Property Testing and its connection to Learning and Approximation (preliminary version)

Oded Goldreich[1] Shafi Goldwasser[2] Dana Ron[3]

October 3, 1996

[1]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. E-mail: oded@wisdom.weizmann.ac.il. On sabbatical leave at LCS, MIT.

[2]Laboratory for Computer Science, MIT, 545 Technology Sq., Cambridge, MA 02139. E-mail: shafi@theory.lcs.mit.edu.

[3]Laboratory for Computer Science, MIT, 545 Technology Sq., Cambridge, MA 02139. E-mail: danar@theory.lcs.mit.edu. Supported by an NSF postdoctoral fellowship.

Abstract

In this paper, we consider the question of determining whether a function f has property P or is ϵ-far from any function with property P. A *property testing* algorithm is given a sample of the value of f on instances drawn according to some distribution. In some cases, it is also allowed to query f on instances of its choice. We study this question for different properties and establish some connections to problems in learning theory and approximation.

In particular we focus our attention on testing graph properties. Given access to a graph G in the form of being able to query whether an edge exists or not between a pair of vertices, we devise algorithms to test whether the underlying graph has properties such as being bipartite, k-colorable, or having a ρ-clique (clique of density ρ w.r.t the vertex set). Our graph property testing algorithms are probabilistic and make assertions which are correct with high probability, utilizing only a constant number of queries into the graph. Moreover, the property testing algorithms can be used to efficiently (i.e., in time linear in the number of vertices) construct partitions of the graph which correspond to the property being tested, if it holds for the input graph.

Contents

Chapter 1

Introduction

We are interested in the following general question of Property Testing:

> Let P be a fixed property of functions, and f be an unknown function. Our goal is to determine (possibly probabilistically) if f has property P or if it is far from any function which has property P, where distance between functions is measured with respect to some distribution D on the domain of f. Towards this end, we are given examples of the form $(x, f(x))$, where x is distributed according to D. We may also be allowed to query f on instances of our choice.

The problem of testing properties emerges naturally in the context of program checking and probabilistically checkable proofs as applied to multi-linear functions or low-degree polynomials [BLR93, Lip89, BFL91, BFLS91, FGL$^+$91, GLR$^+$91, RS96, AS92, ALM$^+$92, BGLR93, BS94, BCH$^+$95, BGS95]. Property testing *per se* was considered in [RS96, Rub94]. Our definition of property testing is inspired by the PAC learning model [Val84]. It allows the consideration of arbitrary distributions rather than uniform ones, and of testers which utilize randomly chosen instances only (rather than being able to query instances of their own choice).

We believe that property testing is a natural notion whose relevance to applications goes beyond program checking, and whose scope goes beyond the realm of testing algebraic properties. Firstly, in some cases one may be merely interested in whether a given function, modeling an environment, (resp. a given program) possess a certain property rather than be interested in learning the function (resp. checking that the program computes a specific function correctly). In such cases, learning the function (resp., checking the program) as means of ensuring that it satisfies the property may be an over-kill. Secondly, learning algorithms work under the postulation that the function (representing the environment) belongs to a particular class. It may be more efficient to test this postulation first before trying to learn the function (and possibly failing when the postulation is wrong). Similarly, in the context of program checking, one may choose to test that the program satisfies certain properties before checking that it computes a specified function. This paradigm has been followed both in the theory of program checking [BLR93, RS96] and in practice where often programmers first test their programs by verifying that the programs satisfy properties that are known to be satisfied by the function they compute. Thirdly, we show how to apply property testing to the domain of graphs by considering several classical graph properties. This, in turn, offers a new perspective on approximation problems as discussed below.

THE RELEVANT PARAMETERS. Let $\mathcal{F}$ be the class of functions which satisfy property P. Then, testing property P corresponds to testing membership in the class $\mathcal{F}$. The two parameters relevant to property testing are the permitted distance, ϵ, and the desired confidence, δ. We require the tester to accept each function in $\mathcal{F}$ and reject every function which is further than ϵ away from any function in $\mathcal{F}$. We allow the tester to be probabilistic and make incorrect positive and negative assertions with probability at most δ. The complexity measures we focus on are the *sample complexity* (the number of examples of the function's values that the tester requires), the *query complexity* (the number of function queries made – if at all), and the *running time* of the tester.

1.1 Property Testing and Learning Theory

As noted above, our formulation of testing mimics the standard frameworks of learning theory. In both cases one is given access to an unknown *target* function (either in the form of random instances accompanied by the function values or in the form of oracle access to the function). A semantic difference is that, for sake of uniformity, even in case the functions are Boolean, we refer to them as functions rather than concepts. However, there are two important differences between property testing and learning. Firstly, the goal of a learning algorithm is to *find* a good approximation to the target function $f \in \mathcal{F}$, whereas a testing algorithm should only *determine* whether the target function is in $\mathcal{F}$ or is far away from it. This makes the task of the testing seem easier than that of learning. On the other hand, a learning algorithm should perform well only when the target function belongs to $\mathcal{F}$ whereas a testing algorithm must perform well also on functions far away from $\mathcal{F}$. Furthermore, (non-proper) learning algorithms may output an approximation $\tilde{f}$ of the target $f \in \mathcal{F}$ so that $\tilde{f} \notin \mathcal{F}$.

We show that the relation between learning and testing is non-trivial. On one hand, proper (representation dependent) learning implies testing. On the other hand, there are function classes for which testing is harder than (non-proper) learning, provided $\mathcal{NP} \not\subseteq \mathcal{BPP}$. Nonetheless, there are also function classes for which testing is much easier than learning. Further details are given in Subsection 2.2. In addition, the graph properties discussed below provide a case where testing (with queries) is much easier than learning (also with queries).

1.2 Testing Graph Properties

In the main technical part of this paper, we focus our attention on testing graph properties. We view graphs as Boolean functions on pairs of vertices, the value of the function representing the existence of an edge. We mainly consider testing algorithms which use queries and work under the uniform distribution. That is, a testing algorithm for graph property P makes queries of the form "is there an edge between vertices u and v" in an unknown graph G. It then decide whether G has property P or is "ϵ-away" from any graph with property P, and is allowed to err with probability 1/3. Distance between two N-vertex graphs is defined as the fraction of vertex-pairs which are adjacent in one graph but not in the other.

We present algorithms of poly$(1/\epsilon)$ query-complexity and running-time[1] at most $\exp(\tilde{O}(1/\epsilon^3))$ for testing the following graph properties:

k-Colorability for any fixed $k \geq 2$. (Here the query-complexity is poly(k/ϵ), and for $k = 2$ the running-time is $\tilde{O}(1/\epsilon^3)$.)

ρ-Clique for any $\rho > 0$. That is, does the N-vertex graph has a clique of size ρN.

[1] Here and throughout the paper, we consider a RAM model in which trivial manipulation of vertices (e.g., reading/writing a vertex name and ordering vertices) can be done in constant time.

ρ-CUT for any $\rho > 0$. That is, does the N-vertex graph has a cut of size at least ρN^2. A generalization to k-way cuts works within query-complexity poly$((\log k)/\epsilon)$.

ρ-Bisection for any $\rho > 0$. That is, does the N-vertex graph has a bisection of size at most ρN^2.

Furthermore:

- For all the above properties, in case the graph has the desired property, the testing algorithm outputs some auxiliary information which allows to construct, in poly$(1/\epsilon) \cdot N$-time, a partition which approximately obeys the property. For example, for ρ-CUT, we can construct a partition with at least $(\rho - \epsilon)N^2$ crossing edges.
- Except for Bipartite (2-Colorability) testing, running-time of poly$(1/\epsilon)$ is unlikely, as it will imply $\mathcal{NP} \subseteq \mathcal{BPP}$.
- None of these properties can be tested without queries when using $o(\sqrt{N})$ random examples.
- The k-Colorability tester has one-sided error: it always accepts k-colorable graphs. Furthermore, when rejecting a graph, this tester always supplies a poly$(1/\epsilon)$-size subgraph which is not k-colorable. All other algorithms have two-sided error, and this is unavoidable within $o(N)$ query-complexity.
- Our algorithms for k-Colorability, ρ-Clique and ρ-Cut can be easily extended to provide testers with respect to *product distributions*: that is, distributions $\Pi : \mathrm{V(G)}^2 \mapsto [0,1]$ of the form $\Pi(u,v) = \pi(u) \cdot \pi(v)$, where $\pi : \mathrm{V(G)} \mapsto [0,1]$ is a distribution on the vertices. In contrast, it is not possible to test any of the graph properties discussed above in a distribution-free manner.

General Graph Partition. All the above property testing problems are special cases of the General Graph Partition Testing Problem, parameterized by a set of lower and upper bounds. In this problem one needs to determine whether there exists a k-partition of the vertices so that the number of vertices in each side as well as the number of edges between each pair of sides falls between the corresponding lower and upper bounds (in the set of parameters). We present an algorithm for solving the above problem. The algorithm uses $\tilde{O}(k^2/\epsilon)^{k+O(1)}$ queries, runs in time exponential in its query-complexity, and makes two-sided error. Approximating partitions, if existing, can be efficiently constructed in this general case as well. We comment that the specialized algorithms perform better than the general algorithm with the appropriate parameters.

Other Graph Properties. Going beyond the general graph partition problem, we remark that there are graph properties which are very easy to test (e.g., Connectivity, Hamiltonicity, and Planarity). On the other hand, there are graph properties in $\mathcal{NP}$ which are extremely hard to test; namely, any testing algorithm must inspect at least $\Omega(N^2/\log N)$ of the vertex pairs. In view of the above, we believe that providing a characterization of graph properties according to the complexity of testing them may not be easy.

Our Techniques. Our algorithms share some underlying ideas. The first is the uniform selection of a small sample and the search for a *suitable* partition of this sample. In case of k-Colorability certain k-colorings of the subgraph induced by this sample will do, and are found by k-coloring a slightly augmented graph. In case of the other algorithms we exhaustively try all possible partitions. This is reminiscent of the *exhaustive sampling* of Arora *et. al.* [AKK95], except that the partitions considered by us are always directly related to the combinatorial structure of the problem. We show how each possible partition of the sample induces a partition of the entire graph so that the following holds. If the tested graph has the property in question then, with high probability over the choice of the sample, there exists a partition of the sample which induces a partition of the entire graph so that the latter partition approximately satisfies the requirements established by the property in question. For example, in case the graph has a ρ-cut there exists a 2-way-partition of the sample inducing a partition of the entire graph with $(\rho-\epsilon)N^2$ crossing edges. On the other hand, if the graph should be rejected by the test, then by definition no partition of the entire graph (and in particular none of the induced partitions) approximately obeys the requirements.

The next idea is to use an additional sample to approximate the quality of each such induced partition of the graph, and discover if at least one of these partitions approximately obeys the requirements of the property in question. An important point is that since the first sample is small (i.e., of size $\mathrm{poly}(1/\epsilon)$), the total number of partitions it induces is only $\exp \mathrm{poly}(1/\epsilon))$. Thus, the additional sample must approximate only these many partitions (rather than all possible partitions of the entire graph) and it suffices that this sample be of size $\mathrm{poly}(1/\epsilon)$,

The difference between the various algorithms is in the way in which partitions of the sample induce partitions of the entire graph. The simplest case is in testing Bipartiteness. For a partition $(\mathrm{S}_1, \mathrm{S}_2)$ of the sample, all vertices in the graph which have a neighbor in S_1 are placed on one side, and the rest of the vertices are placed on the other side. In the other algorithms the induced partition is less straightforward. For example, in case of ρ-Clique, a partition $(\mathrm{S}_1, \mathrm{S}_2)$ of the sample S with $|\mathrm{S}_1| \approx \rho|\mathrm{S}|$, induces a candidate clique roughly as follows. Consider the set T of graph vertices each neighboring all of S_1. Then the candidate clique consists of the ρN vertices with the highest degree in the subgraph induced by T. In the Bisection and General Partition testing algorithms, auxiliary guesses which are implemented by exhaustive search are used.

1.3 Testing Graph Properties and Approximation

The relation of testing graph properties to approximation is best illustrated in the case of Max-CUT. A tester for the class ρ-cut, working in time $T(\epsilon, N)$, yields an algorithm for approximating the maximum cut in an N-vertex graph, up to additive error ϵN^2, in time $\frac{1}{\epsilon} \cdot T(\epsilon, N)$. Thus, for any constant $\epsilon > 0$, we can approximate the size of the max-cut to within ϵN^2 in constant time. This yields a constant time approximation scheme (i.e., to within any constant relative error) for dense graphs, improving on Arora *et. al.* [AKK95] and de la Vega [dlV94] who solved this problem in polynomial-time ($O(N^{1/\epsilon^2})$–time and $\exp(\tilde{O}(1/(\epsilon^2))) \cdot N^2$–time, respectively). In both works the problem is solved by actually constructing approximate max-cuts. Finding an approximate max-cut does not seem to follow from the mere existence of a tester for ρ-cut; yet, as mentioned above, our tester can be used to find such a cut in time linear in N ($\tilde{O}(1/\epsilon^3) \cdot N + \exp(\tilde{O}(1/\epsilon^3))$–time).

One can turn the question around and ask whether approximation algorithms for dense instances can be transformed into corresponding testers as defined above. In several cases this is possible. For example, using some ideas from our work, the Max-CUT algorithm of [dlV94] can be transformed into a tester of complexity comparable to ours. We do not know whether the same is true with respect to the algorithms in [AKK95]. Results on testing graph properties can be derived also from work by Alon *et. al.* [ADL+94]. That paper proves a constructive version of the Regularity Lemma of Szemerédi, and obtains from it a polynomial-time algorithm that given an N-vertex graph, $\epsilon > 0$ and $k \geq 3$, either finds a subgraph of size $f(\epsilon, k)$ which is not k-colorable, or omits at most ϵN^2 edges and k-colors the rest. Noga Alon has observed that the analysis can be modified to yield that almost all subgraphs of size $f(\epsilon, k)$ are not k-colorable, which in turn implies a tester for k-Colorability. In comparison with our k-Colorability Tester, which takes a sample of $O(k^2 \log k/\epsilon^3)$ vertices, the k-colorability tester derived (from [ADL+94]) takes a much bigger sample of size equaling a tower of $(k/\epsilon)^{20}$ exponents (i.e., $\log^* f(\epsilon, k) = (k/\epsilon)^{20}$).

A DIFFERENT NOTION OF APPROXIMATION FOR MAX-CLIQUE. Our notion of ρ-Clique Testing differs from the traditional notion of Max-Clique Approximation.[2] When we talk of testing "ρ-Cliqueness", the task is to distinguish the case in which an N-vertex graph has a clique of size ρN from the case in which it is ϵ-far from the class of N-vertex graphs having a clique of size ρN. On the other hand, traditionally, when one talks of approximating the size of Max-Clique, the task is to distinguish the case in which the max-clique has size at least ρN from, say, the case in which the max-clique has size at most $\rho N/2$. Whereas the latter problem is NP-Hard, for $\rho \leq 1/64$ (see [BGS95, Sec. 3.9]), we've shown that the former problem can be solved in $\exp(O(1/\epsilon^2))$-time, for any $\rho, \epsilon > 0$. Furthermore, Arora *et. al.* [AKK95] showed that the "dense-subgraph" problem, a generalization of ρ-cliqueness, has a polynomial-time approximation scheme (PTAS) for dense instances, and our General Partition algorithm (with the appropriate parameters) improves on their result.

TESTING k-COLORABILITY VS. APPROXIMATING k-COLORABILITY. Petrank has shown that it is NP-Hard to distinguish 3-colorable graphs from graphs in which every 3-partition of the vertex set violates at least a constant fraction of the edges [Pet94]. In contrast, our k-Colorability Tester implies that solving the same promise problem is easy *for dense graphs*, where by dense graphs we mean N-vertex graphs with $\Omega(N^2)$ edges. This is the case since, for every $\epsilon > 0$, our tester can distinguish, in $\exp(k^2/\epsilon^3)$-time, between k-colorable N-vertex graphs and N-vertex graphs which remain non-k-colorable even if one omits at most ϵN^2 of their edges.[3]

We note that deciding k-colorability even for N-vertex graphs of minimum degree at least $\frac{k-3}{k-2} \cdot N$ is NP-complete (cf., Edwards [Edw86]). On the other hand, Edwards also gave a polynomial-time algorithm for k-coloring k-colorable N-vertex graphs of minimum degree at least αN, for any constant $\alpha > \frac{k-3}{k-2}$.

1.4 Other Related Work

PROPERTY TESTING IN THE CONTEXT OF PCP: Property testing plays a central role in the construction of PCP systems. Specifically, the property tested is being a codeword with respect to a specific code. This paradigm explicitly introduced in [BFLS91] has shifted from testing codes defined by low-degree polynomials [BFLS91, FGL$^+$91, AS92, ALM$^+$92] to testing Hadamard codes [ALM$^+$92, BGLR93, BS94, BCH$^+$95], and recently to testing the "long code" [BGS95]. See also discussion in [RS96, Kiw96].

PROPERTY TESTING IN THE CONTEXT OF PROGRAM CHECKING: There is an immediate analogy between program self-testing [BLR93] and property-testing *with queries*. The difference is that in self-testing, a function f (represented by a program) is tested for being close to a fully specified function g, whereas in property-testing the test is whether f is close to any function in a function class $\mathcal{G}$. Interestingly, many self-testers [BLR93, RS96] work by *first* testing that the program satisfies some properties which the function it is supposed to compute satisfies (and only then checking that the program satisfies certain constraints specific to the function). Rubinfeld and Sudan [RS96] defined property testing, under the uniform distribution and using queries, and related

[2] In fact, our notion is analogous to *dual approximation* where one seeks a "super optimal" solution which is "almost feasible"; cf., [HS87, HS88].

[3] As noted by Noga Alon, similar results, alas with much worse dependence on ϵ, can be obtained by using the results of Alon et. al. [ADL$^+$94].

it to their notion of Robust Characterization. Rubinfeld [Rub94] focuses on property testing as applied to properties which take the form of functional equations of various types.

Property Testing in the context of Learning Theory: Departing from work in Statistics regarding the classification of distributions (e.g., [HW58, Cov73, ZK91]), Ben-David [BD92] and Kulkarni and Zeitouni [KZ93] considered the problem of classifying an unknown function into one of two classes of functions, given labeled examples. Ben-David studied this classification problem in the limit (of the number of examples), and Kulkarni and Zeitouni studied it in a PAC inspired model. For any fixed ϵ, the problem of testing the class $\mathcal{F}$ with distance parameter ϵ can be casted as such a classification problem (with $\mathcal{F}$ and the set of functions ϵ-away from $\mathcal{F}$ being the two classes). A different variant of the problem was considered by Yamanishi [Yam95].

Testing Graph Properties. Our notion of testing a graph property P is a *relaxation* of the notion of *deciding the graph property* P which has received much attention in the last two decades [LY91]. In the classical problem there are no margins of error, and one is required to accept all graphs having property P and reject all graphs which lack it. In 1975 Rivest and Vuillemin [RV76] resolved the Aanderaa–Rosenberg Conjecture [Ros73], showing that any deterministic procedure for deciding any non-trivial monotone N-vertex graph property must examine $\Omega(N^2)$ entries in the adjacency matrix representing the graph. The query complexity of randomized decision procedures was conjectured by Yao to be $\Omega(N^2)$. Progress towards this goal was made by Yao [Yao87], King [Kin91] and Hajnal [Haj91] culminating in an $\Omega(N^{4/3})$ lower bound. Our results, that some non-trivial monotone graph properties can be tested by examining a constant number of random locations in the matrix, stand in striking contrast to all of the above.

Approximation in Dense Graphs. As stated previously, Arora *et. al.* [AKK95] and de la Vega [dlV94] presented PTAS for dense instances of Max-CUT. The approach of Arora *et. al.* uses Linear Programming and Randomized Rounding, and applies to other problems which can be casted as a "smooth" Integer Programs.[4] The methods of de la Vega [dlV94] are purely combinatorial and apply also to similar graph partition problems. Following the approach of Alon *et. al.* [ADL$^+$94], but using a modification of the regularity Lemma (and thus obtaining much improved running times), Frieze and Kannan [FK96] devise PTAS for several graph partition problems such as Max-Cut and Bisection. We note that compared to all the above results, our respective graph partitioning algorithms have better running-times. Like de la Vega, our methods use elementary combinatorial arguments related to the problem at hand. Still our methods suffice for dealing with the General Graph Partition Problem.

[4] In [AFK96], the approach of [AKK95] is extended to other problems, such as Graph Isomorphism, using a new rounding procedure for the Assignment Problem.

Chapter 2

General Definitions and Observations

2.1 Definitions

Let $\mathcal{F} = \{\mathcal{F}_n\}$ be a parameterized class of functions, where the functions[1] in $\mathcal{F}_n$ are defined over $\{0,1\}^n$ and let $\mathcal{D} = \{D_n\}$ be a corresponding class of distributions (i.e., D_n is a distribution on $\{0,1\}^n$). We say that a function f defined on $\{0,1\}^n$ is ϵ-*close* to $\mathcal{F}_n$ with respect to D_n if there exists a function $g \in \mathcal{F}_n$ such that

$$\mathrm{Prob}_{x \sim D_n}[f(x) \neq g(x)] \leq \epsilon \ . \tag{2.1}$$

Otherwise, f is ϵ-*far* from $\mathcal{F}_n$ (with respect to D_n).

We shall consider several variants of testing algorithms, where the most basic one is defined as follows.

Definition 2.1.1 (property testing): *Let $\mathcal{A}$ be an algorithm which receives as input a size parameter n, a distance parameter $0 < \epsilon < 1$, and a confidence parameter $0 < \delta < 1/2$. Fixing an arbitrary function f and distribution D_n over $\{0,1\}^n$, the algorithm is also given access to a sequence of f-*labeled examples*, $(x_1, f(x_1)), (x_2, f(x_2)), ...$, where each x_i is independently drawn from the distribution D_n. We say that $\mathcal{A}$ is a* property testing algorithm *(or simply a* testing algorithm*) for the class of functions $\mathcal{F}$ if for every n, ϵ and δ and for every function f and distribution D_n over $\{0,1\}^n$ the following holds*

- *if $f \in \mathcal{F}_n$ then with probability at least $1 - \delta$ (over the examples drawn from D_n and the possible coins tosses of $\mathcal{A}$), $\mathcal{A}$* accepts *f (i.e., outputs 1);*
- *if f is ϵ-far from $\mathcal{F}_n$ (with respect to D_n) then with probability at least $1 - \delta$, $\mathcal{A}$* rejects *f (i.e., outputs 0).*

The sample complexity *of $\mathcal{A}$ is a function of n, ϵ and δ bounding the number of labeled examples examined by $\mathcal{A}$ on input (n, ϵ, δ).*

Though it was not stated explicitly in the definition, we shall usually also be interested in bounding the running time of a property testing algorithm (as a function of the parameters n, δ, ϵ, and in some case of a complexity measure of the class $\mathcal{F}$).

We consider the following variants of the above definition:

[1] The range of these functions may vary and for many of the results and discussions it suffices to consider Boolean function.

1. D_n may be a specific distribution which is known to the algorithm. In particular, we shall be interested in testing with respect to the uniform distribution.

2. D_n may be restricted to a known class of distributions (e.g., product distributions).

3. The algorithm may be given access to an *oracle* for the function f, which when queried on $x \in \{0,1\}^n$, returns $f(x)$. In this case we refer to the number of queries made by $\mathcal{A}$ (which is a function of n, ϵ, and δ), as the *query complexity* of $\mathcal{A}$.

4. In some cases the algorithm might have the additional feature that whenever it outputs fail it also provides a *certificate* to the fact that $f \notin F$. Certificates are defined with respect to a *verification algorithm* which *accepts* a sequence of labeled examples whenever there exists $f \in \mathcal{F}_n$ which is consistent with the sequence. (We do not require that the algorithm reject each sequence which is not consistent with some $f \in \mathcal{F}_n$.) A certificate for $f \notin \mathcal{F}_n$ is an f-labeled sequence which is rejected by the verification algorithm.

5. The algorithm is given two distance parameters, ϵ_1 and ϵ_2, and is required to pass with high probability every f which is ϵ_1-close to F_n, and fail every f which is ϵ_2-far from F_n.

2.2 On the Relation between Property Testing and PAC Learning

A *Probably Approximately Correct* (PAC) learning algorithm [Val84] works in the same framework as that described in Definition 2.1.1 except for the following (crucial) differences:

1. It is given a *promise* that the unknown function f (referred to as the *target* function) belongs to $\mathcal{F}$;

2. It is required to output (with probability at least $1 - \delta$) a *hypothesis* function h which is ϵ-close to f, where closeness is as defined in Equation (2.1) (and ϵ is usually referred to as the *approximation* parameter).

Note that the differences pointed out above effect the tasks in opposite directions. Namely, the absence of a promise makes testing potentially harder than learning, whereas deciding whether a function belongs to a class rather than finding the function may make testing easier.

In the learning literature, a distinction is made between *proper* (or *representation dependent*) learning and *non-proper* learning [PV88]. In the former model, the hypothesis output by the learning algorithm is required to belong to the same function class as the target function f, i.e. $h \in \mathcal{F}$, while in the latter model, no such restriction is made. We stress that a proper learning algorithm (for $\mathcal{F}$) may either halt without output or output a function in $\mathcal{F}$, but it may not output functions not in $\mathcal{F}$.[2] There are numerous variants of PAC learning (including learning with respect to specific distributions, and learning with access to an oracle for the target function f). Unless stated otherwise, whenever we refer in this chapter to PAC learning we mean the *distribution-free no-query model* described above. The same is true for references to property testing. In addition, apart from one example, we shall restrict our attention to classes of Boolean functions.

TESTING IS NOT HARDER THAN PROPER LEARNING.

[2]We remark that in case the function is $\mathcal{F}$ have an easy to recognize representation, one can easily guarantee that the algorithm never outputs a function not in $\mathcal{F}$. Standard classes considered in works on proper learning typically have this feature.

Proposition 2.1 *If a function class $\mathcal{F}$ has a* proper *learning algorithm $\mathcal{A}$, then $\mathcal{F}$ has a property testing algorithm $\mathcal{A}'$ such that $m_{\mathcal{A}'}(n, \epsilon, \delta) = m_{\mathcal{A}}(n, \epsilon/2, \delta/2) + O(\log(1/\delta)/\epsilon)$. Furthermore, the same relation holds between the running times of the two algorithm.*

Proof: In order to test if $f \in \mathcal{F}$ or ϵ-far from it, we first run the learning algorithm $\mathcal{A}$ with confidence parameter $\delta/2$, and approximation parameter $\epsilon/2$. If $\mathcal{A}$ does not output a hypothesis, then we reject f. If $\mathcal{A}$ outputs a hypothesis h (which must be in $\mathcal{F}$ since $\mathcal{A}$ is a proper learning algorithm), then we approximate the distance between h and f by drawing an additional sample of size $O(\log(1/\delta)/\epsilon)$. If the approximated distance is less than $3\epsilon/4$ then we accept, otherwise we reject.

In case $f \in \mathcal{F}$, with probability at least $1 - \delta/2$, h is $\epsilon/2$-close to f, and a simple Chernoff bound tells us that with probability at least $1 - \delta/2$ over the additional sample, we shall not reject it. In case f is ϵ-far from $\mathcal{F}$, any hypothesis $h \in \mathcal{F}$ is at least ϵ-far from f, and with probability at least $1 - \delta/2$ over the additional sample, f is rejected. ■

In particular, the above proposition implies that if for every n, $\mathcal{F}_n$ has polynomial (in n) VC-dimension [VC71, BEHW89][3], then $\mathcal{F}$ has a tester whose sample complexity is polynomial in n, $1/\epsilon$, and $\log(1/\delta)$. The reason is that classes with polynomial VC-dimension can be properly learned from a sample of the above size [BEHW89]. However, the running time of such a proper learning algorithm, and hence of the resulting testing algorithm might be exponential in n.

Corollary 2.2 *Every class which is learnable with a* $\text{poly}(n/\epsilon)$ *sample (and thus has a* $\text{poly}(n)$ *VC dimension [BEHW89]) is testable with a* $\text{poly}(n/\epsilon)$ *sample (in at most exponential time).*

Testing may be harder than Learning. In contrast to Proposition 2.1 and to Corollary 2.2, we show that there are classes which are efficiently learnable (though not by a proper learning algorithm) but are not efficiently testable. This is proven by observing that many hardness results for proper learning (*cf.* [PV88, BR89, PW93]) actually establish the hardness of testing (for the same classes). Furthermore, we believe that it is more natural to view these hardness results as referring to testing and derive the hardness for proper learning via Proposition 2.1. Thus, the separation between efficient learning and efficient proper learning translates to a separation between efficient learning and efficient testing.

Proposition 2.3 *If $\mathcal{NP} \not\subseteq \mathcal{BPP}$ then there exist function classes which are not* $\text{poly}(n/\epsilon)$*-time testable but are* $\text{poly}(n/\epsilon)$*-time (non-properly) learnable.*

Proof: The proposition follows from the fact that many of the representation dependent hardness results (*cf.* [Gol78, Ang78, PV88, BR89, PW93]) have roughly the following form. An NP-complete problem is reduced to the following decision problem: Given a set S of labeled examples, does there exist a function in $\mathcal{F}$ which is consistent with S? A learning algorithm is forced to find a consistent function if one exists by letting the support of the distribution D (which is allowed to be arbitrary) lie solely on S, and setting ϵ to be smaller than $1/|S|$. Actually, since the consistency problem is that of deciding if there exists a consistent function and not necessarily of finding such a function, it follows that the corresponding testing problem (using the same D and ϵ) is hard as well. Details follow.

Let $\mathcal{F}$ be a fixed class of functions and suppose that the following decision problem is NP-complete

[3] The Vapnik Chervonenkis (VC) dimension of a class $\mathcal{F}_n$ is defined to be the size d of the largest set $X \in \{0,1\}^n$ for which the following holds. For each (of the 2^d) partitions (X_0, X_1) of X there exists a function $f \in \mathcal{F}_n$ such that for every $x \in X_0$, $f(x) = 0$, and for every $x \in X_1$, $f(x) = 1$. A set X that has this feature is said to be *shattered* by $\mathcal{F}_n$.

input: a sequence $(x_1, \sigma_1), ..., (x_t, \sigma_t)$, where $x_i \in \{0,1\}^n$, $\sigma_i \in \{0,1\}$, and $t = \text{poly}(n)$.

question: is there a function $f \in \mathcal{F}_n$ so that $f(x_i) = \sigma_i$, for all $i \in [t]$.

Assuming that there exists a property testing algorithm, denoted $\mathcal{A}$, for the class $\mathcal{F}$, we decide the above problem. We invoke $\mathcal{A}$ with parameters n, ϵ and δ, where $\epsilon < 1/m$ (say $\epsilon = 1/2m$) and say $\delta = 1/3$. Suppose that $\mathcal{A}$ requires $m \stackrel{\text{def}}{=} m_{\mathcal{A}}(n, \epsilon, \delta)$ samples. We uniformly select m indices, denoted $i_1, ..., i_m$, (possibly with repetitions) out of $[t]$ and feed $\mathcal{A}$ with the labeled sample $(x_{i_1}, \sigma_{i_1}), ..., (x_{i_m}, \sigma_{i_m})$. We decide according to $\mathcal{A}$'s output.

We analyze the performance of our algorithm by relating it to the performance of the property testing algorithm on the distribution, $\mathcal{D}_n$, which is uniform on the set $\{x_i : i \in [t]\}$. Suppose first that there exists $f \in \mathcal{F}_n$ so that $f(x_i) = \sigma_i$, for all $i \in [t]$. In this case, we provide $\mathcal{A}$ with a random sample labeled by a function in $\mathcal{F}_n$ and thus with probability at least $1 - \delta$ the test must accept. Thus, our decision procedure accepts yes-instances with probability at least $1 - \delta$.

Suppose now that there exists no function $f \in \mathcal{F}_n$ such that $f(x_i) = \sigma_i$, for all $i \in [t]$. This implies that the function f defined by $f(x_i) \stackrel{\text{def}}{=} \sigma_i$, for all $i \in [t]$ (and $f(x) = 0$ for $x \notin \{x_i : i \in [t]\}$), is at distance at least $\frac{1}{t} > \epsilon$ from $\mathcal{F}$ (with respect to the distribution $\mathcal{D}_n$). Since we provide $\mathcal{A}$ with a random sample labeled by this f, the test must reject with probability at least $1 - \delta$. Hence, our decision procedure rejects no-instances with probability at least $1 - \delta$.

This establishes, in particular, that testing the class of k-Term DNF (where k is a constant) is NP-Hard (see [PV88]). On the other hand, k-Term DNF (for constant k) is efficiently learnable (using the hypothesis class of k-CNF) [Val84, PV88]. ∎

We stress that while Proposition 2.1 generalizes to learning and testing under specific distributions, and to learning and testing with queries, the proof of Proposition 2.3 uses the premise that the testing (or proper learning) algorithm works for any distribution and does not make queries.

Testing may be Easier than Learning. We start by presenting a function class which is easy to test but cannot be learnedΦΦ with polynomial sample complexity, regardless of the running-time.

Proposition 2.4 *There exist function classes $\mathcal{F}$ such that:*

- *$\mathcal{F}$ has a property testing algorithm whose sample complexity and running time are $O(\log(1/\delta)/\epsilon)$;*
- *Any learning algorithm for $\mathcal{F}$ must have sample complexity exponential in n.*

Proof: It is possible to come up with quite a few examples of functions classes for which the above holds. We give one example below. For each n let $\mathcal{F}_n$ include all functions f over $\{0,1\}^n$, such that for every $y \in \{0,1\}^{n-1}$, $f(1y) = 1$ (and if the first bit of the input is 0 then no restriction is made).

Given $m = O(\log(1/\delta)/\epsilon)$ examples, labeled by an unknown f and drawn according to an arbitrary distribution D_n, the testing algorithm will simply verify that for all examples x whose first bit is 1, $f(x) = 1$. If $f \in \mathcal{F}_n$, it will always accept it, and if f is ϵ-far from F_n (with respect to D_n) then the probability that it does not observe even a single example of the form $(1y, 0)$ (and as a consequence, accepts f), is bounded by $(1-\epsilon)^m < \delta$. On the other hand, the VC-dimension of $\mathcal{F}_n$ is 2^{n-1} (since the set $\{0y : y \in \{0,1\}^{n-1}\}$ is shattered by $\mathcal{F}_n$). By [BEHW89], learning this class requires a sample of size $\Omega(2^n)$. ∎

The impossibility of learning the function class in Proposition 2.4 is due to its exponential VC-dimension, (i.e., it is a pure information theoretic consideration). We now turn to function classes of exponential (rather than double exponential) size. Such classes are always learnable with a polynomial sample, the question is whether they are learnable in polynomial-time. We present a function class which is easy to test but cannot be learned in polynomial-time (even under the uniform distribution), provided trapdoor one-way permutations exist (*e.g.,* factoring is intractable).

Proposition 2.5 *If there exist trapdoor one-way permutations then there exists a family of functions which can be tested in* $\mathrm{poly}(n/\epsilon)$*-time but can not be learned in* $\mathrm{poly}(n/\epsilon)$*-time, even with respect to the uniform distribution. Furthermore, the functions can be computed by* $\mathrm{poly}(n)$*-size circuits.*

Proof: Let $\{p_\alpha : D_\alpha \mapsto D_\alpha\}$ be a family of trapdoor permutations (see [Gol95, Sec. 2.4]). We first assume, for simplicity, that $D_\alpha = \{0,1\}^{|\alpha|}$. Loosely speaking, this family should satisfy three properties. Firstly, there exists a polynomial-time algorithm which given α and x outputs $p_\alpha(x)$. Secondly, no polynomial-time algorithm can, given α and x, output $p_\alpha^{-1}(x)$ with non-negligible success probability (taken over all possible choices of $\alpha, x \in \{0,1\}^n$). Thirdly, for each $\alpha \in \{0,1\}^*$ (equivalently, for each p_α) there exists a $\mathrm{poly}(|\alpha|)$-size circuit which inverts p_α. We let $\mathcal{OW} = \{\mathcal{OW}_n\}$, where $\mathcal{OW}_n$ consists of the multi-valued functions f_α, so that $f_\alpha(x) \stackrel{\mathrm{def}}{=} (\alpha, p_\alpha^{-1}(x))$ for every $\alpha, x \in \{0,1\}^n$.

To test if $f \in \mathcal{OW}$, we merely examine sufficiently many f-labeled examples. For each labeled example, $(x, (\alpha, y))$, if $p_\alpha(y) \neq x$ then we reject f. In addition we also reject if we see two examples, $(x_1, (\alpha_1, y_1))$ and $(x_2, (\alpha_2, y_2))$, so that $\alpha_1 \neq \alpha_2$. Otherwise we accept f. Note that this test works for any distribution on the examples and so the class $\mathcal{OW}$ is efficiently testable. On the other hand, it is infeasible to learn the class $\mathcal{OW}$ under the uniform distribution (since any such learning algorithm yields an algorithm for inverting the one-way permutations.

In general the D_α's may be arbitrary, but there exists a probabilistic polynomial-time sampling algorithm, S, that on input α selects at random an element in D_α. Let $S(\alpha, r)$ denote the output of the sampling algorithm S, on input α and coin tosses r. The construction of $\mathcal{OW}$ is modified in the natural manner; that is, $f_\alpha(r) \stackrel{\mathrm{def}}{=} (\alpha, p_\alpha^{-1}(S(\alpha, r)))$. ■

The class presented in Proposition 2.5 consists of multi-valued functions. We leave it as an open problem whether a similar result holds for a class of Boolean functions.

Learning and Testing with queries (under the uniform distribution). Let the class of parity functions, $\mathcal{PAR} = \{\mathcal{PAR}_n\}$, where $\mathcal{PAR}_n \stackrel{\mathrm{def}}{=} \{f_S : S \subseteq [n]\}$ and $f_S : \{0,1\}^n \mapsto \{0,1\}$ so that $f_S(x) = \sum_{i \in S} x_i \bmod 2$. Work on linearity testing [BLR93, BFL91, FGL+91, BGLR93, BS94], culminating in the result of Bellare *et. al.* [BCH+95], implies that there exists a testing algorithm for the class $\mathcal{PAR}$, under the uniform distribution, whose query complexity is $O(\log(1/\delta)/\epsilon)$. The running-time is a factor n bigger, merely needed to write down the queries. On the other hand, any learning algorithm for this class must use at least n queries (or examples). The reason being that any query (or example) gives rise to a single linear constraint on the coefficients of the linear function, and with less than n such constraints the function is not uniquely defined. Furthermore, every two linear functions disagree with probability 1/2 on a uniformly chosen input.

An example of a testable (with queries) class which is not learnable even with $\mathrm{poly}(n)$ queries is the class of multi-variate polynomials. Specifically, let $\mathcal{POLY} = \{\mathcal{POLY}_n\}$, where $\mathcal{POLY}_n$ consists of n-variate polynomials of total degree n over the field $\mathrm{GF}(q)$, where q is the first prime in the interval $[n^4, 2n^4]$. Work on low-degree testing [BFL91, BFLS91, GLR+91], culminating in the result of Rubinfeld and Sudan [RS96], implies that there exists a testing algorithm for the class $\mathcal{POLY}$, under the uniform distribution, whose query complexity is $O(\min\{n^3, \frac{n}{\epsilon}\} \cdot \log(1/\delta))$. The running-time is a factor $O(n \log n)$ bigger, merely needed to write down the queries and do some simple algebra. It is not hard to show that one cannot possibly learn $\mathcal{POLY}_n$, under the uniform distribution, using only poly(n) queries. Again, the reason is that any query (or example) gives rise to a single linear constraint on the coefficients of the polynomial. Since there are exponentially (in n) many coefficients, this leaves the polynomial not uniquely defined. Finally, one invokes Schwarz's Lemma [Sch80], by which two such degree n polynomials can agree on at most $\frac{n}{q} < \frac{1}{n^3}$ fraction of the domain.

Agnostic Learning and Testing. In a variant of PAC learning, called *Agnostic* PAC learning [KSS92], there is no promise concerning the target function f. Instead, the learner is required to output a hypothesis h from a certain hypothesis class $\mathcal{H}$, such that h is ϵ-close to the function in $\mathcal{H}$ which is closest to f. The absence of a promise makes agnostic learning closer in spirit to property testing than basic PAC learning. In particular, agnostic learning with respect to a hypothesis class $\mathcal{H}$ implies proper learning of the class $\mathcal{H}$ and thus property testing of $\mathcal{H}$.

Learning and Testing Distributions. A *distribution learning* algorithm for a class of distributions, $\mathcal{D} = \{\mathcal{D}_n\}$, receives (in addition to the parameters n, ϵ and δ) an (unlabeled) sample of strings in $\{0,1\}^n$, distributed according to an unknown distribution $D \in \mathcal{D}_n$. The algorithm is required to output a distribution D' (either in form of a machine which generates strings according to D', or in form of a machine that on input $x \in \{0,1\}^n$ outputs $D'(x)$), such that with probability at least $1-\delta$, the variation distance between D and D' is at most ϵ. (For further details see [KMR$^+$94].) In contrast, a *distribution testing* algorithm, upon receiving a sample of strings in $\{0,1\}^n$ drawn according to an unknown distribution D, is required to accept D, with probability at least $1-\delta$, if $D \in \mathcal{D}_n$, and to reject (with probability $\geq 1-\delta$) if D is ϵ-far from $\mathcal{D}_n$ (with respect to the variation distance).

The context of learning and testing distributions offers a dramatic demonstration to the importance of a promise (*i.e.*, the fact that the learning algorithm is required to work only when the target belongs to the class, whereas the testing algorithm needs to work for all targets which are either in the class or far away from it).

Proposition 2.6 *There exist distribution classes which are efficiently learnable (in both senses mentioned above) but cannot be tested with a subexponential sample (regardless of the running-time).*

Proof: Consider the class of distributions $\mathcal{D} = \{\mathcal{D}_n\}$ consisting of all distributions, D_n^p, which are generated by n independent tosses of a coin with bias p. Clearly, this class can be efficiently learned (by merely approximating the bias p of the target distribution). However, a tester cannot distinguish the case in which a sample of subexponential size is taken from the uniform distribution $D_n^{1/2}$ (and thus should be accepted), and the case in which such a sample is taken from a 'typically bad' distribution B_n^S which is uniform over $S \subset \{0,1\}^n$, where $|S| = 2^{n-1}$. Formally, we consider the behavior of the test when given a sample from $D_n^{1/2}$ versus its behavior when given a sample from B_n^S, where S is uniformly chosen among all subsets of size 2^{n-1}. ■

Observing that the above proof holds for any distribution class which contains the uniform distribution and is far from distributions such as the B_n^S's, we feel discouraged to continue the study of testing distributions.

2.3 Other Observations

Property Testing may be very hard.

Proposition 2.7 *There exists a function class $\mathcal{F} = \{\mathcal{F}_n\}$ for which any testing algorithm must inspect the value of the function at a constant fraction of the inputs (i.e., on $\Omega(2^n)$ inputs). This holds even for testing with respect to the uniform distributions, for any distance parameter $\epsilon < 1/2$ and confidence parameter $\delta < 1/2$, and when allowing the algorithm to make queries and use unlimited computing time.*

Proof: Suppose for simplicity that $\epsilon = 1/4$ and $\delta = 1/5$. We will use the Probabilistic Method to demonstrate the existence of a function class, $\mathcal{F} = \{\mathcal{F}_n\}$, satisfying the claim so that $\mathcal{F}_n$ consists of $2^{\frac{1}{10}\cdot 2^n}$ Boolean functions operating on $\{0,1\}^n$. We'll show that there exists a class $\mathcal{F}$ so that a uniformly selected function is both far from it and indistinguishable from it when observing $o(2^n)$ values.

First we show that, with high probability, a uniformly selected function $g : \{0,1\}^n \mapsto \{0,1\}$ is ϵ-far from any set, $\mathcal{F}_n$, of $2^{\frac{1}{10}\cdot 2^n}$ functions. Let $N \stackrel{\text{def}}{=} 2^n$ and U_N be uniformly distributed on $\{0,1\}^N$. Then,

$$\begin{aligned}
\text{Prob}_g[g \text{ is } \epsilon\text{-close to } \mathcal{F}_n] &\leq \text{Prob}_g[\exists f \in \mathcal{F}_n \text{ s.t. } g(x) \neq f(x) \text{ for more that } \epsilon N\ x\text{'s}] \\
&\leq |\mathcal{F}_n| \cdot \text{Prob}[U_N \text{ has at most } \epsilon N \text{ 1's}] \\
&\leq 2^{N/10} \cdot 2e^{-N/8} \\
&= \exp(-\Omega(N))
\end{aligned}$$

Thus, with overwhelmingly high probability (over the choices of g) the function g is ϵ-far from the class $\mathcal{F}_n$.

We now consider any fixed sequence, S, of $T \stackrel{\text{def}}{=} N/20$ inputs and compare the values assigned to them by g versus the values assigned to them by a uniformly chosen function in $\mathcal{F}_n$. Clearly, in the first case the values are uniformly distributed. Let $\delta_S(\mathcal{F}_n)$ denote the statistical difference between the uniform distribution and the distribution of function-values induced by a uniformly selected function in $\mathcal{F}_n$. That is,

$$\delta_S(\mathcal{F}_n) \stackrel{\text{def}}{=} \frac{1}{2} \cdot \sum_{\alpha \in \{0,1\}^T} |\text{Prob}_{f\in\mathcal{F}_n}[f(S) = \alpha] - 2^{-T}|$$

where $f(\{i_1, ..., i_t\}) = f(i_1) \cdots f(i_t)$. We consider the probability, taken over all possible choices of $\mathcal{F}_n$ (consisting of $2^{\frac{1}{10}\cdot 2^n}$ functions), that $\delta_S(\mathcal{F}_n) > 1/2$. We get

$$\begin{aligned}
\text{Prob}_{\mathcal{F}_n}[\delta_S(\mathcal{F}_n) > 1/2] &\leq \text{Prob}_{\mathcal{F}_n}[\exists \alpha \in \{0,1\}^T \text{ s.t. } |\text{Prob}_{f\in\mathcal{F}_n}(f(S) = \alpha) - 2^{-T}| > 2^{-(T+1)}] \\
&\leq 2^T \cdot 2e^{-\frac{1}{3}\cdot(\frac{1}{2})^2 \cdot 2^{-T}|\mathcal{F}_n|} \\
&= 2^{N/100} \cdot 2e^{-\frac{1}{12}\cdot 2^{0.05N}} \\
&= \exp(-2^{\Omega(N)})
\end{aligned}$$

Summing the probabilities over all possible $\binom{N}{T}$ sequences we conclude that with overwhelmingly high probability, over the choice of $\mathcal{F}_n$, all δ_S's are bounded above by 1/2. Consequently, the difference between the acceptance probability of a truly random g and the acceptance probability of a uniformly selected $f \in \mathcal{F}_n$ is at most 1/2. This does not allow to both accept every $f \in \mathcal{F}_n$ with probability at least 0.8 and accept a random g with probability 0.21 (the extra 0.01 over-compensates for the case that a random g is ϵ-close to $\mathcal{F}_n$). ■

THE ALGEBRA OF PROPERTY TESTING. Suppose that two function classes are testable within certain complexity. What can we of their INTERSECTION, UNION and COMPLEMENTATION? Unfortunately, in general, we can only say that their union is testable within comparable complexity. That is,

Proposition 2.8 *Let $\mathcal{F}' = \{\mathcal{F}'_n\}$ and $\mathcal{F}'' = \{\mathcal{F}''_n\}$ be function classes testable within complexities $c'(n, \epsilon.\delta)$ and $c''(n, \epsilon, \delta)$. Then, the class $\mathcal{F} = \{\mathcal{F}_n\}$, where $F_n = \mathcal{F}'_n \cup \mathcal{F}''_n$, is testable within complexities $c(n, \epsilon, 2\delta) = c'(n, \epsilon, \delta) + c''(n, \epsilon, \delta)$.*

Proof: The testing algorithm for $\mathcal{F}$ consists of testing for membership in both $\mathcal{F}'$ and $\mathcal{F}''$ and accepting iff one of the tests has accepted. The validity of this test relies on the fact that if f is ϵ-far from $\mathcal{F} = \mathcal{F}' \cup \mathcal{F}''$ then it is ϵ-far from both $\mathcal{F}'$ and $\mathcal{F}''$. ■

The fact that an analogous claim does not hold for intersection is the reason that an analogous tester does not work for the intersection class. That is, it may be the case that f is far from $\mathcal{F} = \mathcal{F}' \cap \mathcal{F}''$ yet it is very close to both $\mathcal{F}'$ and $\mathcal{F}''$. Thus, if a function may be close to both $\mathcal{F}'$ and $\mathcal{F}''$, pass both the corresponding property tests, but still may be far from $\mathcal{F}$. For example, $\mathcal{F}'$ may contain functions which are pairwise far apart and $\mathcal{F}''$ may consists of functions obtained from $\mathcal{F}'$ by very few modifications. An efficient algorithm accepting functions in $\mathcal{F}'$ is likely to accept also functions in $\mathcal{F}''$ and so will accept all functions in $\mathcal{F}' \cup \mathcal{F}''$ although there are far from $\mathcal{F}' \cap \mathcal{F}''$ (which is actually empty). Finally, we observe that property testing is not preserved under negation. That is,

Proposition 2.9 *There exists a function class $\mathcal{F} = \{\mathcal{F}_n\}$ which is trivially testable so that the class of functions not in $\mathcal{F}$ is not testable in subexponential complexity. We say that $\mathcal{F}$ is* trivially testable *if for every $\epsilon \geq 2^{-n}$ an algorithm which accepts every n-bit Boolean function satisfies the testing requirements with respect to $\mathcal{F}$.*

Proof: Consider the function class $\mathcal{F}$ used in the proof of Proposition 2.7. As shown there, this class is not testable in subexponential complexity. Now, let $\mathcal{H} = \{\mathcal{H}_n\}$ consists of all functions not in $\mathcal{F}$. It is easy to see that each function $f : \{0,1\}^n \mapsto \{0,1\}$ is at most 2^{-n}-far from $\mathcal{H}$. Thus, as long as $\epsilon \geq 2^{-n}$, the trivial algorithm which accepts all functions constitutes a tester for $\mathcal{H}$. ■

Chapter 3

Testing Graph Properties

In this chapter we concentrate on testing graph properties using queries and with respect to the uniform distribution. In Subsection 3.7.1, we discuss some extensions beyond this basic model.

General graph notations. We consider undirected, simple graphs (no multiple edges or self-loops). For a simple graph G, we denote by $\mathrm{V(G)}$ its vertex set and assume, without loss of generality, that $\mathrm{V(G)} = \{1, ..., |\mathrm{V(G)}|\}$. Graphs are represented by their (symmetric) adjacency matrix. Thus, graphs are associated with the Boolean function corresponding to this matrix (i.e., the value of a pair $(u,v) \in \mathrm{V(G)}^2$ indicates whether $(u,v) \in \mathrm{E(G)}$). This brings us to associated undirected graphs with directed graphs where each edge in the undirected graph is associated with a pair of anti-parallel edges.[1] Specifically, for a graph G, we denote by $\mathrm{E(G)}$ the set of ordered pairs which correspond to edges in G (i.e., $(u,v) \in \mathrm{E(G)}$ iff there is an edge between u and v in G). In the sequel, whenever we say 'edge' we mean a directed edge.

The distance between two N-vertex graphs, G_1 and G_2, is defined as the number of entries $(u,v) \in [N]^2$ ($[N] \stackrel{\mathrm{def}}{=} \{1, ..., N\}$) which are in the symmetric difference of $\mathrm{E(G_1)}$ and $\mathrm{E(G_2)}$. Dividing this quantity by N^2 we get

$$\mathrm{dist}(\mathrm{G}_1, \mathrm{G}_2) \stackrel{\mathrm{def}}{=} \frac{|(\mathrm{E(G_1)} \setminus \mathrm{E(G_2)}) \cup (\mathrm{E(G_2)} \setminus \mathrm{E(G_1)})|}{N^2}$$

This notation is extended naturally to a set, $\mathcal{C}$, of N-vertex graphs; that is, $\mathrm{dist}(\mathrm{G}, \mathcal{C}) \stackrel{\mathrm{def}}{=} \min_{\mathrm{G}' \in \mathcal{C}} \{\mathrm{dist}(\mathrm{G}, \mathrm{G}')\}$. Another notation used extensively in this chapter is the set of neighbors of a vertex v; that is, $\Gamma(v) \stackrel{\mathrm{def}}{=} \{u : (v,u) \in \mathrm{E(G)}\}$. This notation is extended to sets of vertices in the natural manner; i.e., $\Gamma(S) \stackrel{\mathrm{def}}{=} \cup_{v \in S} \Gamma(v)$.

Organization of this chapter: We present testers for Bipartiteness, k-Colorability (for $k \geq 3$), ρ-Clique, Max-CUT (and Max-k-CUT), Bisection and the General Graph Partition property. The latter generalizes all the former ones, but yields worse complexity bounds for the special cases. Also, Max-CUT (resp., Max-k-CUT) generalizes Bipartiteness (resp., k-Colorability), yet it can only be tested with two-sided error whereas the coloring properties have one-sided error testers. In view of all the above, we chose to make the exposition in each section as self-contained as possible. The only dependencies among the sections are the usage of Lemma 3.3.5 (stated for ρ-Clique) in all subsequent sections, and the Bisection section which builds on the Max-CUT section.

[1] Our convention makes the correspondence between graphs and functions more evident. In some places it also makes the analysis more natural; however, in other places it results in doubling certain quantities.

3.1 Testing Bipartiteness

In this section we describe an algorithm for testing the class, $\mathcal{B}$, of bipartite graphs. This is a special case of testing k-Colorability, considered in the next subsection. We choose to present the case of $k = 2$ separately because it is both simpler to describe, and it served as a good prelude to the next section. Moreover, the algorithm presented here has lower query complexity (in terms of its dependence on the distance parameter, ϵ) than the one described in the next section.
We start by describing a testing algorithm whose query complexity is $O\left(\frac{\log^2(1/\delta\epsilon)}{\epsilon^4}\right)$. We later point out how this algorithm can be slightly modified so that its query complexity decreases to $O\left(\frac{\log^2(1/\delta\epsilon)}{\epsilon^3}\right)$.

Bipartite Testing Algorithm

1. Choose uniformly $O\left(\frac{\log(1/(\delta\epsilon)}{\epsilon^2}\right)$ vertices. Let the set of vertices chosen be denoted by X.
2. For every pair of vertices $v_1, v_2 \in$ X, query if $(v_1, v_2) \in$ E(G). Let $\mathrm{G_X}$ be the induced subgraph.
3. If $\mathrm{G_X}$ is a bipartite graph then output *accept*, otherwise output *reject*.

Before stating the main theorem of this section, we introduce the following definitions.

Definition 3.1.1 (violating edges and good partitions): *We say that an edge* $(u, v) \in$ E(G) *is a* violating *edge with respect to a partition* $(\mathrm{V_1}, \mathrm{V_2})$ *of* V(G) *if either* $u, v \in \mathrm{V_1}$ *or* $u, v \in \mathrm{V_2}$. *If a partition* $(\mathrm{V_1}, \mathrm{V_2})$ *has at most* ϵN^2 *violating edges then we say that it is* ϵ-good. *Otherwise, it is* ϵ-bad. *A partition that has no violating edges is called* perfect.

Thus, if G is bipartite, then there exists a perfect partition of V(G), and if G is ϵ-far from bipartite then *every* partition of V(G) is ϵ-bad.

Theorem 3.1 *The* Bipartite Testing Algorithm *is a property testing algorithm for the class of bipartite graphs whose edge-query complexity and running time are* $O\left(\frac{\log^2(1/\delta\epsilon)}{\epsilon^4}\right)$. *Furthermore, if the tested graph* G *is bipartite then it is accepted with probability* 1, *and, with probability at least* $1 - \delta$ *(over the choice of the sampled vertices), it is possible to construct an* ϵ*-good partition of* V(G) *in time* $O\left(\frac{\log(1/\delta\epsilon)}{\epsilon} \cdot N\right)$*,*

3.1.1 Proof of Theorem 3.1

It is clear that if G is bipartite then *any* subgraph of G is bipartite and hence G will always be accepted. Since it is possible to determine if $\mathrm{G_X}$ is bipartite by simply performing a breadth-first-search (BFS) on $\mathrm{G_X}$, the bound on the running time of the testing algorithm directly follows. Note that if $\mathrm{G_X}$ is bipartite then the BFS provides us with a perfect partition of X, while if it is not bipartite, then it gives a certificate that $G \notin \mathcal{B}$. This certificate is in form of a cycle of odd length in $\mathrm{G_X}$, (which is also a cycle in G). Thus the heart of this proof is to show that if G is ϵ-far from bipartite then the test will reject it with probability at least $1 - \delta$. To this end we prove the counter-positive of the previous statement: For any graph G, if the Bipartite Testing Algorithm accepts G with probability greater than δ then V(G) must have an ϵ-good partition.

We view the set of sampled vertices X as a union of two disjoint sets U and S, where $t \stackrel{\mathrm{def}}{=} |\mathrm{U}| = O\left(\frac{\log(1/(\delta\epsilon)}{\epsilon}\right)$, and $m \stackrel{\mathrm{def}}{=} |\mathrm{S}| = O\left(\frac{t+\log(1/\delta)}{\epsilon}\right)$. The role of U, or more precisely of a given partition $(\mathrm{U_1}, \mathrm{U_2})$ of U, is to define a partition of *all* of V(G). In particular, if the test accepts G, then we know that X has a perfect partition, and we shall be interested in the partition of U induced by this perfect partition of X. The role of S is to *test* the partitions of V(G) defined by the partition of U in the following sense. If a certain partition (V_1, V_2) of V(G), defined by a partition (U_1, U_2) of U, is ϵ-bad, then with very high probability there is *no* partition (S_1, S_2) of S such that $(U_1 \cup S_1, U_2 \cup S_2)$ is a perfect partition of X. We next make the above notions more precise.

Given a partition (U_1, U_2) of U we define the following partition (V_1, V_2) of V(G): let $V_1 \stackrel{\rm def}{=} \Gamma(U_2)$, and $V_2 \stackrel{\rm def}{=} V(G) \setminus \Gamma(U_2)$. That is, V_1 is the set of neighbors of U_2, and V_2 all neighbors of U_1 (which are not neighbors of U_2), as well as the rest of the vertices - namely those which do not have a neighbor in U. Note that the partition of U is not relevant to the placement of vertices which have no neighbor in U. Thus, we first ensure that most vertices in V(G) (or most "influential" vertices in V(G)) have a neighbor in U.

Definition 3.1.2 (influential vertices and covering sets): *We say that a vertex $v \in V(G)$ is* influential *if it has degree at least $\frac{\epsilon}{3}N$. Recall that a degree of a vertex is the sum of its in-degree and its out-degree (which is twice its degree in the undirected representation of the graph). We call* U *a* covering *set for* V(G) *if all but at most $\frac{\epsilon}{6}N$ of the influential vertices in* V(G) *have a neighbor in* U *(here each neighbor from which there is one outgoing edge, and one ingoing edge, is counted once).*

Claim 3.1.3 *With probability at least $1 - \delta/2$, a uniformly chosen set* U *of size $t = O\left(\frac{\log(1/(\delta\epsilon))}{\epsilon}\right)$ is a covering set for* V(G).

Proof: For a given influential vertex v, the probability that v does not have any neighbor in a uniformly chosen set U of size t is at most

$$(1 - \epsilon/6)^t \le \exp(-t \cdot \epsilon/6) = O(\delta \cdot \epsilon) . \tag{3.1}$$

Hence the expected number of influential vertices which do not have a neighbor in a random set U is $O(\delta \cdot \epsilon) \cdot N$, and by Markov's inequality (using appropriate constants), the probability that there are more than $\frac{\epsilon}{6}N$ such vertices is less than $\delta/2$. ■

Given a covering set U and a partition of U, we can concentrate on the violating edges between vertices in $\Gamma(U_i)$, for $i = 1, 2$. The reason being that the total number of edges incident to vertices not in $\Gamma(U)$ is small. This motivates the following definition.

Definition 3.1.4 (useful partitions): *Let $U \subset V(G)$. A partition (U_1, U_2) of* U *is called* **ϵ**-useful *(or just* useful*) if*

$$|\{(v, v') \in E(G) \ : \ \exists i \in \{1, 2\} \ \ s.t. \ \ v, v' \in \Gamma(U_i)\}| < \frac{\epsilon}{3} \cdot N^2 . \tag{3.2}$$

Otherwise it is **ϵ**-unuseful.

In other words, a partition of U is unuseful if there are too many violating edges among the neighbors either of U_1 or of U_2 in the corresponding partition defined on V(G). As the following claim shows, if (U_1, U_2) is an unuseful partition, then with high probability we shall see evidence to its unusefulness in the sample S. The evidence is in form of an edge $(v, v') \in S \times S$ between neighbors of vertices in, say, U_1. Let $u \in U_1$ (resp., $u' \in U_1$) be a neighbor of v (resp., v'). In case $u = u'$ there is a triangle in G_X (which means that the test would reject). In case $u \neq u'$ the edge (v, v') forces to place u and u' on opposite sides of any perfect partition of $X = U \cup S$.

Claim 3.1.5 *Let* U *be a set of size* t *and let* $(\mathrm{U}_1, \mathrm{U}_2)$ *be a fixed* ϵ*-unuseful partition of* U. *Then for a uniformly chosen set* $\mathrm{S} \subset \mathrm{V}(G)$ *of cardinality* $m = O\left(\frac{t+\log(1/\delta)}{\epsilon}\right)$,

$$\mathrm{Prob}_{\mathrm{S}}\left[\nexists v, v' \in \mathrm{S}, i \in \{1,2\} \text{ s.t. } (v,v') \in \mathrm{E(G)} \cap (\Gamma(\mathrm{U}_i) \times \Gamma(\mathrm{U}_i))\right] \; < \; \delta \cdot 2^{-(t+1)} \; .$$

The claim says that if $(\mathrm{U}_1, \mathrm{U}_2)$ is an ϵ-unuseful partition of U then with very high probability there exists no partition of S so that the combined partition of $\mathrm{X} = \mathrm{U} \cup \mathrm{S}$ is a perfect partition of the subgraph induced by X.
Proof: If $(\mathrm{U}_1, \mathrm{U}_2)$ is an unuseful partition, then by Eq. (3.2):

$$\mathrm{Prob}_{v,v'}[\exists i \in \{1,2\} \text{ s.t. } (v,v') \in \mathrm{E(G)} \cap (\Gamma(\mathrm{U}_i) \times \Gamma(\mathrm{U}_i))] \geq \epsilon/3 \; . \tag{3.3}$$

Since S can be chosen by drawing $\frac{m}{2}$ independent random pairs of vertices (v, v'),

$$\begin{aligned} \mathrm{Prob}_{\mathrm{S}}[\forall v, v' \in \mathrm{S}, i \in \{1,2\} \; : \; (v,v') \notin \mathrm{E(G)} \cap (\Gamma(\mathrm{U}_i) \times \Gamma(\mathrm{U}_i))] \;&\leq\; (1-\epsilon/3)^{m/2} \\ &=\; \exp\left(-O(t + \log(1/\delta))\right) \\ &<\; \delta \cdot 2^{-(t+1)} \end{aligned}$$

■

As a corollary to Claim 3.1.5, we have

Corollary 3.1.6 *For every set* U *of size* t*, if all partitions of* U *are* ϵ*-unuseful, then with probability at least* $1 - \delta/2$ *there is no perfect partition of* X. *(In such a case* $\mathrm{G_X}$ *is found to be non-bipartite, and the test rejects* G*).*

On the other hand, if U has a useful partition, denoted $(\mathrm{U}_1, \mathrm{U}_2)$, and U is a covering set for V(G), then we have the following lemma.

Lemma 3.1.7 *For every graph* G*, if there exists a covering set* U *of* V(G)*, which has an* ϵ*-useful partition* $(\mathrm{U}_1, \mathrm{U}_2)$*, then* G *is* ϵ*-close to bipartite. In particular, the following partition* $(\mathrm{V}_1, \mathrm{V}_2)$ *of* V(G) *is* ϵ*-good:* $\mathrm{V}_1 \stackrel{\text{def}}{=} \Gamma(U_2)$, $V_2 \stackrel{\text{def}}{=} \mathrm{V(G)} \setminus \Gamma(U_2)$.

Proof: Let us count the number of violating edges with respect to the partition $(\mathrm{V}_1, \mathrm{V}_2)$:

- Edges incident to non-influential vertices: there are at most N such vertices and by definition each has at most $\frac{\epsilon}{3}N$ incident edges, giving a total of $\frac{\epsilon}{3}N^2$.
- Edges incident to influential vertices which do not have neighbors in U: there are at most $\frac{\epsilon}{6}N$ such vertices and each has at most $2N$ incident edges, totaling to $\frac{\epsilon}{3}N^2$.
- Violating edges which are incident to neighbors of U. We consider two cases
  - Edges of the form $(v, v') \in \mathrm{E}(G) \cap (\mathrm{V}_1 \times \mathrm{V}_1)$. Since $\mathrm{V}_1 = \Gamma(\mathrm{U}_2)$, these edge have both end-points in $\Gamma(\mathrm{U}_2)$.
  - Edges of the form $(v, v') \in \mathrm{E}(G) \cap (\mathrm{V}_2 \times \mathrm{V}_2)$. By definition of V_2, both v and v' are not in $\Gamma(\mathrm{U}_2)$. However, since $u, u' \in \Gamma(\mathrm{U})$ it follows that these edges have both end-points in $\Gamma(\mathrm{U}_1)$.

  By the ϵ-usefulness of $(\mathrm{U}_1, \mathrm{U}_2)$, there are at most $\frac{\epsilon}{3}N^2$ such vertices.

Thus we have a total of at most ϵN^2 violating edges, as required. ■

Combining Lemma 3.1.7 with Claim 3.1.3 and Corollary 3.1.6, we complete the proof of Theorem 3.1 as follows. If G is accepted with probability greater than δ, then, by Claim 3.1.3, the probability that G is accepted *and* U is a covering set is greater than $\delta/2$. Thus, there exists a covering set $U \in V(G)$, such that if U is chosen, then G is accepted with probability greater than $\delta/2$ (where here the probability is taken only over the choice of S). But in such a case it follows from Corollary 3.1.6 that (the covering set) U must have a useful partition, and we can apply Lemma 3.1.7 to show that G must be ϵ-close to bipartite.

Finally, let G be a bipartite graph, and let (V_1, V_2) be a perfect partition of V(G). Then, for every covering set U of $V(G)$ (where such a set is chosen with probability at least $1 - \delta/2$), there exists a partition (U_1, U_2) of U, such that $U_1 \in V_1$ and $U_2 \in V_2$. Thus, necessarily, (U_1, U_2) is a useful partition of U which by Lemma 3.1.7 defines an ϵ-good partition of V(G). Furthermore, given such a partition of a covering set U, for *every* set S there exists a perfect partition of $U \cup S$, of the form $(U_1 \cup S_1, U_2 \cup S_2)$. On the other hand, by Claim 3.1.5, for any set U, with probability at least $1 - \delta/2$ over the choice of S, there will be no perfect partition of $U \cup S$ which induces an unuseful partition of U. Therefore, with probability at least $1 - \delta$ over the choice of U and S, the testing algorithm (using BFS) will find a perfect partition of $U \cup S$ that induces a useful partition of U, which can then be used to construct an ϵ-good partition of $V(G)$ (as defined in Lemma 3.1.7) in time $O(|U| \cdot N)$.

3.1.2 Remarks

Improving the query complexity. We can save a factor of $1/\epsilon$ in the query complexity of the testing algorithm. This is done simply by observing that we do not need to perform edge-queries for all pairs of vertices in S. Instead we can choose S to be a uniformly distributed random sample of *$m/2$ pairs of vertices*. We then need only to query which of these $\frac{m}{2} = O\left(\frac{\log(1/\delta\epsilon)}{\epsilon^2}\right)$ pairs are edges, as well as query all $m \cdot t = O\left(\frac{\log^2(1/\delta\epsilon)}{\epsilon^3}\right)$ pairs (u, v) where $u \in U$ and $v \in S$. Note that the proof of Theorem 3.1 does not refer to any edges between vertices in S, except for the $\frac{m}{2}$ pairs mentioned above (which are used for establishing Claim 3.1.5).

Impossibility of testing without queries. A natural question that may arise is if edge-queries are really necessary for testing bipartiteness, or perhaps it might be possible to test this property from a random labeled sample (of pairs of vertices) alone. We show that queries are in fact necessary in the sense that any testing algorithm which uses only a random sample must have very large sample complexity. More precisely:

Proposition 3.2 *Any property testing algorithm for the class of bipartite graphs which observes only a random labeled sample, must have sample complexity $\Omega(\sqrt{N})$.*

Proof: Consider the following two classes of graphs. $\mathcal{G}^1$ is the class of all complete bipartite graphs G in which both sides are of equal cardinality. That is, $V(G) = V_1 \dot\cup V_2$, $|V_1| = |V_2| = N/2$, and $E(G) = \{(v_1, v_2) : v_1 \in V_i, v_2 \in V_j,\ i \neq j\}$. $\mathcal{G}^2$ is the class of graphs which consist of two disjoint cliques of size $N/2$. That is, $V(G) = V_1 \cup V_2$, $|V_1| = |V_2| = N/2$, and $E(G) = \{(v, v') : v, v' \in V_1 \text{ or } v, v' \in V_2\}$. Clearly, all graphs in $\mathcal{G}^2$ are 1/2-far from bipartite. Note that all graphs in both classes have the same edge density, since every vertex has degree N. What we would essentially like to show is that if the edge-labeled sample is not large enough then a hypothetical property testing algorithm cannot distinguish between random samples labeled by graphs in $\mathcal{G}^1$ and random sample labeled by graphs in $\mathcal{G}^2$.

For simplicity, let us fix δ to be 1/4. Then, by definition, a property testing algorithm for the class of bipartite graphs should accept each $G \in \mathcal{G}^1$ with probability at least 3/4, and should accept each $G \in \mathcal{G}^2$ with probability less than 1/4. Therefore, the difference in acceptance probability between an arbitrary $G \in \mathcal{G}^1$ and an arbitrary $G \in \mathcal{G}^2$ must be greater than 1/2. Since the above should be true for any pair of graphs taken from the two classes, it should hold for a random pair of graphs chosen from the two classes. Suppose we first draw an unlabeled random sample of m pairs of vertices, and then label it by a graph G chosen randomly either from the class $\mathcal{G}^1$ or from the class $\mathcal{G}^2$. Assume first that the sample is such that no vertex appears in more than one pair in the sample. Then, regardless of whether G was chosen uniformly in $\mathcal{G}^1$ or in $\mathcal{G}^2$, each of the 2^m possible labeling of the sample has equal probability. If the sample does include two pairs that share a vertex, then we cannot make such a claim. Let us say in this case that the sample is *informative*. However, the probability that a random sample of size m is informative is at most $\binom{m}{2} \cdot \frac{2}{N} < \frac{m^2}{N}$. By the argument made above on non-informative samples, the difference between the acceptance probability of a random graph in $\mathcal{G}^1$ and the acceptance probability of a random graph in $\mathcal{G}^2$ is at most the probability that a random sample is informative. But in order that this probability be greater than 1/2, the size of the random sample must be $\Omega(\sqrt{N})$, ■

3.2 Testing k-Colorability

In this subsection we present an algorithm for testing the k-Colorability property for any given k. Namely, we are interested in determining if the vertices of a graph G can be colored by k colors so that no two adjacent vertices are colored by the same color, or if any k-partition of the graph has at least ϵN^2 violating edges (i.e. edges between pairs of vertices which belong to the same side of the partition).

The test itself is analogous to the bipartite test described in the previous section: We sample from the vertices of the graph, query all pairs of vertices in the sample to find which are edges in G, and check if the induced subgraph is k-Colorable. The edge-query complexity of the algorithm is polynomial in $1/\epsilon$, $\log(1/\delta)$ and k. In lack of efficient algorithms for k-Colorability, for $k \geq 3$, we use the obvious exponential-time algorithm on the induced subgraph (which is typically small). Note that the number of queries made is larger than in the Bipartite Tester (i.e., by a factor of $\tilde{O}(k^4/\epsilon^4)$).

k-Colorability Testing Algorithm

1. Choose uniformly $O\left(\frac{k^2(\log(k/\delta)}{\epsilon^3}\right)$ vertices. Let the set of vertices chosen be denoted by X.
2. For every pair of vertices $v_1, v_2 \in X$, query if $(v_1, v_2) \in E(G)$. Let G_X be the induced subgraph.
3. If G_X is k-Colorable then output *accept*, otherwise output *reject*.

Similarly to the bipartite case, we define violating edges and good k-partitions.[2]

Definition 3.2.1 (violating edges and good k-partitions): *We say that an edge $(u, v) \in E(G)$ is a* violating *edge with respect to a k-partition $\pi : V(G) \to [k]$ if $\pi(u) = \pi(v)$. We shall say that a*

[2] k-partitions are associated with mappings of the vertex set into the canonical k-element set $[k]$. The partition associated with $\pi : V(G) \to [k]$ is $(V_1 \overset{\text{def}}{=} \pi^{-1}(1), \ldots, V_k \overset{\text{def}}{=} \pi^{-1}(k))$. We shall use the mapping notation π, and the explicit partition notation $(V_1, \ldots, V_k)$, interchangibly.

k-partition is ϵ-good *if it has at most ϵN^2 violating edges (otherwise it is* ϵ-bad*). The partition is* perfect *if it has no violating edges.*

Theorem 3.3 *The* k-Colorability Testing Algorithm *is a property testing algorithm for the class of k-Colorable graphs whose edge-query complexity is $O\left(\frac{k^4(\log^2(k/\delta))}{\epsilon^6}\right)$ and whose running time is* $\exp\left(O\left(\frac{k^2(\log(k/\delta))}{\epsilon^3}\right)\right)$. *If the tested graph* G *is k-Colorable, then it is accepted with probability* 1, *and with probability at least $1-\delta$ (over the choice of the sampled vertices), it is possible to construct an ϵ-good k-partition of* V(G) *in time $O\left(\frac{k^2(\log(k/\delta))}{\epsilon^3}\cdot N\right)$.*

Proof: If G is k-Colorable then every subgraph of G is k-Colorable, and hence G will always be accepted. As in the bipartite case, the crux of the proof is to show that every G which is ϵ-far from the class of k-Colorable graphs, denoted $\mathcal{G}_k$, is rejected with probability at least $1-\delta$. We establish this claim by proving its counter-positive. Namely, that every G which is accepted with probability greater than δ, must be ϵ-close to $\mathcal{G}_k$. This is done by giving a (constructive) proof of the existence of an ϵ-good k-partition of V(G). Hence, in case G $\in \mathcal{G}_k$, we also get an efficient probabilistic procedure for finding an ϵ-good k-partition of V(G). Note that if the test rejects G then we have a certificate that G $\notin \mathcal{G}_k$, in form of the (small) subgraph induced by X which is not k-colorable.

We view the set of sampled vertices X as a union of two disjoint sets U and S, where U is a union of ℓ (disjoint) sets $\mathrm{U}^1,\ldots,\mathrm{U}^\ell$, each of size m. The size of S is m as well, where $m = O((\ell\log(k/\delta))/\epsilon)$ and $\ell = 4k/\epsilon$. The roles of U and S are analogous to their roles in the bipartite case. The set U (or rather a k-partition of U) is used to define a k-partition of V(G). The set S ensures that with high probability, the k-partition of U which is induced by the perfect k-partition of X = U $\cup$ S, defines an ϵ-good partition of V(G).

In order to define a k-partition of V(G) given a k-partition of U, we first introduce the notion of a *clustering* of the vertices in V(G) with respect to this partition of U. More precisely, we define the clustering based on the k-partition of a subset $\mathrm{U}' \subset \mathrm{U}$, where this partition, denoted $(\mathrm{U}'_1,\ldots,\mathrm{U}'_k)$, is the one induced by the k-partition of U. The clustering is defined so that vertices in the same cluster have neighbors on the same sides of the partition of U$'$. For every $A \subseteq [k]$, the A-cluster, denoted C_A, contains all vertices in V(G) which have neighbors in U'_i for every $i \in A$ (and do not have vertices in the other U'_i's). The clusters impose restrictions on possible extensions of the partition of U$'$ to partitions $(\mathrm{V}_1,\ldots,\mathrm{V}_k)$ of all V(G), which do not have violating edges incident to vertices in U$'$. Namely, vertices in C_A should not be placed in any V_i such that $i \in A$. As a special case, $C_\emptyset$ is the set of vertices that do not have any neighbors in U$'$ (and hence can be put on any side of the partition). In the other extreme, $C_{[k]}$ is the set of vertices that in any extension of the partition of U$'$ will cause violations. For each i, the vertices in $C_{[k]\setminus\{i\}}$ are *forced* to be put in V_i, and thus are easy to handle. In the bipartite case we focused on the clusters $C_{\{1\}}$ and $C_{\{2\}}$, where vertices in $C_{\{i\}}$ were forced to the side opposite to i. (The cluster $C_\emptyset$ was explicitly shown to be unimportant and the cluster $C_{[2]}$ was dealt with implicitly.) In the case of k-coloring the situation is more complex. In particular, the clusters C_A where $|A| < k-1$ do not force a placement of vertices.

Definition 3.2.2 (clusters): *Let* U$'$ *be a set of vertices, and let π' be a perfect k-partition of* U$'$. *Define* $\mathrm{U}'_i \stackrel{\rm def}{=} \{v \in \mathrm{U} : \pi'(v) = i\}$. *For each subset $A \subseteq [k]$ we define the A-*cluster with respect to π' *as follows:*

$$C_A \stackrel{\rm def}{=} \left(\bigcap_{i\in A}\Gamma(\mathrm{U}'_i)\right) \setminus \left(\bigcup_{i\notin A}\Gamma(\mathrm{U}'_i)\right) . \tag{3.4}$$

The relevance of the above clusters becomes clear given the following definitions of extending and consistent partitions.

Definition 3.2.3 (consistent extensions): *Let* U' *and* π' *be as above. We say that a* k*-partition* π *of* $\mathrm{V(G)}$ extends *a* k*-partition* π' *of* U' *if* $\pi(u) = \pi'(u)$ *for every* $u \in \mathrm{U}'$*. An extended partition* π *is* consistent *with* π' *if* $\pi(v) \neq \pi'(u)$ *for every* $u \in \mathrm{U}'$ *and* $v \in \Gamma(u) \setminus C_{[k]}$*, where* $C_{[k]}$ *is the* $[k]$*-cluster w.r.t* π'*.*

Thus, each vertex v in the cluster C_A (w.r.t π' defined on U') is forced to satisfy $\pi(v) \in \bar{A} \stackrel{\mathrm{def}}{=} [k] \setminus A$, for every k-partition π which extends π' in a consistent manner. There are no restrictions regarding vertices in $C_\emptyset$ and vertices in $C_{[k]}$ (the latter is guaranteed artificially in the definition and the consequences will have to be treated separately). For $v \in C_{[k]-\{i\}}$ the consistency condition forces $\pi(v) = i$, but unlike the bipartite case we cannot ignore the A-clusters with $|A| < k - 1$.

We now focus on the main problem of the analysis. Given a k-partition of U, what is a good way to define a k-partition of $\mathrm{V}(G)$? Our main idea is to claim that with high probability the set U contains a subset U' so that the clusters with respect to the induced k-partition of U' determine whatever needs to be determined. That is, if these clusters allow to place some vertex on a certain side of the partition, then doing so does not introduce too many violating edges. The first step in implementing this idea is the notion of a *restricting* vertex.

Definition 3.2.4 (restricting vertex): *A pair* (v, i)*, where* $v \in C_A$*,* $A \neq [k]$ *and* $i \in \bar{A}$ *is said to be* restricting *with respect to a* k*-partition* π' *of* U' *if* v *has at least* $\frac{\epsilon}{4}N$ *neighbors in* $\cup_{B: i \notin B} C_B$*. Otherwise,* (v, i) *is* non-restricting*. A vertex* $v \in C_A$*,* $A \neq [k]$*, is* restricting *with respect to* π' *if for every* $i \in \bar{A}$*, the pair* (v, i) *is restricting. Otherwise,* v *is* non-restricting*. As always, the clusters are with respect to* π'*.*

Thus, a vertex $v \in C_A$ is restricting if for every $i \in \bar{A}$, adding v to U'_i (and thus to U') will cause may of its neighbors to move to a cluster corresponding to a bigger subset. That is, v's neighbors in the B-cluster (w.r.t $(\mathrm{U}'_1, \ldots, \mathrm{U}'_k)$) move to the $(B \cup \{i\})$-cluster (w.r.t $(\mathrm{U}'_1, \ldots, \mathrm{U}'_i \cup \{v\}, \ldots, \mathrm{U}'_k)$).

Given a perfect k-partition of U, we construct U' in steps starting with the empty set. At step j we add to U' a vertex $u \in \mathrm{U}^j$ (recall that $\mathrm{U} = \mathrm{U}^1 \cup \ldots \cup \mathrm{U}^\ell$), which is a restricting vertex with respect to the k-partition of the current set U'. If no such vertex exists, the procedure terminates. When the procedure terminates (and as we shall see it must terminate after at most ℓ steps), we will be able to define, based on the k-partition of the final U', an ϵ-good k-partition of $\mathrm{V(G)}$. The procedure defined below is viewed at this point as a mental experiment. Namely, it is provided in order to show that with high probability there exists a subset U' of U with certain desired properties (which we later exploit). We later discuss how to implement this procedure when we are actually interested in choosing U' for the purposes of partitioning all of $\mathrm{V(G)}$ efficiently.

Restriction Procedure (Construction of U')
Input: a perfect k-partition of $\mathrm{U} = \mathrm{U}^1, \ldots, \mathrm{U}^\ell$.

1. $\mathrm{U}' \leftarrow \emptyset$.

2. For $j = 1, 2, \ldots$ do the following. Consider the current set U' and its partition π' (induced by the perfect k-partition of U).

   - If there are less than $(\epsilon/8)N$ restricting vertices with respect to π' then halt and output U'.
   - If there are at least $(\epsilon/8)N$ restricting vertices but there is no restricting vertex in U^j, then halt and output error.

- Otherwise (there is a restricting vertex in U^j), add the first (by any fixed order) restricting vertex to U'.

Claim 3.2.5 *For every* U *and a perfect k-partition of* U*, after at most $\ell = 4k/\epsilon$ iterations, the Restriction Procedure halts and outputs either* U' *or* error.

Proof: If the procedure has not halted and output error, then in each iteration at least $(\epsilon/4)N$ vertices in V(G) move to a cluster corresponding to a bigger subset. Since each vertex can be moved at most k times (before it belongs to $C_{[k]}$), the maximal number of iterations before the procedure halts is $4k/\epsilon$. ■

Before we show how U' can be used to define a k-partition π of V(G), we need to ensure that with high probability, the restriction procedure in fact outputs a set U' and not error. To this end we first extend the notion of *covering set* to the context of k-coloring. Though the notion here may seem somewhat remote from the one used in the bipartite case, it can be shown that the two are related.

Definition 3.2.6 (covering sets – for k-coloring): *We say that* U *is a* covering set *for* V(G)*, if for* every *perfect k-partition of* U*, the Restriction Procedure, given this partition as input, halts with an output* $\mathrm{U}' \subset \mathrm{U}$ *(rather than an error message).*

In other words, U is such that for every perfect k-partition of U and for each of the at most ℓ iterations of the procedure, if there exist at least $(\epsilon/8)N$ restricting vertices with respect to the current partition of U', then U^j will include at least one such restricting vertex.

Lemma 3.2.7 *With probability at least $1-\delta/2$, a uniformly chosen set* U *of size $\ell \cdot m = O\left(\frac{k^2\log(k/\delta)}{\epsilon^3}\right)$ is a covering set.*

Proof: Let us first consider a single iteration of the Restriction Procedure. If there are at least $(\epsilon/8)N$ restricting vertices with respect to the partition π' of the current U', then the probability that in a uniformly chosen sample of size m ($\stackrel{\rm def}{=} |U^j|$) there will be no restricting vertex with respect to π', is at most $(1-(\epsilon/8))^m$. By our choice of $m = O((\ell\log(k/\delta))/\epsilon)$, the latter is bounded by $\frac{\delta}{2}k^{-\ell}$. Thus, the lemma reduces to proving that, for every j, the number of possible pairs (U', π') that we need to consider for the $j^{\rm th}$ iteration is at most k^{j-1}.

We shall prove the above claim inductively. Let the set U' in iteration j (before adding a new restricting vertex to it) be denoted by $\mathrm{U}'(j)$, and let its partition be denoted by π'_j. For the base case, $j = 1$, the set $\mathrm{U}'(1)$ is empty and the claim trivially holds. Assuming the claim holds for j, we now prove it for $j+1$. In the $j^{\rm th}$ iteration, for each of the possible pairs $(\mathrm{U}'(j), \pi'_j)$, such that there exist at least $(\epsilon/8)N$ restricting vertices with respect to π'_j, the vertex $u_j \in \mathrm{U}^j$ which is the *first* restricting vertex in U^j, is uniquely defined[3]. Hence, for each such pair $(\mathrm{U}'(j), \pi'_j)$, there is a single possible extension $\mathrm{U}'(j+1)$ of $\mathrm{U}'(j)$, namely, $\mathrm{U}'(j+1) = \mathrm{U}'(j) \cup \{u_j\}$. The new partition, π'_{j+1} which extends π'_j can be one of at most k possibilities (depending only on $\pi'_{j+1}(u_j)$). ■

Definition 3.2.8 (closed partitions): *Let* U' *be a set and* π' *a k-partition of it. We call* (U', π') closed *if there are less than $(\epsilon/8)N$ restricting vertices with respect to* π'.

[3]It may be the case that no such restricting vertex exists in U^j, but the probability for this event has been bounded in dealing with the $j^{\rm th}$ iteration. Here we look at the $j^{\rm th}$ iteration only to see which possible pairs could emerge from it effecting the $j+1^{\rm st}$ iteration.

Clearly, if the Restriction Procedure outputs a set U' then this set together with its its (induced) partition are closed. If (U', π') is closed, then most of the vertices in $\mathrm{V(G)}$ are non-restricting. Recall that a non-restricting vertex v, belonging to a cluster C_A, $A \neq [k]$, has the following property. There exists at least one index $i \in \bar{A}$, such that (v, i) is non-restricting. It follows from Definition 3.2.4 that for every consistent extension of π' to π which satisfies $\pi(v) = i$ there are at most $\frac{\epsilon}{2}N$ violating edges incident to v.[4] However, even if v is non-restricting there might be indices $i \in \bar{A}$ such that (v, i) is restricting, and hence there may exist a consistent extensions of π' to π which satisfies $\pi(v) = i$ in which there are more than $\frac{\epsilon}{2}N$ violating edges incident to v. Therefore, we need to define for each vertex its set of *forbidden* indices which will not allow to have $\pi(v) = i$ for a restricting pair (v, i).

Definition 3.2.9 (forbidden sets): *Let (U', π') be closed and consider the clusters with respect to π'. For each $v \in \mathrm{V}(G) \setminus \mathrm{U}'$ we define the* forbidden set of *v, denoted F_v, as the smallest set satisfying*

- *$F_v \supseteq A$, where $v \in C_A$.*
- *For every $i \in \bar{A}$, if v has at least $(\epsilon/4)N$ neighbors in the clusters C_B for which $i \notin B$, then i is in F_v.*

For $u \in \mathrm{U}'$, define $F_u = [k] \setminus \{\pi'(u)\}$.

Lemma 3.2.10 *Let (U', π') be an arbitrary closed pair and F_v's be as in Definition 3.2.9. Then:*

1. *$|\{v : (v \notin C_{[k]}) \wedge (F_v = [k])\}| \leq \frac{\epsilon}{8}N$.*
2. *Let π be any k-partition of $\mathrm{V(G)} \setminus \{v : F_v = [k]\}$ such that $\pi(v) \notin F_v$, for every $v \in \mathrm{V}(G)$. Then, the number of edges $(v, v') \in \mathrm{E(G)}$ for which $\pi(v) = \pi(v')$ is at most $(\epsilon/2)N^2$.*

The lemma can be thought of as saying that any k-partition which respects the forbidden sets is good (i.e., does not have many violating edges). However, the partition applies only to vertices for which the forbidden set is not $[k]$. The first item tells us that there cannot be many such vertices which do not belong to the cluster $C_{[k]}$. We deal with vertices in $C_{[k]}$ at a later stage.

Proof: The first item follows from the closeness of (U', π'). Namely, if $F_v = [k]$ and $v \notin C_{[k]}$ then by the second item of Definition 3.2.9 it follows that for every $i \in \bar{A}$, vertex v has at least $(\epsilon/4)N$ neighbors in clusters C_B such that $i \notin B$. But in this case, it is a restricting vertex with respect to π'. By Definition 3.2.8 as applied to (U', π'), there are at most $(\epsilon/8)N$ such vertices.

For the second item, consider a vertex v such that $\pi(v) = i$. All edges (v, u) and (u, v) such that $u \in C_B$ and $i \in B$ cannot be violating edges since $i \in F_u$ (by the first item in Definition 3.2.9). As for edges (v, u) and (u, v) where $u \in C_B$ and $i \notin B$, vertex v can have at most $(\epsilon/2)N$ such edges (according to the second item in Definition 3.2.9). The total of violating edges is hence at most $(\epsilon/2)N^2$. ■

We next need to show that with high probability over the choice of S, the k-partition π' of U' (induced by the k-partition of $\mathrm{U} \cup \mathrm{S}$) is such that $C_{[k]}$ is small. This implies that all the vertices in $C_{[k]}$ (which were left out of the partition in the previous lemma) can be placed in any side without contributing too many violating edges (which are incident to them).

[4]First note that by definition of a consistent extension no vertex in cluster C_B, where $i \in B$, can have π-value i. Thus, all violated edges incident to v are incident to vertices in clusters C_B so that $i \notin B$. Using the definition of a restricting vertex, we are done.

Definition 3.2.11 (useful k-partitions): *We say that a pair* (U', π') *is* ϵ-useful *if* $|C_{[k]}| < \frac{\epsilon}{8}N$. *Otherwise it is* ϵ-unuseful.

The next claim directly follows from our choice of m and the above definition.

Claim 3.2.12 *Let* U' *be a fixed set of size* ℓ *and* π' *be a fixed* k*-partition of* U' *so that* (U', π') *is* ϵ*-unuseful. Let* S *be a uniformly chosen set of size* m*. Then, with probability at least* $\frac{\delta}{2}k^{-\ell}$*, there exists no perfect* k*-partition of* $\mathrm{U}' \cup \mathrm{S}$ *which extends* π'*.*

By the same argument applied in the proof of Lemma 3.2.7, we have that the number of possible closed pairs (U', π') determined by all possible k-partitions of U is at most k^{ℓ}. Therefore we get the following corollary to the above claim:

Corollary 3.2.13 *If all closed pairs* (U', π') *which are determined by all possible* k*-partitions of* U *are unuseful, then with probability at least* $1-\delta/2$ *over the choice of* S*, there is no perfect* k*-partition of* $\mathrm{X} = \mathrm{U} \cup \mathrm{S}$*.*

We can now wrap up the proof of Theorem 3.3. If G is accepted with probability greater than δ, then by Lemma 3.2.7, the probability that it is accepted and U is a covering set is greater than $\delta/2$. In particular, there must exist at least one covering set U, such that if U is chosen then G is accepted with probability greater than $\delta/2$ (with respect to the choice of S). That is, (with probability greater than $\delta/2$) there exists a perfect partition of $\mathrm{U} \cup \mathrm{S}$. But in such a case (by applying Corollary 3.2.13), there must be a useful closed pair (U', π') (where $\mathrm{U}' \subset \mathrm{U}$). If we now partition $\mathrm{V}(G)$ as described in Lemma 3.2.10, where vertices with forbidden set $[k]$ are placed arbitrarily, then from the two items of Lemma 3.2.10 and the usefulness of (U', π') it follows that there are at most ϵN^2 violating edges with respect to this partition. This completes the main part of the proof and the rest refers to the efficient procedure for finding ϵ-good partitions.

Similarly to the bipartite case, if $\mathrm{G} \in \mathcal{G}_k$, then with probability at least $1-\delta$ (over the choice of U and S), the k-coloring of G_{X} (recall that $\mathrm{X} = \mathrm{U} \cup \mathrm{S}$) is such that the induced (perfect) coloring of U determines a useful pair (U', π') which can be used to partition $\mathrm{V}(G)$. Details are omitted.

Hence, as a final point, we address the question of efficiently implementing the Restricting Procedure (i.e., constructing U') and the definition of forbidding sets. We first observe, that in the Restricting Procedure we do not actually need to determine (in each iteration) if there are more or less than $(\epsilon/8)N$ restricting vertices. Since we know that with high probability U^j contains a restricting vertex if many such vertices exist, we need only scan U^j in search for such a vertex. Note that no harm is done when despite the fact that there are too few restricting vertices in iteration j nevertheless U^j contains one. This is true since the bound on the number of iterations performed by the Restriction Algorithm, is unrelated to the actual number of restricting vertices in each iteration. In order to recognize a restricting vertex, it suffices to uniformly choose a set Y of $\mathrm{poly}(k\log(1/\delta)/\epsilon)$ vertices and examine their neighborhood relation to the vertex and to the current U'. We may sometimes take vertices which are very close to being restricting, but this does not harm us either. A more subtle point is that with respect to this efficient implementation the first restricting vertex in an iteration is not determined by the partition of U. However, once we have chosen the additional sample Y, for each fixed partition of U, the first restricting (or "close to restricting") vertex is determined. Note that whenever a vertex u is added to U', only the neighbors of u might move to different clusters, and hence the process of updating the clusters takes time $O(\ell \cdot N)$. As for implementing the definition of forbidden sets, here each vertex v must sample its neighbors to determine F_v. Wrong decisions in marginal cases (i.e., when the number of neighbors in relevant clusters is approximately $\frac{\epsilon}{2}N$) do not matter here either. Also, non-marginally wrong decisions on few vertices do not matter. Hence the total running time is $\mathrm{poly}(k\log(1/\delta)/\epsilon)\cdot N$. ■ **(Theorem 3.3)**

3.3 Testing Max-Clique

Let $\omega(\mathrm{G})$ denote the size of the largest clique in graph G, and $\mathcal{C}_\rho \stackrel{\rm def}{=} \{\mathrm{G} : \omega(\mathrm{G}) \geq \rho\cdot|\mathrm{V(G)}|\}$ be the set of graphs having cliques of density at least ρ. The main result of this section is:

Theorem 3.4 *There exists a property testing algorithm, $\mathcal{A}$, for the class $\mathcal{C}_\rho$ whose edge-query complexity is $O\left(\frac{\log^2(1/\epsilon\delta)\rho^2}{\epsilon^6}\right)$ and whose running time is $\exp\left(O\left(\frac{\log(1/\epsilon\delta)\rho}{\epsilon^2}\right)\right)$. In particular, $\mathcal{A}$ uniformly selects $O\left(\frac{\log^2(1/\epsilon\delta)\rho^2}{\epsilon^4}\right)$ vertices in G and queries the oracle only on the existence of edges between these vertices. In case $\mathrm{G}\in\mathcal{C}_\rho$, one can also retrieve in time $O\left(\frac{\log^2(1/\epsilon\delta)\rho^2}{\epsilon^4}\right)\cdot|\mathrm{V(G)}|$ a set of $\rho\cdot|\mathrm{V(G)}|$ vertices in G which is almost a clique (in the sense that it lacks at most $\epsilon\cdot|\mathrm{V(G)}|^2$ edges to being a clique).*

Theorem 3.4 is proven by presenting a seemingly unnatural algorithm/tester (see below). However, as a corollary, we observe that "the natural" algorithm, which uniformly selects $\mathrm{poly}(\log(1/\delta)/\epsilon)$ many vertices and accepts iff they induce a subgraph with a clique of density $\rho-\epsilon/2$, is a valid $\mathcal{C}_\rho$-tester as well.

Corollary 3.5 *Let $m=\mathrm{poly}(1/\epsilon)$ and let R be a uniformly selected set of m vertices in $V(\mathrm{G})$. Let $\mathrm{G_R}$ be the subgraph (of G) induced by R. Then,*

- *if* $\mathrm{G}\in\mathcal{C}_\rho$ *then* $\mathrm{Prob_R}[\omega(\mathrm{G_R}) > (\rho-\epsilon/2)\cdot m] > \frac{2}{3}$.
- *if* $\mathrm{dist}(\mathrm{G},\mathcal{C}_\rho) > \epsilon$ *then* $\mathrm{Prob_R}[\omega(\mathrm{G_R}) \leq (\rho-\epsilon/2)\cdot m] > \frac{2}{3}$.

Proof: The first item is quite obvious, provided $m=\Omega(1/\epsilon^2)$. To prove the second item, we use the algorithm $\mathcal{A}$ guaranteed by Theorem 3.4.[5] Suppose that on density parameter $\rho'=\rho-\epsilon/2$, distance parameter $\epsilon'=\epsilon/2$ and confidence parameter $\delta'=1/5$, algorithm $\mathcal{A}$ takes a sample of $s=s(\rho',\epsilon',\delta')$ vertices and let $m=O(s^2)$. Let G be an arbitrary graph so that $\mathrm{dist}(\mathrm{G},\mathcal{C}_\rho)>\epsilon$. Observe that $\mathrm{dist}(\mathrm{G},\mathcal{C}_{\rho'})>\epsilon/2$ (as otherwise there exists $\mathrm{G}'\in\mathcal{C}_{\rho'}$ so that $\mathrm{dist}(\mathrm{G},\mathrm{G}')\leq\epsilon/2$ whereas $\mathrm{dist}(\mathrm{G}',\mathcal{C}_\rho)\leq\rho\epsilon/2$ for all $\mathrm{G}'\in\mathcal{C}_{\rho'}$). Now, on one hand, we know that for any G so that $\mathrm{dist}(\mathrm{G},\mathcal{C}_{\rho'})>\epsilon'$, algorithm $\mathcal{A}$ (with parameters $p'=(\rho',\epsilon',\delta')$) accepts G with probability at most 1/5. That is,

$$\mathrm{Prob}[\mathcal{A}(p',\mathrm{G})=1] \;\leq\; \frac{1}{5}$$

On the other hand, assuming the second item is false (for G), we get

$$\begin{aligned}\mathrm{Prob_R}[\mathcal{A}(p',\mathrm{G_R})=1] &\geq \mathrm{Prob_R}[\omega(\mathrm{G_R})>\rho'\cdot m]\cdot\min_{\mathrm{G}':\omega(\mathrm{G}')>\rho'\cdot m}\{\mathrm{Prob}[\mathcal{A}(p',\mathrm{G}')=1]\}\\ &\geq \frac{1}{3}\cdot\frac{4}{5}>\frac{1}{5}+0.06\end{aligned}$$

[5] Our presentation presupposes that $\mathcal{A}$ is given oracle access to a graph the vertices of which may be an arbitrary subset of [V(G)]. If one insists that $\mathcal{A}$ only tests graphs with $|\mathrm{V(G)}|$ vertices then another auxiliary trick is needed. Instead of providing $\mathcal{A}$ with oracle access to $\mathrm{G_R}$ we provide it with oracle access to a graph in which each vertex of $\mathrm{G_R}$ is duplicated $|\mathrm{V(G)}|/|\mathrm{R}|$ times and edges are duplicated in the natural manner.

However, the statistical difference between a uniformly selected sample of s vertices from V(G) and a uniformly selected sample of s vertices from a set R of size $10 \cdot s^2$, uniformly selected in V(G), is bounded by 0.05 (the difference being due to the difference in collision probabilities: $\binom{s}{2} \cdot \frac{1}{N} > 0$ versus $\binom{s}{2} \cdot \frac{1}{10s^2} < 0.05$). Thus, $\mathrm{Prob}[\mathcal{A}(p', \mathrm{G}) = 1]$ and $\mathrm{Prob}_{\mathrm{R}}[\mathcal{A}(p', \mathrm{G}_{\mathrm{R}}) = 1]$ cannot differ by more than 0.05 and so contradiction follows. ■

The rest of this section is devoted to the presentation and analysis of the algorithm, asserted in Theorem 3.4, which we call the **clique-degree tester**. We start with a motivating discussion. Recall that $N = |\mathrm{V(G)}|$ denotes the number of vertices in G.

3.3.1 The Clique-Degree Tester

Our first idea is to select at random a small sample U of V(G) and to consider all subsets U$'$ of size $\frac{\rho}{2} \cdot |\mathrm{U}|$ of U where $|\mathrm{U}| = \mathrm{poly}(1/\epsilon)$. For each U$'$ let T(U$'$) be the set of all vertices which neighbor every vertex in U$'$ (i.e., $\mathrm{T(U')} = \cap_{u \in \mathrm{U'}} \Gamma(u)$). In the subgraph induced by T(U$'$), consider the set Y(U$'$) of ρN vertices with highest degree in the induced subgraph. Clearly, if G is ϵ-far from $\mathcal{C}_\rho$, then Y(U$'$) misses at least ϵN^2 edges to being a clique (for every choice of U and U$'$). On the other hand, we show that if G has a clique C of size ρN then, with high probability over the choice of U, there exists a subset $\mathrm{U'} \subset \mathrm{U}$ such that Y(U$'$) misses at most $(\epsilon/3)N^2$ to being a clique (in particular, $\mathrm{U'} \subseteq \mathrm{C} \cap \mathrm{U}$ will do).

Assume that for any fixed U$'$ we could sample the vertices in Y(U$'$) and perform edge queries on pairs of vertices in this sample. Then, a sample of $O(t/\epsilon^2)$ vertices (where $t = |\mathrm{U}|$) suffices for approximating the edge density in Y(U$'$) to within an $\epsilon/3$ fraction with probability $1 - O(2^{-t})$. In particular a sample can distinguish between a set Y(U$'$) which is far from being a clique and a set Y(U$'$) which is almost a clique. The point is that we need only consider $\binom{|\mathrm{U}|}{|\mathrm{U'}|} < 2^t$ possible sets Y(U$'$), where t is only a polynomial in $1/\epsilon$.

The only problem which remains is how to sample from Y(U$'$). Certainly, we can sample $\mathrm{T} = \mathrm{T(U')}$, by sampling V(G) and testing membership in T, but how do we decide which vertex is among those of highest degree? The first idea is to estimate the degrees of vertices in T using an additional sample, denoted W. Thus, instead of considering the ρN vertices of highest degree in T, we consider the ρN vertices in T having the most neighbors in $\mathrm{T} \cap \mathrm{W}$. The second idea is that we can sample T, order vertices in this sample according to the number of neighbors in $\mathrm{T} \cap \mathrm{W}$, and take the ρ fraction with the most such neighbors.

The resulting *clique-degree tester* is described in Figure 3.1. The sets W$'$ and S$'$, defined in Steps (1) and (2), correspond to samples of the set $\mathrm{T(U')} = \cap_{u \in \mathrm{U'}} \Gamma(u)$. The set W$'$ is used, in Step (3), to approximate the degrees of vertices in the subgraph induced by T(U$'$). The set $\widehat{\mathrm{C}} \subseteq \mathrm{S'}$, defined in Step (3), will be shown to be a good sample of the vertices with the highest such degrees. Condition (b) in Step (4) is more query-efficient and easier to analyze than the obvious alternative of checking all pairs of vertices in $\widehat{\mathrm{C}}$.

Organization of the proof of Theorem 3.4. We start by analyzing what happens, in case $\mathrm{G} \in \mathcal{C}_\rho$, when one takes the ρN vertices of about the highest degree in T(U$'$), where U is uniformly chosen and U$'$ is in its intersection with a fixed clique of size ρN in G. Loosely speaking, we show that these vertices are likely to induce a subgraph which is close to a clique. We then turn to the analysis of the actual clique-degree tester. This analysis also yields the algorithm for finding an approximate ρN-clique in G (as required in the last sentence of Theorem 3.4). Throughout the analysis we assume that $\epsilon < \rho^2$ (as otherwise $\mathrm{dist}(G, \mathcal{C}_\rho) \leq \rho^2 \leq \epsilon$ for every graph G).

Clique-Degree Tester for G

Let $\epsilon_1 = \Theta(\epsilon/\rho)$, $t = \Theta\left(\frac{\log(1/(\epsilon\delta))\cdot\rho}{\epsilon^2}\right)$, $r = \Theta(\frac{\log(1/\epsilon\delta)\cdot\rho}{\epsilon^2})$, *and* $m = \Theta(\frac{t+\log(1/\delta)}{\epsilon^2})$.
Uniformly select three independent samples, U, W *and* $\mathrm{S} = \{s_1, ..., s_m\}$, *of sizes* t, r *and* m, *respectively.*
For each $\mathrm{U}' \subset \mathrm{U}$ *of cardinality* $\frac{\rho}{2}\cdot t$, *perform the following steps:*

1. *Let* $\mathrm{W}' \subseteq \mathrm{W}$ *be the subset of vertices which neighbor all vertices in* U' (*i.e.,* $\mathrm{W}' \stackrel{\mathrm{def}}{=} \{v \in \mathrm{W} : \Gamma(v) \supseteq \mathrm{U}'\} = \mathrm{W} \cap (\bigcap_{u\in\mathrm{U}'}\Gamma(u))$).
2. *Let* $\mathrm{S}' \subseteq \mathrm{S}$ *be the subset of vertices which neighbor all vertices in* U' (*i.e.,* $\mathrm{S}' = \mathrm{S} \cap (\bigcap_{u\in\mathrm{U}'}\Gamma(u))$).
3. *For each* $v \in \mathrm{S}'$, *compute* $\hat{d}(v) \stackrel{\mathrm{def}}{=} 2\cdot|\Gamma(v)\cap\mathrm{W}'|$. *Let* $\widehat{\mathrm{C}} \subseteq \mathrm{S}'$ *be a set of* ρm *vertices of the highest* $\hat{d}(\cdot)$ *value in* S'. *(Ties are broken by lexicographic order, and in case* $|\mathrm{S}'| < \rho m$ *we let* $\widehat{\mathrm{C}} \stackrel{\mathrm{def}}{=} \mathrm{S}'$.*)*
4. *If the following two conditions hold then* accept *and halt.*

   Condition (a): $|\widehat{\mathrm{C}}| \geq (\rho - \epsilon_1)m$.

   Condition (b): $|\{i : (s_{2i-1}, s_{2i}) \in (\widehat{\mathrm{C}}\times\widehat{\mathrm{C}}) \setminus \mathrm{E}(\mathrm{G})\}| \leq \frac{2\epsilon}{3}\cdot\frac{m}{2}$.

If none of these iterations made the algorithm accept G *then it halts and* rejects G.

Figure 3.1: Clique-Degree Tester

3.3.2 Yes Instances: Preliminaries

Let C be a clique of size ρN in G (where $N = |\mathrm{V(G)}|$). We say that a set $\mathrm{U}' \subset \mathrm{C} \subset \mathrm{V(G)}$ is ϵ_2-representative (w.r.t., C) if for all but $\epsilon_2 N$ of the vertices, $v \in \mathrm{V(G)}$,

$$\text{if } |\Gamma(v)\cap\mathrm{C}| < (\rho-\epsilon_2)N \text{ then } \Gamma(v)\cap\mathrm{U}' \neq \mathrm{U}' \qquad (3.5)$$

For every $v \in \mathrm{C}$ the above condition holds for all $\epsilon_2 \geq 0$ (since $\Gamma(v) \supseteq \mathrm{C} \supset \mathrm{U}'$).

Lemma 3.3.1 *Let* $t = \Omega(\frac{\log(1/(\epsilon_2\delta))}{\epsilon_2\rho})$. *Let* U *be a uniformly chosen set of* t *vertices in* G. *Then, with probability at least* $1-\delta/2$, *the set* U *contains an* ϵ_2*-representative subset of size* $\frac{\rho}{2}t$.

Proof: Using a multiplicative Chernoff Bound we obtain that $|\mathrm{U}\cap\mathrm{C}| \geq \frac{1}{2}\rho t$ with probability at least $1-\exp(-\Omega(\rho t)) > 1-\delta/4$. Let us now consider a uniformly selected subset $\mathrm{U}' \subset \mathrm{C}$ of cardinality $t' \stackrel{\mathrm{def}}{=} \frac{\rho}{2}t$. Then, by Chernoff Bound, we have for each $v \in \mathrm{V(G)}\setminus\mathrm{C}$

$$\begin{aligned}\mathrm{Prob}_{\mathrm{U}'}[\text{Eq. (3.5) does not hold for } v] &= (1-\epsilon_2)^{t'} \\ &< \frac{\delta\epsilon_2}{4}\end{aligned}$$

where the last inequality is due to $t' > \rho t/2$ and the hypothesis regarding t. Thus, the expected number of vertices which violate Eq. (3.5) is bounded by $\frac{\delta}{4}\cdot\epsilon_2 N$. Applying Markov's Inequality we conclude that with probability at least $1-\delta/4$ there are at most $\epsilon_2 N$ vertices which violate Eq. (3.5). The lemma follows. ■

NOTATION: Let C_N^ρ denote the class of N-vertex graphs consisting of a clique of size ρN and $(1-\rho)\cdot N$ isolated vertices. In the sequel, we denote by $\mathrm{dist}(\mathrm{G}', C_N^\rho)$ the relative distance (as a fraction of N^2) between a graph G′ and C_N^ρ. In case G′ contains less than N vertices we augment it by $N - |\mathrm{V(G')}|$ isolated vertices. In all cases $|\mathrm{V(G')}| \leq N$. With slight abuse of notation, for a set $\mathrm{X} \subseteq \mathrm{V(G)}$, we let $\mathrm{dist}(\mathrm{X}, C_N^\rho)$ denote the relative distance between the subgraph of G induces by X and C_N^ρ.

Lemma 3.3.2 *Let* $\epsilon_1 < \epsilon/\rho$ *and* $\epsilon_2 = \epsilon_1^2$. *Let* U' *be* ϵ_2*-representative* (w.r.t., C) *and* $\mathrm{T} \stackrel{\mathrm{def}}{=} \cap_{u\in\mathrm{U}'}\Gamma(u)$. *Let* α *be such that* αN *is the degree of the* $(\rho-\epsilon_1)\cdot N^{\mathrm{th}}$ *vertex of highest degree in* T. (*We stress that we consider degrees in the subgraph of* G *induced by* T.) *Suppose that* $\widetilde{\mathrm{C}} \subseteq \mathrm{T}$ *has size* $\rho\cdot N$ *and contains at least* $(\rho-3\epsilon_1)\cdot N$ *vertices of degree at least* $(\alpha-2\epsilon_1)\cdot N$ (*in the subgraph induced by* T). *Then,* $\widetilde{\mathrm{C}}$ *satisfies* $\mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho) = 14\epsilon_1\rho$.

Proof: Clearly $\mathrm{T} \supseteq \mathrm{C}$ (as U' is a subset of the clique C). Let $\mathrm{H} \stackrel{\mathrm{def}}{=} \{v \in \mathrm{V(G)}\setminus\mathrm{C} : |\Gamma(v)\cap\mathrm{C}| \geq (\rho-\epsilon_2)\cdot N\}$ be the set of vertices outside of C having many neighbors in C. Let $\mathrm{R} \stackrel{\mathrm{def}}{=} \mathrm{T}\setminus(\mathrm{C}\cup\mathrm{H})$ (i.e., the rest of T). Since U' is ϵ_2-representative, it follows that $|\mathrm{R}| < \epsilon_2 N$ (since a vertex not in $\mathrm{C}\cup\mathrm{H}$ may enter T only if it neighbors all vertices in U' whereas it neighbors less than $(\rho-\epsilon_2)N$ vertices in C). Denoting by $\deg_{\mathrm{T}}(v)$ the degree of vertex v in the subgraph induced by T we get

$$\sum_{v\in\mathrm{C}} \deg_{\mathrm{T}}(v) \;\geq\; |\mathrm{C}|^2 + 2\cdot|\mathrm{H}|\cdot(\rho-\epsilon_2)\cdot N \;\geq\; \rho N\cdot\left(\rho N + 2\cdot|\mathrm{H}| - \frac{2\epsilon_2}{\rho}N\right)$$

On the other hand, the maximum value $\deg_{\mathrm{T}}(v)$ for $v\in\mathrm{C}$ is bounded by $2|\mathrm{T}| = 2(|\mathrm{C}|+|\mathrm{H}|+|\mathrm{R}|) \leq 2(\rho N + |\mathrm{H}| + \epsilon_2 N)$. Applying Markov's Inequality, we obtain, for every γ,

$$\left|\left\{v\in\mathrm{C} : \deg_{\mathrm{T}}(v) < \left(\rho N + 2|\mathrm{H}| - \frac{2\epsilon_2}{\rho}N\right) - \gamma\left(\rho + 2\epsilon_2 + \frac{2\epsilon_2}{\rho}\right)\cdot N\right\}\right| < \frac{|\mathrm{C}|}{\gamma}$$

Setting $\gamma = \rho/\epsilon_1$, we get $|\mathrm{C}|/\gamma = \epsilon_1 N$ and $\gamma(\rho + 2\epsilon_2 + \frac{2\epsilon_2}{\rho}) \leq 5\gamma\rho = 5\epsilon_1$. The latter follows from $\epsilon_2 \leq \epsilon_2/\rho$ and $\epsilon_2/\rho = \epsilon_1^2/\rho < \epsilon^2/\rho^3 < \rho$. Thus, at least $(\rho-\epsilon_1)\cdot N$ vertices in C have degree (in the subgraph induced by T) of at least $\rho N + 2|\mathrm{H}| - \frac{2\epsilon_2}{\rho} - 5\epsilon_1 N$. Since $\frac{2\epsilon_2}{\rho} = \frac{2\epsilon_1^2}{\rho}$, $\epsilon_1 < \frac{\epsilon}{\rho}$, and $\epsilon < \rho^2$, we have that α as defined in the lemma satisfies

$$\alpha \;\geq\; \rho + \frac{2|\mathrm{H}|}{N} - 7\epsilon_1 \;=\; \rho + \frac{2(|\mathrm{T}| - (|\mathrm{R}| + |C|))}{N} - 7\epsilon_1 \;\geq\; \frac{2|\mathrm{T}|}{N} - \rho - 9\epsilon_1 \tag{3.6}$$

By the lemma's hypothesis, we have $|\widetilde{\mathrm{C}}| = \rho\cdot N$. Also, denoting by $\mathrm{Z}\subseteq\widetilde{\mathrm{C}}$ the set of vertices v for which $\deg_{\mathrm{T}}(v) \geq (\alpha-2\epsilon_1)\cdot N$, we have by the lemma's hypothesis $|\mathrm{Z}| \geq (\rho-3\epsilon_1)\cdot N$. By Eq. (3.6) we also have, for each $v\in\mathrm{Z}$,

$$\begin{aligned}
\deg_{\widetilde{\mathrm{C}}}(v) &\geq \deg_{\mathrm{T}}(v) - 2\cdot|\mathrm{T}\setminus\widetilde{\mathrm{C}}| \\
&\geq [(2|\mathrm{T}| - \rho N - 9\epsilon_1 N) - 2\epsilon_1 N] - 2\,[|\mathrm{T}| - \rho\cdot N] \\
&= (\rho - 11\epsilon_1)\cdot N
\end{aligned}$$

Summing up the degrees (in $\widetilde{\mathrm{C}}$) of all vertices in $\widetilde{\mathrm{C}}$, we obtain

$$\begin{aligned}
\sum_{v\in\widetilde{\mathrm{C}}} \deg_{\widetilde{\mathrm{C}}}(v) &\geq \sum_{v\in\mathrm{Z}} \deg_{\widetilde{\mathrm{C}}}(v) \\
&\geq |\mathrm{Z}|\cdot((\rho-11\epsilon_1)\cdot N) \\
&\geq (\rho-3\epsilon_1)\cdot(\rho-11\epsilon_1)\cdot N^2 \\
&> (\rho^2 - 14\epsilon_1\rho)\cdot N^2 \\
&= |\widetilde{\mathrm{C}}|^2 - 14\epsilon_1\rho\cdot N^2
\end{aligned}$$

It follows that $\widetilde{\mathrm{C}}$ is $14\epsilon_1\rho$-close to being a clique and so the subgraph induced by it satisfies the claim of the lemma. ■

3.3.3 Analysis of the Clique-Degree Tester

Let $\mathrm{U}' \subset \mathrm{V(G)}$, of cardinality $t' \geq (\rho - \epsilon')t$, be fixed throughout this subsection and let $\mathrm{T} = \mathrm{T}(\mathrm{U}') \stackrel{\mathrm{def}}{=} \cap_{u \in \mathrm{U}'} \Gamma(u)$. We first prove that $\deg_{\mathrm{W}'}(\cdot)$ $(= \hat{d}(\cdot))$ provides a good estimates of $\deg_{\mathrm{T}}(\cdot)$.

Claim 3.3.3 *Let $r = \Omega(\frac{\log(1/\epsilon\delta)}{\epsilon_1^2 \rho})$. Suppose that $|\mathrm{T}| \geq \rho N$ and that* W *is a uniformly selected subset of r vertices in* V(G). *Then, with probability at least $1 - \frac{\delta}{4}$, all but an ϵ_1 fraction of the vertices $v \in \mathrm{V(G)}$ satisfy*

$$\left| \frac{\deg_{\mathrm{W} \cap \mathrm{T}}(v)}{r} - \frac{\deg_{\mathrm{T}}(v)}{N} \right| < \epsilon_1 \tag{3.7}$$

where $\deg_Q(v)$ is the degree of v in the subgraph of G *induced by Q.*

Proof: By applying a multiplicative Chernoff Bound, we get

$$\mathrm{Prob}_{\mathrm{W}}\left[|\mathrm{W} \cap \mathrm{T}| < \frac{\rho}{2} r\right] \; = \; \exp(-\Omega(\rho r)) \; < \; \frac{\delta}{8}$$

We now consider a uniformly chosen $\mathrm{W}' \subset \mathrm{T}$ of size $r' \geq \frac{\rho}{2} r$. By applying a (additive) Chernoff Bound, we get for any fixed $v \in \mathrm{V(G)}$,

$$\mathrm{Prob}_{\mathrm{W}'}\left[\left| \frac{\deg_{\mathrm{W}'}(v)}{r} - \frac{\deg_{\mathrm{T}}(v)}{N} \right| \geq \epsilon_1\right] \; = \; \exp(-\Omega(\epsilon_1^2 \rho r)) \; < \; \frac{\delta \epsilon_1}{8}$$

Applying Markov's Inequality, the claim follows. ■

As a corollary to Lemma 3.3.2, we get

Corollary 3.3.4 *Let* C *be a ρN-clique in* G. *Suppose that $\mathrm{U}' \subset \mathrm{C}$ is ϵ_2-representative w.r.t.* C, *and that* W *is such that for all but at most $\epsilon_1 N$ vertices Eq. (3.7) holds. Then, the set of ρN vertices of highest $\hat{d}(\cdot)$ value (in $\mathrm{T}(\mathrm{U}') \supseteq \mathrm{C}$) is $14\epsilon_1\rho$-close to being a ρN-clique. Recall that $\hat{d}(v) \stackrel{\mathrm{def}}{=} 2 \cdot |\Gamma(v) \cap \mathrm{W}'| = d_{\mathrm{W}'}(v)$.*

Thus, once we determine good subsets U′ and W′, we can produce an approximate clique in time $\mathrm{poly}(\frac{\log(1/\delta)}{\epsilon}) \cdot N$.[6]

Proof: Let $\widetilde{\mathrm{C}}$ be the set of ρN vertices with highest $\hat{d}(\cdot)$ value in $\mathrm{T} = \mathrm{T}(\mathrm{U}')$ (where ties are broken arbitrarily). Let α be as defined in Lemma 3.3.2 (i.e., $\deg_{\mathrm{T}}(v) \geq \alpha N$ for at least $(\rho - \epsilon_1)N$ vertices in T). By the hypothesis that Eq. (3.7) holds for all but $\epsilon_1 N$ vertices, it follows that at least $(\rho - 2\epsilon_1)N$ of the vertices v of T satisfy

$$\frac{1}{r}\hat{d}(v) \; \geq \; \frac{1}{N}\deg_{\mathrm{T}}(v) - \epsilon_1 \; \geq \; \alpha - \epsilon_1$$

Since $\widetilde{\mathrm{C}}$ contains vertices with highest $\hat{d}(\cdot)$ value, it must contain at least $(\rho - 2\epsilon_1)N$ vertices of $\hat{d}(\cdot)$ value at least $(\alpha - \epsilon_1)r$. Using the hypothesis regarding Eq. (3.7) again, we conclude that $\widetilde{\mathrm{C}}$ contains at least $(\rho - 3\epsilon_1)N$ vertices of $\deg_T(\cdot)$ value at least $(\alpha - 2\epsilon_1)N$. Using the hypothesis that U′ is ϵ_2-representative w.r.t. C, we may now invoke Lemma 3.3.2 and the corollary follows. ■

The correctness of the clique-degree algorithm now follows by two observations: (1) with high probability there exist an iteration where the sets U′ and W′ are as required in Corollary 3.3.4; and (2) the set S is a "good" sample of T(U′). We start by formulating the second observation.

[6]Note that it is possible to find the ρN vertices with highest $\hat{d}(\cdot)$ value in time linear in N (as opposed to $O(N \log N)$) since the number of values $\hat{d}(\cdot)$ takes is only $\mathrm{poly}(\frac{\log(1/\delta)}{\epsilon})$.

Lemma 3.3.5 *For a fixed* U' *and* W, *let* $\mathrm{T} = \mathrm{T}(U')$, *and let* $\widetilde{\mathrm{C}}$ *be the set of* $\min(\rho N, |\mathrm{T}|)$ *vertices with highest* $\hat{d}(\cdot)$ *value in* T, *where* $\hat{d}(\cdot)$ *is defined with respect to* $\mathrm{W}' = \mathrm{W} \cap \mathrm{T}$. *Let* $\epsilon_1 \leq \epsilon/(20\rho)$ *and* $\mathrm{S} = \{s_1, ..., s_m\}$ *be a uniformly selected set of size* $m = \Omega(\frac{t+\log(1/\delta)}{\epsilon^2})$. *Then*

1. *For any fixed set* A, $\mathrm{Prob}_\mathrm{S}\left[\left|\frac{|\mathrm{S}\cap A|}{m} - \frac{|A|}{N}\right| > \epsilon_1\right] < \frac{\delta}{16} \cdot 2^{-t}$

2. *Assume* $|\widetilde{\mathrm{C}}| \geq (\rho - 2\epsilon_1) \cdot N$. *Let* $\widehat{\mathrm{C}} \subseteq \mathrm{S}'$ *be the set of* $\min(\rho m, |\mathrm{S}'|)$ *vertices in* $\mathrm{S}' \stackrel{\mathrm{def}}{=} \mathrm{S} \cap \mathrm{T}$ *with highest* $\hat{d}(\cdot)$ *value. Then*

$$\mathrm{Prob}_\mathrm{S}\left[\left|\frac{\left|\{i\,:\,(s_{2i-1}, s_{2i}) \in (\widehat{\mathrm{C}} \times \widehat{\mathrm{C}}) \setminus \mathrm{E(G)}\}\right|}{m/2} - \mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho)\right| > \frac{\epsilon}{3}\right] \;<\; \frac{\delta}{8} \cdot 2^{-t}$$

Proof: Part (1) of the lemma follows from a simple application of Hoeffding's Inequality (i.e., an additive Chernoff bound).

Let Z be the set of $\min\{(\rho - \epsilon_1)N, |\mathrm{T}|\}$ vertices with highest $\hat{d}(\cdot)$ value in T (or, equivalently $\widetilde{\mathrm{C}}$). In proving Part (2), we assume that

$$|\mathrm{S} \cap \mathrm{Z}| \;=\; \left(\frac{|\mathrm{Z}|}{N} \pm \epsilon_1\right) \cdot m$$

By Part (1) this holds with probability at least $\frac{\delta}{16} \cdot 2^{-t}$. Recall that $\widehat{\mathrm{C}} \subseteq \mathrm{S} \cap \mathrm{T}$ has size $\min\{\rho m, |\mathrm{S} \cap \mathrm{T}|\}$ and takes elements in Z before taking elements in $\mathrm{T} \setminus \mathrm{Z}$. Thus, $\widehat{\mathrm{C}}$ contains all of $\mathrm{S} \cap \mathrm{Z}$. Furthermore, $|\widehat{\mathrm{C}} \setminus \mathrm{Z}| \leq 2\epsilon_1 m$.

For $i = 1, 2, ..., m/2$, let ζ_i be random variable which is 1 if $(s_{2i-1}, s_{2i}) \in (\widehat{\mathrm{C}} \times \widehat{\mathrm{C}}) \setminus \mathrm{E(G)}$ and 0 otherwise. Similarly, let ξ_i be random variable which is 1 if $(s_{2i-1}, s_{2i}) \in (\widehat{\mathrm{C}} \times \widehat{\mathrm{C}}) \setminus (\mathrm{Z} \times \mathrm{Z})$ and 0 otherwise. Finally, let $\zeta_i' = \zeta_i$ if $\xi_i = 1$ and be determined arbitrarily (i.e., by an adversary) otherwise. To prove Part (2) it suffices to prove

$$\mathrm{Prob}_\mathrm{S}\left[\left|\frac{\sum_{i=1}^{m/2} \zeta_i'}{m/2} - \mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho)\right| > \frac{\epsilon}{3}\right] \;<\; \frac{\delta}{8} \cdot 2^{-t} \qquad (3.8)$$

Clearly, $\sum_{i=1}^{m/2} \zeta_i' = \sum_{i=1}^{m/2}(1 - \xi_i)\zeta_i + X$, where X is determined by the adversary in the range $[0, \sum_{i=1}^{m/2} \xi_i]$. By the above, with probability at least $1 - \frac{\delta}{16} \cdot 2^{-t}$, we have $|(\widehat{\mathrm{C}} \times \widehat{\mathrm{C}}) \setminus (\mathrm{Z} \times \mathrm{Z})| \leq 2 \cdot |\widehat{\mathrm{C}}| \cdot |\widehat{\mathrm{C}} \setminus \mathrm{Z}| \leq 2 \cdot \rho m \cdot 2\epsilon_1 m$. Conditioned on the above event, we may think of the ordering of the vertices (in S) as being random and thus of the $m/2$ tested pairs as being chosen at random. Applying a Chernoff bound, we get $\mathrm{Prob}[\sum_{i=1}^{m/2} \xi_i > 5\epsilon_1 \rho m] < \frac{3\delta}{32} \cdot 2^{-t}$. Thus it is left to bound the behavior of $\sum_{i=1}^{m/2}(1 - \xi_i)\zeta_i$. Let $\chi_i \stackrel{\mathrm{def}}{=} (1 - \xi_i) \cdot \zeta_i$. Note that the χ_i are independent 0-1 random variables each being 1 iff $(s_{2i-1}, s_{2i}) \in (\mathrm{Z} \times \mathrm{Z}) \setminus \mathrm{E}(G)$. Thus, their expected value is $\mathrm{dist}(\mathrm{Z}, C_N^{\rho'})$, where $\rho' = |\mathrm{Z}|/N$. Recall that either $|\mathrm{Z}| = (\rho - \epsilon_1)N$ or $\mathrm{Z} = \widetilde{\mathrm{C}}$ (and $\rho' < \rho - \epsilon_1$). In the latter case, we have $\mathrm{dist}(\mathrm{Z}, C_N^{\rho'}) < \mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho)$, and using the hypothesis $|\widetilde{\mathrm{C}}| \geq (\rho - 2\epsilon_1)N$, we have $\mathrm{dist}(\mathrm{Z}, C_N^{\rho'}) \geq \mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho) - 2(\rho - \rho')\rho \geq \mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho) - 4\epsilon_1\rho$. In the former case we also have $\mathrm{dist}(\mathrm{Z}, C_N^{\rho'}) \leq \mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho)$, but here $\mathrm{dist}(\mathrm{Z}, C_N^{\rho'}) \geq \mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho) - 2\epsilon_1\rho$. Applying a Chernoff bound we get

$$\begin{aligned}\mathrm{Prob}_\mathrm{S}\left[\left|\frac{\sum_{i=1}^{m/2} \chi_i}{m/2} - \mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho)\right| > \frac{\epsilon}{3}\right] \;&\leq\; \mathrm{Prob}_\mathrm{S}\left[\left|\frac{\sum_{i=1}^{m/2} \chi_i}{m/2} - \mathrm{dist}(\mathrm{Z}, C_N^{\rho'})\right| > \frac{\epsilon}{3} - 5\epsilon_1\rho\right] \\ &<\; \exp(-\Omega(\epsilon^2 m))\end{aligned}$$

where the last inequality uses the hypothesis $5\epsilon_1\rho < \frac{\epsilon}{4}$. Using the bound on m, we bound $\exp(-\Omega(\epsilon^2 m))$ by $\frac{\delta}{32} \cdot 2^{-t}$ and the lemma follows. ■

Corollary 3.3.6 *Let $\epsilon_1 = \frac{\epsilon}{52\rho}$ and $\mathcal{A}$ be the algorithm of Figure 3.1.*

1. *If* $\mathrm{G} \in \mathcal{C}_\rho$ *then* $\mathrm{Prob}[\mathcal{A}(\mathrm{G}) = \mathsf{accept}] > 1 - \delta$.
2. *If* $\mathrm{dist}(\mathrm{G}, \mathcal{C}_\rho) > \epsilon$ *then* $\mathrm{Prob}[\mathcal{A}(\mathrm{G}) = \mathsf{accept}] < \delta$.

The main part of Theorem 3.4 follows from Corollary 3.3.6. As for the construction of an approximate clique, it follows by using the sets U' and W' which made the tester accept in Corollary 3.3.4.

Proof: In proving Part (1), we let C be an arbitrary ρN-clique in G. By Lemma 3.3.1, with probability at least $1 - \frac{\delta}{2}$, the set U contains an ϵ_2-representative (w.r.t. C) subset of size $\frac{\rho}{2}t$. Let us denote this subset by U' and recall that $\mathrm{U}' \subset \mathrm{C}$. We now consider the execution of Steps (1)–(4) with this U'. By Claim 3.3.3, with probability at least $1 - \frac{\delta}{4}$, the set W is such that Eq. (3.7) holds for all but $\epsilon_1 N$ vertices. Let $\widetilde{\mathrm{C}}$ denote the set of ρN vertices of highest $\hat{d}(\cdot)$ value in $\mathrm{T}(\mathrm{U}')$. By Corollary 3.3.4 and $14\epsilon_1\rho = \epsilon/3$, the set $\widetilde{\mathrm{C}}$ is $\frac{\epsilon}{3}$-close to being a ρN-clique. Applying Part (1) of Lemma 3.3.5 to T, Condition (a) of Step (4) holds with probability greater than $1 - \frac{\delta}{8}$. ¿From Part (2) of that lemma, Condition (a) of Step (4) also holds with probability greater than $1 - \frac{\delta}{8}$. Summing up the error probabilities Part (1) of this corollary follows.

We now turn to prove Part (2) For any fixed choice of U' and W, we consider the set, denoted $\widetilde{\mathrm{C}}$, of $\min(\rho N, |\mathrm{T}(\mathrm{U}')|)$ vertices of highest $\hat{d}(\cdot)$ value in $\mathrm{T}(\mathrm{U}')$. If $|\widetilde{\mathrm{C}}| < (\rho - 2\epsilon_1)N$ then necessarily $\mathrm{T} = \widetilde{\mathrm{C}}$ applying Part (1) of Lemma 3.3.5 to T, Condition (a) of Step (4) is violated with probability greater than $1 - 2^{-t} \cdot \delta$. Otherwise, we apply Part (2) of Lemma 3.3.5. Since $\mathrm{dist}(\widetilde{\mathrm{C}}, C_N^\rho) \geq \epsilon$, with probability greater than $1 - 2^{-t} \cdot \delta$, Condition (b) of Step (4) is violated. We conclude that for every possible choice of U' and W, Step (4) accepts with probability bounded by $2^{-t}\delta$. Recalling that U and W are selected at random and less than 2^t possible $\mathrm{U}' \subset \mathrm{U}$ are tried, Part (2) follows. ■

3.4 Testing Max-CUT

For a given partition $(\mathrm{V}_1, \mathrm{V}_2)$ of $\mathrm{V}(G)$, let $\mu(\mathrm{V}_1, \mathrm{V}_2)$ denote the edge density of the cut defined by $(\mathrm{V}_1, \mathrm{V}_2)$. Namely,

$$\mu(\mathrm{V}_1, \mathrm{V}_2) \stackrel{\mathrm{def}}{=} \frac{|\{(v, v') \in \mathrm{E}(\mathrm{G}) : \ \text{for } j \neq j',\ v \in \mathrm{V}_j \ \&\ v' \in V_{j'}\}|}{|\mathrm{V}(G)|^2}$$

Let $\mu(\mathrm{G})$ denote the edge density of the largest cut in G. The main results of this section are summarized below.

Theorem 3.6

1. *There exists an algorithm that on input ϵ and δ and oracle access to a graph* G, *makes* $O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^7}\right)$. *edge-queries to* G, *and with probability at least* $1-\delta$, *after time* $\exp\left(O\left(\frac{\log(1/\epsilon\delta)}{\epsilon^3}\right)\right)$, *outputs a value $\hat{\mu}$ such that* $|\hat{\mu} - \mu(\mathrm{G})| \leq \epsilon$.
2. *There exists an algorithm that on input* G, ϵ, *and* δ, *runs in time* $\exp\left(O\left(\frac{\log(1/\epsilon\delta)}{\epsilon^3}\right)\right) + O\left(\frac{\log(1/\epsilon\delta)}{\epsilon^3}\right) \cdot N$ *and with probability at least* $1 - \delta$ *outputs a partition* $(\mathrm{V}_1, \mathrm{V}_2)$ *of* $\mathrm{V}(\mathrm{G})$ *such that* $\mu(\mathrm{V}_1, \mathrm{V}_2) \geq \mu(\mathrm{G}) - \epsilon$.

Item (1) yields a property tester for the class $\mathcal{MC}_\rho \stackrel{\text{def}}{=} \{G : \mu(G) \geq \rho\}$, for every $0 \leq \rho \leq 1$, with query complexity $O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^7}\right)$, and running time $\exp\left(O\left(\frac{\log(1/\epsilon\delta)}{\epsilon^3}\right)\right)$. A more natural property tester follows as in previous cases:

Corollary 3.7 *Let $m = \text{poly}(\log(1/\delta)/\epsilon)$ and let R be a uniformly selected set of m vertices in $V(G)$. Let G_R be the subgraph (of G) induced by R. Then,*

- *if $G \in \mathcal{MC}_\rho$ then $\text{Prob}_R[\mu(G_R) > (\rho - \epsilon/2)] > \frac{2}{3}$.*
- *if $\text{dist}(G, \mathcal{MC}_\rho) > \epsilon$ then $\text{Prob}_R[\mu(G_R) \leq (\rho - \epsilon/2)] > \frac{2}{3}$.*

Our algorithms can be easily generalized to approximate and test Max-k-way-CUT for $k > 2$. Furthermore, since maximizing the density of cut edges effectively minimizes the density of edges inside the different sides of the partition, the approximation algorithm for Max-CUT (and Max-k-way-CUT) can be used to test Bipartiteness (and respectively k-Colorability) as well. However, as opposed to our bipartite (resp., k-colorability) testing algorithm, here we achieve a two-sided error (rather one-side error). That is, even if the input graph is bipartite (resp., k-colorable) it might be rejected (here) with probability greater than 0. Furthermore, for constant k, the sample complexity and running time of the algorithms presented here, are also worse than those specifically intended to test bipartiteness and k-colorability.

ORGANIZATION: We start by presenting a quadratic-time partitioning algorithm, which given a graph G constructs a cut (i.e. a partition the vertices of the graph into two disjoint sets) of edge density at least $\mu(G) - \frac{3}{4}\epsilon$. This algorithm runs in time $O\left(\exp\text{poly}\left(\frac{\log(1/\delta)}{\epsilon}\right) \cdot N^2\right)$ and is the basis for the approximation algorithm of Item (1) of Theorem 3.6. The algorithm claimed in Item (2) follows by combining the two algorithms. The extension to k-way cuts is presented at the end of this section.

3.4.1 A Preliminary Graph Partitioning Algorithm

Let $\ell = \frac{4}{\epsilon}$, and let $(\mathrm{V}^1, \ldots, \mathrm{V}^\ell)$ be a fixed partition of V(G) into ℓ sets of (roughly) equal size (say, according to the lexicographical order of the vertices in V(G)). In the Graph Partitioning Algorithm given below we describe how to construct a partition (V_1, V_2) of V(G) in ℓ iterations, where in the i^{th} iteration we construct a partition $(\mathrm{V}_1^i, \mathrm{V}_2^i)$ of V^i. The algorithm is essentially based on the following observation.
Let $(\mathrm{H}_1, \mathrm{H}_2)$ be any fixed partition of V(G). (In particular, we may consider a partition which defines a maximum cut). Let $v \in \mathrm{H}_1$ and assume, without loss of generality, that v has at least as many neighbors in H_1 as it has in H_2 (i.e., $|\Gamma(v) \cap \mathrm{H}_1| \geq |\Gamma(v) \cap \mathrm{H}_2|$). Then by moving v from H_1 to H_2 we cannot decrease the edge density of the cut (and we might even increase it). Namely,

$$\mu(\mathrm{H}_1 \setminus \{v\}, \mathrm{H}_2 \cup \{v\}) \geq \mu(\mathrm{H}_1, \mathrm{H}_2)$$

Furthermore,

$$\mu(\mathrm{H}_1 \setminus \{v\}, \mathrm{H}_2 \cup \{v\}) - \mu(\mathrm{H}_1, \mathrm{H}_2) = \frac{2 \cdot (|\Gamma(v) \cap \mathrm{H}_1| - |\Gamma(v) \cap \mathrm{H}_2|)}{N^2}. \qquad (3.9)$$

Taking this observation one step further, we next show how we can define a new partition of V based on some information concerning $(\mathrm{H}_1, \mathrm{H}_2)$, where we move $\Theta(\epsilon N)$ vertices between the sides of the partition. While we can not ensure that the size of the cut does not decrease, as was the case when moving a single vertex, we can show that the the size of the cut decreases by $O(\epsilon^2 N^2)$ (note that in the worst case, by moving $\Theta(\epsilon N)$ vertices, the size of a cut may decrease by $\Theta(\epsilon N^2)$). We then show how such a process could be used by a graph partitioning algorithm given such information (which can be viewed as access to certain oracles), and finally we show how the process can be implemented approximately (without any additional information).

An "Ideal" Procedure

let X be a subset of V(G) of size $\frac{N}{\ell} = \frac{1}{4}\epsilon N$, let $\mathrm{W} \overset{\text{def}}{=} \mathrm{V} \setminus \mathrm{X}$, and let $(\mathrm{W}_1, \mathrm{W}_2)$ be the partition of W induced by $(\mathrm{H}_1, \mathrm{H}_2)$. That is $\mathrm{W}_1 \overset{\text{def}}{=} \mathrm{H}_1 \cap \mathrm{W}$, and $\mathrm{W}_2 \overset{\text{def}}{=} \mathrm{H}_2 \cap \mathrm{W}$. Assume we knew for every vertex $x \in \mathrm{X}$ how many neighbors it has on each side of the partition $(\mathrm{W}_1, \mathrm{W}_2)$. In such a case, define X_{UB} to be the set of *unbalanced* vertices in X with respect to $(\mathrm{W}_1, \mathrm{W}_2)$. That is, X_{UB} is the set of vertices which have significantly (say $\frac{1}{8}\epsilon N$) more neighbors on one side of the partition than it has in the other. Analogously, define $\mathrm{X}_{\mathrm{B}} = \mathrm{X} \setminus \mathrm{X}_{\mathrm{UB}}$ to be the set of *balanced* vertices with respect to $(\mathrm{W}_1, \mathrm{W}_2)$.

Assume we partition X into $(\mathrm{X}_1, \mathrm{X}_2)$ as follows: Vertices in X_{UB} which have more neighbors in W_1, are put in X_2; vertices in X_{UB} which have more neighbors in W_2, are put in X_1; and vertices in X_{B} are placed arbitrarily. Based on this partition of X we define a new partition of V: $(\mathrm{H}'_1, \mathrm{H}'_2) = (\mathrm{W}_1 \cup \mathrm{X}_1, \mathrm{W}_2 \cup \mathrm{X}_2)$, which differs from $(\mathrm{H}_1, \mathrm{H}_2)$ only in the placement of vertices in X. Then clearly, the difference between $\mu(\mathrm{H}'_1, \mathrm{H}'_2)$ and $\mu(\mathrm{H}_1, \mathrm{H}_2)$, is only due to the change in the number of edges between vertices in W and vertices in X, and between pairs of vertices in X. By definition of X_{UB}, and the way it was partitioned, the number of cut edges between vertices in W and vertices in X_{UB} could not have decreased. By definition of X_{B}, the arbitrary placement of these vertices decreased the number of cut edges between W and X_{B} by at most $|\mathrm{X}_{\mathrm{B}}| \cdot 2 \cdot \frac{1}{8}\epsilon N \leq \frac{\epsilon}{4\ell} N^2$, and the number of cut edges between pairs of vertices in X decreased by at most $|\mathrm{X}|^2 = \frac{1}{\ell^2} N^2 \leq \frac{\epsilon}{4\ell} N^2$.

Now, let X be V^1 (i.e. the first N/ℓ vertices in lexicographical order), let $(\mathrm{H}_1, \mathrm{H}_2)$ define a maximum cut, and let the partition resulting from the process defined above be denoted by $(\mathrm{H}_1^1, \mathrm{H}_2^1)$. Thus the information needed in order to partition V^i, is the (approximate) number of neighbors each vertex in V^i has on each side of the partition of $\mathrm{V} \setminus V^1$ induced by $(\mathrm{H}_1, \mathrm{H}_2)$. Assume we continue iteratively, where in stage i we perform the above partitioning process for V^i, given the partition $(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1})$ determined in stage $i-1$. That is, in stage i we assume we know which vertices in V^i are unbalanced with respect to the partition of $\mathrm{V} \setminus \mathrm{V}^i$ induced by $(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1})$. Then we can apply the same argument used above to each pair of consecutive partitions $(\mathrm{H}_1^i, \mathrm{H}_2^i)$ and $(\mathrm{H}_1^{i+1}, \mathrm{H}_2^{i+1})$, and get that $\mu\,(\mathrm{H}_1^\ell, \mathrm{H}_2^\ell)$ is smaller than $\mu(\mathrm{H}_1, \mathrm{H}_2) = \mu(\mathrm{G})$ by no more than $\frac{3}{4}\epsilon$.

The Actual Algorithm

The graph partitioning algorithm, depicted in Figure 3.2, approximately implements the iterative procedure described above, starting from a partition $(\mathrm{H}_1^0, \mathrm{H}_2^0)$ which defines a maximum cut. Clearly, we do not have a clue as to what $(\mathrm{H}_1^0, \mathrm{H}_2^0)$ is, and hence, in particular, we have no direct way of determining for a given vertex v in V^1 whether it is balanced or unbalanced with respect to the partition $(\mathrm{W}_1^0, \mathrm{W}_2^0)$ of $\mathrm{W}^0 \overset{\text{def}}{=} \mathrm{V} \setminus \mathrm{V}^1$ induced by this partition. However, we can approximate the number of neighbors v has on each side of $(\mathrm{W}_1^0, \mathrm{W}_2^0)$ by sampling. Namely, if we uniformly choose a set of vertices U^1 of size $t = \mathrm{poly}\left(\frac{\log(1/\delta)}{\epsilon}\right)$ in W^0, then (as we later prove formally), with high probability over the choice of U^1 *there exists* a partition $(\mathrm{U}_1^1, \mathrm{U}_2^1)$ of U^1, which is *representative* with respect to $(\mathrm{W}_1^0, \mathrm{W}_2^0)$ and V^1 in the following sense. For all but a small fraction of vertices v in V^1, the number of neighbors v has in U_1^1 (U_2^1), relative to the size of U^1, is approximately the

Graph Partitioning Algorithm (for MaxCUT)

```
1. Choose $\ell = \frac{4}{\epsilon}$ sets $U^1, \ldots, U^\ell$ each of size $t = \Theta\left(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon\delta}\right)$, where $U^i$ is chosen uniformly in $V \setminus V^i$.
   Let $U$ be the union of these sets.
2. For each of the partitions $\pi(U) = \left(\bigcup_{i=1}^{\ell} U_1^i, \bigcup_{i=1}^{\ell} U_2^i\right)$ of $U$ do:
   (a) For $i = 1 \ldots \ell$, partition $V^i$ into two disjoint sets $V_1^i$ and $V_2^i$ as follows:
       For each $v \in V^i$,
         i. If $|\Gamma(v) \cap U_1^i| \geq |\Gamma(v) \cap U_2^i|$ then put $v$ in $V_2^i$.
         ii. Otherwise put $v$ in $V_1^i$.
   (b) Let $V_1^{\pi(U)} = \bigcup_{i=1}^{\ell} V_1^i$, and let $V_2^{\pi(U)} = \bigcup_{i=1}^{\ell} V_2^i$.
3. Among all partitions $\left(V_1^{\pi(U)}, V_2^{\pi(U)}\right)$, created in Step (2), let $(V_1^U, V_2^U)$ be the one which defines
   the largest cut, and output it.
```

Figure 3.2: Graph Partitioning Algorithm for Max-CUT

same as the number of neighbors v has in W_1^0 (W_2^0), relative to the size of V(G). Clearly such an approximation suffices since what is important when deciding where to put the vertices in V^1 is to determine where to put the unbalanced vertices. If U^1 has a representative partition, then we say that U^1 is *good*. Since we do not know which of the 2^t partitions of U^1 is the representative one (assuming one exists), we simply try them all.

The choice of U^1 together with each of its partitions determines a partition of V^1. While we must consider all partitions (U_1^1, U_2^1) of U^1, we are only interested in the (hopefully representative) partition for which $U_1^1 \subset W_1^0$ and $U_2^1 \subset W_2^0$. Denote this partition by (U_1^1, U_2^1). Let (V_1^1, V_2^1) be the partition of V^1 which is determined by this partition of U^1, and let (H_1^1, H_2^1) be the resulting partition of V(G). Namely, (H_1^1, H_2^1) is the same as (H_1^0, H_2^0) *except* for the placement of the vertices in V^1, which is done according to (V_1^1, V_2^1). If in fact (U_1^1, U_2^1) is representative (with respect to (W_1^0, W_2^0) and V^1), then $\mu(H_1^1, H_2^1)$ is not much smaller than $\mu(H_1^0, H_2^0) = \mu(G)$. We continue in the same manner, where in stage i we randomly pick a set U^i, and for each of its partitions we determine a partition of V^i. Therefore, we are actually constructing $(2^t)^\ell = 2^{\ell \cdot t}$ possible partitions of V(G), one for each partition of all the U^i's. However, in order to show that at least one of these partitions defines a cut which is not much smaller than the maximum cut, we only need to ensure that for each i, with high probability, U^i is good with respect to (W_1^{i-1}, W_2^{i-1}), where the latter partition is determined by the choice of $U^1, \ldots, U^{i-1}$, and their representative partitions, $(U_1^1, U_2^1), \ldots, (U_1^{i-1}, U_2^{i-1})$. The actual code is depicted in Figure 3.2. In the following lemma we formalize the intuition given previously as to why the partitioning algorithm works.

Lemma 3.4.1 *Let* (H_1, H_2) *be a fixed partition of* V(G)*. Then with probability at least* $1 - \delta/2$ *over the choice of* U*, there exists a partition* $\pi(U)$*, such that* $\mu(V_1^{\pi(U)}, V_2^{\pi(U)}) \geq \mu(H_1, H_2) - \frac{3}{4}\epsilon$.

Proof: For a given partition $\pi(U)$ of U, we consider the following $\ell + 1$ *hybrid* partitions. the hybrid (H_1^0, H_2^0) is simply (H_1, H_2). The hybrid partition (H_1^i, H_2^i) is defined as follows:

$$H_1^i \stackrel{\text{def}}{=} W_1^{i-1} \cup V_1^i$$

and

$$H_2^i \stackrel{\text{def}}{=} W_2^{i-1} \cup V_2^i$$

where for $j \in \{1,2\}$, $\mathrm{W}_j^{i-1} \stackrel{\text{def}}{=} \mathrm{H}_j^{i-1} \cap \mathrm{W}^{i-1}$, and $\mathrm{W}^{i-1} \stackrel{\text{def}}{=} \mathrm{V} \setminus \mathrm{V}^i$. Note, that in particular, $(\mathrm{H}_1^\ell, \mathrm{H}_2^\ell)$ is the partition $\left(\mathrm{V}_1^{\pi(\mathrm{U})}, \mathrm{V}_2^{\pi(\mathrm{U})}\right)$. Since the partition of each V^i is determined by the choice of U^i and its partition, the i^{th} hybrid partition is determined by the choice of $\mathrm{U}^1, \ldots, \mathrm{U}^{i-1}$ and their partitions *but not* by the choice nor the partitions of $\mathrm{U}^i, \ldots, \mathrm{U}^\ell$. We shall show that for every $1 \le i \le \ell$, for any fixed choice and partitions of $\mathrm{U}^1, \ldots, \mathrm{U}^{i-1}$, with probability at least $1 - \frac{\delta}{2\ell}$ over the choice of U^i, there exists a partition $(\mathrm{U}_1^i, \mathrm{U}_2^i)$ of U^i such that

$$\mu\left(\mathrm{H}_1^i, \mathrm{H}_2^i\right) \ge \mu\left(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1}\right) - \frac{3\epsilon}{4\ell}$$

The lemma will directly follow.

For the $i-1$ hybrid partition $(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1})$, or more precisely, for the partition it induces on W^{i-1}, and a sample set U^i, let

$$\mathrm{U}_1^i \stackrel{\text{def}}{=} \mathrm{W}_1^{i-1} \cap \mathrm{U}^i$$

and

$$\mathrm{U}_2^i \stackrel{\text{def}}{=} \mathrm{W}_2^{i-1} \cap \mathrm{U}^i .$$

We say that U^i is *good* with respect to $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$ and V^i if $(\mathrm{U}_1^i, \mathrm{U}_2^i)$ is *representative* with respect to $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$ and V^i. That is, $(\mathrm{U}_1^i, \mathrm{U}_2^i)$ is such that for all but $\frac{1}{8}\epsilon$ of the vertices v in V^i the following holds:

$$\text{For each } j \in \{1,2\}, \qquad \frac{\left|\Gamma(v) \cap \mathrm{U}_j^i\right|}{t} = \frac{\left|\Gamma(v) \cap \mathrm{W}_j^{i-1}\right|}{N} \pm \frac{\epsilon}{32} \tag{3.10}$$

Assume that in fact for each i, the set U^i is good with respect to $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$ and V^i. As was previously defined, we say that a vertex v is *unbalanced* with respect to $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$ if

$$\text{for } j, j' \in \{1,2\},\ j \ne j' \qquad \left|\Gamma(v) \cap \mathrm{W}_j^{i-1}\right| \ge \left|\Gamma(v) \cap \mathrm{W}_{j'}^{i-1}\right| + \frac{1}{8}\epsilon N . \tag{3.11}$$

Thus, if $v \in \mathrm{V}^i$ is an unbalanced vertex with respect to $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$ for which Equation (3.10) is satisfied, then $\left|\Gamma(v) \cap \mathrm{U}_j^i\right| \ge \left|\Gamma(v) \cap \mathrm{U}_{j'}^i\right| + \frac{1}{16}\epsilon t$. We are hence guaranteed (by Steps (2)(a)(i) and (2)(a)(ii) of the algorithm) that when the partition $(\mathrm{U}_1^i, \mathrm{U}_2^i)$ is used then v is put opposite the majority of its neighbors in W^{i-1} (according to their position in $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$). If v is balanced then it might be placed on either side of the partition. The same is true for the (at most $\frac{\epsilon}{8} \cdot \frac{N}{\ell}$) vertices for which Equation (3.10) does not hold.

As was noted previously, the decrease in the size of the cut is only due to changes in the number of edges between vertices in W^{i-1} and vertices in V^i, and between pairs of vertices in V^i. In particular:

1. The number of cut edges between vertices in W^{i-1} and unbalanced vertices in V^i for which Equation (3.10) is satisfied can not decrease.

2. The number of cut edges between vertices in W^{i-1} and balanced vertices in V^i, decreases by at most $|\mathrm{V}^i| \cdot 2 \cdot \frac{1}{8}\epsilon N \le \frac{\epsilon}{4\ell} N^2$.

3. The number of cut edges between vertices in W^{i-1} and unbalanced vertices in V^i for which Equation (3.10) is *not* satisfied decreases by at most $\frac{\epsilon}{8} \cdot |\mathrm{V}^i| \cdot 2N \le \frac{\epsilon}{4\ell} N^2$.

4. The number of cut edges between pairs of vertices in V^i decreased by at most $|V^i|^2 = \frac{1}{\ell^2}N^2 \leq \frac{\epsilon}{4\ell}N^2$.

The total decrease is bounded by $\frac{3\epsilon}{4\ell}N^2$.

It remains to prove that with high probability a chosen set U^i is good (with respect to (W_1^{i-1}, W_2^{i-1}) and V^i). We first fix a vertex $v \in V^i$. Let $U^i = \{u_1, \ldots, u_t\}$. Recall that U^i is chosen uniformly in $W^{i-1} \stackrel{\text{def}}{=} V \setminus V^i$. For $j \in \{1,2\}$, and for $1 \leq k \leq t$, define a Bernoulli random variable, ξ_j^k, which is 1 if u_k is a neighbor of v and $u_k \in W_j^{i-1}$, and is 0 otherwise. By definition. for each j, the sum of the ξ_j^k's is simply the number of neighbors v has in U_j^i $(= U^i \cap W_j^{i-1})$ and the probability that $\xi_j^k = 1$ is $\frac{1}{N}\left|\Gamma(v) \cap W_j^{i-1}\right|$. By Hoeffding's inequality and our choice of t, for each $j \in \{1, \ldots, k\}$,

$$\mathrm{Prob}_{U^i}\left[\left|\frac{\left|\Gamma(v) \cap U_j^i\right|}{t} - \frac{\left|\Gamma(v) \cap W_j^{i-1}\right|}{N}\right| > \frac{\epsilon}{32}\right] = O(\exp(-\epsilon^2 t)) = O\left(\frac{\epsilon\delta}{\ell}\right).$$

By Markov's inequality, for each $j \in \{1,2\}$, with probability at least $1 - \frac{\delta}{4\ell}$ over the choice of U^i, for all but $\frac{1}{8}\epsilon$ of the vertices in V^i, Equation (3.10) holds (for that j), and thus with probability at least $1 - \frac{\delta}{2\ell}$, U^i is good as required. ■

Applying Lemma 3.4.1 to a maximum cut of G, we get

Corollary 3.4.2 *With probability at least* $1 - \frac{\delta}{2}$ *over the choice of* U, $\mu(V_1^U, V_2^U) \geq \mu(G) - \frac{3}{4}\epsilon$.

3.4.2 The Max-CUT Approximation Algorithm

Given the graph partitioning algorithm described above, the Max-CUT approximation algorithm is quite straightforward. We uniformly choose a set S of size $m = \Theta\left(\frac{\ell \cdot t + \log(1/\delta)}{\epsilon^2}\right)$, and run the graph partitioning algorithm restricted to this set. The only small difference from what might be expected is that we do not necessarily output the size of largest cut among the cuts defined by the resulting partitions of S (i.e. those determined by the partitions of U). Instead, we view S as a multiset of $m/2$ (ordered) pairs of vertices, and we choose the cut which maximizes the number of such pairs which are edges in the cut. The exact code is given in Figure 3.3.

Lemma 3.4.3 *For any fixed set* U*, with probability at least* $1-\delta/2$ *over the choice of* S, $\hat{\mu}(S_1^U, S_2^U) = \mu(V_1^U, V_2^U) \pm \frac{1}{4}\epsilon$, *where* $\hat{\mu}(\cdot,\cdot)$ *is as defined in step* 4 *of the Max-CUT approximation algorithm.*

Proof: Consider first a particular partition of U, $\pi(U)$. The key observation is that for every $s \in S$, and for $j \in \{1,2\}$, $s \in S_j^{\pi(U)}$ if and only if $s \in V_j^{\pi(U)}$. Thus for each partition of U we are effectively sampling from $\left(V_1^{\pi(U)}, V_2^{\pi(U)}\right)$. Furthermore, by viewing S as consisting of $m/2$ pairs of vertices (s_{2k-1}, s_{2k}), and counting the number of such pairs which are on opposite sides of the partition and have an edge in between, we are able to approximate the density of the cut edges. For $1 \leq k \leq m/2$, let Let ξ_k be a Bernoulli random variable which is 1 if $(s_{2k-1}, s_{2k}) \in E(G)$, and for $j \neq j'$, $s_{2k-1} \in S_j^{\pi(U)}$ and $s_{2k} \in S_{j'}^{\pi(U)}$. Then, by definition, $\hat{\mu}\left(S_1^{\pi(U)}, S_2^{\pi(U)}\right) = \frac{2}{m}\sum_k^{m/2} \xi_k$, and the probability that $\xi_k = 1$ is $\mu\left(V_1^{\pi(U)}, V_2^{\pi(U)}\right)$. Hence, by Hoeffding's inequality and our choice of m,

$$\mathrm{Prob}_S\left[\left|\hat{\mu}\left(S_1^{\pi(U)}, S_2^{\pi(U)}\right) - \mu\left(V_1^{\pi(U)}, V_2^{\pi(U)}\right)\right| > \frac{1}{8}\epsilon\right] = O(\exp(-\epsilon^2 m)) = O\left(\delta 2^{-\ell \cdot t}\right) \qquad (3.12)$$

Max-CUT Approximation Algorithm

1. *As Step (1) of Figure 3.2.*
2. *Uniformly choose a set* $S = \{s_1, \ldots, s_m\}$ *of size* $m = \Theta\left(\frac{\ell \cdot t + \log(1/\delta)}{\epsilon^2}\right)$. *For* $1 \le i \le \ell$, *let* $S^i \overset{\text{def}}{=} V^i \cap S$.
3. *Analogously to Step (2) of Figure 3.2, for each of the partitions* $\pi(U) = \left(\bigcup_{i=1}^{\ell} U_1^i, \bigcup_{i=1}^{\ell} U_2^i\right)$ *of* U, *partition each* S^i *into two disjoint sets* S_1^i *and* S_2^i, *and let* $S_j^{\pi(U)} = \bigcup_{i=1}^{\ell} S_j^i$ *(for* $j = 1, 2$*).*
4. *Among all partitions* $\left(S_1^{\pi(U)}, S_2^{\pi(U)}\right)$, *let* (S_1^U, S_2^U) *be the one which maximizes the number of cut edges between pairs of vertices* (s_{2k-1}, s_{2k}). *Namely, define*

$$\hat{\mu}\left(S_1^{\pi(U)}, S_2^{\pi(U)}\right) \overset{\text{def}}{=} \frac{\left|\left\{(s_{2k-1}, s_{2k}) \in E(G) : \text{ for } j \neq j',\ s_{2k-1} \in S_j^{\pi(U)} \ \&\ s_{2k} \in S_{j'}^{\pi(U)}\right\}\right|}{m/2}$$

Output $\max_\pi \hat{\mu}\left(S_1^{\pi(U)}, S_2^{\pi(U)}\right)$.

Figure 3.3: Max-CUT Approximation Algorithm

Since there are $2^{\ell \cdot t}$ partitions of U, with probability at least $1 - \delta/2$, for every partition $\pi(U)$, $\hat{\mu}\left(S_1^{\pi(U)}, S_2^{\pi(U)}\right) = \mu\left(V_1^{\pi(U)}, V_2^{\pi(U)}\right) \pm \frac{1}{8}\epsilon$, and hence $\hat{\mu}(S_1^U, S_2^U) = \mu(V_1^U, V_2^U) \pm \frac{1}{4}\epsilon$, ■

Combining Corollary 3.4.2 and Lemma 3.4.3, Part (1) of Theorem 3.6 follows.[7] As per Part (2) it is proved in the next subsection.

3.4.3 An Improved Graph Partitioning Algorithm

The improved graph partitioning algorithm starts by invoking the Max-CUT approximation algorithm of Figure 3.3, and recording the set U uniformly selected in Step (1) and the partition $\Pi = \pi(U)$ selected in Step (4). Using this specific partition Π, the algorithm executes a single iteration of Step (2) of the Graph Partitioning Algorithm of Figure 3.2 an obtains the partition $\left(V_1^{\pi(U)}, V_2^{\pi(U)}\right)$. By Lemma 3.4.1, we have that with probability at least $1 - \delta/2$ over the choice of U, there exists a partition $\pi(U)$, such that $\mu(V_1^{\pi(U)}, V_2^{\pi(U)}) \ge \mu(G) - \frac{3}{4}\epsilon$. ¿From the proof of Lemma 3.4.3 we have that for a fixed set U, with probability at least $1 - \delta/2$ over the choice of S, $\hat{\mu}\left(S_1^{\pi(U)}, S_2^{\pi(U)}\right)$ is within $\frac{\epsilon}{8}$ from $\mu\left(V_1^{\pi(U)}, V_2^{\pi(U)}\right)$ for every partition Π of U. It follows that with probability at least $1 - \delta$, the recorded partition Π is such that $\mu\left(V_1^{\pi(U)}, V_2^{\pi(U)}\right) \ge \mu(G) - \epsilon$, as required.

3.4.4 Generalization to k-way CUTs

For a k-way partition $(V_1, \ldots, V_k)$ of $V(G)$, we denote by $\mu_k(V_1, \ldots, V_k)$ the edge density of the cut defined by $(V_1, \ldots, V_k)$. Namely,

$$\mu_k(V_1, \ldots, V_k) \overset{\text{def}}{=} \frac{|\{(v, v') \in E(G) : \text{ for } j \neq j',\ v \in V_j \ \&\ v' \in V_{j'}\}|}{N^2}$$

Let $\mu_k(G)$ denote the edge density of the largest k-way cut in G.

[7] Actually, we need to deal separately with the trivial case in which $\rho > 1/2$. In this case the class $\mathcal{MC}_\rho$ is empty and so we need to reject all graphs (as none is close to $\mathcal{MC}_\rho$).

Theorem 3.8

1. *There exists an algorithm that on input k, ϵ and δ and oracle access to a graph* G, *makes* $O\left(\frac{\log^2(k/\epsilon\delta)}{\epsilon^7}\right)$ *edge-queries to* G, *and with probability at least* $1-\delta$, *after time* $\exp\left(O\left(\frac{\log(k/\epsilon\delta)}{\epsilon^3}\right)\right)$, *outputs a value* $\hat{\mu_k}$ *such that* $|\hat{\mu_k} - \mu_k(\mathrm{G})| \le \epsilon$.
2. *There exists an algorithm that on input* G, k, ϵ, *and* δ, *runs in time* $\exp\left(O\left(\frac{\log(k/\epsilon\delta)}{\epsilon^3}\right)\right) + O\left(\frac{\log(k/\epsilon\delta)}{\epsilon^3}\right) \cdot N$, *and with probability at least* $1-\delta$ *outputs a k-way partition* $(\mathrm{V}_1, \ldots, \mathrm{V}_k)$ *of* $\mathrm{V}(\mathrm{G})$ *such that* $\mu_k(\mathrm{V}_1, \ldots, \mathrm{V}_k) \ge \mu_k(\mathrm{G}) - \epsilon$.

The graph k-way partitioning algorithm (resp., Max-k-way-CUT approximation algorithm and testing algorithms), are obtained from the 2-way partitioning algorithm by the following simple modifications

- Instead of considering all two-way partitions of each U^i, we consider all its k-way partitions.
- For each such partition, $(\mathrm{U}^i_1, \ldots, \mathrm{U}^i_k)$, we partition V^i (into k disjoint sets) as follows. For each vertex $v \in \mathrm{V}^i$, and for $1 \le j \le k$, let $d^i_j(v) = \left|\Gamma(v) \cap \mathrm{U}^i_j\right|$, and let $j_{\min} = \mathrm{argmin}_j\{d^i_j(v)\}$. Then we put v in $\mathrm{V}^i_{j_{\min}}$.

The following (minor) changes suffice for the adapting the analysis to the modified algorithm. Let $(\mathrm{H}^0_1, \ldots, \mathrm{H}^0_k)$ be a k-way partition of $\mathrm{V}(\mathrm{G})$ which defines a maximum k-way cut. For a given choice and k-way partitions of $\mathrm{U}^1, \ldots, \mathrm{U}^{i-1}$, let $(\mathrm{H}^{i-1}_1, \ldots, \mathrm{H}^{i-1}_k)$ be the $i-1$ hybrid k-way partition (defined analogously to the two-way cut case) which is determined by this choice and partitions of $\mathrm{U}^1, \ldots, \mathrm{U}^{i-1}$. Using the same notation introduced in the two-way cut case, for a set U^i and for each $j \in \{1, \ldots, k\}$ let $\mathrm{U}^i_j \stackrel{\mathrm{def}}{=} \mathrm{W}^{i-1}_j \cap \mathrm{U}^i$ where $\mathrm{W}^{i-1}_j \stackrel{\mathrm{def}}{=} \mathrm{H}^{i-1}_j \setminus \mathrm{V}^i$. We shall say that U^i is *good* with respect to $(\mathrm{W}^{i-1}_1, \ldots, \mathrm{W}^{i-1}_k)$ and V^i, if for all but $\frac{1}{8}\epsilon$ of the vertices v in V^i, Equation (3.10) holds for every $j \in \{1, \ldots, k\}$. It follows that in order to ensure that each U^i be good (with respect to $(\mathrm{W}^{i-1}_1, \ldots, \mathrm{W}^{i-1}_k)$ and V^i), we need to choose $t = |\mathrm{U}^i|$ to be $\log k$ times larger than in the two-way cut case.

The notion of *unbalanced* vertices is generalized as follows. We say that $v \in \mathrm{V}^i$ is *unbalanced* with respect to $(\mathrm{W}^{i-1}_1, \ldots, \mathrm{W}^i_k)$, if there exists $j \in \{1, \ldots, k\}$ such that for every $j' \ne j$ $\left|\Gamma(v) \cap \mathrm{W}^{i-1}_{j'}\right| \ge \left|\Gamma(v) \cap \mathrm{W}^{i-1}_j\right| + \frac{1}{8}\epsilon N$. Thus, it is still the case that if v is an unbalanced vertex for which Equation (3.10) holds for every $j \in \{1, \ldots, k\}$, then v will be put on the side in which it has the minimal number of neighbors in W^{i-1}. As a consequence, the number of cut edges between vertices in W^{i-1} and unbalanced vertices in V^i can not decrease. If v is not unbalanced it does not mean (as in the two-way cut case) that it has roughly the same number of neighbors on each side of the partition $(\mathrm{W}^{i-1}_1, \ldots, \mathrm{W}^{i-1}_k)$. Rather, it means that there exist at least two sides of the partition on which v has roughly the same number of neighbors, and furthermore, one of these sides includes its smallest number of neighbors. We shall thus refer to such a vertex v as *partially balanced* rather than *balanced*. However, as for unbalanced vertices, if v is a partially balanced vertex for which Equation (3.10) holds for every $j \in \{1, \ldots, k\}$, then v will be placed on a side of the partition in which its number of neighbors is not much far from minimum. The key point is that if a vertex is moved from side j (in $(\mathrm{H}^{i-1}_1, \ldots, \mathrm{H}^{i-1}_k)$ to side j' (in $(\mathrm{H}^i_1, \ldots, \mathrm{H}^i_k)$), all edges that it has with vertices on sides $j'' \ne j, j'$ remain cut edges, and only the number of cut-edges between side j and side j' might change. Thus, when moving a partially balanced vertex (for which Equation (3.10) holds for every $j \in \{1, \ldots, k\}$), the number of cut edges between this vertex and vertices in $\mathrm{V} \setminus \mathrm{V}^i$ does not decrease by much. As in the case of $k = 2$, we are essentially "giving up" on all cut edges between pairs of vertices in V^i.

Finally, since the number of k-way partitions of all the U^i's is $k^{t\cdot\ell}$, we must choose m (the size of S in the Max-k-way-CUT approximation algorithm) to be $\approx \Theta(\frac{\ell t}{\epsilon^2}\cdot\log k)$ rather than $\approx \Theta(\frac{\ell t}{\epsilon^2})$ (as our choice in the case of two-way cuts).

3.5 Testing Bisection

In this section we study a variant of the Max-CUT problem in which both sides of the partition are required to be of equal size. Namely, using the notation presented in Section 3.4, let

$$\mu^{(\frac{1}{2},\frac{1}{2})}(\mathrm{G}) \stackrel{\mathrm{def}}{=} \max_{\mathrm{V}_1\subset \mathrm{V}(G), |\mathrm{V}_1|=N/2} \mu(\mathrm{V}_1, \mathrm{V}(\mathrm{G})\setminus V_1)\ .$$

A partition $(\mathrm{V}_1,\mathrm{V}_2)$ of $\mathrm{V}(\mathrm{G})$, such that $|\mathrm{V}_1| = |\mathrm{V}_2| = N/2$ is called a *bisection*.[8] We first consider the less standard problem of maximizing the (number of edges crossing the) bisection. The (more standard) case of minimization is handled analogously (see Subsection 3.5.4). The main result of this section is

Theorem 3.9

1. *There exists an algorithm that on input ϵ and δ and oracle access to a graph* G*, makes* $O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^7}\right)$*. edge-queries to* G*, and with probability at least* $1-\delta$*, after time* $\exp\left(O\left(\frac{\log(1/\epsilon\delta)}{\epsilon^3}\right)\right)$*, outputs a value* $\hat{\mu}^{(\frac{1}{2},\frac{1}{2})}$ *such that* $|\hat{\mu}^{(\frac{1}{2},\frac{1}{2})} - \mu^{(\frac{1}{2},\frac{1}{2})}(\mathrm{G})| \le \epsilon$*.*

2. *There exists an algorithm that on input* G*,* ϵ*, and* δ*, runs in time* $\exp\left(O\left(\frac{\log(1/\epsilon\delta)}{\epsilon^3}\right)\right) + O\left(\frac{\log(1/\epsilon\delta)}{\epsilon^3}\right)\cdot N$ *and with probability at least* $1-\delta$ *outputs a bisection* $(\mathrm{V}_1,\mathrm{V}_2)$ *of* $\mathrm{V}(\mathrm{G})$ *such that* $\mu(\mathrm{V}_1,\mathrm{V}_2) \ge \mu^{(\frac{1}{2},\frac{1}{2})}(\mathrm{G}) - \epsilon$*.*

Item (1) yields a property tester for the class $\mathcal{MC}_\rho^{(\frac{1}{2},\frac{1}{2})} \stackrel{\mathrm{def}}{=} \{G : \mu^{(\frac{1}{2},\frac{1}{2})}(G) \ge \rho\}$, for every $0\le\rho\le 1$, with query complexity $O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^7}\right)$ and running time $\exp\left(O\left(\frac{\log(1/\epsilon\delta)}{\epsilon^3}\right)\right)$. A more natural property tester follows as in previous cases:

Corollary 3.10 *Let* $m = \mathrm{poly}(\log(1/\delta)/\epsilon)$ *and let R be a uniformly selected set of m vertices in $V(G)$. Let G_R be the subgraph (of G) induced by R. Then,*

- *if* $G\in\mathcal{MC}_\rho^{(\frac{1}{2},\frac{1}{2})}$ *then* $\mathrm{Prob}_R[\mu^{(\frac{1}{2},\frac{1}{2})}(G_R) > (\rho-\epsilon/2)] > \frac{2}{3}$*.*

- *if* $\mathrm{dist}(G,\mathcal{MC}_\rho^{(\frac{1}{2},\frac{1}{2})}) > \epsilon$ *then* $\mathrm{Prob}_R[\mu^{(\frac{1}{2},\frac{1}{2})}(G_R) \le (\rho-\epsilon/2)] > \frac{2}{3}$*.*

Our proof of Theorem 3.9 follows the outline of the proof of Theorem 3.6 (i.e., the analysis of the Max-CUT algorithms). However, there is one crucial difference between the problem of constructing (resp., approximating the size of) a maximum cut and the problem of constructing (resp., approximating the size of) a bisection: In a bisection both sides of the cut must be of equal size. This has the following consequence. Recall that in in case of Max-CUT, it is always beneficial (and possible) to relocate a vertex so that it is in the side opposite to the majority of its neighbors. In contrast, when restricted to bisections, this property no longer holds: a maximum size bisection may have vertices which belong to the same side of the bisection as the majority of their neighbors.

[8] We assume throughout this section that N is even. In case N is odd, one may require $|\mathrm{V}_1| = |\mathrm{V}_2|+1 = (N+1)/2$.

Thus, when partitioning a subset V^i, we can not simply put all vertices (or all unbalanced vertices) on the side opposite the majority of their neighbors.

However, we can still use information concerning the unbalance of vertices with respect to a given bisection (and some additional information) in order to define a new bisection whose cut is not much smaller. Consider an arbitrary bisection (H_1, H_2), and and arbitrary set of vertices X of size $O(\epsilon N)$. Assume we are told how many neighbors each vertex in X has in $W_1 \stackrel{\rm def}{=} H_1 \setminus X$ and how many in $W_2 \stackrel{\rm def}{=} H_2 \setminus X$. Further assume that we know $|H_1 \cap X|$ and $|H_2 \cap X|$. Let us related with each vertex in X and *unbalance* value, with is simply the fraction of neighbors it has in W_2 (among all N possible neighbors) minus the fraction of neighbors it has in W_1. This value (which ranges from 1 to -1) tries to capture our "preference" of placing a vertex on side 1. Assume we now repartition the vertices in X so that: (1) there are $|H_1 \cap X|$ vertices on side 1 and $|H_2 \cap X|$ on side 2; (2) all vertices on side 1 have unbalance value which is greater or equal to the unbalance value of any vertex on side 2; then the following holds. First, the resulting partition is clearly a bisection. Secondly, the size of the cut decreases by at most $|X|^2 = O(\epsilon^2 N^2)$. The latter is due to the fact that for any other partition of X with $|H_1 \cap X|$ vertices on one side (and the rest on the other), there cannot be more cut edges between vertices in X and vertices in $V \setminus X$ than in the partition defined above. The decrease in the size of the cut is hence due to the decrease in the number of cut edges between pairs of vertices in X. Our graph bisection algorithm is based on this observation.

3.5.1 A High Level Description of the Bisection Algorithm

As was done in Subsection 3.4, we start by describing an algorithm which is aided by certain oracles, and then show how to simulate these oracles. Similarly to the (oracle aided) graph partitioning algorithm described in Section 3.4 for Max-CUT, the (oracle aided) graph bisection algorithm proceeds in $\ell = O(1/\epsilon)$ iterations where in the $i^{\rm th}$ iteration the vertices in V^i are partitioned into two subsets, V_1^i and V_2^i. Here too we think of the algorithm as defining *hybrid* partitions. Starting from the zero hybrid partition $\mathcal{H}^0 = \{H_1^0, H_2^0\}$, which is a maximum bisection, the $i^{\rm th}$ hybrid partition $\mathcal{H}^i = (H_1^i, H_1^i)$ is a hybrid of the partition (V_1^i, V_2^i) of V^i (constructed in the $i^{\rm th}$ iteration), and the partition $\mathcal{W}^{i-1} = \{W_1^{i-1}, W_2^{i-1}\}$ of $W^{i-1} \stackrel{\rm def}{=} V \setminus V^i$ induced by the $i-1$ hybrid partition, $\mathcal{H}^{i-1} = \{H_1^{i-1}, H_2^{i-1}\}$. However, differently from the Max-CUT graph partitioning algorithm, here we might place vertices of V^i which are *unbalanced* with respect to $\mathcal{W}^{i-1}$ on the same side of the partition as the majority of their neighbors. This is done so to maintain the desired proportion of vertices of V^i on each side of the new hybrid. That is, for each $i \in \{1, \ldots, \ell\}$, let $\beta^i = \frac{\ell}{N} \cdot |V^i \cap H_1^{i-1}|$ be the fraction of vertices in V^i which belong to H_1^{i-1}. Assume we knew all β^i's. If in each iteration, i, we make sure to put β^i of the vertices in V^i on side 1 and $(1-\beta^i)$ on side 2, then since $\mathcal{H}^0$ is a bisection, so will each hybrid partition be, and in particular the final partition which the algorithm outputs. Indeed, we assume here that we know the β^i's.

Further assume that in each iteration of the algorithm we knew *exactly* how many neighbors each vertex in V^i has on each side of the partition. In such a case we could compute for each vertex v its *unbalance* value:

$$\mathrm{ub}(v) \stackrel{\rm def}{=} \frac{|\Gamma(v) \cap W_2^{i-1}| - |\Gamma(v) \cap W_1^{i-1}|}{N}\,. \tag{3.13}$$

Let $L \stackrel{\rm def}{=} N/\ell$ and $v_{(1)}, \ldots, v_{(L)}$ be an ordering of the vertices in V^i according to their unbalance value; that is, $\mathrm{ub}(v_{(k)}) \geq \mathrm{ub}(v_{(k+1)})$. Consider the following partition (V_1^i, V_2^i) of V^i: $V_1^i \stackrel{\rm def}{=} \{v_{(1)}, \ldots, v_{(\beta^i L)}\}$, and $V_2^i \stackrel{\rm def}{=} \{v_{(\beta^i L+1)}, \ldots, v_{(L)}\}$. Let $(H_1^i, H_2^i) \stackrel{\rm def}{=} (W_1^{i-1} \cup V_1^i, W_2^{i-1} \cup V_2^i)$. Then the number of cut edges in $(\mathrm{H}_1^i, \mathrm{H}_2^i)$ between vertices in W^{i-1} and vertices in V^i is at least as large as in $(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1})$. This is true since our partition of V^i, by definition, maximizes the number of such cut edges among all partitions which are a hybrid between $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$, and a partition of V^i into two subsets of size $\beta^i L$ and $(1-\beta^i)L$ respectively. The decrease in the size of the cut is hence at most $|\mathrm{V}^i|^2 = O(\frac{\epsilon}{\ell}N^2)$.

We next remove the assumptions that we know β^i as well as $\mathrm{ub}(v)$, for every $i \in \{1,...,\ell\}$ and $v \in \mathrm{V}^i$. Firstly, we note that approximations (up to $O(\epsilon)$) to these values are good enough. Actually, we can afford having bad approximations of $\mathrm{ub}(v)$ for $O(\epsilon)$ fraction of the vertices in V^i. As in the Max-CUT partitioning algorithm, we use sample sets U^i to obtain the approximations $\widehat{\mathrm{ub}}(v) = \mathrm{ub}(v) \pm O(\epsilon)$. The approximations $\hat\beta^i = \beta^i \pm O(\epsilon)$ are obtained by simply trying all integer multiples of $\epsilon/16$ which sum up to $\frac{1}{2}$.[9] Each possible setting of $\hat\beta^1,\ldots,\hat\beta^\ell$ and sequence of partitions of $\mathrm{U}^1,\ldots,\mathrm{U}^\ell$ gives rise to a different bisection of V, and we choose the resulting bisection whose cut is maximized.

We show that with high probability over the choice of the U^i's, there exist partitions of these sets, and there always exists a setting of the β^i's, so that at least one of the resulting bisection is close to having the maximum number of crossing edges. In particular, let $\hat v_{(1)},\ldots,\hat v_{(L)}$ be an ordering of the vertices in V^i according to $\widehat{\mathrm{ub}}(v)$. As we prove in Lemma 3.5.1, if we put the vertices $\{\hat v_{(1)},\ldots,\hat v_{(\lceil\hat\beta^i L\rceil)})\}$ on side 1, and the vertices $\{\hat v_{(\lceil\hat\beta^i L\rceil+1)},\ldots,\hat v_{(L)}\}$ on side 2, then the number of crossing edges in the resulting hybrid partition is not much smaller than that defined by the previous hybrid partition.

A detailed description of the graph bisection algorithm is given in Figure 3.4, and its formal analysis is provided in Lemma 3.5.1.

Graph Bisection Algorithm

1. *Choose $\ell = \frac{4}{\epsilon}$ sets $\mathrm{U}^1,\ldots,\mathrm{U}^\ell$ each of size $t = \Theta\left(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon\delta}\right)$, where U^i is chosen uniformly in $\mathrm{V}\setminus\mathrm{V}^i$. Let U be the union of these sets.*
2. *For each of the partitions $\Pi = \left(\bigcup_{i=1}^\ell \mathrm{U}_1^i, \bigcup_{i=1}^\ell \mathrm{U}_2^i\right)$ of U and for each of the ℓ-tuples $\hat\beta = \left(\hat\beta^1,\ldots,\hat\beta^\ell\right)$, where each $\hat\beta^i \in [0,1]$ is an integer multiple of $\frac{\epsilon}{16}$, and $\sum_i \hat\beta^i = 1/2$, construct a bisection $\left(\mathrm{V}_1^{\Pi,\hat\beta}, \mathrm{V}_2^{\Pi,\hat\beta}\right)$ as follows:*
   - *(a) For $i = 1\ldots\ell$ do:*
     - *i. For each $v \in \mathrm{V}^i$ let $\widehat{\mathrm{ub}}(v) \stackrel{\rm def}{=} \frac{1}{t}\left(|\Gamma(v)\cap U_2^i| - |\Gamma(v)\cap U_1^i|\right)$;*
     - *ii. Let $\hat v_{(1)},\ldots,\hat v_{(L)}$ be an ordering of the vertices in V^i such that $\widehat{\mathrm{ub}}(\hat v_{(k)}) \geq \widehat{\mathrm{ub}}(\hat v_{(k+1)})$ (ties are broken according to lexicographical order).*
     - *iii. $\mathrm{V}_1^i \leftarrow \{\hat v_{(1)},\ldots,\hat v_{(\lceil\hat\beta^i L\rceil)}\}$, and $\mathrm{V}_2^i \leftarrow \{\hat v_{(\lceil\hat\beta^i L\rceil+1)},\ldots,\hat v_{(L)}\}$*
   - *(b) Let $\mathrm{V}_1^{\Pi,\hat\beta} = \bigcup_{i=1}^\ell \mathrm{V}_1^i$, and let $\mathrm{V}_2^{\Pi,\hat\beta} = \bigcup_{i=1}^\ell \mathrm{V}_2^i$.*
3. *Among all bisections, $\left(\mathrm{V}_1^{\Pi,\hat\beta}, \mathrm{V}_2^{\Pi,\hat\beta}\right)$, let $(\mathrm{V}_1^{\mathrm{U}}, \mathrm{V}_2^{\mathrm{U}})$ be the one with maximum number of crossing edges.*

Figure 3.4: Graph Bisection algorithm

[9]This is always possible in case $1/\epsilon$ is an integer. Otherwise, we can try all integer multiples of $\epsilon'/16$ which sum up to $1/2$, where $\epsilon' = 1/(\lceil 1/\epsilon\rceil)$. Since $\epsilon' > \epsilon/2$, for simplicity we assume that $1/\epsilon$ is in fact an integer.

Lemma 3.5.1 *Let* $(\mathrm{H}_1, \mathrm{H}_2)$ *be a fixed bisection of* $\mathrm{V(G)}$. *Then, with probability at least* $1 - \delta/2$ *over the choice of* U, *there exists a partition* Π *of* U, *and an* ℓ*-tuple* $\hat{\beta}$, *such that* $\mu(\mathrm{V}_1^{\Pi,\hat{\beta}}, \mathrm{V}_2^{\Pi,\hat{\beta}}) \geq \mu(\mathrm{H}_1, \mathrm{H}_2) - \frac{3}{4}\epsilon$.

Proof: For a given set U, a fixed partition Π of U, and a fixed $\hat{\beta}$, let the i^{th} hybrid partition $(\mathrm{H}_1^i, \mathrm{H}_2^i)$ determined by Π and $\hat{\beta}$ be as defined in Lemma 3.4.1. ¿From this point on, let $\hat{\beta}$ be such that for every i, $\hat{\beta}^i = \beta^i \pm \epsilon/16$ where $\beta^i \stackrel{\mathrm{def}}{=} |\mathrm{H}_1 \cap \mathrm{V}^i|$. The existence of such $\hat{\beta}^i$ follows from the resolution of the values taken by $\hat{\beta}^i$ and the fact that all possibilities (to within this resolution) were tried. Similarly to what was observed in Lemma 3.4.1, for $\hat{\beta}$ fixed as above, an i^{th} hybrid partition is actually determined by the partitions of $\mathrm{U}^1, \ldots, \mathrm{U}^i$, and does not depend on the partitions of $\mathrm{U}^{i+1}, \ldots, \mathrm{U}^\ell$. Similarly to the analysis of the graph partition algorithm for Max-CUT, we shall show that for every $1 \leq i \leq \ell$, and for a fixed choice and partitions of $\mathrm{U}^1, \ldots, \mathrm{U}^{i-1}$, with probability at least $1 - \frac{\delta}{2\ell}$ over the choice of U^i, there exists a partition $(\mathrm{U}_1^i, \mathrm{U}_2^i)$ of U^i such that

$$\mu\left(\mathrm{H}_1^i, \mathrm{H}_2^i\right) \geq \mu\left(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1}\right) - \frac{3\epsilon}{4\ell}\,. \tag{3.14}$$

By induction on i we have that the ℓ^{th} hybrid partition (which is necessarily a bisection since $\sum_i \hat{\beta}^i = 1/2$) has a cut whose size is at most $\frac{3\epsilon}{4}N^2$ smaller than the 0 hybrid partition.

Let us define a *good* sample set U^i and a *representative* partition $(\mathrm{U}_1^i, \mathrm{U}_2^i)$, similarly to the way they were defined in Lemma 3.4.1 except that here we make the slightly stronger requirement that for all but $\frac{\epsilon}{16}$ of the vertices v in V^i

$$\text{For each } j \in \{1,2\}, \qquad \frac{\left|\Gamma(v) \cap \mathrm{U}_j^i\right|}{t} = \frac{\left|\Gamma(v) \cap \mathrm{W}_j^{i-1}\right|}{N} \pm \frac{\epsilon}{64} \tag{3.15}$$

As was shown in the proof of Lemma 3.4.1, for our choice of $t = |\mathrm{U}^i|$, with probability at least $1 - \delta/2$, $\mathrm{U}^1, \ldots, \mathrm{U}^\ell$ are good with respect to the respective partitions. Assume from now on that $\mathrm{U}^1, \ldots, \mathrm{U}^\ell$ are good and for each i let $\widehat{\mathrm{ub}}(v)$ be defined with respect to the representative partition of U^i.

For a fixed i, let $v_{(1)}, \ldots, v_{(L)}$ be the ordering of the vertices in V^i according to their (correct) unbalance value $\mathrm{ub}(v)$ with respect to $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$. As was noted previously, if we put $v_{(1)}, \ldots, v_{(\beta^i L)}$ on side 1 (and the rest of the vertices on side 2), then the number of cut edges between W^{i-1} and V^i does not decrease, and the decrease in the size of the cut is at most $|\mathrm{V}^i|^2 = L^2 = \frac{\epsilon}{4\ell}N^2$. We refer to this partition of V^i as the *ideal* partition, and bound the additional decrease in the size of the cut due to the change in the number of edges between W^{i-1} and V^i in the actual partition of V^i constructed by the algorithm. We do this by comparing the actual partition to the ideal one. Assume first that $\beta^i = \hat{\beta}^i$, and let Y^i be the set of *misplaced* vertices in V^i which are put on a different side in $(\mathrm{H}_1^i, \mathrm{H}_2^i)$ than they are put in the ideal partition described above (given the correct unbalance values of the vertices). We shall show below that in such a case all but $\frac{\epsilon}{16}L$ of the vertices in Y^i have approximately the same unbalance value (i.e., within $\frac{\epsilon}{16}N$). This will imply the following bound on the decrease in the number of cut edges between W^{i-1} and V^i.

By our assumption that $\hat{\beta}^i = \beta^i$ (which we remove momentarily) we know that in the actual algorithm we put exactly the same number of vertices on each side of the partition of V^i as in the ideal partition. It follows that the number of misplaced vertices on each side of the partition is the same, and we can pair the misplaced vertices and view these pairs as having switched sides. Whenever we switch sides between pairs of vertices whose unbalance value differs by at most $\frac{\epsilon}{16}N$, the decrease in the number of cut edges between these two vertices and vertices in W^{i-1} is at most

$\frac{\epsilon}{8}$. The contribution of all such pairs is at most $\frac{L}{2} \cdot \frac{\epsilon}{8} N = \frac{\epsilon}{16\ell} N^2$. The number of cut edges between W^{i-1} and the at most $\frac{\epsilon}{16} L$ vertices in Y^i which differ significantly in their unbalance value from the rest, decreases by at most $\frac{\epsilon}{32} L \cdot 4N = \frac{\epsilon}{8\ell} N^2$. Finally, if we do not assume that $\hat{\beta}^i = \beta^i$, then since $|\hat{\beta}^i - \beta^i| \leq \frac{\epsilon}{16}$, there are at most $\frac{\epsilon}{16} L$ additional misplaced vertices which cause an additional decrease of at most $\frac{\epsilon}{16} L \cdot 2N = \frac{\epsilon}{8\ell} N^2$ in the size of the cut. Summing all contributions we get that the total decrease in the size of the cut is bounded by $\frac{3\epsilon}{4\ell} N^2$, as required.

It remains to prove our claim concerning the set of misplaced vertices, Y^i under the assumption that $\hat{\beta}^i = \beta^i$. Consider a grouping of the vertices in V^i into $K = 1/\epsilon'$ *unbalance bins* according to their (correct) unbalance value where $\epsilon' = \epsilon/32$. For $k = -1, \ldots, K-1$, the kth bin, denoted B_k, is defined as follows:

$$B_k \stackrel{\text{def}}{=} \{v \in \mathrm{V}^i : \mathrm{ub}(v) \in [k \cdot \epsilon' N, (k+1) \cdot \epsilon' N)\} \ .$$

Let g be the index of the bin which $v_{(\beta^i L)}$ belongs to. By definition of the bins, all vertices in B_g have approximately the same unbalance value. Since we only have approximations of the unbalance values we also group the vertices according to their approximated unbalance values. Namely, For $k = -1, \ldots, K-1$,

$$\widehat{B}_k \stackrel{\text{def}}{=} \left\{v \in \mathrm{V}^i : \widehat{\mathrm{ub}}(v) \in [k \cdot \epsilon' N, (k+1) \cdot \epsilon' N)\right\} \ ,$$

By our assumption on the representativeness of $(\mathrm{U}_1^i, \mathrm{U}_2^i)$, at most $\frac{\epsilon}{16}$ of the vertices in V^i belong to a bin $\widehat{B}_{k'}$ whose index differs by more than 1 from their correct bin B_k (and vertices in the same, or in neighboring bins have approximately the same unbalance value).

Let $\hat{v}_{(1)}, \ldots, \hat{v}_{(L)}$ be an ordering of the vertices in V^i according to their approximate unbalance value $\widehat{\mathrm{ub}}(v)$, and let $\hat{g}$ be the index of the bin which $\hat{v}_{(\hat{\beta}^i L)}$ belongs to. It follows that if $\hat{g} = g \pm 1$, then all but $\frac{\epsilon}{16} L$ of the misplaced vertices have unbalance value $(g \pm 2\epsilon')N = gN \pm \frac{\epsilon}{16} N$, as required. Thus assume that $\hat{g} \geq g + 2$ (the case $\hat{g} \leq g - 2$ is analogous). In such a case, necessarily, all but $\frac{\epsilon}{16} L$ of the vertices $v_{(1)}, \ldots, v_{(\beta^i L)}$ (which belong to bins $B_1, \ldots, B_g$), are put on side 1 (as they should). But since we put only $\hat{\beta}^i L = \beta^i L$ vertices on side 1, the total number of misplaced vertices is bounded by $\frac{\epsilon}{16} L$. ■

3.5.2 The Bisection Approximation Algorithm

Similarly to the Max-CUT approximation algorithm, the Bisection Approximation Algorithm (described in Figure 3.5) performs the same steps as the algorithm described in Figure 3.4, but does so only on a small sample S. The analysis of this Bisection approximation algorithm given the correctness of the graph bisection algorithm, is the same as that of the Max-CUT approximation algorithm (Lemma 3.4.3) except for the following detail. Here we need to take into account that it is not necessarily the case that for a given U, a partition Π of U and $\hat{\beta}$, for each $s \in \mathrm{S}$, $s \in \mathrm{S}_j^{\Pi, \hat{\beta}}$, i.f.f. $s \in \mathrm{V}_j^{\Pi, \hat{\beta}}$. This is due to to the possibility that for some vertices $v \in \mathrm{S}^i$, vertex v appears before (resp., after) the $\hat{\beta}^i N$th vertex in the ordering of V^i, but after (resp., before) the $\hat{\beta}^i m$th vertex in the ordering of S^i. However, by applying arguments analogous to Lemma 3.3.5, it can be shown that the fraction of such vertices is small, and that their contribution to the approximation error is small too. Part (1) of Theorem 3.9 follows.

3.5.3 The improved Graph Bisection Algorithm

The improved graph bisection algorithm (whose running time is as stated in Theorem 3.9, Part (2)) starts by invoking the Bisection approximation algorithm of Figure 3.5, and recording the set U

Bisection Approximation Algorithm

1. *As Step (1) of Figure 3.4.*
2. *Uniformly choose a set* $\mathrm{S} = \{s_1, \ldots, s_m\}$ *of size* $m = \theta\left(\frac{\ell \cdot t + \log(1/\delta)}{\epsilon^2}\right)$. *For* $1 \le i \le \ell$, *let* $\mathrm{S}^i \stackrel{\mathrm{def}}{=} \mathrm{V}^i \cap \mathrm{S}$.
3. *Analogously to Step (2) of Figure 3.4, for each of the partitions* $\Pi = \left(\bigcup_{i=1}^{\ell} \mathrm{U}_1^i, \bigcup_{i=1}^{\ell} \mathrm{U}_2^i\right)$ *of* U, *and for each of the* ℓ*-tuples* $\hat{\beta} = \left(\hat{\beta}^1, \ldots, \hat{\beta}^\ell\right)$, *construct a bisection* $\left(\mathrm{S}_1^{\Pi,\hat{\beta}}, \mathrm{S}_2^{\Pi,\hat{\beta}}\right)$ *of* S. *Specifically, in the* i^{th} *iteration of Substep (a),* S_1^i *is assigned the* $\lceil \hat{\beta}^i m/\ell \rceil$ *vertices with the biggest* $\widehat{\mathrm{ub}}(\cdot)$ *value.*
4. *Among all partitions* $\left(\mathrm{S}_1^{\Pi,\hat{\beta}}, \mathrm{S}_2^{\Pi,\hat{\beta}}\right)$, *let* $(\mathrm{S}_1^{\mathrm{U}}, \mathrm{S}_2^{\mathrm{U}})$ *be the one which maximizes the number of cut edges between pairs of vertices* (s_{2k-1}, s_{2k}). *Namely, define*

$$\hat{\mu}\left(\mathrm{S}_1^{\Pi,\hat{\beta}}, \mathrm{S}_2^{\Pi,\hat{\beta}}\right) \stackrel{\mathrm{def}}{=} \frac{\left|\left\{(s_{2k-1}, s_{2k}) \in \mathrm{E(G)} : \ \textit{for } j \neq j',\ s_{2k-1} \in \mathrm{S}_j^{\Pi,\hat{\beta}},\ \&\ s_{2k} \in \mathrm{S}_{j'}^{\Pi,\hat{\beta}}\right\}\right|}{m/2}.$$

Output $\hat{\mu}^{\frac{1}{2},\frac{1}{2}}(\mathrm{G}) = \max_{\pi,\hat{\beta}} \hat{\mu}\left(\mathrm{S}_1^{\Pi,\hat{\beta}}, \mathrm{S}_2^{\pi(\mathrm{U},\hat{\beta})}\right)$.

Figure 3.5: Bisection Approximation Algorithm

uniformly selected in Step (1), the partition $\Pi = \pi(\mathrm{U})$ selected in Step (4), and the ℓ-tuple $\hat{\beta}$ used for it. Using these specific Π and $\hat{\beta}$, the algorithm executes a single iteration of Step (2) of the Graph Bisection Algorithm (of Figure 3.4). It is easy to see that the resulting algorithm satisfies Part (2) of Theorem 3.9.

3.5.4 Variations

An easy modification suffices for finding (resp., approximating the size of) a nearly minimum bisection rather than a nearly maximum one. In each iteration of the algorithm(s), instead of placing in side 1 the first $\hat{\beta}^i$ vertices in decreasing order of (approximate) unbalance, we would do the opposite. Namely, since we would like to minimize the size of the cut, we try and put vertices on the size opposite the *minority* of their neighbors. While we might not be able to do so for all vertices (since we are restricted to constructing a bisection), analogously to the maximization problem, there exists one side in which all vertices have a smaller (i.e., more negative) unbalance value than all those on the other side. Thus, in the i^{th} iteration we order all vertices in V^i (or S^i in the approximation algorithm) according to *increasing* unbalance value $\widehat{\mathrm{ub}}(v)$ (where $\widehat{\mathrm{ub}}(v)$ is as defined in the maximization algorithms), and put the first $\hat{\beta}^i$ vertices on side 1 and the rest on side 2.

We can easily generalize the algorithms to construct (resp., approximate the size of) partitions with other given proportion of vertices on each side. A key observation is that the main steps of our algorithms are oblivious of the bisection requirement. In fact, the main steps produce partitions with maximum (resp., minimum) number of crossing edges per each proportion of vertices on each side (up to some resolution). Thus, all we need to do is modify the last step of these algorithms so that they select the best partition among those with specific vertex proportions (or alternatively, first modify each partition to have the desired vertex proportion). Thus, for example, we can approximate quantities such as $\mathrm{opt}_{|\mathrm{V}_1|=N/3}\{\mu(\mathrm{V}_1, \mathrm{V(G)} \setminus V_1)\}$ or $\mathrm{opt}_{\frac{N}{3} \le |\mathrm{V}_1| \le 2\frac{N}{3}}\{\mu(\mathrm{V}_1, \mathrm{V(G)} \setminus V_1)\}$, where $\mathrm{opt} \in \{\max, \min\}$.

3.6 The General Partition Problem

The following framework of a general partition problem generalizes all properties considered in previous sections. In particular, it captured any graph property which requires the existence of partitions satisfying certain fixed density constraints. These constraints may refer both to the number of vertices on each side of the partition and to the number of edges between each pair of sides.

Let $\Phi \stackrel{\rm def}{=} \left\{\rho_j^{\rm lb}, \rho_j^{\rm ub}\right\}_{j=1}^{k} \cup \left\{\varrho_{j,j'}^{\rm lb}, \varrho_{j,j'}^{\rm ub}\right\}_{j,j'=1}^{k}$ be a set of non-negative parameters so that $\rho_j^{\rm lb} \leq \rho_j^{\rm ub}$ $(\forall j)$ and $\varrho_{j,j'}^{\rm lb} \leq \varrho_{j,j'}^{\rm ub}$ $(\forall j, j')$. Let $\mathcal{GP}_\Phi$ be the class of graphs which have a k-way partition $({\rm V}_1, \ldots, {\rm V}_k)$ with the following properties[10]:

$$\forall j,\;\; \rho_j^{\rm lb} \cdot N \;\leq\; |{\rm V}_j| \;\leq\; \rho_j^{\rm ub} \cdot N\;, \tag{3.16}$$

and

$$\forall j, j'\;\; , \varrho_{j,j'}^{\rm lb} \cdot N^2 \;\leq\; |{\rm E}({\rm V}_j, {\rm V}_{j'})| \;\leq\; \varrho_{j,j'}^{\rm ub} \cdot N^2\;, \tag{3.17}$$

where ${\rm E}({\rm V}_j, {\rm V}_{j'})$ is the set of edges between vertices in ${\rm V}_j$ and vertices in ${\rm V}_{j'}$ (where we include edges going in both directions). That is, Eq. (3.16) places lower and upper bounds on the relative sizes of the various parts; whereas Eq. (3.17) imposes lower and upper bounds on the density of edges among the various pairs of parts. (LB stands for Lower Bound, and UB stands for Upper Bound.)

In this section we describe a testing algorithm for the class $\mathcal{GP}_\Phi$ (for any given set of parameters $\Phi = \{\rho_j^{\rm xx}\} \cup \{\varrho_{j,j'}^{\rm xx}\}$). The testing algorithm is based on a randomized *partitioning* algorithm for the related partition problem. Namely, given a graph G, a set of parameters Φ, an approximation parameter ϵ and a confidence parameter δ, so that G has a k-way partition which obeys Equations (3.16) and (3.17), the partitioning algorithm constructs a partition $({\rm V}_1, \ldots, {\rm V}_k)$ of G for which the following holds with probability at least $1 - \delta$:

$$\forall j,\;\; (\rho_j^{\rm lb} - \epsilon) \cdot N \;\leq\; |{\rm V}_j| \;\leq\; (\rho_j^{\rm ub} + \epsilon) \cdot N, \tag{3.18}$$

and

$$\forall j, j',\;\; (\varrho_{j,j'}^{\rm lb} - \epsilon) \cdot N^2 \;\leq\; |{\rm E}({\rm V}_j, {\rm V}_{j'})| \;\leq\; (\varrho_{j,j'}^{\rm ub} + \epsilon) \cdot N^2\;, \tag{3.19}$$

A partition obeying (3.18) and (3.19) is called an *ϵ-approximation* for the partitioning problem defined by the set of parameters Φ.

As stated above, all properties considered in previous sections can be casted as special cases of the general partition problem. For example, k-colorability is expressed by setting $\varrho_{j,j}^{\rm ub} = 0$ for every j (and setting $\rho_j^{\rm lb} = 0$, $\rho_j^{\rm ub} = 1$, and similarly setting the $\varrho_{j,j'}^{\rm xx}$'s for $j' \neq j$). In case we are interested in maximizing or minimizing a parameter (e.g. maximizing ${\rm E}({\rm V}_1, {\rm V}_2)$ in the case of Max-CUT) we can simply run the general partitioning (resp., testing) algorithm on all values of this parameter which are multiples of ϵ, and find the maximum/minimum value attainable.[11] However, as can be

[10] Actually, to avoid integrability problems, we consider generalized (or fractional) k-way partitions in which up to $k-1$ vertices may be split among several parts. Had we not followed this convention, the set of N-vertex graphs in $\mathcal{GP}_\Phi$ could be empty for some values of N and non-empty for others. For example, if $\rho_1^{\rm lb} = \rho_1^{\rm ub} = 1/3$ then only graphs with $3N'$ vertices may be in $\mathcal{GP}_\Phi$. In such a case, the tester must reject any graph on $3N'+1$ vertices (as it is 1-away from $\mathcal{GP}_\Phi$) whereas it must accept some $3N'$-vertex graphs, and consequently it must count the number of vertices in the graph. These integrability problems have nothing to do with the combinatorial structure which we wish to investigate and thus we avoid them by taking this somewhat unnatural convention.

[11] Actually, our partitioning algorithm works by producing a set of partitions of the graph vertices and then searching among them for one which is an ϵ-approximation of the partitioning problem. The testing algorithm runs a similar procedure on a sample set of vertices. The procedure for producing these partitions depends on k, ϵ, and δ, but not on the particular set of parameters Φ, and therefore we do not actually need to run the algorithm more than once.

seen from our theorems below, this generality has a price: the query complexity and running times of our algorithms (for the general partition problem) are quite large. More efficient algorithms for specific problems such as Max-Clique, Max-CUT, and Bisection, were presented in previous sections. Though some of the ideas used in the above algorithms are also applied here, we choose to present the algorithms in this section in a self-contained manner. Throughout the rest of this section, we always assume that the parameters in Φ are given to within an accuracy of ϵ.

Theorem 3.11 *There exists an algorithm $\mathcal{A}$ such that for every given set of parameters Φ, algorithm $\mathcal{A}$ is a property testing algorithm for the class $\mathcal{GP}_\Phi$ with query complexity $\left(\frac{O(k^2)}{\epsilon}\right)^{k+5} \cdot k\log^2(k/\epsilon\delta)$, and running time $\exp\left(\left(\frac{O(k^2)}{\epsilon}\right)^{k+1} \cdot k\log(k/\epsilon\delta)\right)$.*

Theorem 3.12 *There exists a graph partitioning algorithm that on input* G, Φ, ϵ, *and* δ, *runs in time* $\exp\left(\left(\frac{O(1)}{\epsilon}\right)^{k+1} \cdot k\log(k/\epsilon\delta)\right) + O\left(\frac{1}{\epsilon^2}\log(k/\epsilon\delta)\right)\cdot N$, *and if* G *has a k-way partition satisfying Equations (3.16) and (3.17) then with probability at least* $1-\delta$ *the graph partitioning algorithm outputs a partition which satisfies Equations (3.18) and (3.19).*

We start by describing the graph partitioning algorithm. The running time of this algorithm, as described below is is $\exp\left(\left(\frac{1}{\epsilon}\right)^{k+1} k\log(k/(\epsilon\delta))\right)\cdot N^2$. However, as we later show, by first running the testing algorithm, with high probability we can construct such a partition (if one exists) in time as stated in Theorem 3.12.

3.6.1 High Level Description of the Partitioning Algorithm

The algorithm is based on the following observation. Let $\mathcal{H} = (\mathrm{H}_1, \ldots, \mathrm{H}_k)$ be any fixed partition of V. In particular, we may want to consider a partition which obeys Equations (3.16) and (3.17). Let X be a set of vertices of small size (i.e., of size $O(\epsilon N)$) and suppose that all vertices in X have approximately the same (i.e., $\pm O(\epsilon N)$) number of neighbors in each $\mathrm{H}_j \setminus \mathrm{X}$ (i.e., $|\Gamma(u) \cap (H_j \setminus \mathrm{X})| = |\Gamma(v) \cap (H_j \setminus \mathrm{X})| \pm O(\epsilon N)$, for every $u, v \in \mathrm{X}$ and j). The observation is that if we redistribute the vertices of X among the k parts (i.e., H_j's) while maintaining the number of vertices in each part, then the number of edges between every pair of parts is approximately maintained. That is, let $(\mathrm{X}_1, \ldots, \mathrm{X}_k)$ be an arbitrary partition of X so that $|\mathrm{X}_j| = |\mathrm{X} \cap \mathrm{H}_j| \pm O(\epsilon^2 N)$, and let $\mathrm{H}'_j = (\mathrm{H}_j \setminus \mathrm{X}) \cup \mathrm{X}_j$. Then, $||\mathrm{H}_j| - |\mathrm{H}'_j|| = O(\epsilon^2 N)$ and furthermore $||\mathrm{E}(\mathrm{H}_j, \mathrm{H}_{j'})| - |\mathrm{E}(\mathrm{H}'_j, \mathrm{H}'_{j'})|| = O(|\mathrm{X}| \cdot \epsilon N + |\mathrm{X}|^2) = O(\epsilon^2 N)$, for every j, j', where the last equality is due to the size of X. The bound on the difference of the number of edges follows by considering the three types of edges. Note first that edges with both endpoints NOT in X are on the same side in $\mathcal{H}'$ as they were in $\mathcal{H}$, and thus are not effected by the redistribution of X. The number of edges with both endpoints in X is bounded by $|\mathrm{X}|^2$. Finally we get to the interesting edges; that is, those with exactly one endpoint in X. Here we use the hypothesis that all vertices in X have about the same "neighboring profile" with respect to $\mathcal{H} \setminus \mathrm{X}$ and thus switching vertices of X among the parts (while maintaining the number of vertices in each part) does not effect the edge-count (among the various parts) by much. In particular, each vertex in X may contribute a change of at most $O(\epsilon N)$ and thus the claim follows. Note that it suffices to have *almost* all vertices in X (i.e., all but $O(\epsilon|\mathrm{X}|)$ of the vertices) have approximately the same "neighboring profile" with respect to $\mathcal{H} \setminus \mathrm{X}$.

Going with this observation one step further, let $\mathcal{H}$ be as defined above, let Y be any given set of vertices of size $O(\epsilon N)$, and let $\mathcal{W} = (\mathrm{W}_1, \ldots, \mathrm{W}_k)$ be defined by $\mathrm{W}_j \stackrel{\mathrm{def}}{=} \mathrm{H}_j \setminus \mathrm{Y}$ (for each j). Let us first *cluster* the vertices in Y according to the number of neighbors they have in each W_j.

That is, in each (disjoint) cluster all vertices have approximately the same number of neighbors in each W_j. Suppose we now partition the vertices in each cluster X into k parts, $(\mathrm{X}_1, \ldots, \mathrm{X}_k)$ in an arbitrary way so that the number of vertices in each X_j is approximately $|\mathrm{X} \cap \mathrm{H}_j|$, and add each X_j to W_j, defining a hybrid partition, $\mathcal{H}' = (\mathrm{H}'_1, \ldots, \mathrm{H}'_k)$. Then the previous argument (for a single cluster) easily generalizes to show that $\mathcal{H}'$ has approximately the same vertex and edge densities as $\mathcal{H}$.

The Oracle Aided Procedure In view of the above observation, we are almost ready to describe the partitioning algorithm. We first describe a mental experiment in which we assume the algorithm has access to certain oracles (which it actually does not have direct access to). We later show how we can approximately simulate these oracles. The algorithm works in ℓ iterations, where $\ell = \frac{4}{\epsilon}$. In the i^{th} iteration the algorithm considers a fixed subset of N/ℓ vertices, denoted V^i, and produces a k-way partition of V^i, denoted $(\mathrm{V}^i_1, \ldots, \mathrm{V}^i_k)$. The subsets $\mathrm{V}^1, \ldots, \mathrm{V}^\ell$ are defined according to (increasing) lexicographical order of the vertices. The final partition constructed, $(\mathrm{V}_1, \ldots, \mathrm{V}_k)$, is simply defined by $\mathrm{V}_j \stackrel{\text{def}}{=} \bigcup_{i=1}^{\ell} \mathrm{V}^i_j$, for each $j \in \{1, \ldots, k\}$.

Let $\mathcal{H}^0 = (\mathrm{H}^0_1, \ldots, \mathrm{H}^0_k)$ be a k-way partition which obeys Equations (3.16) and (3.17). Let $\mathcal{W}^0 = (\mathrm{W}^0_1, \ldots, \mathrm{W}^0_k)$ be the partition induced on $\mathrm{V} \setminus \mathrm{V}^1$ by $\mathcal{H}^0$. That is, for each j, $\mathrm{W}^0_j \stackrel{\text{def}}{=} \mathrm{H}^0_j \setminus \mathrm{V}^1$. For any given vertex $v \in \mathrm{V}^1$ and for every $j \in \{1, \ldots, k\}$, let $\gamma_j(v) \stackrel{\text{def}}{=} \frac{|\Gamma(v) \cap \mathrm{W}^0_j|}{N}$. That is, $\gamma_j(v)$ is the fraction of neighbors v has in W^0_j (normalized by N, the maximum number of neighbors v may have).

Assume we had an oracle which, given $v \in \mathrm{V}^i$ and j, returns a value $\hat{\gamma}_j(v)$ so that for all but an $O(\epsilon)$ fraction of the vertices v in V^1 it holds that for every j, $\hat{\gamma}_j(v) = \gamma_j(v) \pm \epsilon/32$. Using this oracle we could cluster the vertices in V^1 according to the number of neighbors they have on each side of $\mathcal{W}^0$ as approximated by the oracle: For every possible $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_k \rangle$ where each α_j ranges over all integer multiples of $\epsilon/16$, let

$$\mathrm{V}^{1,\vec{\alpha}} \stackrel{\text{def}}{=} \{v \in V^1 : \forall j,\ \alpha_j - \epsilon/32 < \hat{\gamma}_j(v) \le \alpha_j + \epsilon/32\} \ .$$

We refer to the $\vec{\alpha}$'s as the *cluster names*, since each $\vec{\alpha}$ uniquely defines a different cluster of V^1 (or similarly, of any given set of vertices).

Assume further that the algorithm also had access to an oracle which for every cluster $\mathrm{V}^{1,\vec{\alpha}}$ and for each j, returns an approximation, up to an error of $\epsilon/16$, of the fraction of vertices in $\mathrm{V}^{1,\vec{\alpha}}$ which belong to H^0_j. Let this approximate fraction be denoted $\beta_j^{1,\vec{\alpha}}$. We refer to $\langle \beta_1^{1,\vec{\alpha}}, \ldots, \beta_1^{1,\vec{\alpha}} \rangle$ as the *quantitative partition* of $\mathrm{V}^{1,\vec{\alpha}}$, since it only determines how many vertices from $\mathrm{V}^{1,\vec{\alpha}}$ should be on each side of the partition (and does not specify which vertices should be on each side). However, by the above observation, the quantitative partition of $\mathrm{V}^{1,\vec{\alpha}}$ is all that matters, and we may as well partition $\mathrm{V}^{1,\vec{\alpha}}$ in an arbitrary way as long as the quantitative partition is satisfied. Let $\left(\mathrm{V}_1^{1,\vec{\alpha}}, \ldots, \mathrm{V}_k^{1,\vec{\alpha}}\right)$ be such a partition; that is, $\lfloor \beta_j^{1,\vec{\alpha}} |\mathrm{V}^{1,\vec{\alpha}}| \rfloor \le |\mathrm{V}_j^{1,\vec{\alpha}}| \le \lceil \beta_j^{1,\vec{\alpha}} |\mathrm{V}^{1,\vec{\alpha}}| \rceil$, for every j.

Let $(\mathrm{V}^1_1, \ldots, \mathrm{V}^1_k)$ be defined by $\mathrm{V}^1_j \stackrel{\text{def}}{=} \bigcup_{\vec{\alpha}} \mathrm{V}_j^{1,\vec{\alpha}}$, for each j. Consider the partition $\mathcal{H}^1 = (\mathrm{H}^1_1, \ldots, \mathrm{H}^1_k)$, where $\mathrm{H}^1_j \stackrel{\text{def}}{=} \mathrm{W}^0_j \cup \mathrm{V}^1_j$, for each j. We view $(\mathrm{H}^1_1, \ldots, \mathrm{H}^1_k)$ as a hybrid of the partition $(\mathrm{W}^0_1, \ldots, \mathrm{W}^0_k)$ (of $\mathrm{V} \setminus \mathrm{V}^1$), and the partition $(\mathrm{V}^1_1, \ldots, \mathrm{V}^1_k)$ (of V^1), and refer to it as the 1st *hybrid partition*. Combining our hypothesis concerning $\mathcal{H}^0$ (i.e., that $\mathcal{H}^0$ obeys Equations (3.16) and (3.17)) and our observation above (which can be applied since $|\mathrm{V}^1| = O(\epsilon N)$), we conclude that $\mathcal{H}^1$ is an $O(\epsilon^2)$-approximation of the partitioning problem as defined by Equations (3.16) and (3.17).

We have just described how the algorithm (with the aid of oracles) produces a k-way partition of V^1. In general, in the i^{th} iteration, the algorithm performs an analogous procedure for partitioning V^i. The starting point of the i^{th} iteration is the hybrid partition resulting from the previous iteration, which is denoted $\mathcal{H}^{i-1} = (\mathrm{H}_1^{i-1}, \ldots, \mathrm{H}_k^{i-1})$. Assume that for every $i \in \{1, \ldots, \ell\}$, in the $i^{\rm th}$ iteration we had access to the following two oracles. The first is an oracle for approximating $\gamma_j(v) \stackrel{\rm def}{=} \frac{1}{N} |\Gamma(v) \cap \mathrm{W}_j^{i-1}|$ for every given $v \in \mathrm{V}^i$ and j, where $\mathrm{W}_j^{i-i} \stackrel{\rm def}{=} \mathrm{H}_j^{i-i} \backslash \mathrm{V}^i$. The second is an oracle which for every cluster name $\vec{\alpha}$ and for every j, returns a fraction $\beta_j^{i,\vec{\alpha}}$ which approximates $|\mathrm{V}^{i,\vec{\alpha}} \cap \mathrm{H}_j^{i-1}|/N$ (up to $\epsilon/16$). Then we can partition each cluster of V^i as described for V^1 obtaining the $i^{\rm th}$ hybrid, denoted $\mathcal{H}^i$. That is, $\mathcal{H}^i$ is a hybrid of $(\mathrm{W}_1^{i-1}, \ldots, \mathrm{W}_k^{i-1})$, and the partition $(\mathrm{V}_1^i, \ldots, \mathrm{V}_k^i)$ (produced in the $i^{\rm th}$ iteration). We show that in each iteration the resulting hybrid partition does not differ by much from the previous hybrid partition in terms of the sizes of the different sides and the edge densities in and between the different sides. It follows that the final partition is an ϵ-approximation of the partitioning problem as defined by Equations (3.16) and (3.17).

Simulating the Oracles We next need to get rid of the oracles used in each iteration. It is not hard to see that we do not actually need an oracle to give us the $\beta_j^{i,\vec{\alpha}}$'s. This is true since i takes on $\ell = O(1/\epsilon)$ values, and for each i, there are $(O(1)/\epsilon)^k$ possible values of $\vec{\alpha}$. Finally, for each i, $\vec{\alpha}$, and $j \in \{1, \ldots, k\}$, there are $O(1/\epsilon)$ possible values of $\beta_j^{i,\vec{\alpha}}$. Thus, we can simply try all possible $((O(1)/\epsilon)^k)^{(O(1)/\epsilon)^k \cdot \ell)} < \exp\left((O(1)/\epsilon)^{k+1} \cdot k \cdot \log(1/\epsilon)\right)$ (vectors of) values for the quantitative partitions of the clusters. Each one gives rise to a different partition of V, and we can search among these partitions for an ϵ-approximation of the partitioning problem.

We now turn to the oracles for $\gamma_j(v)$. Consider the oracle needed in the first iteration. This oracle gives approximations $\hat{\gamma}_j(v)$, for $v \in \mathrm{V}^1$ with respect to $\mathcal{W}^0 = (\mathrm{W}_1^0, \ldots, \mathrm{W}_k^0)$, where $\mathrm{W}_j^0 \stackrel{\rm def}{=} \mathrm{H}_j^0 \setminus \mathrm{V}^1$, and $\mathcal{H}^0$ is assumed to be a partition which obeys Equations (3.16) and (3.17). Clearly we do not know of any such partition, or else we would be done. However, if we take a sample U of size $\mathrm{poly}((1/\epsilon)\log(k/\delta))$ chosen uniformly from $\mathrm{V} \setminus \mathrm{V}^1$, then with high probability the following holds. Let $(\mathrm{U}_1, \ldots, \mathrm{U}_k)$ be the partition of U defined by $\mathrm{U}_j \stackrel{\rm def}{=} \mathrm{U} \cap \mathrm{W}_j^0$ and for each $v \in \mathrm{V}^1$ let $\hat{\gamma}_j(v) \stackrel{\rm def}{=} |\Gamma(v) \cap \mathrm{U}_j|/|\mathrm{U}|$. That is $\hat{\gamma}_j(v)$ is the fraction of neighbors v has in U_j, among all vertices in U, and U_j is essentially a random sample of vertices from W_j^0. Then, with high probability over the choice of U, for most vertices v in V^1, $\hat{\gamma}_j(v)$ is a good approximation of $\gamma_j(v)$ (with respect to $(\mathrm{W}_1^0, \ldots, \mathrm{W}_k^0)$) for all j. However, we do not know which partition $(\mathrm{U}_1, \ldots, \mathrm{U}_k)$ to use, but this is not a problem since we may simply try all $(k^{|\mathrm{U}|})$ possible k-way partitions of U.

In general, in order to simulate all oracles which yield, for every i, v, and j, an approximate value $\hat{\gamma}_j(v)$ of $\gamma_j(v)$ (for vertices $v \in \mathrm{V}^i$ and with respect to with $\mathcal{W}^{i-1}$), we do the following. We choose ℓ sets, $\mathrm{U}^1, \ldots, \mathrm{U}^\ell$, each of size $t = \Theta((1/\epsilon^2)\log(k/\epsilon\delta))$, where U^i is chosen uniformly in $\mathrm{V} \setminus \mathrm{V}^i$. We then consider all possible k-way partitions of each U^i. For each possible sequence of partitions, $((\mathrm{U}_1^1, \ldots, \mathrm{U}_k^1), \ldots, (\mathrm{U}_1^\ell, \ldots, \mathrm{U}_k^\ell))$, (where there are $k^{\ell \cdot t} = \exp\left(O((1/\epsilon)^3 \cdot \log(k/\epsilon\delta)\log k)\right)$ such possible partitions), and for each possible setting of all the $\beta_j^{i,\vec{\alpha}}$'s, we run the (oracle aided) procedure described previously, where for $\hat{\gamma}_j(v)$ in iteration i we use $|\Gamma(v) \cap \mathrm{U}_j^i|/t$. The idea is that with high probability over the choice of the U^i's, there exist (correct, or representative) partitions of these sets for which $\hat{\gamma}_j(v)$ is in fact a good approximation of $\gamma_j(v)$, where $\gamma_j(v)$ in each iteration is defined with respect to the $i-1$ hybrid partition. This partition is the one determined by the previous iteration (using the representative partition of U^{i-1} and the correct setting of $\vec{\beta}^{i-1}$). While we must try all possible partitions of the U^i's and all possible settings of the $\beta_j^{i,\vec{\alpha}}$'s since we do not know which is the correct one, with high probability over the choice of $\mathrm{U}^1, \ldots, \mathrm{U}^k$, there exists at least one execution of the partitioning algorithm which simulates the oracle procedure correctly.

3.6.2 The preliminary Partitioning Algorithm

A detailed description of the graph partitioning algorithm is given in Figure 3.6, and its formal analysis is provided in Lemma 3.6.1.

```
Graph Partitioning Algorithm
Let $\Lambda = \{\vec{\alpha} : \vec{\alpha} = \langle \alpha_1, \ldots, \alpha_k \rangle,\ \alpha_i \in [0,1]$ is an integer multiple of $\epsilon/16$ $\}$.
  1. Choose $\ell = \frac{4}{\epsilon}$ sets $U^1, \ldots, U^\ell$ each of size $t = \Theta\left(\frac{1}{\epsilon^2}\log\frac{k}{\epsilon\delta}\right)$, where $U^i$ is chosen uniformly in $V \setminus V^i$.
     Let U be the union of these sets.
  2. For every setting of $\vec{\beta} = \vec{\beta}^1, \ldots, \vec{\beta}^\ell$, where $\vec{\beta}^i = \left\langle \beta_1^{i,\vec{\alpha}}, \ldots, \beta_k^{i,\vec{\alpha}} \right\rangle_{\vec{\alpha}\in\Lambda}$ and $\beta_j^{i,\vec{\alpha}} \in [0,1]$ is an integer
     multiple of $\epsilon/8$, and for each of the partitions $\Pi = \left(\bigcup_{i=1}^{\ell} U_1^i, \ldots, \bigcup_{i=1}^{\ell} U_k^i\right)$ of U,
     construct a partition $\mathcal{V}^{\Pi,\vec{\beta}} = \left(V_1^{\Pi,\vec{\beta}}, \ldots, V_k^{\Pi,\vec{\beta}}\right)$ as follows:
     (a) For $i = 1 \ldots \ell$ do:
         i. For each $v \in V^i$, and for $j \in \{1, \ldots, k\}$, let $\hat{\gamma}_j(v) \stackrel{\text{def}}{=} \frac{1}{t}\left(|\Gamma(v) \cap U_j^i|\right)$.
         ii. For each $\vec{\alpha} \in \Lambda$ let the $\vec{\alpha}$-cluster of $V^i$, denoted $V^{i,\vec{\alpha}}$, be defined as follows:
             $V^{i,\vec{\alpha}} \stackrel{\text{def}}{=} \{v \in V^i : \forall j,\ \alpha_j - \epsilon/32 < \hat{\gamma}_j(v) \le \alpha_j + \epsilon/32\}$.
         iii. For each cluster $V^{i,\vec{\alpha}}$, $\vec{\alpha} \in \Lambda$, let $V_1^{i,\vec{\alpha}}$ be the first $\beta_1^{i,\vec{\alpha}} \cdot |V^{i,\vec{\alpha}}|$ vertices in $V^{i,\vec{\alpha}}$, let $V_2^{i,\vec{\alpha}}$
              be the next $\beta_2^{i,\vec{\alpha}} \cdot |V^{i,\vec{\alpha}}|$ vertices in $V^{i,\vec{\alpha}}$, and in general, let $V_j^{i,\vec{\alpha}}$ be the first $\beta_j^{i,\vec{\alpha}} \cdot |V^{i,\vec{\alpha}}|$
              vertices in $V^{i,\vec{\alpha}} \setminus \bigcup_{j'<j} V_{j'}^{i,\vec{\alpha}}$.
              That is, the $\vec{\alpha}$-cluster of $V^i$ is partitioned according to the (guess of the) corresponding
              quantitative partition (i.e., the $\beta_j^{i,\vec{\alpha}}$'s).
         iv. For each $j \in \{1, \ldots, k\}$, let $V_j^i = \bigcup_{\alpha\in\Lambda} V_j^{i,\vec{\alpha}}$.
     (b) For each $j \in \{1, \ldots, k\}$, let $V_j = \bigcup_{i=1}^{\ell} V_j^i$.
         Since each $V_j$ actually depends on $\Pi$ and $\vec{\beta}$ we denote it by $V_j^{\Pi,\vec{\beta}}$.
     (c) If $\mathcal{V}^{\Pi,\vec{\beta}} = \{V_1^{\Pi,\vec{\beta}}, \ldots, V_k^{\Pi,\vec{\beta}}\}$ obeys Equations (3.18) and (3.19) then Output $\mathcal{V}^{\Pi,\vec{\beta}}$.
```

Figure 3.6: Graph Partitioning Algorithm

Lemma 3.6.1 *Let $\mathcal{H} = (H_1, \ldots, H_k)$ be a fixed partition of $V(G)$. Then with probability at least $1 - \delta$ over the choice of U, there exists a partition Π of $U = U^1, \ldots, U^\ell$, and a setting of $\vec{\beta}$, such that*

$$\forall j, \quad \left| \frac{|V_j^{\Pi,\vec{\beta}}| - |H_j|}{N} \right| \le \epsilon\ ,$$

and

$$\forall j, j', \quad \left| \frac{\left|E\left(V_j^{\Pi,\vec{\beta}}, V_{j'}^{\Pi,\vec{\beta}}\right)\right| - |E(H_j, H_{j'})|}{N^2} \right| \le \epsilon\ .$$

Proof: For a fixed partition Π of U and a fixed setting $\vec{\beta} = \vec{\beta}^1, \ldots, \vec{\beta}^\ell$, we consider the following $\ell + 1$ *hybrid* partitions: The hybrid $\mathcal{H}^0 = (H_1^0, \ldots, H_k^0)$ is simply $\mathcal{H}$; The hybrid partition $\mathcal{H}^i = (H_1^i, \ldots, H_k^i)$ is defined as follows:

$$H_j^i \stackrel{\text{def}}{=} W_j^{i-1} \cup V_j^i$$

where the partition $\mathcal{W}^{i-1} = (\mathrm{W}_1^{i-1}, \ldots, \mathrm{W}_k^{i-1})$ of $\mathrm{V} \setminus \mathrm{V}^i$, is defined by $\mathrm{W}_j^{i-1} \stackrel{\rm def}{=} \mathrm{H}_j^{i-1} \setminus \mathrm{V}^i$, and V_j^i's are determined as in Figure 3.6. We shall show that for every $1 \leq i \leq \ell$, for a given choice and partitions of $\mathrm{U}^1, \ldots, \mathrm{U}^{i-1}$, and for a given setting of $\vec{\beta}^1, \ldots, \vec{\beta}^{i-1}$, there always exists a setting of $\vec{\beta}^i$ (referred to as the *correct* setting), and with probability at least $1 - \frac{\delta}{\ell}$ over the choice of U^i, there exists a partition $(\mathrm{U}_1^i, \ldots, \mathrm{U}_k^i)$ of U^i such that

$$\forall j, \quad \left| \frac{\left|\mathrm{H}_j^i\right| - \left|\mathrm{H}_j^{i-1}\right|}{N} \right| \leq \frac{\epsilon}{\ell}\,, \tag{3.20}$$

and

$$\forall j, j', \quad \left| \frac{\left|\mathrm{E}\left(\mathrm{H}_j^i, \mathrm{H}_{j'}^i\right)\right| - \left|\mathrm{E}\left(\mathrm{H}_j^{i-1}, \mathrm{H}_{j'}^{i-1}\right)\right|}{N^2} \right| \leq \frac{\epsilon}{\ell}\,. \tag{3.21}$$

The lemma follows by simple induction on i.

Let $(\mathrm{U}_1^i, \ldots, \mathrm{U}_k^i)$ be the partition of U^i induces by $\mathcal{W}^{i-1}$. Namely, for each j, $\mathrm{U}_j^i \stackrel{\rm def}{=} \mathrm{W}_j^{i-1} \cap \mathrm{U}^i$. We say that U^i is *good* with respect to $(\mathrm{W}_1^{i-1}, \ldots, \mathrm{W}_k^{i-1})$ and V^i if the following holds. For all but $\epsilon/8$ of the vertices v in V^i,

$$\text{For each } j \in \{1, \ldots, k\} \quad \frac{\left|\Gamma(v) \cap \mathrm{U}_j^i\right|}{t} = \frac{\left|\Gamma(v) \cap \mathrm{W}_j^{i-1}\right|}{N} \pm \frac{\epsilon}{32} \tag{3.22}$$

If the above holds then we say that $(\mathrm{U}_1^i, \ldots, \mathrm{U}_k^i)$ is *representative* with respect to $(\mathrm{W}_1^{i-1}, \ldots, \mathrm{W}_k^{i-1})$ and V^i.

Proving Eq. (3.20) and (3.21) for a Good U^i Assume first that $(\mathrm{U}_1^i, \ldots, \mathrm{U}_k^i)$ is representative with respect to $(\mathrm{W}_1^{i-1}, \ldots, \mathrm{W}_k^{i-1})$ and V^i, and let the clusters $\mathrm{V}^{i,\vec{\alpha}}$ be as defined in Figure 3.6, Step 2(a)ii, where $\hat{\gamma}_j(v)$ (for every $v \in \mathrm{V}^i$ and j) is defined as in Step 2(a)i. Let $\vec{\beta}^i = \left\langle \beta_1^{i,\vec{\alpha}}, \ldots, \beta_k^{i,\vec{\alpha}} \right\rangle_{\vec{\alpha} \in \Lambda}$ be such that for every $\alpha \in \Lambda$, and for each j, $\beta_j^{i,\vec{\alpha}} = \frac{\left|\mathrm{W}_j^{i-1} \cap \mathrm{V}^{i,\vec{\alpha}}\right|}{\left|\mathrm{V}^{i,\vec{\alpha}}\right|} \pm \epsilon/16$. Since each $\beta_j^{i,\vec{\alpha}}$ takes on all values which are multiples of $\epsilon/8$, there is such a setting of $\vec{\beta}^i$'s. It follows that when the vertices in V^i are partitioning using this $\vec{\beta}^i$, then Equation (3.20) holds. We now show that Equation (3.21) holds as well. Towards this end we fix arbitrary j, j' and consider the contribution of three types of edges to the l.h.s. of Equation (3.21):

1. The contribution of edges with both endpoints NOT in V^i. Since $v \in \mathrm{H}_j^i$ iff $v \in \mathrm{H}_j^{i-1}$, for every $v \notin \mathrm{V}^i$, such edges do not contribute to the difference (i.e., to the l.h.s. of Equation (3.21)).

2. The contribution of edges with both endpoints IN V^i. There are at most $\left|\mathrm{V}^i\right|^2$ such edges. Using $\left|\mathrm{V}^i\right| = \frac{N}{\ell} = \frac{\epsilon}{4}N$, the potential contribution of these edges is bounded by $\frac{\epsilon}{4\ell}N$.

3. The contribution of edges with exactly ONE endpoint in V^i. We distinguish two cases.

   (a) Edges incident to vertices in V^i for which Equation (3.22) does NOT hold. Since $(\mathrm{U}_1^i, \ldots, \mathrm{U}_k^i)$ is representative with respect to $(\mathrm{W}_1^{i-1}, \ldots, \mathrm{W}_k^{i-1})$ and V^i, there are at most $\frac{\epsilon}{8\ell}N$ such vertices. Thus, the contribution of these edges to the l.h.s. of Equation (3.21) is bounded above by $\frac{\epsilon}{8\ell}N \cdot 2N = \frac{\epsilon}{4\ell}N^2$

   (b) Edges incident to vertices in V^i for which Equation (3.22) DOES hold. The contribution of these edges is due to two types of approximation errors. The first approximation error is due to the clustering itself. That is, vertices in V^i which belong to the same cluster, might differ by $\frac{\epsilon}{8}N$ in the number of neighbors they have in each W_j^{i-1} (i.e., for every v, v' which belong to the same cluster $\mathrm{V}^{i,\vec{\alpha}}$ and for each j, we have $|\gamma_j(v) - \gamma_j(v')| \leq |\hat{\gamma}_j(v) - \hat{\gamma}_j(v')| + |\hat{\gamma}_j(v) - \gamma_j(v)| + |\hat{\gamma}_j(v') - \gamma_j(v')| \leq (\epsilon/8)N$.) The second approximation error is due to the fact that the fractional partition is specified with bounded precision (i.e., $\beta_j^{i,\vec{\alpha}} = \frac{|\mathrm{W}_j^{i-1} \cap \mathrm{V}^{i,\vec{\alpha}}|}{|\mathrm{V}^{i,\vec{\alpha}}|} \pm \epsilon/16$). Thus, the contribution of these edges to the l.h.s. of Equation (3.21) is bounded above by $\frac{3\epsilon}{8\ell}N^2$.

Summing both cases we get a contribution bounded above by $\frac{\epsilon}{\ell}N^2$.

Bounding the Probability that U^i is not Good We first fix a vertex $v \in \mathrm{V}^i$. Let $\mathrm{U}^i = \{u_1, \ldots, u_t\}$. For $j \in \{1 \ldots, k\}$, and for $1 \leq s \leq t$, define a Bernoulli random variable, ξ_j^s, which is 1 if u_s is a neighbor of v and $u_s \in \mathrm{W}_j^{i-1}$, and is 0 otherwise. By definition, for each particular j, the sum of the ξ_j^s's is simply the number of neighbors v has in U_j^i, and the probability that $\xi_j^s = 1$ is $\frac{1}{N}|\Gamma(v) \cap \mathrm{W}_j^{i-1}|$. By Hoeffding's inequality and our choice of t, for each $j \in \{1, \ldots, k\}$,

$$\mathrm{Prob}_{\mathrm{U}^i}\left[\left|\frac{|\Gamma(v) \cap \mathrm{U}_j^i|}{t} - \frac{|\Gamma(v) \cap \mathrm{W}_j^{i-1}|}{N}\right| > \frac{\epsilon}{32}\right] = \exp(-O(\epsilon^2 t)) = O\left(\frac{\epsilon\delta}{k\ell}\right) .$$

By Markov's inequality, for each $j \in \{1, \ldots, k\}$, with probability at least $1 - \frac{\delta}{k\ell}$ over the choice of U^i, for all but $\frac{1}{8}\epsilon$ of the vertices in V^i, Equation (3.22) holds (for that j), and thus with probability at least $1 - \frac{\delta}{\ell}$, U^i is good as required. ■

Lemma 3.6.1 implies that if $\mathrm{G} \in \mathcal{GP}_\Phi$, then with high probability the Graph Partitioning Algorithm described in Figure 3.6 will find an ϵ-approximation for the partitioning problem defined by Φ. As was mentioned previously, we show below how to produce such a partitioning in a more efficient way by first running the testing algorithm for $\mathcal{GP}_\Phi$.

3.6.3 The Testing Algorithm

The testing algorithm for $\mathcal{GP}_\Phi$ (described in Figure 3.7) essentially performs the same steps as the graph partitioning algorithm on a small sample, S, and it is analyzed below. An important technical detail is that (in Step 4) the tester checks whether the sampled densities are close to an *admissible set of densities* (defined below), rather than test if they satisfy inequalities analogous to Equations (3.18) and (3.19) hold.

ADMISSIBLE SET OF DENSITIES: A set of (non-negative) reals, $\{a_j\} \cup \{b_{j,j'}\}$, is called admissible with respect to Φ ($= \{\rho_j^{\mathrm{lb}}, \rho_j^{\mathrm{ub}}\} \cup \{\varrho_{j,j'}^{\mathrm{lb}}, \varrho_{j,j'}^{\mathrm{ub}}\}$) if it satisfies the following inequalities

$$\sum_{j=1}^{k} a_j = 1 \text{ and } \sum_{j,j'=1}^{k} b_{j,j'} \leq 1 \tag{3.23}$$

$$\rho_j^{\mathrm{lb}} \leq a_j \leq \rho_j^{\mathrm{ub}} \ (\forall j\ ) \quad \text{and} \quad \varrho_{j,j'}^{\mathrm{lb}} \leq b_{j,j'} \leq \varrho_{j,j'}^{\mathrm{ub}} \ (\forall j, j'\ ) \tag{3.24}$$

and

$$b_{j,j} \leq a_j^2 \ (\forall j\ ) \quad \text{and} \quad b_{j,j'} \leq 2 \cdot a_j \cdot a_{j'} \ (\forall j \neq j'\ ) \tag{3.25}$$

We may assume that Φ is such that there are admissible sets of reals with respect to it. In Step 4 of Figure 3.7 we check that a set of densities $\{\rho_j^{\Pi,\vec{\beta}}\} \cup \{\varrho_{j,j'}^{\Pi,\vec{\beta}}\}$ is $2\epsilon'$-close to an admissible set. That is, we ask whether there are non-negative a_j's and $b_{j,j'}$'s which, in addition to the above, also satisfy $|a_j - \rho_j^{\Pi,\vec{\beta}}| \leq 2\epsilon'$ $(\forall j)$ and $|b_{j,j'} - \varrho_{j,j'}^{\Pi,\vec{\beta}}| \leq 2\epsilon'$ $(\forall j, j')$. This comes instead of checking if $\rho_j^{\Pi,\vec{\beta}} \in [(\rho_j^{\text{lb}} - \frac{\epsilon}{k}), (\rho_j^{\text{ub}} + \frac{\epsilon}{k})]$ $(\forall j)$ and $\varrho_{j,j'}^{\Pi,\vec{\beta}} \in [(\varrho_{j,j'}^{\text{lb}} - \frac{\epsilon}{k}), (\varrho_{j,j'}^{\text{ub}} + \frac{\epsilon}{k})]$ $(\forall j, j')$. The reason for this choice will become clear in the proof below. We note that the problem of whether a set of densities is $2\epsilon'$-close to an admissible set for Φ can be certainly solved in $\exp(\text{poly}(k) \cdot L)$-time, where L is the length of the encoding (in binary) of Φ and ϵ (see Appendix A.2). Note that this time-bound is dominated by the number of partitions examined in Step 4 (of Figure 3.7).

Testing Algorithm for $\mathcal{GP}_\Phi$, where $\Phi = \{\rho_j^{\text{lb}}, \rho_j^{\text{ub}}\}_{j=1}^k \cup \{\varrho_{j,j'}^{\text{lb}}, \varrho_{j,j'}^{\text{ub}}\}_{j,j'=1}^k$

```
Let $\epsilon' = \frac{\epsilon}{3k^2}$.
  1. Uniformly choose a set $S = \{s_1, \ldots, s_m\}$ of size $m = \Theta\left(\left(\frac{O(1)}{\epsilon'}\right)^{k+3} \cdot k \cdot \log(k/(\epsilon'\delta))\right)$.
     For $1 \leq i \leq \ell$, let $S^i \overset{\text{def}}{=} V^i \cap S$.
  2. Choose $\ell = \frac{4}{\epsilon'}$ sets $U^1, \ldots, U^\ell$ each of size $t = \Theta\left(\frac{1}{\epsilon'^2} \log \frac{k}{\epsilon'\delta}\right)$, where $U^i$ is chosen uniformly in $V \setminus V^i$.
     Let U be the union of these sets.
  3. For every setting of $\vec{\beta} = \vec{\beta}^1, \ldots, \vec{\beta}^\ell$ and for each of the partitions $\Pi = \left(\bigcup_{i=1}^\ell U_1^i, \ldots, \bigcup_{i=1}^\ell U_k^i\right)$ of
     U (as in Figure 3.6),
     construct a partition $\mathcal{S}^{\Pi,\vec{\beta}} = \left(S_1^{\Pi,\vec{\beta}}, \ldots, S_k^{\Pi,\vec{\beta}}\right)$ as follows:
     (a) For $i = 1 \ldots \ell$ do:
          i. For each $v \in S^i$, and for $j \in \{1, \ldots, k\}$, let $\hat{\gamma}_j(v) \overset{\text{def}}{=} \frac{1}{t}\left(\left|\Gamma(v) \cap U_j^i\right|\right)$.
         ii. For each $\vec{\alpha} \in \Lambda$ let the $\vec{\alpha}$-cluster of $S^i$, denoted $S^{i,\vec{\alpha}}$ be defined as follows:
             $S^{i,\vec{\alpha}} \overset{\text{def}}{=} \{v \in S^i : \forall j,\ \alpha_j - \epsilon'/32 < \hat{\gamma}_j(v) \leq \alpha_j + \epsilon'/32\}$.
        iii. For each cluster $S^{i,\vec{\alpha}}$, $\vec{\alpha} \in \Lambda$, let $S_1^{i,\vec{\alpha}}$ be the first $\lfloor \beta_1^{i,\vec{\alpha}} \cdot |S^{i,\vec{\alpha}}| \rfloor$ vertices in $S^{i,\vec{\alpha}}$, let $S_2^{i,\vec{\alpha}}$
             be the next $\lfloor \beta_2^{i,\vec{\alpha}} \cdot |S^{i,\vec{\alpha}}| \rfloor$ vertices in $S^{i,\vec{\alpha}}$ and in general, for $j < k$ let $S_j^{i,\vec{\alpha}}$ be the first
             $\lfloor \beta_j^{i,\vec{\alpha}} \cdot |S^{i,\vec{\alpha}}| \rfloor$ vertices in $S^{i,\vec{\alpha}} \setminus \bigcup_{j'<j} S_{j'}^{i,\vec{\alpha}}$, and let $S_k^{i,\vec{\alpha}}$ be the remaining vertices in $S^{i,\vec{\alpha}}$.
         iv. For each $j \in \{1, \ldots, k\}$, let $S_j^i = \bigcup_{\alpha \in \Lambda} S_j^{i,\vec{\alpha}}$.
     (b) For each $j \in \{1, \ldots, k\}$, let $S_j = \bigcup_{i=1}^\ell S_j^i$.
  4. For each partition, $\mathcal{S}^{\Pi,\vec{\beta}}$, let $\rho_j^{\Pi,\vec{\beta}} \overset{\text{def}}{=} \frac{1}{m} \cdot |\mathcal{S}^{\Pi,\vec{\beta}}|$ $(\forall j)$
     and $\varrho_{j,j'}^{\Pi,\vec{\beta}} \overset{\text{def}}{=} \frac{1}{m/2} \cdot |\{(s_{2r-1}, s_{2r}) \in E : s_{2r-1} \in S_j^{\Pi,\vec{\beta}} \text{ and } s_{2r} \in S_{j'}^{\Pi,\vec{\beta}} \text{ or vica versa }\}|$ $(\forall j, j')$.
     If for some $\Pi, \vec{\beta}$ the set of densities $\{\rho_j^{\Pi,\vec{\beta}}\} \cup \{\varrho_{j,j'}^{\Pi,\vec{\beta}}\}$ is $2\epsilon'$-close to an admissible set w.r.t. $\Phi$
     then output Accept, otherwise output Reject.
```

Figure 3.7: Testing Algorithm for $\mathcal{GP}_\Phi$

Proof of Theorem 3.11 Recall that $\epsilon' = \frac{\epsilon}{3k^2}$. We first note that if $G \in \mathcal{GP}_\Phi$, then by Lemma 3.6.1, with probability $1 - \delta/2$ over the choice of U, there exist Π and $\vec{\beta}$ such that $\mathcal{V}^{\Pi,\vec{\beta}}$ is an ϵ'-approximation of the partition problem. Furthermore, the densities related to this partition are ϵ'-close to an admissible set for Φ. On the other hand, if G is ϵ-far from $\mathcal{GP}_\Phi$, then no k-way partition of V has densities which are $3\epsilon'$-close to an admissible set w.r.t. Φ.[12] Suppose, on the

[12] Here is where the notion of admissible solution plays an important role. We could not have concluded that then no k-way partition of V is a $3\epsilon'$-good approximation (of the partition problem). To demonstrate the point consider the parameters $\rho_1^{\text{lb}} = \rho_2^{\text{lb}} = \frac{1}{2} + \epsilon'$ (and all other parameters are trivial; i.e., upper-bounds are 1 and lower bounds

contrary, that V has a k-way partition with densities that are $3\epsilon'$-close to an admissible set w.r.t. Φ. Then we can modify this partition by moving up to $3k\epsilon' \cdot N$ vertices (according to the guaranteed admissible densities) and omitting/adding up to $3k^2\epsilon' \cdot N^2 = \epsilon N^2$ edges, and obtain a graph in $\mathcal{GP}_\Phi$ in contradiction to the hypothesis.

For a fixed set U, we now consider the partitions $\mathcal{V}^{\Pi,\vec{\beta}}$'s which would result in running the graph partitioning algorithm with an approximation parameter $\epsilon' = \epsilon/3k^2$, and a confidence parameter $\delta/2$. We relate these partitions to the the partitions $\mathcal{S}^{\Pi,\vec{\beta}}$'s that are generated by the Testing algorithm when using the same set U. We claim that, with high probability over the choice of S, for each partition Π of U and for each setting of $\vec{\beta}$, the following holds:

1. If the densities of the partition $\mathcal{V}^{\Pi,\vec{\beta}} = \left(\mathrm{V}_1^{\Pi,\vec{\beta}}, \ldots, \mathrm{V}_k^{\Pi,\vec{\beta}}\right)$ are ϵ'-close to an admissible set for the problem, then the densities of the partition $\mathcal{S}^{\Pi,\vec{\beta}} = \left(\mathrm{S}_1^{\Pi,\vec{\beta}}, \ldots, \mathrm{S}_k^{\Pi,\vec{\beta}}\right)$ are $2\epsilon'$-close to an admissible set for the problem. It follows that the testing algorithm will accept G (cf., Step 4 in Figure 3.7).

2. If the densities in $\mathcal{V}^{\Pi,\vec{\beta}}$ are not $3\epsilon'$-close to an admissible set then the densities in $\mathcal{S}^{\Pi,\vec{\beta}}$ are not $2\epsilon'$-close to an admissible set. It follows that the testing algorithm will reject G.

To prove the claim, consider a particular Π and $\vec{\beta}$. Assume first that for each j, $\mathrm{S}_j^{\Pi,\vec{\beta}}$ is exactly $\mathrm{V}_j^{\Pi,\vec{\beta}} \cap \mathrm{S}$. Then we can view $\mathcal{S}^{\Pi,\vec{\beta}}$ as *sampling* $\mathcal{V}^{\Pi,\vec{\beta}}$. Namely, for a given side $j \in \{1, \ldots, k\}$ and for each $r \in \{1, \ldots, m\}$ we can define a Bernoulli random variable, ξ_j^r, which is 1 if $s_r \in \mathrm{V}_j^{\Pi,\vec{\beta}}$ and 0 otherwise. Thus, $\mathrm{Prob}(\xi_j^r = 1) = \left|\mathrm{V}_j^{\Pi,\vec{\beta}}\right|/N$, and with high probability, by Hoeffding's inequality, the sum of the ξ_j^r's does not deviate from its expectation by more than ϵ'. Similarly, for a given j, j' we can define a Bernoulli random variable, $\xi_{j,j'}^r$ for every $r \in \{1, \ldots, m/2\}$ which is 1 if $(s_{2r-1}, s_{2r}) \in \mathrm{E}$, $s_{2r-1} \in \mathrm{V}_j^{\Pi,\vec{\beta}}$ and $s_{2r} \in \mathrm{S}_{j'}^{\Pi,\vec{\beta}}$ (or vica versa), and is 0 otherwise. The expected value of the sum of the $\xi_{j,j'}^r$'s is the edge density between $\mathrm{V}_j^{\Pi,\vec{\beta}}$ and $\mathrm{V}_{j'}^{\Pi,\vec{\beta}}$, and for large enough m we do not deviate by more than ϵ' from this expected value for every j and j'. Since we want these estimations of expected values to be good for all Π and $\vec{\beta}$, the size of S must grow logarithmically with the number of possible partitions Π and settings of $\vec{\beta}$.

The only detail that we need to take care of is that when partitioning $\mathrm{S}^{i,\vec{\alpha}} \subset \mathrm{V}^{i,\vec{\alpha}}$ for some i and α, (and a fixed Π and $\vec{\beta}$), some s_r's which belong to $\mathrm{S}^{i,\vec{\alpha}}$ might be put in a particular $\mathrm{S}_j^{i,\vec{\alpha}}$, while in the graph partition they are put in $\mathrm{V}_{j'}^{i,\vec{\alpha}}$, for $j' \neq j$. However, by applying arguments analogous to Lemma 3.3.5, it can be shown that the fraction of such vertices is small, and that their contribution to the approximation error is small too. ■

Proof of Theorem 3.12 In order to get a partitioning algorithm whose running time is as stated in the theorem, we first run the testing algorithm. The idea is that by first running the testing algorithm we find the correct choice of Π and $\vec{\beta}$ (in time independent of N), and then we can use these choices to partition all of V. As we have shown in the proof of Theorem 3.11, with high probability, a partition $\mathcal{S}^{\Pi,\vec{\beta}}$ will cause the test to accept if $\mathcal{V}^{\Pi,\vec{\beta}}$ is an ϵ'-approximation of the partitioning problem, and if $\mathcal{V}^{\Pi,\vec{\beta}}$ is not an $3\epsilon'$-approximation, then $\mathcal{S}^{\Pi,\vec{\beta}}$ will not cause the test to accept. Thus, with high probability if the testing algorithm accepted G due to a certain setting of Π and $\vec{\beta}$, then $\mathcal{V}^{\Pi,\vec{\beta}}$ is an ϵ-approximation of the partitioning problem. ■

are 0). In this case the class $\mathcal{GP}_\Phi$ is empty and thus all graphs are ϵ-far from it. Yet, any bisection of any graph is an ϵ'-good approximation of the partition problem with parameters Φ.

3.7 Various Comments

3.7.1 Extensions and Limitations

We have presented several Graph Property Testers which use queries and are evaluated with respect to the uniform distribution (on pairs of vertices). We now comment on several extensions, variations and considerations.

Impossibility of testing without queries. Proposition 3.2 shows that queries are essential for testing Bipartiteness. The construction used in the proof actually establishes the same for testing the class of graphs having cliques of density at least 1/2, and for approximating the MaxCUT (of dense graphs up to an $N^2/8$ additive term). Furthermore, the proof can be easily modified to yield the same result for testing k-Colorability, for any $k \geq 3$.

Extension to Directed Graphs. Some of the problems we study have analogies in directed graphs. In particular this is true for ρ-Cut, where we are interested in testing whether a directed graph has a two-way partition $(\mathrm{V}_1, \mathrm{V}_2)$ such that the number of edges crossing *from* V_1 *to* V_2 is at least ρN^2. Note that directed graphs in general do not have a symmetric adjacency matrix and the existence of a directed edge from v_1 to v_2 does not necessarily imply the existence of an edge from v_2 to v_1. Similarly to the undirected case, the algorithm is essentially based on the following observation. Consider a vertex v and a partition $(\mathrm{W}_1, \mathrm{W}_2)$ of $\mathrm{V} \setminus \{v\}$. Let $\mathrm{E}(v, \mathrm{W}_2)$ be the set of edges going from v to vertices in W_2, and let $\mathrm{E}(\mathrm{W}_1, v)$ be the set of edges going from vertices in W_1 to v. In case $|\mathrm{E}(v, \mathrm{W}_2)| > |\mathrm{E}(\mathrm{W}_1, v)|$, then it is preferable to put v on side 1, i.e. $\mu(\mathrm{W}_1 \cup \{v\}, \mathrm{W}_2) > \mu(\mathrm{W}_1, \mathrm{W}_2 \cup \{v\})$, and in case $|\mathrm{E}(\mathrm{W}_1, v)| > |\mathrm{E}(v, \mathrm{W}_2)|$, we should put v on side 2. The notion of *unbalance* is hence slightly modified, but as in the case of undirected cuts, the above observation generalizes to sets of vertices of size $O(\epsilon N)$. Thus the ρ-Directed-Cut testing algorithm is very similar to the ρ-Cut testing algorithm, the only difference is that when deciding where to put a vertex $v \in \mathrm{V}^i$ given a fixed partition $(\mathrm{U}_1, \mathrm{U}_2)$ of the uniformly selected set U, we compare the number of edges going from v to vertices in U_2 to the number of edges going from U_1 to v. The algorithms for k-way-Cut and Bisection are modified similarly.

It is also possible to extend the definition of the general partition problem to directed graphs by allowing the bounds $\varrho^{\mathrm{lb}}_{j,j'}$ and $\varrho^{\mathrm{ub}}_{j,j'}$ to differ from $\varrho^{\mathrm{lb}}_{j',j}$ and $\varrho^{\mathrm{ub}}_{j',j}$, respectively for $j \neq j'$. That is, there are separate requirements on the number of edges crossing from V_j to $\mathrm{V}_{j'}$ and the number of edges crossing from $\mathrm{V}_{j'}$ to V_j. In such a case, the graph partitioning and testing algorithms for directed graphs differ from the algorithms for undirected graphs only in their definition of *clusters* (but the clusters are partitioned by the algorithms as in the undirected case). A cluster of vertices which is defined with respect to a fixed partition $(\mathrm{W}_1, \ldots, \mathrm{W}_k)$, is a set of vertices Z such that for all $v_1, v_2 \in \mathrm{Z}$, and for every $j \in \{1, \ldots, k\}$, $\mathrm{E}(v_1, \mathrm{W}_j) \approx \mathrm{E}(v_2, \mathrm{W}_j)$, and $\mathrm{E}(\mathrm{W}_j, v_1) \approx \mathrm{E}(\mathrm{W}_j, v_2)$.

Extension to product distributions. Our algorithms for k-Colorability, ρ-Clique and ρ-Cut can be easily extended to provide testers with respect to "product distributions". We call the distribution $\Pi : \mathrm{V}(\mathrm{G})^2 \mapsto [0,1]$ a *product distribution* if there exists a distribution on vertices $\pi : \mathrm{V}(\mathrm{G}) \mapsto [0,1]$ so that $\Pi(u,v) = \pi(u) \cdot \pi(v)$. Recall that each of our algorithms takes a uniform sample of vertices, and queries the graph only on the existence of edges between these vertices. Instead, in case we need to test G w.r.t the product distribution Π, we sample V(G) according to the distribution π. Note that the algorithm need not "know" π; it suffices that the algorithm has access to a source of vertices drawn from this distribution (or to a source of pairs drawn according to Π). To prove that this extension is valid consider an auxiliary graph G' consisting of vertex-components so that each component correspond to a vertex in G. Complete bipartite graphs will be placed between pairs of components which correspond to edges in G. The size of the component will be related to the probability measure of the corresponding vertex in G. Thus, uniform distribution on V(G′) is almost the same as distribution Π on V(G). For the analysis of the ρ-Clique tester, we add also edges between vertices residing in the same component. (This is certainly NOT done in analyzing the k-Colorability and Max-CUT testers.) It can be shown that testing G for any of the above mentioned properties with respect to the product distribution Π corresponds to testing G′ (for the respective property) under the uniform distribution. Suppose, for example, that G is k-Colorable. Then so is G′ and thus the k-Colorability Tester will always accept G′ (and thus always accept G). On the other hand, if the k-Colorability Tester, run with parameters ϵ, δ, accepts G′ with probability $1 - \delta$, then there is a k-coloring of G′ which violates less than an ϵ fraction of vertex pairs. To obtain a k-coloring of G we randomly assign each vertex in G a color according to the proportions of colors assigned to the corresponding vertices in G′. It follows that the expected probability mass (according to Π) assigned to violated edges is less than ϵ.

IMPOSSIBILITY OF DISTRIBUTION-FREE TESTING. In contrast to the above extension, it is not possible to test any of the graph properties discussed above in a distribution-free manner (even with queries). For simplicity, let us consider the case of Bipartite Testing. Consider the following class of distributions on pairs of vertices of an N-vertex graph. Each distribution is defined by a partition of $[N]$ into $\frac{N}{4}$ 4-tuples. The distribution assigns probability $\frac{1}{3N}$ to each (ordered) pair of distinct vertices which belong to the same 4-tuple. Pairs residing in different 4-tuples are assigned probability 0. For each distribution, we consider two graphs. The first graph consists of $N/4$ paths of length 3, each residing in a different 4-tuple; whereas the second graph consists of $N/4$ triangles, each residing in a different 4-tuple. Clearly, the first graph is bipartite whereas the second is not. Furthermore, the second graph is 1/6-far (w.r.t the above distribution) from being bipartite. Still, no tester which works in $o(\sqrt{N})$ time can tell these two graphs apart, even if it gets samples drawn from the distribution and is allowed queries. The reason being, that t samples drawn from the distribution will, with probability at least $1 - 4t^2/N$, come from different 4-tuples. In this case, any query made by the tester, unless if it is on a pair which has appeared in the sample, is likely to be on a pair which is not from the same 4-tuple and thus a non-edge (in both graphs). (Formally, one may consider a fixed distribution drawn at random from the above class and the likelihood claim is made with respect to the choice of distribution.)

ON THE POSSIBILITY OF WORKING IN poly$(1/\epsilon)$ TIME. The algorithm for Bipartite Testing works in poly$(1/\epsilon)$-time (see Thm. 3.1), whereas all the other testers we present work in time exponential in poly$(1/\epsilon)$. In fact, it seems that one cannot hope for much better. For example, we claim that if one can test 3-Colorability with distance parameter ϵ in time poly$(1/\epsilon)$ then $\mathcal{NP} \subseteq \mathcal{BPP}$. To test if a graph G is 3-colorable, simply set $\epsilon = 1/|\mathrm{V(G)}|^2$ (and $\delta = 1/3$) and run the property testing. Clearly, if G is 3-colorable then the test will accept with probability at least 2/3, whereas if G is not 3-colorable it must be ϵ-far from being 3-colorable and thus be rejected by the test with probability at least 2/3. We remark that a similar argument can be made when using a relatively bigger value of ϵ. For example, to decide if G is 3-Colorable consider an auxiliary graph, G′, with $2^{|\mathrm{V(G)}|}$ vertices which are grouped into $n \stackrel{\mathrm{def}}{=} |\mathrm{V(G)}|$ components each consisting of $2^n/n$ vertices. These huge components will correspond to vertices in G and complete bipartite graphs will be placed between pairs of components which correspond to edges in G. Thus, we can set $\epsilon = 1/n^2$ and apply the same reduction as above this time showing that deciding 3-colorability of G reduces to 3-colorability testing of G′ with distance parameter $\epsilon = 1/\mathrm{poly}\log|\mathrm{V(G')}|$.

ON ONE-SIDED FAILURE PROBABILITY. The testers for Bipartite and k-Colorability always accept graphs which have the property. In contrast, all other testers we present may fail to accept a yes-instance (yet, with probability at most δ). We claim that non-zero failure probability, in this case, is inevitable. Consider the execution of a potential Clique Tester given access to a graph with no edges at all. Clearly, the tester must reject with probability at least $1-\delta$. Fix any sequence of coin tosses (for the tester) which makes it reject. This determines a sequence of queries into the graph (all queries are answered by a 0). Assuming that the number of queries is less than $(1-\rho)N$, there exists an N-vertex graph having a clique of size ρN in which all the queried pairs are non-edges. It follows that the Clique Tester rejects this yes-instance with positive probability.[13] We conclude that there is a fundamental difference between testing k-Colorability and testing ρ-Clique.

Impossibility of learning with queries. Finally, we remark that all the above classes are not learnable, not even under uniform distribution and when allowing queries. Furthermore, this holds also if we allow unlimited computing power as long as we restrict the number of queries to $o(N)$. Intuitively, there are "too" many graphs in each family. Formally, we may consider attempts to learn a random bipartite N-vertex graph and may even fix the 2-partition, say, place vertices $\{1, ..., N/2\}$ on one side. Then, each uninspected pair (i,j), with $i \leq N/2$ and $j > N/2$, is equally likely to be either an edge or a non-edge.

Extension to Weighted Graphs (with bounded weights). Let $\mathrm{G} = (\mathrm{V},\mathrm{E})$ be a (simple undirected) graph, and $\omega : \mathrm{E} \mapsto [0,B]$ be a weight function assigning each edge $e \in \mathrm{E}$ a non-negative value (bounded by B). We assume that B, the bound on the weights of the edges, is known. We associate with G a function $f_{\mathrm{G},\omega} : \mathrm{V} \times \mathrm{V} \to [0,B]$, where $f_{\mathrm{G},\omega}(v_1,v_2) = 0$ if $(v_1,v_2) \notin \mathrm{E}$ and is $\omega(v_1,v_2)$ otherwise. When performing a query on a pair of vertices, (v_1,v_2), our testing algorithms receive the value of $f_{\mathrm{G}}(v_1,v_2)$. The notion of distance between graphs is slightly different from the unweighted case: We define the distance between two weighted graphs (G,ω) and (G',ω'), or equivalently between $f_{\mathrm{G},\omega}$ and $f_{\mathrm{G}',\omega}$, to be $\frac{1}{N^2}\sum_{v_1,v_2\in\mathrm{V}} |f_{\mathrm{G},\omega}(v_1,v_2) - f_{\mathrm{G}',\omega}(v_1,v_2)|$. The distance between a weighted graph and a class of weighted graphs is defined in the obvious manner. The generalization to weighted graphs may affect the testing problems in two ways:

Affect on the objective: Consider, for example, a generalization of ρ-CUT to weighted graphs. A weighted graph $\mathrm{G} = (\mathrm{V},\mathrm{E},\omega)$ has a cut of weight at least ρ, if there exists a partition $(\mathrm{V}_1,\mathrm{V}_2)$ of V such that $\frac{1}{N^2}\sum_{v_1\in\mathrm{V}_1,v_2\in\mathrm{V}_2} 2f_{\mathrm{G}}(v_1,v_2) \geq \rho$. Thus, the class ρ-CUT is a class of weighted graphs (rather than of graphs). Still, our algorithms for unweighted ρ-CUT are easily adapted to the weighted case; yet, the query complexity (resp., running-time) of our tester will depend polynomially (resp., exponentially) on the bound B.[14] The **k-Cut**, **Bisection**, and **General Partition** properties and algorithms generalize similarly.

Affect on distance: Consider, for example, testing k-colorability of weighted graphs. Clearly, the property of being k-colorable has nothing to do with the weights of the edges; yet, the distance from the class of k-Colorable graph does depend on these weights. However, the latter dependency can be easily waived by replacing each non-zero weighted edge by an edge with weight B. Note that this replacement does not affect k-colorability and that it only increases the distance of non-

[13] An analogous argument can be used to show that any Clique Tester must accept some no-instance with positive probability. (Start by considering an execution on the complete graph.)

[14] Specifically, all that need be changed is the notion of *balanced vertices*: We say that a vertex v is *unbalanced* with respect to a partition $(\mathrm{W}_1,\mathrm{W}_2)$ if $\frac{1}{N}|\sum_{w_1\in\mathrm{W}_1} f_{\mathrm{G}}(v,w_1) - \sum_{w_2\in\mathrm{W}_2} f_{\mathrm{G}}(v,w_2)|$ is non-negligible. Similarly to the unweighted case we shall use partitions $(\mathrm{U}_1,\mathrm{U}_2)$ of a uniformly selected sample U to approximate this difference (and use an additional sample S to approximate weights of cuts). The only difference in the analysis is that instead of using sums of 0/1 random variables to approximate expected values, we are using sums of random variables whose value lies in $[0,B]$, and hence we'll get a polynomial dependence on B in the sample complexity, and an exponential dependency in the running time.

k-colorable weighted graph from the class of k-colorable weighted graphs. We stress that in the resulting graph all edges have weight B. Thus, testing a weighted graph for k-Colorability with distnace paprameter ϵ, reduces to testing the underlying (unweighted) graph (for k-colorability) with distance parameter $\frac{\epsilon}{B}$. Again, the bound B turns out to affect the query complexity and running-time as in the first case.

3.7.2 Testing Graph Properties and Approximation

k-COLORABILITY TESTING VS. APPROXIMATING k-COLORABILITY. Petrank has defined *Approximating k-Colorability* as determining the number of edges which are "violated" under the best k-partition of the graph [Pet94]. (By *violated edges* w.r.t a k-partition we mean edges with two end-points in the same side of the partition.) Furthermore, he has suggested to concentrate on the "genuine" special case of the problem where one has to distinguish k-colorable graphs from graphs for which every k-partition violates a constant fraction of the edges. Petrank has shown that solving this promise problem is NP-Hard [Pet94], for $k = 3$. In contrast, our k-Colorability Tester implies that solving the same promise problem is easy *for dense graphs*, where by dense graphs we mean N-vertex graphs with $\Omega(N^2)$ edges. This is the case since, for every constant $\epsilon > 0$, our tester can distinguish, in $\exp(k^2/\epsilon^3)$-time, between k-colorable N-vertex graphs and N-vertex graphs which remain non-k-colorable even if one omits at most ϵN^2 of their edges.[15]

Another application of our k-Colorability Tester uses the fact that, for every $\epsilon > 0$, in case the N-vertex graph is k-colorable, the tester may retrieve in $\text{poly}(k/\epsilon) \cdot N$-time a k-partition which violates at most ϵN^2 edges. Thus, we may reduce the general problem of coloring k-colorable graphs to the same problem restricted to non-dense graphs (i.e., N-vertex graphs with $o(N^2)$ edges). (The reduction produces a coloring for dense graphs with k times more colors than the number used by the coloring of the non-dense graphs, but a factor of k seems small at the current state of art for this problem [KMS94].) We remark that some known algorithms for this task, seem to perform better when the maximum degree of vertices in the graph is smaller [KMS94]. Furthermore, deciding k-colorability even for N-vertex graphs of minimum degree at least $\frac{k-3}{k-2} \cdot N$ is NP-complete (cf., Edwards [Edw86]). On the other hand, Edwards also gave a polynomial-time algorithm for k-coloring k-colorable N-vertex graphs of minimum degree at least αN, for any constant $\alpha > \frac{k-3}{k-2}$.

ρ-CLIQUE TESTING VS. APPROXIMATING THE MAXCLIQUE. Our notion of ρ-Clique Testing differs from the traditional notion of MaxClique Approximation. Our ρ-Clique tester refers to the distance of a graph from the family of N-vertex graphs having cliques of size ρN, say, for $\rho = 1/4$. It accepts N-vertex graphs having a clique of size $N/4$ and reject graphs which are, say, $0.01 \cdot N^2$ edges away from having a clique of size $N/4$. On the other hand, when approximating the size of MaxClique, one is given an N-vertex graph and is asked to distinguish, for example, the case in which the max-clique has size at most $N/4$ from the case in which the max-clique has size at most $N/5$. Distinguishing the cases in the latter example is NP-Hard [BGS95]. Still the graphs constructed by the FGLSS reduction [FGL+91] are very close to having a clique of size $N/4$, also in the case where the max-clique has size smaller than $N/5$. Furthermore, a random graph with high edge density is very close to being a clique, whereas with very high probability its max-clique is quite small In general, a random N-vertex graph with edge density $1 - p$ is only p-away (i.e., misses "only" a p fraction of all possible edges) from being an N-clique, whereas the probability that it contains a clique of size t is bounded above by $\binom{N}{t} \cdot (1-p)^{\binom{t}{2}}$). (For example, with $p = N^{-1/10}$ the maxclique

[15] Similar results, alas with much worse dependence on ϵ in the running-time, can be obtained by using the results of Alon et. al. [ADL+94]. This was observed by Noga Alon (private communication, April 1996).

is very likely to be smaller than $\sqrt[4]{N}$.)

3.7.3 Testing other Graph Properties

The following remarks and observations are meant to indicate that providing a characterization of graph properties according to the complexity of testing them may not be easy.

We first recall that testing k-Colorability seems to be fundamentally different from testing ρ-Clique since the first can be done without ever rejecting yes-instances whereas the second cannot be done without two-sided failure probability.

EASY TO TEST GRAPH PROPERTIES. Next, we observe that any graph property which can be made to hold by adding or omitting few edges from any graph can be tested very easily. Namely,

Proposition 3.13 (testing almost trivial classes): *Let $\alpha > 0$ be a constant and $\mathcal{C}$ a class of graphs so that for every graph,* G*,*

$$\mathrm{dist}(\mathrm{G},\mathcal{C}) \leq |\mathrm{V}(\mathrm{G})|^{-\alpha}$$

Then, $\mathcal{C}$ can be tested by using at most $\mathrm{poly}(\log(1/\delta)/\epsilon)$ *labeled random examples, where ϵ is the distance parameter. Furthermore, the test always accepts graphs in $\mathcal{C}$.*

Classes which satisfy the hypothesis of the proposition include: *Connected Graphs* (add $\leq |\mathrm{V}(\mathrm{G})|-1$ edges), *Eulerian Graphs* (make connected and add $\leq |\mathrm{V}(\mathrm{G})|/2$ edges), *Hamiltonian Graphs* (add $\leq |\mathrm{V}(\mathrm{G})| - 1$ edges), *Graphs with $K(\cdot)$-vertex Dominating Set* (add $\leq |\mathrm{V}(\mathrm{G})| - K(|\mathrm{V}(\mathrm{G})|)$ edges), *Graphs having Perfect Matching* (add $\leq |\mathrm{V}(\mathrm{G})|/2$ edges), and *Graphs containing a subgraph* H (add $\leq |\mathrm{E}(\mathrm{H})|$ edges).[16] We remark that the above also holds with respect to some of the complementing classes such as *UnConnected Graphs*, *Non-Hamiltonian Graphs*, *Non-Eulerian Graphs*, *Graphs without $K(\cdot)$-vertex Dominating Set* and *Graphs not having Perfect Matching* (e.g., by removing edges to make one vertex isolated).

Proof: Let $\Delta(N) \stackrel{\mathrm{def}}{=} \max_{\mathrm{G}:|\mathrm{V}(\mathrm{G})|=N}\{\mathrm{dist}(\mathrm{G},\mathcal{C})\}$ and suppose that $\Delta(N)$ or a good upper bound on it is known. (One may always use $\Delta(N) = N^{-\alpha}$.) On input an N-vertex graph G (and distance parameter ϵ), if $\epsilon > \Delta(N)$ then the tester always accepts. Otherwise, the tester inspects all $N^2 = \mathrm{poly}(1/\epsilon)$ edges and decides accordingly. Actually, we may take a sample of $O(\log(N/\delta)\cdot N^2)$ labeled random examples and accept iff either the sample does not cover all possible vertex pairs or the sample "reveals" a graph in $\mathcal{C}$. The main point is that in case $\epsilon > \Delta(N)$ every N-vertex graph is ϵ-close to $\mathcal{C}$. ■

We stress that as long as the distance parameter (i.e., ϵ) is not too small, the tester is trivial (i.e., it accepts any graph without performing any checking). Slightly more is required in order to check any graph property which holds only on very sparse graphs. Here, as long as ϵ is not too small, the tester need only check that random examples of vertex-pairs have no edge between them.

Proposition 3.14 (testing classes of sparse graphs): *Let $\alpha > 0$ be a constant and $\mathcal{C}$ a class of graphs so that $\mathcal{C} \subseteq \{\mathrm{G} : |\mathrm{E}(\mathrm{G})| < |\mathrm{V}(\mathrm{G})|^{2-\alpha}\}$. Then, $\mathcal{C}$ can be tested by using at most* $\mathrm{poly}(\log(1/\delta)/\epsilon)$ *labeled random examples, where ϵ and δ are the distance and confidence parameters.*

[16]The latter can be generalized to *Graphs containing a subgraph which can be contracted to one of the graphs in* $\{\mathrm{H}_1, ..., \mathrm{H}_t\}$. The Non-Planar Graphs are a special case.

Classes which satisfy the hypothesis of the proposition include: *Trees*, *Forests* and *Planar Graphs.*

Proof: Let $\rho(N) \stackrel{\rm def}{=} \max_{G\in\mathcal{C}:|V(G)|=N}\{|E(G)|/N^2\}$ and suppose that $\rho(N)$ or a good upper bound on it is known. (One may always use $\rho(N) = N^{-\alpha}$.) On input an N-vertex graph G (and parameters ϵ, δ), if $\epsilon > 3\rho(N)$ then the tester takes a sample of $t \stackrel{\rm def}{=} O(\log(1/\delta)/\epsilon^2)$ labeled random examples and accepts iff it has seen at most $(\rho(N) + (\epsilon/3)) \cdot t$ edges. Otherwise, the tester inspects all $N^2 = \mathrm{poly}(1/\epsilon)$ edges and decides accordingly. (Again, the actual implementation is by a sample of $O(\log(N/\delta) \cdot N^2)$ edges which is very likely to cover all vertex pairs.) The main point is that, in case $\epsilon > 3\rho(N)$, the graphs which are ϵ-away from $\mathcal{C}$ have edge density at least $\epsilon - \rho(N) > \rho(N) + 2\epsilon/3$. We conclude by noting that, with probability at least $1 - \delta$ the number of edges seen by the tester provides a good estimate (i.e., $\epsilon/3$ deviation) to the density of the graph. ■

We stress that both the above testers make no queries. In contrary, the following slightly more involved tester does make queries in order to check that vertices drawn at random have about the same degree. This algorithm is a tester for the class of Regular Graphs, and its correctness is established based on a theorem due to Noga Alon (private communication, 13th April 1996).

Proposition 3.15 (testing regular graphs): *The class of regular graphs can be tested by using at most* $\mathrm{poly}(\log(1/\delta)/\epsilon)$ *queries, where* ϵ *and* δ *are the distance and confidence parameters.*

Proof: On input an N-vertex graph G (and parameters ϵ, δ), the test takes a sample, S, of $O(\log(1/\delta)/\epsilon)$ many vertices and for each $v \in S$ makes $O(\log(1/\epsilon\delta)/\epsilon^2)$ queries of the form "is $(v, w) \in E(G)$", where $w \in V(G)$ is uniformly chosen. Thus, the test estimates the degrees of all vertices in S. The test accept iff all the estimated degrees (divided by N) are within $\epsilon/3$ of one another.

Clearly, if G is regular then, with probability at least $1 - \delta$, the test accepts it. Assume, on the other hand, that the test accepts with probability greater than δ. Then, for some ρ and $\epsilon' = \epsilon/10$, it must be the case that all but an ϵ' fraction of the vertices in G have degree $(\rho \pm \epsilon')N$. By omitting/adding edges to the few vertices with degree outside the above interval, we obtain a graph G$'$ so that

1. $\mathrm{dist}(G, G') \leq \epsilon'$.
2. every vertex in G$'$ has degree $(\rho \pm 2\epsilon')N$.

At this point we invoke a theorem due to Noga Alon (see Appendix A.1) which asserts that a graph G$'$ in which the difference between the maximum and minimum degree is bounded above by $\epsilon''|V(G')|$ is at most $(3 + o(1)) \cdot \epsilon''$-away from the class of regular graphs. Thus, G is at most ϵ-away from this class, and the proposition follows. ■

Estimating vertex degree also suffices to test that the minimum cut in the graph is above some threshold. That is,

Proposition 3.16 (testing min-cut): *The class of graphs* G *with minimum cut at least* $K = K(|V(G)|)$ *can be tested by using at most* $\mathrm{poly}(\log(1/\delta)/\epsilon)$ *queries, where* ϵ *and* δ *are the distance and confidence parameters.*

Proof: If $\epsilon = O(\log(N)/N)$ then we examine the entire graph. Otherwise, we merely test via a $\mathrm{poly}(1/\epsilon)\log(1/\delta)$ sample that all vertices seem to have degree (approximately) above K. That is, to test that the minimum cut is at least $K = K(|V(G)|)$ we sample sufficiently many vertices, approximate their degree according to the sample, and accept iff all estimated degrees are above, say, $K - \frac{\epsilon}{2}|\mathrm{V(G)}|$. The analysis utilizes the observation that at most $O(N \log N)$ edges must be added to an N-vertex graph of minimum degree K in order to make it have min-cut at least K. This observation is proved by a random construction.

The basic idea is to consider the immediate neighborhoods of each of the N vertices in the graph. We get a collection of subsets, each having cardinality at least K. Thus, all that is needed is to guarantee K edge-disjoint paths between each pair of such subsets. This can be done easily, by designating K special vertices in the graph, and randomly connect each vertex in the graph to the designated set by $O(\log N)$ random edges. Consider one specific neighborhood (out of the N). With probability greater than $1 - \frac{1}{N}$, the random edges (from its vertices to the designated set) contain a K-matching. Thus, we obtain K edge-disjoint paths between each pair of neighborhoods, which implies that the augmented graph id K-edge-connected. ■

Testing graph properties using the Regularity Lemma. As noted above, much less efficient testers for k-Colorability and other graph properties can be obtained by using the Regularity Lemma of Szemerédi [Sze78]. Interestingly, the Regularity Lemma yields the following result about testing, which we do not know to obtain without it. Let H be an arbitrary fixed graph (e.g., the triangle K_3) and consider the class of graphs which have no H subgraphs. Using the Regularity Lemma, Noga Alon (private communication) observed that there exist testers for H-freeness, with query complexity which is a tower of $\mathrm{poly}(1/\epsilon)$ exponents. Recall that ϵ is our distance parameter (i.e., Alon's tester rejects a graph if it is ϵ-far from being H-free). Alon expressed the opinion that proving a result like this without the Regularity Lemma (and hence getting better bounds) would be, indeed, very challenging, and would probably have some very nice combinatorial applications.

Hard to test graph properties. Analogously to Proposition 2.7, we show that there are graph properties requiring inspection of a constant fraction of all possible vertex-pairs.

Proposition 3.17 *There exists a class of graphs, $\mathcal{G}$, for which any testing algorithm must inspect a constant fraction of the vertex pairs. This holds even for testing with respect to the uniform distributions, for any distance parameter $\epsilon < 1/2$ and confidence parameter $\delta < 1/2$, and when allowing the algorithm to make queries and use unlimited computing time.*

Proof: In adapting the proof of Proposition 2.7, we introduce for each N a random subset of $2^{\frac{1}{20}N^2}$ N-vertex graphs. Each graph is specified by the lower triangle of the corresponding adjacency matrix. This allows at most $N!$ representations of the same graph (i.e., all its automorphism). The multiple representation only effects the first part of the proof; that is, the bound on the probability that a uniformly selected graph is ϵ-close to $\mathcal{G}$. However, the extra factor of $N!$ is easily eliminated by the probability $\exp(-\Omega(N^2))$ that a random graph is ϵ-close to a specific graph. The second part of the proof (i.e., the distance between the two observed distributions) remains almost unchanged. ■

Actually, a similar result holds with respect to graph properties which are in $\mathcal{NP}$; that is, classes of graphs which constitute $\mathcal{NP}$ sets.

Proposition 3.18 *There exists an NP set of graphs, $\mathcal{G}$, for which any testing algorithm must inspect at least $\Omega(N^2)$ of the vertex pairs, where N is the number of vertices in the graph. This holds even for testing with respect to the uniform distributions, for any distance parameter $\epsilon < 1/2$ and confidence parameter $\delta < 1/2$, and when allowing the algorithm to make queries and use unlimited computing time.*

Proof: We adapt the proof of Proposition 3.17, by considering, for each N, all graphs which arise for particular "pseudorandom" sequences. Specifically, we consider $\binom{N}{2}$-long sequences taken from an $\epsilon \cdot 2^{-t}$-biased sample space (cf., [NN93] or [AGHP92]), where $t \stackrel{\text{def}}{=} \frac{1}{20}N^2$. Efficiently constructible sample spaces of size $\text{poly}(2^t/\epsilon)$ having the above property can be found in [NN93, AGHP92]. Graphs are now specified, as before, by letting each such sequence define (the lower triangle of) the corresponding adjacency matrix. The first part of the proof remains unchanged (since all that matters is the number of graphs in the class). The second part of the proof is actually simplified since any t observed bits in the a random sequence as above deviates (in max-norm) from the uniform distribution by at most ϵ. All which remains is to be convinced that we have constructed an NP set. This follows by letting the NP-witness of the membership of a graph in the set be the isomorphism to the canonical representation (i.e., the representation corresponding to the almost unbiased sequence). $\blacksquare$

Acknowledgments

We wish to thank Noga Alon, Michel Goemans, Ravi Kannan, David Karger and Madhu Sudan for useful discussions.

Bibliography

[ADL$^+$94] N. Alon, R. A. Duke, H. Lefmann, V. Rodl, and R. Yuster. The algorithmic aspects of the regularity lemma. *Journal of Algorithms*, 16:80–109, 1994.

[AFK96] S. Arora, A. Frieze, and H. Kaplan. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. In *Proceedings of the Thirty-Seventh Annual Symposium on Foundations of Computer Science*, 1996.

[AGHP92] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Journal of Random structures and Algorithms*, 3(3):289–304, 1992.

[AKK95] S. Arora, D. Karger, and M Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 284–293, 1995.

[ALM$^+$92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proceedings of the Thirty-Third Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[Ang78] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39:337–350, 1978.

[AS92] S. Arora and S. Safra. Probabilistic checkable proofs: A new characterization of NP. In *Proceedings of the Thirty-Third Annual Symposium on Foundations of Computer Science*, pages 1–13, 1992.

[BCH$^+$95] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science*, pages 432–441, 1995.

[BD92] S. Ben-David. Can finite samples detect singularities of real-valued functions? In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 390–399, 1992.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, October 1989.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 294–304, 1993.

[BGS95] M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps and non-approximability – towards tight results. In *Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science*, pages 422–431, 1995. Full version available from *ECCC*, `http://www.eccc.uni-trier.de/eccc/`.

[BLR93] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–595, 1993.

[BR89] A. Blum and R. Rivest. Training a 3-node neural network is NP-complete. In *Advances in Neural Information Processing Systems I*, pages 494–501, 1989.

[BS94] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 184–193, 1994.

[Cov73] T. M. Cover. On determining the rationality of the mean of a random variable. *Annals of Statistics*, 1:862–871, 1973.

[dlV94] W. F. de la Vega. MAX-CUT has a randomized approximation scheme in dense graphs. To appear in *Random Structures and Algorithms*, 1994.

[Edw86] K. Edwards. The complexity of colouring problems on dense graphs. *Theoretical Computer Science*, 43:337–343, 1986.

[FGL$^+$91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the Thirty-Second Annual Symposium on Foundations of Computer Science*, pages 2–12, 1991.

[FK96] A. Frieze and R. Kanan. The regularity lemma and approximation schemes for dense problems. In *Proceedings of the Thirty-Seventh Annual Symposium on Foundations of Computer Science*, 1996.

[GLR$^+$91] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 32–42, 1991.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer–Verlag, 1988.

[Gol78] M. E. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[Gol95] O. Goldreich. Foundations of Cryptography – Fragments of a Book. Available from *ECCC*, `http://www.eccc.uni-trier.de/eccc/`, 1995.

[Haj91] P. Hajnal. An $\Omega(n^{4/3})$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(2):131–144, 1991.

[HS87] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the Association for Computing Machinery*, 34(1):144–162, January 1987.

[HS88] D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for machine scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[HW58] W. Hoeffding and J. Wolfowitz. Distinguishability of sets of distributions. *Annals of Mathematical Statistics*, 29:700–718, 1958.

[Kin91] V. King. An $\Omega(n^{5/4})$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(1):23–32, 1991.

[Kiw96] M. Kiwi. *Probabilistically Checkable Proofs and the Testing of Hadamard-like Codes.* PhD thesis, Massachusetts Institute of Technology, 1996.

[KMR$^+$94] M. J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *The 25th Annual ACM Symposium on Theory of Computing*, pages 273–282, 1994.

[KMS94] D. R. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 2–13, 1994.

[KSS92] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 341–352, 1992.

[KZ93] S. R. Kulkarni and O. Zeitouni. On probably correct classification of concepts. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 111–116, 1993.

[Lip89] R. J. Lipton. New directions in testing. Unpublished manuscript, 1989.

[LY91] L. Lovász and N. Young. Lecture notes on evasiveness of graph properties. Technical Report TR–317–91, Princeton University, Computer Science Department, 1991.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[Pet94] E. Petrank. The hardness of approximations: Gap location. *Computational Complexity*, 4:133–157, 1994.

[PV88] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of the Association for Computing Machinery*, 35(4):965–984, October 1988.

[PW93] L. Pitt and M. K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the Association for Computing Machinery*, 40(1):95–142, January 1993.

[Ros73] A. L. Rosenberg. On the time required to recognize properties of graphs: A problem. *SIGACT News*, 5:15–16, 1973.

[RS96] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[Rub94] R. Rubinfeld. Robust functional equations and their applications to program testing. In *Proceedings of the Thirty-Fifth Annual Symposium on Foundations of Computer Science*, 1994.

[RV76] R. L. Rivest and J. Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3:371–384, 1976.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27:701–717, 1980.

[Sze78] E. Szemeŕedi. Regular partitions of graphs. In *Proceedings, Colloque Inter. CNRS*, pages 399–401, 1978.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, 17(2):264–280, 1971.

[Yam95] K. Yamanishi. Probably almost discriminative learning. *Machine Learning*, 18:23–50, 1995.

[Yao87] A. C. C. Yao. Lower bounds to randomized algorithms for graph properties. In *Proceedings of the Twenty-Eighth Annual Symposium on Foundations of Computer Science*, pages 393–400, 1987.

[ZK91] O. Zeitouni and S. R. Kulkarni. A general classification rule for probability measures. To appear in Annals of Statistics, 1991.

Appendix A

Technical Appendix

A.1 A note on Regular Graphs (by Noga Alon)

The following theorem is due to Noga Alon. We stress that the theorem refers to simple undirected graphs (i.e., with no self-loops and no parallel edges).

Theorem A.1 *Let* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ *be a graph on* N *vertices with maximum degree* D *and minimum degree* d*, where* $D - d \leq \epsilon N$*. Then there is a regular graph on* N *vertices obtained from* G *by omitting and adding at most* $3\epsilon N^2 + 2N$ *edges.*

Proof: By replacing G, if needed, with its complement, we may and will assume that $D \leq \frac{1}{2}(1 + \epsilon)N$. It is convenient to first reduce to the case in which the maximum degree is a bit smaller than $0.5N$. This can be done as follows. Delete the maximum possible number of edges from G subject to keeping its minimum degree d. Let us denote the resulting graph by G' and the maximum degree in it by D'. In case $D' = d$ we are done. Otherwise, we consider the set of vertices of degree D' and observe that it is an independent set (since the existence of an edge between two vertices of degree greater than d violates the maximality of G'). Thus, for each set A of vertices of maximum degree in G', $|\Gamma(A)| > |A|$, since the number of edges from A to $\Gamma(A)$ equals $|A| \cdot D'$ as well as is bounded above by $|\Gamma(A)| \cdot (D' - 1)$. Therefore, by Hall's theorem, there is a matching in G' that saturates all vertices of degree D'. Omitting such a matching, we obtain a graph, denoted G_1, of maximum degree $D_1 = D' - 1$ and minimum degree $d_1 \in \{d, d-1\}$. Iterating this procedure (of first omitting edges between vertices of maximum degree and then omitting an appropriate matching) we obtain a sequence of graphs $\mathrm{G}_1, \mathrm{G}_2, ..., \mathrm{G}_t$, stopping if either G_t is regular or if $t = \frac{\epsilon}{2}N + 2$. Let $(D_1, d_1), ..., (D_t, d_t)$ be the corresponding sequence of maximum and minimum degrees in the G_i's. Since $d_t \geq d - t$ we conclude that during this process we have omitted at most $(D - (d - t)) \cdot N \leq 1.5\epsilon N^2 + 2N$ edges. In case G_t is regular, we are done. Otherwise, we have $D_t - d_t \leq \epsilon N$ and $D_t \leq 0.5N - 2$. We let $H \stackrel{\mathrm{def}}{=} \mathrm{G}_t$, $D' \stackrel{\mathrm{def}}{=} D_t$, $d' \stackrel{\mathrm{def}}{=} d_t$, and let D'' be the smallest even integer which is at least D' (i.e., $D'' = D' + (\lceil D'/2 \rceil - \lfloor D'/2 \rfloor) \in \{D', D' + 1\}$).

To complete the proof we show how to modify H by omitting and adding to it at most $O(\epsilon n^2)$ edges so that the resulting graph will be D''-regular. For each vertex v of H, define $s(v) = D'' - d(v)$, where $d(v)$ is the degree of v in H. Thus the sum $S = \sum_{v \in V} s(v)$ is even, and we have to increase the degree of each vertex v by $s(v)$. We do so in $S/2$ steps, where in each step we increase by 1 the degrees of two (not necessarily distinct) vertices with positive s values, and keep the other degrees invariant. Specifically, in each step we either add one edge or add two edges and remove one edge. In either case, we update the relevant s values. Since $S \leq (D'' - d')N \leq \epsilon N^2$ the desired result follows. Here is a typical step; as long as there are some vertices with positive s values, proceed as follows.

Case 1 *there is a vertex v for which $s(v) \geq 2$.* Let Y denote the set of all its non-neighbors. If some member of Y has a positive s value, connect it to v and update their s values to complete the step. Else, each member of Y has degree D''. Note that $|Y| > N/2$ (since $|Y| \geq N - (D'' - 2) \geq N - D' + 1 \geq 0.5N - 1$). Hence there must be an edge between two members of Y, since otherwise $|Y| \cdot D'' \leq (N - |Y|) \cdot D''$. Let uw be such an edge, omit it from the graph and add the two edges vu and vw, update the s value of v and complete the step.

Case 2 *the only vertices v with positive s values have $s(v) = 1$.* Since the sum of the s-values is even, there are at least two such vertices, say, u and v. Let Y be the set of all non-neighbors of u and Z the set of all non-neighbors of v and note that, as before, if some vertex in either Y or Z has a positive s value we can connect it to either u or v and complete the step. We now claim that there must be an edge yz in the graph with $y \in Y$ and $z \in Z$, since otherwise all edges from Y lead to vertices in $V \setminus Z$, implying that $|Y| \cdot D'' \leq (N - |Z|) \cdot D''$, which is impossible, since both $|Y|$ and $|Z|$ are greater than $N/2$. We can now add the edges uy and zv, remove the edge yz, and update the s values of u and v, completing the step.

Observing the the maximum number of edge modifications in these $S/2$ steps is $\frac{3}{2}S \leq \frac{3}{2}\epsilon N^2$, the theorem follows. ■

A.2 Determining closeness to an Admittable Set

In this appendix we consider the computational problem of determining whether a sequence of densities is $2\epsilon'$-close to an admissible set for Φ. That is,

Input:
- parameters: $\rho_1^{\mathrm{lb}}, \ldots, \rho_k^{\mathrm{lb}}$, $\rho_1^{\mathrm{ub}}, \ldots, \rho_k^{\mathrm{ub}}$, $\varrho_{1,1}^{\mathrm{lb}}, \ldots, \varrho_{k,k}^{\mathrm{lb}}$, $\varrho_{1,1}^{\mathrm{ub}}, \ldots, \varrho_{k,k}^{\mathrm{ub}}$, and ϵ'.
- densities: $\rho_1, \ldots, \rho_k$ and $\varrho_{1,1}, \ldots, \varrho_{k,k}$.

Question: Does the following system of inequalities in x_i's and $y_{i,j}$'s have a solution?

$$\sum_{i=1}^{k} x_i = 1 \quad \text{and} \quad \sum_{i,j=1}^{k} y_{i,j} \leq 1 \tag{A.1}$$

$$\rho_i^{\mathrm{lb}} \leq x_i \leq \rho_i^{\mathrm{ub}} \quad (\forall i\,) \quad \text{and} \quad \varrho_{i,j}^{\mathrm{lb}} \leq y_{i,j} \leq \varrho_{i,j}^{\mathrm{ub}} \quad (\forall i,j\,) \tag{A.2}$$

$$y_{i,i} \leq x_i^2 \quad (\forall i\,) \quad \text{and} \quad y_{i,j} \leq 2 \cdot x_i \cdot x_j \quad (\forall i \neq j\,) \tag{A.3}$$

$$|x_i - \rho_i| \leq 2\epsilon' \quad \text{and} \quad |y_{i,j} - \varrho_{i,j}| \leq 2\epsilon' \quad (\forall i,j\,). \tag{A.4}$$

We first observe that the corresponding lower and upper bounds in Eq. (A.2) and Eq. (A.4) can be combined. We also observe that the above system has a solution if and only if it has a solution in which the $y_{i,j}$ are set to be as small as possible. That is, a solution in which each $y_{i,j}$ has the minimum value that obeys the lower bounds in Equations Eq. (A.2) and Eq. (A.4). This yields the following system of inequalities, where $L_i = \max\{\rho_i^{\mathrm{lb}}, \rho_i - 2\epsilon'\}$, $U_i = \min\{\rho_i^{\mathrm{ub}}, \rho_i + 2\epsilon'\}$, $L_{i,i} = \max\{\varrho_{i,i}^{\mathrm{lb}}, \varrho_{i,i} - 2\epsilon'\}$, and $L_{i,j} = \frac{1}{2} \cdot \max\{\varrho_{i,j}^{\mathrm{lb}}, \varrho_{i,j} - 2\epsilon'\}$ $(i \neq j)$:

$$\sum_{i=1}^{k} x_i = 1 \tag{A.5}$$

$$L_i \leq x_i \leq U_i \quad (\forall i\,) \tag{A.6}$$

$$x_i \cdot x_j \geq L_{i,j} \quad (\forall i, j\,) \tag{A.7}$$

We first observe that Eq. (A.5)–(A.7) constitute a Convex Program; furthermore, its feasibility region, in case it is not empty, is a t-dimensional convex set, where $t \leq k-1$. Next, we observe that this convex set contains any simplex defined by $t+1$ points of general position inside the convex set. This holds, in particular, for points which are on the intersection of t of the boundaries/inequalities (i.e., "vertices" of the body). It can be easily verified that for any such two points and for any coordinate, if the points are different along this coordinate then their difference is bounded below by $2^{-k \cdot L}$, where L is the length of the encoding (in binary) of Φ and ϵ. It follows that the feasibility region, if not empty, contains a t-dimensional ball of radius $2^{-\mathrm{poly}(k) \cdot L}$ (and is contained in $[0,1]^t$). Thus, the feasibility problem can be solved by exhaustive search in $\exp(\mathrm{poly}(k) \cdot L)$-time: First, we reduce the problem to t dimensions, by guessing $k-t$ ("independent") inequalities which are satisfied at equality. Next, we search the resulting t-dimensional space for a feasible solution, by examining all points which reside on a cubic integer lattice spanned by vectors of length $2^{-\mathrm{poly}(k) \cdot L}$.

REMARK: It seems that the feasibility problem can be solvable by the Ellipsoid Method (cf., [GLS88]) in poly(L)-time, but this saving has little affect on our application.