are of the form $(0, 1, 1)$.

2. According to Lemma 13, both in D^1_{ALG} and in D^2_{ALG} the total weight of query-answer histories in which any neighbor query q_t is answered with a vertex that belongs to the knowledge graph G_{t-1}, is $o(1)$. In particular, this means that with probability at least $1 - o(1)$, the knowledge graph at time of termination of ALG contains no cycles.

Let Π_1 be the set of all possible query-answer histories in which a vertex-pair query is answered by $(1, *, *)$, let Π_2 be the set of histories in which a neighbor query is answered with a vertex that appears in the current knowledge graph, and let Π_3 be the set of all remaining histories. Then the statistical distance between D^1_{ALG} and D^2_{ALG} is upper bounded by

$$\sum_{\pi\in\Pi_1}\left|D^1_{\mathrm{ALG}}(\pi) - D^2_{\mathrm{ALG}}(\pi)\right| + \sum_{\pi\in\Pi_2}\left|D^1_{\mathrm{ALG}}(\pi) - D^2_{\mathrm{ALG}}(\pi)\right| + \sum_{\pi\in\Pi_2}\left|D^1_{\mathrm{ALG}}(\pi) - D^2_{\mathrm{ALG}}(\pi)\right| . \qquad (40)$$

Observe that

$$\sum_{\pi\in\Pi_1}\left|D^1_{\mathrm{ALG}}(\pi) - D^2_{\mathrm{ALG}}(\pi)\right| \leq \sum_{\pi\in\Pi_1} D^1_{\mathrm{ALG}}(\pi) + \sum_{\pi\in\Pi_1} D^2_{\mathrm{ALG}}(\pi) = D^1_{\mathrm{ALG}}(\Pi_1) + D^2_{\mathrm{ALG}}(\Pi_2)$$

(and similarly for the sum over Π_2). Hence the first two terms in Equation (40) contribute $o(1)$.

As for the third term, first note that for every fixed history $\langle (q_1, a_1), \cdots, (q_{t-1}, a_{t-1}) \rangle$, the distribution on the next query q_t that ALG performs is the same no matter which process it interacts with. By definition of Π_3, if q_t is a vertex-pair query, then $a_t = (0, 1, 1)$ for both processes. Finally, if $q_t = (u_t, i_t)$ then again by definition of Π_3 the answer $a_t = (v_t, j_t)$ is such that $v_t \notin G_{t-1}$. For both processes, conditioned on $v_t \notin G_{t-1}$, the vertex v_t is uniformly distributed among all vertices not in G_{t-1}, and j_t is uniformly distributed over $\{1, \cdots, d\}$. Therefore, the third term in Equation (40) is bounded by $|D^1_{\text{ALG}}(\Pi_3) - D^2_{\text{ALG}}(\Pi_3)|$ which is $o(1)$ as well, and the lemma follows. ■

Theorem 8 follows by combining Lemma 14 with Lemma 9.

5.1 Self Loops and Multiple Edges

The lower bound proof as stated above is valid for graphs that may contain multiple edges and self loops. In both the distribution $\mathcal{G}(n, d)$ and the distribution $\mathcal{G}(n/2, n/2, d)$ the probability of a multiple edge between vertices u and v is $O(\frac{d^2}{n^2})$, and the probability of a self loop edge incident to vertex v is $O(\frac{d}{n})$. Thus, graphs created according to these distributions contain self loops and multiple edges with probability close to 1. However, with probability $1 - o(1)$ there are at most $O(d^2)$ loops and multiple edges in the graphs created according to these distributions.

We have shown in Lemma 9 that graphs created according to the distribution $\mathcal{G}(n, d)$ are ϵ-far from being bipartite with probability $1 - o(1)$. Hence we can deduce that by removing self-loops and multiple edges from a graph G constructed according to a distribution $\mathcal{G}(n, d)$, the resulting graph is ϵ-far from being bipartite with probability $1 - o(1)$.

In addition, an algorithm ALG that interacts with process P^b, detects a multiple edge with probability $o(1)$, due to the following reason: ALG does not detect any edges by sampling steps with probability $1 - o(1)$. The probability to detect a multiple edge/self loop in a neighbor query is at most the probability that such a query is answered by a vertex in the knowledge graph. This probability was shown (in Lemma 13) to be $o(1)$.

Thus, P^b in the end of the interaction with ALG, can delete all the loops and multiple edges from the resulting graph, so that the resulting graph contain no loops and multiple edges. In the case of P^2 the graph is bipartite as before, and in the case of P^1 the graph is (still) ϵ-far from being bipartite.

As a conclusion we get that our lower bound is valid also for graphs with no multiple edges and loops.

5.2 The Necessity of Both Neighbor Queries and Vertex-Pair Queries

As noted in the introduction, our lower bound proof implies the necessity of both neighbor queries and vertex-pair queries in obtaining an upper bound whose dependence on n and m is $\tilde{O}(\min(\sqrt{n}, n^2/m))$. Specifically, if only neighbor queries are allowed then Lemma 13 implies a lower bound of $\Omega(\sqrt{n})$ for every m, which is higher than $\tilde{O}(n^2/m))$ when $m = \omega(n^{1.5})$. On the other hand, every bipartite tester that uses only vertex-pair queries, has to make $\Omega(n^2/m)$ since otherwise it won't see any edge. This lower bound is above the upper bound of $\tilde{O}(\sqrt{n})$ when $m = o(n^{1.5})$.

5.3 A Lower bound for Testing k-Colorability

It is possible to generalize the lower bound stated for bipartite-testers to a lower bound for k-colorability. By using similar arguments we can get that a graph chosen uniformly from the distribution $\mathcal{G}(n, d)$ is ϵ-far from being k-colorable. In addition, by defining a distribution $\mathcal{G}(n/k, n/k, \cdots, n/k, d)$ in an analogues way to the definition of $\mathcal{G}(n/2, n/2, d)$, and by an analogues definition of the processes $P^b, b \in 1, 2$, we can get a lower bound for k-colorability which is: $\Omega(\min(\sqrt{n}, n^2/m))$.

Acknowledgments

We would like to thank the anonymous reviewers of this paper for very helpful comments.

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. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 700–709, 2003.

[5] N. Alon and J. H. Spencer. *The Probabilistic Method.* John Wiley & Sons, Inc., 1992.

[6] M. Bender and D. Ron. Testing properties of directed graphs: Acyclicity and connectivity. *Random Structures and Algorithms*, pages 184–205, 2002.

[7] 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.

[8] A. Bogdanov and L. Trevisan. Lower bounds for testing bipartiteness in dense graphs. Technical Report TR02-064, Electronic Colloquium on Computational Complexity (ECCC), 2002. Available from `http://www.eccc.uni-trier.de/eccc/`.

[9] B. Bollobas, B. Erdos, M. Simonovitz, and E. Szemeredi. Extremal graphs without large forbidden subgraphs. *Annals of Discrete Mathematics*, 3:29–41, 1978.

[10] A. Czumaj and C. Sohler. Abstract combinatorial programs and efficient property testers. In *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science*, pages 83–92, 2002.

[11] U. Feige. Sampling vertex degrees and estimating network load parameters. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing*, page to appear, 2004.

[12] 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.

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

[14] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[15] O. Goldreich and D. Ron. On estimating the average degree of a graph. Technical Report TR04-013, Electronic Colloquium on Computational Complexity (ECCC), 2004. Available from `http://www.eccc.uni-trier.de/eccc/`.

[16] O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Structures and Algorithms*, 23(1):23–57, 2003.

[17] 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.

[18] Gregory A. Margulis. Explicit constructions of expanders. *Problemy Peredachi Informatsii*, 9(4):71–80, 1973.

[19] M. Mihail. Conductance and convergence of Markov chains - A combinatorial treatment of expanders. In *Proceedings 30th Annual Conference on Foundations of Computer Science*, pages 526–531, 1989.

[20] M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, 20(2):165–183, 2002.

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

[22] 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.

A A formal definition of $\mathrm{M}_{\ell_1}^{\ell_2}(H)$

For every vertex v in H we have a state v in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$. For simplicity, we shall continue referring to these states as vertices. Let the *border* of H, denoted $B(H)$, be the set of vertices in H that have at least one neighbor in G that is not in H. Then, for every vertex $v \in B(H)$, we have a set $a_{v,1}, \cdots, a_{v,\ell_1}$ of *auxiliary* states. Let $p^H_{v,u}(t)$ denote the probability of a walk of length t that starts at v and ends at u without passing through any other vertex in H. Namely, it is the sum over all such walks w, of the product, taken over all steps in w, of the transition probabilities of these steps. In particular, $p^H_{v,v}(1) \geq \frac{1}{2}$ (where equality holds in case v has degree d), and for every $u \in \Gamma(v)$, $p^H_{v,u}(1) = \frac{1}{2d}$ (here we assume that we can choose a random neighbor of a vertex with in time which is $O(1)$). The transition probabilities, $q_{x,y}$, in $\mathrm{M}_{\ell_1}^{\ell_2}(H)$ are defined as follows:

- For every v and u in H, $q_{v,u} = \sum_{t=1}^{\ell_2-1} p^H_{v,u}(t)$.

  Thus, $q_{v,u}$ is a sum of $p^H_{v,u}(1)$ and $\sum_{t=2}^{\ell_2-1} p^H_{v,u}(t)$. The first term implies that for every v in H, $q_{v,v} \geq \frac{1}{2}$ and for every pair of neighbors v and u, $q_{v,u} \geq \frac{1}{2d}$. The second term, which we refer to as the *excess* probability is due to walks of length less than ℓ_2 (from v to u) passing through vertices outside of H, and can be viewed as *contraction* of these walks.

Hence, for every pair of vertices v and u, $q_{v,u} = q_{u,v}$.

- For every $v \in B(H)$, $q_{v,(a_{v,1})} = \sum_{u\in H} \sum_{t\geq \ell_2} p^H_{v,u}(t)$; for every ℓ, $1 \leq \ell < \ell_1$, $q_{(a_{v,\ell}),(a_{v,\ell+1})} = 1$; and for every $u \in H$, $q_{(a_{v,\ell_1}),u} = \frac{1}{q_{v,(a_{v,1})}} \cdot \sum_{t\geq \ell_2} p^H_{v,u}(t)$. (The parentheses added in the notation above (e.g., $q_{(a_{v,\ell}),(a_{v,\ell+1})}$) are only for sake of readability.)

  In other words, $q_{v,(a_{v,1})}$ is the probability that a random walk in G that starts from v takes at least ℓ_2 steps outside of H before returning to H, and $q_{(a_{v,\ell_1}),u}$ is the conditional probability of reaching u in such a walk. Thus, the auxiliary states form auxiliary paths in $\mathrm{M}^{\ell_2}_{\ell_1}(H)$, where these paths correspond to walks of length at least ℓ_2 outside of H.

We shall restrict our attention to walks of length at most ℓ_1 in $\mathrm{M}^{\ell_2}_{\ell_1}(H)$, and hence any walk that starts at a vertex of H and enters an auxiliary path never returns to vertices of H.

For any two states y, z in $\mathrm{M}^{\ell_2}_{\ell_1}(H)$ let $q_{y,z}(t)$ be the probability that a walk of length t starting from y ends at z. In particular $q_{y,z} \equiv q_{y,z}(1)$, and for any two vertices u and v and any integer t, we have $q_{u,v}(t) = q_{v,u}(t)$. We further let the parity of the lengths of paths corresponding to walks in G be carried on to $\mathrm{M}^{\ell_2}_{\ell_1}(H)$. That is, each transition between vertices v and u that corresponds to walks outside of H consists of two transitions – one due to even-length paths corresponding to walks from v to u outside of H, and one to odd-length paths. For any two vertices in H we let $q^b_{v,u}(t)$ denote the probability in $\mathrm{M}^{\ell_2}_{\ell_1}(H)$ of a walk of length t starting from v, ending at u, and corresponding to a path whose length has parity b.

In all that follows we assume that G is connected. Our analysis can easily be modified to deal with the case in which G is not connected, simply by treating separately each of its connected components. Under the assumption that G is connected, for every v and u in H, there exists a t such that $q_{u,v}(t) > 0$, and hence $\mathrm{M}^{\ell_2}_{\ell_1}(H)$ is irreducible. Furthermore, because for each $v \in H$ $q_{v,v} \geq \frac{1}{2}$, $\mathrm{M}^{\ell_2}_{\ell_1}(H)$ is also aperiodic. Thus it has a unique stationary distribution.

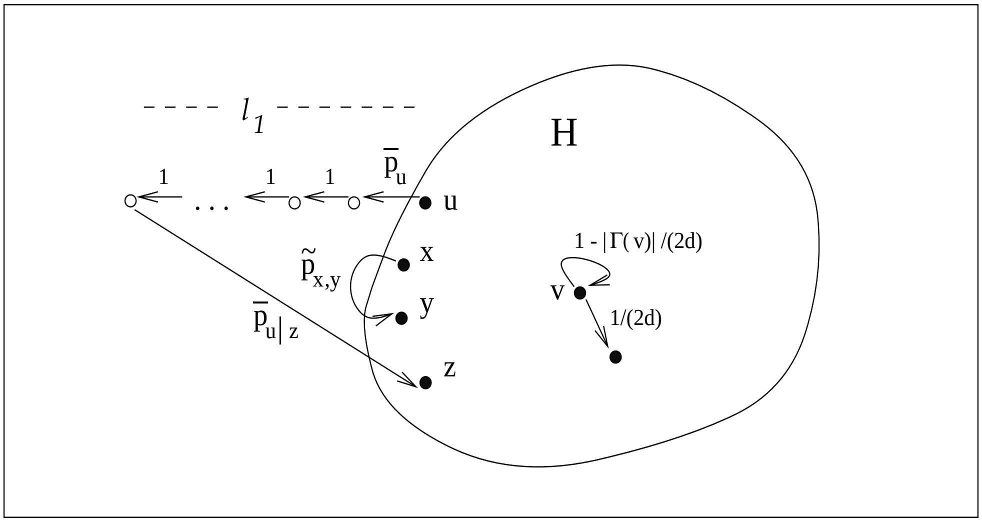

Figure 8: The structure of $\mathrm{M}_{\ell_1}^{\ell_2}(H)$. The states corresponding to vertices of H are depicted as black dots, and the auxiliary states as white ones. Here $\tilde{p}_{x,y}$ denotes the transition probability between any two vertices $x, y \in B(H)$, which equals $\sum_{t=1}^{\ell_2-1} p_{x,y}^{H}(1)$, $\bar{p}_u$ denotes the probability of entering an auxiliary path starting from $u \in B(H)$, which equals $\sum_{z\in H}\sum_{t\geq\ell_2} p_{u,z}^{H}(t)$, and $\bar{p}_{u|z}$ denotes the probability of returning from the last state on this auxiliary path to $z \in B(H)$, which equals $\frac{1}{\bar{p}_u} \cdot \sum_{t\geq\ell_2} p_{u,z}^{H}(t)$.