Property Testing and Its Connection to Learning and Approximation

Oded Goldreich[*] Shafi Goldwasser[†] Dana Ron[‡]

February 13, 1998

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 with respect to the vertex set). Our graph property testing algorithms are probabilistic and make assertions that are correct with high probability, while making a number of queries that is *independent* of the size of 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 that correspond to the property being tested, if it holds for the input graph.

1 Introduction

Property Testing is concerned with the computational task of determining whether a given object has a predetermined property or is "far" from any object having the property. A notion of Property Testing was first formulated in [RS96]. In their formulation, a property testing algorithm for property P is given oracle access to the tested function f. The algorithm must distinguish the case that f has property P from the case that f is far from any function having the property. Distance between functions is measured in terms of the fraction of arguments in the domain on which the functions disagree. Note that property testing so defined is a relaxation of the standard decision task (of distinguishing the case that f has property P from the case f does not have property P). Here we are interested in testers that are far more efficient than the corresponding decision procedure.

Property testing emerges naturally in the context of program checking and probabilistically checkable proofs (PCP). Specifically, in the context of program checking, one may choose to test that

[*]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.

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

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

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. In the context of probabilistically checkable proofs, 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 [BFL91, BFLS91, FGL+91, AS92, ALM+92] to testing Hadamard codes [ALM+92, BGLR93, BS94, BCH+95, Kiw96, Tre97], and recently to testing the "long code" [BGS95, Hås96b, Hås97, Tre97]. All the above has focused on property testing in the sense of [RS96], and on testing algebraic properties such as linearity, multi-linearity and being a low-degree polynomial.

In this work we extend the scope of property testing in two ways:

1. Working within the above framework, we venture into the domain of *combinatorial objects*. In particular, we study property testing as applied to graph properties, demonstrating its relevance to standard and dual notions of approximation, and derive extremely fast algorithms for testing several natural graph properties. These in turn yield results such as a constant-time approximation scheme for Max-CUT in dense graphs. We believe that, in general, property testing offers a new perspective on approximation problems.

2. We generalize the above definition so as to allow an arbitrary probability distribution D over arguments to the function f, as well as the consideration of algorithms that only obtain random labeled examples, of the form $(x, f(x))$, where x is selected according to D. Distance between functions is measured, accordingly, with respect to D. This formulation is inspired by the PAC learning model [Val84], and we indeed relate property testing so formulated to various variants of the PAC model. We believe that property testing offers new perspective with respect to computational learning theory.

We start with the second item.

1.1 General Property Testing

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 that 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.

Let $\mathcal{F}$ be the class of functions that satisfy property P. Then, testing property P corresponds to testing membership in the class $\mathcal{F}$. The two most relevant parameters to property testing are the permitted distance, hereafter denoted ϵ, and the desired confidence, denoted δ. We require the tester to accept each function in $\mathcal{F}$ and reject every function that 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.

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, theoretical analysis of learning algorithms typically works under the postulation that the function (representing the environment) belongs to a particular class.[1] It may be more efficient to test this postulation first before trying to learn the function (and possibly failing when the postulation is wrong).[2] We stress that the generalization of property testing to arbitrary distributions and to algorithm that only obtain random labeled examples is essential to the potential applications mentioned above.

PROPERTY TESTING AND LEARNING THEORY. 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). (An insignificant 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 $\mathcal{F}$.

We show that the relation between learning and testing is non-trivial. On one hand, *proper* learning (i.e., when the hypothesis of the learning algorithm must belong to the same class as the target function) implies testing. On the other hand, there are function classes for which testing is harder than (*non-proper*) learning (i.e. when the hypothesis is *not* required to belong to the same class as the target function), provided $\mathcal{NP} \not\subseteq \mathcal{BPP}$. Nonetheless, there are also function classes for which testing is much easier than learning. In addition, the graph properties discussed below provide a case where testing (with queries) is much easier than learning (also with queries).

The above results as well as additional results regarding the relations between property testing and learning appear in Section 3.

1.2 Testing Graph Properties

Property testing is a natural notion of approximation, and furthermore it is related to standard notions of approximation. This holds even with respect to the restricted notion of property testing where one considers only the uniform distribution and allows the algorithm to make queries of its choice [RS96]. The above assertion is demonstrated within one of the most basic domains of approximation algorithms – the one of graph algorithms. But let us start with a general high level discussion.

[1] Here and throughout the introduction we refer to the standard PAC learning model. We shortly discuss agnostic learning [KSS92], where no assumption is made on the target function, towards the end of Section 3.

[2] 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. As mentioned above, this paradigm has been followed both in the theory and practice of program checking.

1.2.1 Motivating Discussion

Throughout the rest of the introduction, we refer to property testing in the restricted sense of [RS96]. Recall that the definition of property testing is a relaxation of the standard definition of a decision task: The tester is allowed arbitrary behavior when the object does not have the property, and yet is "close" to an object having the property. Thus, a property tester may be far more efficient than a standard decision procedure (for the same property).

In case the object is huge, as when one thinks of a function and algorithms which operate in time polynomial in the length of the arguments to the function, there is actually no other alternative to approximation. That is, it is typically infeasible (i.e., requires exponential time in the length of the arguments) to decide whether such a function has the desired property. A property testing algorithm which operates in time polynomial in the length of the arguments thus offers a feasible approximation to a problem which is intractable in the exact formulation.

Property testers are valuable also in case one deals with objects of feasible size (i.e., size for which scanning the entire object is feasible): If a property tester is much faster than the exact decision procedure then it makes sense to run it before running the decision procedure. In case the object is far from having the property, we may obtain an indication towards this fact, and save the time we might have used running the decision procedure. In case the tester supplies proofs of violation of the property (as in some of the testers discussed below), we obtain an absolutely correct answer without running the decision procedure at all. Thus, we may only need to run the decision procedure on objects which are close to having the property. In some setting where *typical* objects are either good (i.e., have the property) or very bad (i.e., are very far from objects having the property), we may gain a lot. Furthermore, *if* it is *guaranteed* that objects are either good or very bad then we may not even need to run the decision procedure at all. The gain in such a setting is enormous.

Being close to an object which has the property is a notion of approximation which, in certain applications, may be of great value. In some cases, being close to an object having the property translates to a standard notion of approximation. In other cases, it translates to a notion of "dual approximation". This point is clarified and exemplified below (by referring to specific properties). In both cases, a fast property tester which is more efficient than the decision procedure is of value, both if the decision procedure is feasible and more so if it is not.

Alternatively, we may be forced to take action, without having time to run a decision procedure, while given the option of modifying the object in the future, at a cost proportional to the number of modifications of the object. For example, suppose you are given a graph which represents some design problem, where Bipartite graphs corresponds to a good design and changes in the design correspond to edge additions/omissions. Using a Bipartiteness tester you always accept a good design, and reject with high probability designs which will cost a lot to modify. You may still accept bad designs, but then you know that it will not cost you much to modify them later.

1.2.2 Representing graphs as functions.

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. Accordingly, distance between two N-vertex graphs is defined as the fraction (over N^2) of vertex-pairs that are adjacent in one graph but not in the other. A testing algorithm for property P is required to decide whether G has property P or is "ϵ-away" from any graph with property P, and is allowed to err with probability, say, $\delta = 1/3$. (For simplicity, we assume throughout the introduction that $\delta = 1/3$. Standard error-reduction techniques are obviously applicable to property testing. However, better dependencies on δ may be obtained in special cases as can be seen in the rest of the paper.)

A few comments are in place. Firstly, we note that this representation, as well as our results, are applicable both in case the graph is huge and in case it is of feasible size. The reader may thus choose whether to think of the graph as being huge and so accessible only by such queries, or as being explicitly given and inspected in an random-access manner. Secondly, the above adjacency predicate representation is most appropriate for dense graphs, and so the reader may think of the graph as being dense (e.g., having at least ϵN^2 edges). An alternative representation, appropriate for bounded-degree graphs, has been subsequently considered in [GR97a].

1.2.3 Our Algorithms

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

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

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

***ρ*-Cut** for any $\rho > 0$. That is, does the N-vertex graph have a cut of size at least ρN^2. A generalization to k-way cuts works with query-complexity $\mathrm{poly}((\log k)/\epsilon)$, and has running time $\exp(poly(\log k/\epsilon))$.

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

Furthermore:

1. 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 $\mathrm{poly}(1/\epsilon)\cdot N$-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.

2. The k-Colorability tester has one-sided error: it always accepts k-Colorable graphs. Furthermore, when rejecting a graph, this tester always supplies a $\mathrm{poly}(1/\epsilon)$-size subgraph which is not k-Colorable. All other algorithms have two-sided error, and this is unavoidable within $o(N)$ query-complexity.

3. 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 $\Psi : \mathrm{V(G)}^2 \mapsto [0,1]$ of the form $\Psi(u,v) = \psi(u)\cdot\psi(v)$, where $\psi : \mathrm{V(G)} \mapsto [0,1]$ is a distribution on the vertices, which is poly-time sampleable (by the tester). In contrast, it is not possible to test any of the graph properties discussed above in a distribution-free manner.

We comment that, except for Bipartite (2-Colorability) testing, running-time of $\mathrm{poly}(1/\epsilon)$ is unlikely, as it will imply $\mathcal{NP} \subseteq \mathcal{BPP}$. Also, none of these properties can be tested without queries when using $o(\sqrt{N})$ random examples.

[3] 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.

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 component of the partition as well as the number of edges between each pair of components 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 $\widetilde{O}(k^2/\epsilon)^{2k+O(1)}$ 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 that are very easy to test (e.g., Connectivity, Hamiltonicity, and Planarity). The reason being that for these properties either every N-vertex graph is at distance at most $O(1/N)$ from a graph having the desired property (and so for $\epsilon = \Omega(1/N)$ the trivial algorithm which always accepts will do), or the property holds only for sparse graphs (and so for $\epsilon = \Omega(1/N)$ one may reject any non-sparse graph). On the other hand, there are graph properties in $\mathcal{NP}$ that are extremely hard to test; namely, any testing algorithm must inspect at least $\Omega(N^2)$ 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 be very challenging.

1.2.4 Testing versus deciding and approximating

We shortly discuss the relation between testing graph properties and some well-known computational tasks.

Relation to deciding (recognizing) graph properties: Our notion of testing a graph property P is a *relaxation* of the notion of *deciding (recognizing) the graph property* P which has received much attention in the last three 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 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 also $\Omega(N^2)$. Progress towards proving this conjecture was made in [Yao87], [Kin91] and [Haj91] culminating in an $\Omega(N^{4/3})$ lower bound. This stands in striking contrast to the results mentioned above, by which some non-trivial monotone graph properties can be *tested* by examining a constant number of locations in the matrix.

Application to the standard notion of approximation: The relation of testing graph properties to the standard notions of approximation is best illustrated in the case of Max-CUT. Any tester for the class ρ-Cut, working in time $T(\rho, \epsilon, N)$, yields an algorithm for approximating the maximum cut in an N-vertex graph, up to additive error ϵN^2, in time $\frac{1}{\epsilon} \cdot T(\rho, \epsilon, N)$. Thus, for any constant $\epsilon > 0$, using the abovementioned tester we can approximate the size of the max-cut to within ϵN^2 in constant time.[4] This yields a **constant time approximation scheme** (i.e., to within any constant relative error) for dense graphs, improving over previous work of Arora *et. al.* [AKK95]

[4] We comment that due to the specific structure of our tester, the value of the maximum cut is actually approximated in time $T(\rho, \epsilon, N)\mathrm{poly}(1/\epsilon)$, rather than $\frac{1}{\epsilon} \cdot T(\rho, \epsilon, N)$.

and de la Vega [dlV94] who solved this problem in polynomial-time (i.e., in $O(N^{1/\epsilon^2})$–time and $\left(\exp(\widetilde{O}(1/\epsilon^2)) \cdot N^2\right)$–time, respectively). In the latter 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 (i.e., $\left(\widetilde{O}(1/\epsilon^2) \cdot N + \exp(\widetilde{O}(1/\epsilon^3))\right)$–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 of this 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 [ADL$^+$94].

Relation to "dual approximation" (cf., [HS87, HS88]): To illustrate this relation, we consider the ρ-Clique Tester mentioned above. The traditional notion of approximating Max–Clique corresponds to distinguishing 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 $\rho N/2$. On the other hand, 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. This is equivalent to the "dual approximation" task of distinguishing the case in which an N-vertex graph has a clique of size ρN from the case in which any ρN subset of the vertices misses at least ϵN^2 edges. To demonstrate that these two tasks are vastly different we mention that whereas the former task is NP-Hard, for $\rho < 1/4$ (see [BGS95, Hås96b, Hås96a]), the latter task can be solved in $\exp(O(1/\epsilon^2))$-time, for any $\rho, \epsilon > 0$. We believe that there is no absolute sense in which one of these approximation tasks is more important than the other: Each of these tasks may be relevant in some applications and irrelevant in others.

As another illustration of the applicability to "dual approximation" problems, we discuss our results regarding testing k-Colorability. It is known 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.[5]

Another application of our 3-Colorability Tester uses the fact that, for every $\epsilon > 0$, in case the N-vertex graph is 3-colorable, the tester may retrieve in linear time a 3-partition which violates at most ϵN^2 edges. Thus, we may reduce the general problem of coloring 3-Colorable graphs with few edges 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 3 times more colors than the number used by the coloring of the non-dense graphs, but a factor of 3 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.*, [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 $\alpha > \frac{k-3}{k-2}$.

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

1.2.5 Our Techniques

Our algorithms share some underlying ideas. The first is the uniform selection of a small sample of vertices 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* iof [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 at least $(\rho - \epsilon)N^2$ 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 that 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 that are implemented by exhaustive search are used (to induce a partition on the entire graph).

A simple observation which is useful in our analysis is that we do not need the sample to approximate well all relevant quantities (e.g., the degree of all vertices in the graph). It suffices to approximate well most of these quantities. This observation may explain how we can manage with a sample of size independent of the size of the graph.

1.3 Other related work

PROPERTY TESTERS IMPLICIT IN PREVIOUS WORKS. As mentioned above, results on testing graph properties can be derived from [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 with query complexity $f(\epsilon, k)^2$ for k-Colorability. In comparison with our k-Colorability Tester, which takes a sample of $O(\epsilon^{-3}k^2 \log k)$ 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., $\log^* f(\epsilon, k) = (k/\epsilon)^{20}$).

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 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 Kularni 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].

Approximation in Dense Graphs. As stated previously, [AKK95] and [dlV94] presented polynomial-time approximation schemes (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.[6] The methods of de la Vega [dlV94] are purely combinatorial and apply also to similar graph partition problems. Following the approach of [ADL$^+$94], but using a relaxation of the regularity Lemma (and thus obtaining much improved running times), Frieze and Kanan [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 (not to mention that we obtain constant-time approximation schemes for approximating only the value of the partition). 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.

We note that [AKK95] showed that the "dense subgraph" problem, a generalization of ρ-Clique, has a PTAS for dense instances. Our General Graph Partition algorithm (with the appropriate setting of the parameters) improves on their result.

1.4 Subsequent work

As mentioned above, our representation of graphs by their adjacency predicate is most adequate for dense graphs. Another natural representation, most adequate for bounded-degree graphs was subsequently suggested in [GR97a]: An N-vertex graph of degree bound d is represented there by the *incidence function*, $g : \mathrm{V} \times [d] \mapsto \mathrm{V} \cup \{0\}$, so that $g(u,i) = v$ if v is the i^{th} vertex incident at u, and $g(u,i) = 0 \notin \mathrm{V}$ if u has less than i neighbors.

As usual, the choice of representation has a fundamental impact on the potential algorithm.

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

Here the impact is even more dramatic since we seek algorithms which only inspect a relatively small fraction of the object (graph represented by a function). Furthermore, there is another fundamental impact of the choice of representation on the task of property testing. This has to do with our definition of distance, which is relative to the size of the domain of the function. In particular, distance ϵ in the adjacency predicate representation (adopted in this paper) means a symmetric difference of $2\epsilon \cdot N^2$ edges, whereas in the incident function representation (of [GR97a]) this means a symmetric difference of $2\epsilon \cdot dN$ edges. (In both cases, the extra factor 2 is due to the redundant representation which is adopted for sake of simplicity.)

In contrast to our poly$(1/\epsilon)$-query Bipartite tester (for the adjacency predicate representation), it was proven in [GR97a] that testing Bipartiteness in the incident function representation requires $\Omega(\sqrt{N})$ queries. Interestingly, this bound is tight up to a polylogarithmic factor, as shown in [GR97b] which presents a Bipartite tester for the incident function representation working in time $O(\text{poly}(\epsilon^{-1}\log N) \cdot \sqrt{N})$. We mention that [GR97a] also presents poly$(1/\epsilon)$-time algorithms for testing k-Connectivity, for $k \geq 1$, Planarity and other properties (all in the incident function representation).

In recent work, Kearns and Ron [KR98] generalize our definition of property testing, and present testing algorithms that use only random examples for classes of functions that have been studied in the learning literature.

We also mention the work of Ergun et al. [EKK+98], in which they propose the study of *Spot Checker*, which in some contexts coincides with property testing.

1.5 Organization

The paper is organized in two parts. The first part focuses on property testing in general: A definition is given and discussed in Section 2. In Section 3 we explore the relations between property testing and learning. General observations regarding property testing appear in Section 4.

The second part of the paper focuses on testing graph properties: The basic framework is presented in Section 5. In Section 6 we present the Bipartiteness tester, and its generalization to a k-Colorability Tester. In Section 7 we present our tester for ρ-Clique. In Section 8 we present our tester for ρ-CUT, and its generalization to Bisection. The general graph-partition problem is treated in Section 9. We conclude, in Section 10, with comments regarding extensions and limitations of the above algorithms and problems, as well as discuss other graph properties

Appendix A contains a list of some recurring notation, and Appendix B recalls standard probabilistic inequalities which are extensively used.

Part I
General Property Testing

2 Definitions

Let $\mathcal{F} = \{\mathcal{F}_n\}$ be a parameterized class of functions, where the functions[7] in $\mathcal{F}_n$ are defined over $\{0,1\}^n$ and let $\mathcal{D} = \{\mathrm{D}_n\}$ be a corresponding class of distributions (i.e., D_n is a distribution on $\{0,1\}^n$). We use $x \sim \mathrm{D}_n$ to denote that x is distributed according to the distribution D_n. We say

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

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

$$\Pr_{x \sim \mathrm{D}_n}[f(x) \neq g(x)] \leq \epsilon \ . \tag{1}$$

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 (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, *$(x_1, f(x_1)), (x_2, f(x_2)), \ldots$, 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. 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 may have only *one-sided error*. Namely, in case $f \in \mathcal{F}_\backslash$, the algorithm *always* accepts f.
6. 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.

3 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 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 (1) (and ϵ is usually referred to as the *accuracy* 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, $h \in \mathcal{H}$, for some hypothesis class $\mathcal{H}$ such that $\mathcal{F} \subseteq \mathcal{H}$. We assume that a proper learning algorithm (for $\mathcal{F}$) either halts without output or outputs a function in $\mathcal{F}$, but it never outputs any function not in $\mathcal{F}$.[8] 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 (which in the case of boolean functions is referred to as a *membership oracle*)). Unless stated otherwise, whenever we refer in this section to PAC learning we mean the *distribution-free no-query non-proper 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.

Proposition 3.1 *If a function class $\mathcal{F}$ has a* proper *learning algorithm $\mathcal{A}$, then $\mathcal{F}$ has a property testing algorithm $\mathcal{A}'$ with sample complexity*

$$m_{\mathcal{A}'}(n,\epsilon,\delta) = m_{\mathcal{A}}(n,\epsilon/2,\delta/2) + O\left(\frac{\log(1/\delta)}{\epsilon}\right)$$

where $m_{\mathcal{A}}(\cdot,\cdot,\cdot)$ is the sample complexity of $\mathcal{A}$. Furthermore, the same relation holds between the running times of the two algorithm.

Proof: In order to test if $f \in \mathcal{F}$ or is ϵ-far from any function in $\mathcal{F}$, we first run the learning algorithm $\mathcal{A}$ with confidence parameter $\delta/2$, and accuracy parameter $\epsilon/2$, using random examples labeled by f. 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 $O(\epsilon^{-1}\log(1/\delta))$. 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 $1-\delta/2$, $\mathcal{A}$'s output, h, is $\epsilon/2$-close to f, and an additive Chernoff bound (see Appendix B), tells us that with probability at least $1-\delta/2$ over the additional

[8] We remark that in case the functions in $\mathcal{F}$ have an easy to recognize representation, one can easily guarantee that the algorithm never outputs a function not in $\mathcal{F}$, by simply checking the hypothesis' representation. Standard classes considered in works on proper learning (e.g. Decision-Trees) typically have this feature.

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][9], 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 3.2 *Every class that is learnable with constant confidence using a sample of size* $\mathrm{poly}(n/\epsilon)$ *(and thus has a* $\mathrm{poly}(n)$ *VC dimension [BEHW89]), is testable with a* $\mathrm{poly}(n/\epsilon) \cdot \log(1/\delta)$ *sample (in at most exponential time).*

Testing may be Harder than Learning. In contrast to Proposition 3.1 and to Corollary 3.2, 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 3.1. Thus, the separation between efficient learning and efficient proper learning translates to a separation between efficient learning and efficient testing.

Proposition 3.3 *If* $\mathcal{NP} \not\subseteq \mathcal{BPP}$ *then there exist function classes which are not* $\mathrm{poly}(n/\epsilon)$*-time testable but are* $\mathrm{poly}(n/\epsilon)$*-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

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

question is there a function $f \in \mathcal{F}_n$ so that $f(x_i) = \sigma_i$, for all $i \in [t] \stackrel{\mathrm{def}}{=} \{1, \ldots, t\}$.

Assuming that there exists a $\mathrm{poly}(n/\epsilon)$–time property testing algorithm, denoted $\mathcal{A}$, for the class $\mathcal{F}$, we construct a polynomial-time decision procedure for the above problem (contradicting the assumption that NP $\not\subseteq$ BPP). We invoke $\mathcal{A}$ with parameters n, ϵ and δ, where $\epsilon < 1/t$ (say $\epsilon = 1/(2t)$) and say $\delta = 1/3$. Suppose that $\mathcal{A}$ requires $m \stackrel{\mathrm{def}}{=} m_{\mathcal{A}}(n, \epsilon, \delta)$ samples. We uniformly

[9] 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 $\mathcal{F}_n$.

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 uniform distribution over 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 uniform distribution over $\{x_i : i \in [t]\}$). 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 whereas Proposition 3.1 generalizes to learning and testing under specific distributions, and to learning and testing with queries, the proof of Proposition 3.3 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 that is easy to test but cannot be learned with polynomial sample complexity, regardless of the running-time.

Proposition 3.4 *There exist function classes $\mathcal{F}$ such that:*

- *$\mathcal{F}$ has a property testing algorithm whose sample complexity and running time are $O(\epsilon^{-1} \cdot \log(1/\delta))$ (i.e., independent of n);*
- *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 $m = O(\epsilon^{-1} \log(1/\delta))$ 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 3.4 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 that is easy to test but cannot be learned in polynomial-time (even under the uniform distribution), provided certain trapdoor one-way permutations exist (*e.g.,* factoring is intractable). We start by defining this assumption, which in some sense is weaker than the standard one (cf., [Gol95, Sec. 2.4]). (We comment that the standard term “trapdoor permutation” is somewhat misleading since what is being defined is a collection of permutations.)

Definition 3.5 (weak trapdoor permutations with dense domains): *Let $\{p_\alpha : \mathrm{D}_\alpha \mapsto \mathrm{D}_\alpha\}_{\alpha\in\{0,1\}^*}$ be a family of permutations. We say that this family has* dense domains *if for some positive polynomial $q(\cdot)$ and all α's, both $\mathrm{D}_\alpha \subseteq \{0,1\}^{|\alpha|}$ and $|\mathrm{D}_\alpha| \geq \frac{2^{|\alpha|}}{q(|\alpha|)}$ hold. The family is* one-way *if it satisfies*

1. (easy to evaluate) *There exists a polynomial-time algorithm which given α and x outputs $p_\alpha(x)$.*
2. (hard to invert) *No probabilistic polynomial-time algorithm can, given uniformly chosen α and x, output $p_\alpha^{-1}(x)$ with, say, success probability at least $1/n$ (where the probability is taken uniformly over the coin tosses of the algorithm and all possible choices of $\alpha \in \{0,1\}^n$ and $x \in \mathrm{D}_\alpha$).*

The family is said to have weak trapdoors *if for each $\alpha \in \{0,1\}^*$ (equivalently, for each p_α) there exists a* poly$(|\alpha|)$*-size circuit which inverts p_α.*

The weak trapdoor condition is a relaxation of the standard condition which requires also that one can efficiently generate (α,trapdoor)-pairs. The dense domain condition is indeed non-standard, but does hold for all popular candidates (e.g., RSA and Rabin functions). See discussion in [CFGN96]. Another non-standard aspect of the definition is associating a permutation with each string, whereas the standard definition associates permutations only with a subset of all strings. Again, we note that in case of the popular candidates, permutations are associated with a polynomial fraction of all strings of certain length, and so we can modify these constructions so that a permutation is associated with each string.[10]

Proposition 3.6 *If there exist weak trapdoor one-way permutations with dense domains then there exists a family of functions that can be tested in* poly(n/ϵ)*-time but can not be learned in* poly(n/ϵ)*-time, even with respect to the uniform distribution. Furthermore, the functions can be computed by* poly(n)*-size circuits.*

Proof: By [CFGN96], any collection of trapdoor one-way permutations with dense domains can be converted into a collection trapdoor one-way permutations where the domain of each p_α is $\{0,1\}^{|\alpha|}$. Thus, we use such a collection

$$\{p_\alpha : \{0,1\}^{|\alpha|} \mapsto \{0,1\}^{|\alpha|}\}_{\alpha\in\{0,1\}^*}$$

We consider the function class $\mathcal{OW} = \{\mathcal{OW}_n\}$, where $\mathcal{OW}_n$ consists of the multi-valued functions f_α, so that $f_\alpha(x) \stackrel{\rm def}{=} (\alpha, p_\alpha^{-1}(x))$ for every $\alpha, x \in \{0,1\}^n$. Using the weak trapdoor condition, we know that the functions in $\mathcal{OW}_n$ can be computed by poly(n)-size circuits.

To test if $f \in \mathcal{OW}$, we merely examine sufficiently many f-labeled examples. Specifically, $m \stackrel{\rm def}{=} O(\epsilon^{-1}\log(1/\delta))$ examples will do. 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.

We show that this test works for any distribution on the examples and so the class $\mathcal{OW}$ is efficiently testable (in a distribution-free sense). Clearly, the test always accepts $f_\alpha \in \mathcal{OW}_n$. Assuming that f is ϵ-far from $\mathcal{OW}_n$, with respect to some distribution D, we wish to show that f is rejected by the test with high probability. We consider two cases.

[10] One way of doing so proceeds in two steps, and is analogous to the [CFGN96] transformation mentioned subsequently. First one introduces dummy permutations for each string that is not associated with a permutation in the original construction. This may weaken the one-way property but still some "one-wayness" remains. Next, one amplifies the one-wayness by taking a "direct product" – that is, for each $\alpha_1, ..., \alpha_t$ and $x_1, ..., x_t$, we define a new function indexed by the α-sequence that consists of the concatenation of the values of each function associated with an α_i evaluated at the corresponding x_i.

Case 1 Suppose there exists an α such that the probability that the first element of $f(x)$ equals α is at least $1-\frac{\epsilon}{2}$, where the probability is taken over $x \sim \mathrm{D}$. Since f is ϵ-far from $\mathcal{OW}_n$, we have in particular

$$\Pr_{x\sim \mathrm{D}_n}[f(x) = f_\alpha(x)] \; < \; 1-\epsilon$$

and so

$$\begin{aligned} q & \stackrel{\text{def}}{=} \Pr_{x\sim \mathrm{D}_n}[x \neq p_\alpha(y) \text{ where } f(x) = (\alpha, y)] \\ &= \Pr_{x\sim \mathrm{D}_n}[f(x) = (\alpha, y) \text{ for some } y] - \Pr_{x\sim \mathrm{D}_n}[f(x) = f_\alpha(x)] \\ &> \left(1-\frac{\epsilon}{2}\right) - (1-\epsilon) \; = \; \frac{\epsilon}{2} \end{aligned}$$

Hence, the test accepts with probability at most $(1-q)^m < \delta$.

Case 2 Suppose that for every α the probability that the first element of $f(x)$ equals α is at most $1-\frac{\epsilon}{2}$. Then, the probability that all m examples have a label starting with the same α is at most

$$\sum_\alpha \Pr_{x\sim \mathrm{D}_n}[\exists y \text{ s.t. } f(x) = (\alpha, y)]^m$$

which is bounded above by $(1-\frac{\epsilon}{2})^{m-1} < \delta$.

Thus, in both cases, the test rejects ϵ-far functions with sufficiently high probability. We now turn to show that it is infeasible to learn the class $\mathcal{OW}$ under the uniform distribution. This is done by using any efficient learning algorithm, $\mathcal{A}$, in order to construct an algorithm that contradicts the one-way condition. The inverting algorithm operates as follows, on input $\alpha, p_\alpha(x)$. First it uniformly generates an f_α-labeled sample $\{(x_i, (\alpha, y_i))\}$ for $\mathcal{A}$. This is done by uniformly selecting y_i and setting $x_i = p_\alpha(y_i)$. (Here we use the efficient evaluation property of the collection.) Note that indeed $f_\alpha(x_i) = (\alpha, p_\alpha^{-1}(x_i)) = (\alpha, y_i)$, and that x_i is uniformly distributed. When the learning stage of $\mathcal{A}$ is over, the inverter supplies $\mathcal{A}$ with x, asking for its label. With probability at least $(1-\delta)\cdot(1-\epsilon) > 1-\delta-\epsilon$, taken uniformly over all possible $x \in \{0,1\}^{|\alpha|}$ and the internal coin tosses of both the inverter and $\mathcal{A}$, algorithm $\mathcal{A}$ returns the correct label; that is, $(\alpha, p_\alpha^{-1}(x)$. Thus, an efficient learning algorithm is transformed into and efficient inverting algorithm for the family, in contradiction to the one-wayness condition.[11] ■

The class presented in Proposition 3.6 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 $\mathcal{PAR}_n \stackrel{\text{def}}{=} \{f_\mathrm{S} : \mathrm{S} \subseteq \{1,\ldots,n\}\}$ and $f_\mathrm{S} : \{0,1\}^n \mapsto \{0,1\}$ so that $f_\mathrm{S}(x) = \sum_{i\in \mathrm{S}} x_i \bmod 2$. 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 $O(\epsilon^{-1}\log(1/\delta))$. 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.

[11] Actually, the inverter is stronger than what is required to contradict one-wayness: It can invert any p_α on all but a small fraction of the range elements.

An example of a poly(n/ϵ)–time testable (with queries) class which is not learnable with poly(n/ϵ) queries is the class of multi-variate (i.e., n–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 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. However, we were not able to translate this similarity in spirit to anything stronger than the relation between testing and proper learning. Namely, since agnostic learning with respect to a hypothesis class $\mathcal{H}$ implies proper learning of the class $\mathcal{H}$, it also implies 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 $\mathrm{D} \in \mathcal{D}_n$. 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 $\mathrm{D}'(x)$), 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 $\mathrm{D} \in \mathcal{D}_n$, 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 another 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 3.7 *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 $\mathrm{D}_n^{1/2}$ (and thus should be accepted), and the case in which such a sample is taken from a 'typically bad' distribution $\mathrm{B}_n^{\mathrm{S}}$ which is uniform over $S \subset \{0,1\}^n$, where $|\mathrm{S}| = 2^{n-1}$. 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}. ∎

We note that the above proof holds for any distribution class that contains the uniform distribution and is far from distributions such as the B_n^S's.

4 General Observations

PROPERTY TESTING MAY BE VERY HARD.

Proposition 4.1 *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 constant distance parameter $\epsilon < 1/2$ and confidence parameter $\delta < 1/2$, and even when allowing the algorithm to make queries and use unlimited computing time.*

Proof: Suppose for simplicity that $\epsilon = 1/4$ and $\delta = 1/5$. The proof easily generalizes to general constant $\epsilon, \delta < 1/2$ by appropriately modifying the size of $\mathcal{F}_n$ (which is defined below). 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,

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

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 (a random) 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 (on the inputs in S) induced by a uniformly selected function in $\mathcal{F}_n$. That is,

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

where for a sequence $S = x_1, \ldots, x_T$, $f(S) = f(x_1)\cdots f(x_t)$. We consider the probability, taken over all possible choices of $\mathcal{F}_n$ (consisting of $2^{\frac{1}{10}\cdot 2^n}$ functions), that $\delta_S(\mathcal{F}_n) > 1/2$. By using a multiplicative Chernoff bound (with multiplicative factors 1/2 and 3/2 – see Appendix B), we get

$$\begin{aligned}
&\Pr_{\mathcal{F}_n}[\delta_S(\mathcal{F}_n) > 1/2] \\
&\quad \leq \Pr_{\mathcal{F}_n}[\exists \alpha \in \{0,1\}^T \text{ s.t. } |\Pr_{f\in\mathcal{F}_n}(f(S) = \alpha) - 2^{-T}| > 2^{-(T+1)}]
\end{aligned}$$

$$
\begin{aligned}
&\leq \; 2^T \cdot 2\exp\left(-\frac{1}{3}\cdot(\frac{1}{2})^2 \cdot 2^{-T}|\mathcal{F}_n|\right) \\
&= \; 2^{N/20} \cdot 2\exp\left(-\frac{1}{12}\cdot 2^{N/20}\right) \\
&= \; \exp\left(-2^{\Omega(N)}\right)
\end{aligned}
$$

Summing the probabilities over all possible $\binom{N}{T} < 2^N$ 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 one to both accept every $f \in \mathcal{F}_n$ with probability at least 0.8 and accept a random g with probability at most 0.21 (and so to reject ϵ-far functions with probability at least 0.8 – 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 infer about their INTERSECTION, UNION and COMPLEMENT? Unfortunately, in general, we can only say that their union is testable within comparable complexity. That is,

Proposition 4.2 *Let $\mathcal{F}' = \{\mathcal{F}'_n\}$ and $\mathcal{F}'' = \{\mathcal{F}''_n\}$ be function classes testable within complexities $c'(n,\epsilon,\delta)$ and $c''(n,\epsilon,\delta)$, respectively. Then, the class $\mathcal{F} = \{\mathcal{F}_n\}$, where $\mathcal{F}_n = \mathcal{F}'_n \cup \mathcal{F}''_n$, is testable within complexity $c(n,\epsilon,\delta) = c'(n,\epsilon,\frac{\delta}{2}) + c''(n,\epsilon,\frac{\delta}{2})$.*

Proof: The testing algorithm for $\mathcal{F}$ consists of testing for membership in both $\mathcal{F}'$ and $\mathcal{F}''$ and accepting if and only if 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 a claim analogous to the one used in the above proof 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 $\mathcal{F} = \mathcal{F}' \cap \mathcal{F}''$ and yet it is very close to both $\mathcal{F}'$ and $\mathcal{F}''$. Thus, a function (close to both $\mathcal{F}'$ and $\mathcal{F}''$), may pass both the corresponding property tests, but still may be far from $\mathcal{F}$. In particular

Proposition 4.3 *There exist function classes $\mathcal{F}' = \{\mathcal{F}'_n\}$ and $\mathcal{F}'' = \{\mathcal{F}''_n\}$ such that both are trivially testable under the uniform distribution (i.e., by an oblivious algorithm that always accepts provided $\epsilon > 2^{-n}$), whereas the class $\mathcal{F} = \{\mathcal{F}_n\}$, where $\mathcal{F}_n = \mathcal{F}'_n \cap \mathcal{F}''_n$ is not testable under the uniform distribution with query complexity $o(2^n)$ even for constant $\epsilon, \delta < \frac{1}{2}$.*

Proof: Let $\mathcal{F} = \{\mathcal{F}_n\}$ be as guaranteed in Proposition 4.1, and let $\mathcal{F}'_n \stackrel{\rm def}{=} \mathcal{F}_n \cup \{f : f(0^n) = 0\}$, $\mathcal{F}''_n \stackrel{\rm def}{=} \mathcal{F}_n \cup \{f : f(0^n) = 1\}$. Clearly both $\mathcal{F}'$ and $\mathcal{F}''$ are testable as claimed since every function is 2^{-n}-close to both $\mathcal{F}'$ and $\mathcal{F}''$. On the other hand, $\mathcal{F}'_n \cap \mathcal{F}''_n = \mathcal{F}_n$, and so the negative result of Proposition 4.1 applies. ■

Finally, we observe that property testing is not preserved under complementation. That is,

Proposition 4.4 *There exists a function class $\mathcal{G} = \{\mathcal{G}_n\}$ such that for every $\epsilon \geq \frac{1}{2^n-1}$ the algorithm that accepts every function is a tester for $\mathcal{G}$, while the class of functions not in $\mathcal{G}$ is not testable in subexponential complexity.*

Proof: Consider the function class $\mathcal{F}$ used in the proof of Proposition 4.1. As shown there, this class is not testable in subexponential complexity. Furthermore, as we show subsequently, $\mathcal{F}$ can be chosen so that for every n and for every function $f \in \mathcal{F}_n$, there exists at most one function $f' \in \mathcal{F}_n$, such that f and f' differ on exactly one input. In other words, there exist at least $2^n - 1$ functions *not in* $\mathcal{F}_n$ that differ from f on exactly one input. Let $\mathcal{G} = \{\mathcal{G}_n\}$ consists of all functions not in $\mathcal{F}$. Based on our additional claim (concerning differences between functions in $\mathcal{F}_n$), as long as $\epsilon \geq 1/(2^n - 1)$, the trivial algorithm which accepts all functions constitutes a tester for $\mathcal{G}$. This is true since for every $f \notin \mathcal{G}_n$ (that is, $f \in \mathcal{F}_n$), and for every distribution, there exists some function $g \in \mathcal{G}_n$ that is at distance less than $\frac{1}{2^n-1}$ from f. Namely, the total probability mass of the x's for which there exists $g \in \mathcal{G}_n$ such that g and f differ only on x is at most 1, and there are at least $2^n - 1$ such x's.

To prove the claim regarding differences between functions in $\mathcal{F}$, we again use the probabilistic method. Observe first that the probability that a particular function f belongs to $\mathcal{F}_n$, is $\frac{2^{N/10}}{2^N} = 2^{-0.9 \cdot N}$, where $N \stackrel{\text{def}}{=} 2^n$. Thus, for any fixed function f, the probability that there exist two functions f_1 and f_2 in $\mathcal{F}_n$ such that both f_1 and f_2 differ from f on exactly one input is bounded by $N^2 \cdot (2^{-0.9 \cdot N})^2 < 2^{-1.7 \cdot N}$. Finally the probability that this event occurs for some f is at most $2^N \cdot 2^{-1.7 \cdot N} = 2^{-0.7 \cdot N}$, which is extremely small. By adding the negligible probability (bounded in Proposition 4.1) that the uniformly chosen $\mathcal{F}_n$ is not hard to test, Proposition 4.4 follows. ■

Part II
Testing Graph Properties

5 Testing Graph Properties – Preliminaries

In the following sections we concentrate on testing graph properties using queries and with respect to the uniform distribution. In Section 10.1, we discuss some extensions beyond this basic model. We start by defining the basic model and giving an overview of our results.

5.1 General Graph Notation.

We consider undirected, simple graphs (no multiple edges or self-loops). For a simple graph G, we denote by 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 $(u, v) \in \mathrm{V(G)} \times \mathrm{V(G)}$ 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. Specifically, for a graph G, we denote by 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, and the degree of a vertex is the total number of edges incident to it (i.e., the sum of its in-degree and out-degree). For two (not necessarily disjoint) sets of vertices, X_1 and X_2, we let

$$\mathrm{E}(\mathrm{X}_1, \mathrm{X}_2) \stackrel{\text{def}}{=} \{(u, v) \in \mathrm{E(G)} : \ u \in \mathrm{X}_1, v \in \mathrm{X}_2 \text{ or } u \in \mathrm{X}_2, v \in \mathrm{X}_1\}$$

The distance between two N-vertex graphs, G_1 and G_2, is defined as the fraction of entries $(u, v) \in [N]^2$ ($[N] \stackrel{\text{def}}{=} \{1, ..., N\}$), among all N^2 entries, which are in the symmetric difference of $\mathrm{E}(\mathrm{G}_1)$ and $\mathrm{E}(\mathrm{G}_2)$. Namely,

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

PROPERTY	TESTING		PARTITIONING
	Query	Time	Time (on top of testing)
Bipartiteness	$O\left(\frac{\Delta^2}{\epsilon^3}\right)$	$O\left(\frac{\Delta^2}{\epsilon^3}\right)$	$O(\frac{\Delta}{\epsilon}) \cdot N$
k-Colorability	$O\left(\frac{k^4 \cdot \Delta_k^2}{\epsilon^6}\right)$	$\exp\left(O\left(\frac{k^2 \cdot \Delta_k^2}{\epsilon^3}\right)\right)$	$O(\frac{\Delta_k}{\epsilon^2}) \cdot N$
ρ-Clique	$O\left(\frac{\Delta^2 \cdot \rho^2}{\epsilon^6}\right)$	$\exp\left(O\left(\frac{\Delta \cdot \rho}{\epsilon^2}\right)\right)$	$O\left(\frac{\Delta^2}{\epsilon^2}\right) \cdot N$
ρ-Cut	$O\left(\frac{\Delta^2}{\epsilon^7}\right)$	$\exp(O(\frac{\Delta}{\epsilon^3}))$	$O(\frac{\Delta}{\epsilon^2}) \cdot N$
ρ-k-Cut	$O\left(\frac{\Delta_k^2}{\epsilon^7}\right)$	$\exp\left(O\left(\frac{\Delta_k^2}{\epsilon^3}\right)\right)$	$O(\frac{\Delta_k}{\epsilon^2}) \cdot N$
ρ-Bisection	$O\left(\frac{\Delta^2}{\epsilon^8}\right)$	$\exp(O(\frac{\Delta}{\epsilon^3}))$	$O(\frac{\Delta}{\epsilon^2}) \cdot N$
General k-Partition	$\Delta^2 \cdot \left(\frac{O(k^2)}{\epsilon}\right)^{2k+8}$	$\exp\left(\Delta \cdot \left(\frac{O(k^2)}{\epsilon}\right)^{k+1}\right)$	$O\left(\frac{\Delta_k}{\epsilon^2}\right) \cdot N$

Figure 1: Summary of Results. Here Δ and Δ_k denote logarithmic factors. Specifically, $\Delta \stackrel{\text{def}}{=} \log(1/(\epsilon\delta))$ and $\Delta_k \stackrel{\text{def}}{=} \log(k/(\epsilon\delta))$. For simplicity we use Δ_k also in cases where the dependency is slightly better (e.g. in k-Colorability). We also note that in some of the cases and for certain values of δ, one can obtain slightly better complexities (in terms of the dependence on δ). This is done by considering the complexity for constant δ, and then amplifying the confidence by repeating the execution of the algorithm $\log(1/\delta)$ times.

This notation is extended naturally to a set, $\mathcal{C}$, of N-vertex graphs; that is,

$$\text{dist}(\mathrm{G}, \mathcal{C}) \stackrel{\text{def}}{=} \min_{\mathrm{G}' \in \mathcal{C}} \{\text{dist}(\mathrm{G}, \mathrm{G}')\}$$

Another notation used extensively in subsequent sections is the set of neighbors of a vertex v; that is, $\Gamma(v) \stackrel{\text{def}}{=} \{u : (v, u) \in \mathrm{E}(\mathrm{G})\}$. This notation is extended to sets of vertices in the natural manner; i.e., $\Gamma(\mathrm{S}) \stackrel{\text{def}}{=} \cup_{v \in \mathrm{S}} \Gamma(v)$.

5.2 Our Algorithms and their Analyses.

We present testers for Bipartiteness, k-Colorability (for $k \geq 3$), ρ-Clique, ρ-Cut (and ρ-k-Cut), ρ-Bisection and the General Graph Partition property. The latter generalizes all the former ones, but yields worse complexity bounds for the special cases. Also, the testers for ρ-Cut and ρ-k-Cut, which actually work by approximating the size of the maximum cut, generalize Bipartiteness and k-Colorability, respectively. However, the former have two-sided error probability and higher complexity, whereas the latter can be directly tested with one-sided error probability.

For all properties we also present linear (in N) time algorithms that, given a graph that has the property, find a partition that is approximately-good with respect to that property (e.g., in the case of ρ-Clique, the algorithm finds a set of size ρN that misses few edges to being a clique). While these algorithm are clearly beneficial when the corresponding approximation problems are know to be hard, they are also meaningful in the case of Bipartiteness (where an exact efficient algorithm is known). This is true since in order to partition a dense bipartite graph, one has to work in time quadratic in N (i.e., linear in the number of edges), while we offer an approximation algorithm that works in time linear in N. The complexities of our algorithms are summarized in Figure 1.

COMMON THEMES. As mentioned in the introduction, our algorithms and their analyses share a few themes. In particular, all algorithms uniformly select a set of vertices and perform edge-queries only on pairs of vertices in the sample. In all cases the *natural* algorithm for the property, which checks whether the property (approximately) holds on the subgraph induced by the sample of vertices, is indeed a testing algorithm for the property. However, only in the case of Bipartiteness and k-Colorability do we analyze directly the natural algorithm. For the other properties, we present and analyze more complex algorithms. However the correctness of the natural algorithms follows from the correctness of the algorithms that we present, and the resulting complexities of the former are only slightly higher than those of the latter.

For all our testers, the sample of vertices is viewed as consisting of two parts – one part, denoted U, has the role of inducing partitions on V(G) (i.e., all graph vertices), and the other part, denoted S, evaluates these partitions. In the Bipartiteness and k-Colorability testers this view of the sample is only taken in the analysis. In the other algorithms we explicitly use these two parts of the sample differently. In particular, the latter algorithms consider all possible partitions of U. For each such partition of U, the second part of the sample, S, is partitioned according to a property-specific rule that uses the neighborhood relations between vertices in S and vertices in U.[12]

In all cases we want U to be *representative* of certain *good* partitions of V(G), which exist whenever G has the property that is being tested. The notion of a good partition depends on the property being tested, and the sense in which U is representative varies accordingly. For example, when testing ρ-Clique, a good partition of a graph G is $(\mathrm{C}, \mathrm{V(G)} \setminus \mathrm{C})$, where C is a clique of size ρN. In this case U is representative if for *almost all* vertices v in the graph, if v neighbors all vertices in $\mathrm{U} \cap \mathrm{C}$, then it neighbors almost all neighbors in C. Note that we do not require that the above hold for all vertices v, and this feature is common to all our definitions of a representative sample U. Consequently, U need be only of size $\mathrm{poly}(\epsilon^{-1}\log(1/\delta))$, and so independent of N. This is especially important since most of our algorithms run in time exponential in $|\mathrm{U}|$.

Specific Themes. Though the algorithms and their analyses have much in common, there are several themes that are only shared by some of the algorithms. Consider first the Bipartite and the k-Colorability testers. As noted above, for both properties we directly analyze the natural algorithms, which uniformly select a small sample of vertices and check whether the induced subgraph is Bipartite (respectively, k-Colorable). These two algorithms have one-sided error as they always accept graphs that have the property. To show that with high probability the algorithms reject graphs that are far from having the property, we prove the counterpositive. Namely, that if a graph is accepted with probability greater than δ then it is close to having the property. We do so by showing that the partition of the *sampled* vertices found by the algorithm when accepting the graph can be used to define a good partition of all graph vertices (where this definition is constructive.)

For all other properties we first describe an algorithm that actually partitions V(G). This graph-partitioning algorithm produces several (i.e., $\exp\left(\mathrm{poly}\left(\frac{\log(1/\delta)}{\epsilon}\right)\right)$) partitions of V(G) and outputs the best one. The corresponding tester runs the graph-partitioning algorithm on a small sample S and evaluates the resulting partitions of S. Thus S serves as a sample which approximates the quality of the graph-partitions (determined by the corresponding graph-partitioning algorithm). Let us hence focus in this introductory discussion on the graph-partitioning algorithms.

In the case of ρ-Clique, the graph-partitioning algorithm tries to find an approximate ρ-clique (i.e., a set of vertices of size ρN that is close to being a clique). To this end it selects a small sample of vertices U, and for each subset U′ of U having size $\frac{\rho}{2} \cdot |\mathrm{U}|$, it considers all vertices that neighbor every vertex in U′. Assume first that G in fact has a clique C of size ρN. The idea is that in such a case, with high probability over the choice of U, the intersection of U and C will be of size at

[12]In the ρ-Clique tester we slightly deviate from the above formula by using an additional sample of vertices that aids in partitioning S.

least $\frac{\ell}{2} \cdot |\mathrm{U}|$. Therefore, for some subset U' of U, the set of vertices, T, that neighbor every vertex in U' contains the clique C. However, T might contain many other vertices. Nonetheless, we can show that if we order the vertices in T roughly according to their degree in the subgraph induced by T, then the first ρN vertices in this order will be close to being a clique. The above implies that if G has a clique of size ρN then with high probability the algorithm finds an approximate clique. On the other hand, if every subset of size ρN in G is far from having a clique then no such approximate-clique exists.

The graph-partitioning algorithms for ρ-Cut, ρ-Bisection and the General Partition property work in $\ell = O(1/\epsilon)$ stages, and consider a fixed (standard) equal-partition $\mathrm{V}^1, \ldots, \mathrm{V}^\ell$ of V(G).[13] In the i^{th} stage, the set V^i is partitioned. The partition of V^i is determined by a partition of U^i (which is part of the sample U), and possibly other varying parameters. For example, in the case of ρ-Cut (where the partitioning algorithm actually finds a close-to-*maximum* cut) the vertices in V^i are partitioned as follows: The vertices that have more neighbors on side 1 of the partition of U^i are put on side 2 of the partition of V^i, and vica-versa. The fact that the algorithms partition only a set of size $O(\epsilon N)$ in each stage (using a different sample U^i) is essential to the proofs of correctness of the algorithms.

5.3 Two Technical Conventions

As noted above, we represent N-vertex graphs by their $N \times N$ adjacency matrix. This representation is redundant and means that each edge appears twice. In some places this convention simplifies the analysis, but in others it results in the need to artificially double certain natural quantities. Still we believe we made the right choice.

Our algorithms are presented as uniformly selecting sets of vertices of certain sizes (without repetitions). However, the analysis refers to uniform and independent selection of a number of vertices (with possible repetitions). Formally-inclined readers are thus encouraged to consider the algorithms as selecting multisets (rather than sets) of the prescribed size. Other readers may just ignore this point.

6 Testing Bipartiteness and Vertex–Colorability

In Subsection 6.1 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 Subsection 6.2. 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 general case. Moreover, the algorithm presented for$k = 2$ has lower complexity (in terms of its dependence on the distance parameter, ϵ) than the one described in the Subsection 6.2.

6.1 Testing Bipartiteness

We start by describing a testing algorithm whose query complexity is $O\left(\frac{\log^2(1/(\epsilon\delta))}{\epsilon^4}\right)$. We later point out how this algorithm can be slightly modified so that its query complexity decreases to $O\left(\frac{\log^2(1/(\epsilon\delta))}{\epsilon^3}\right)$.

Bipartite Testing Algorithm

[13] In particular, this partition is defined according to lexicographical order (i.e. where each vertex is represented by a string of length $\log_2 N$). We stress that this partition is arbitrary and has nothing to do with the desired partition determined by the property being tested.

1. Choose uniformly a set, denoted X, of $O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^2}\right)$ vertices.
2. For every pair of vertices $v_1, v_2 \in \mathrm{X}$, query if $(v_1, v_2) \in \mathrm{E(G)}$. 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 subsection, we introduce the following definitions. Recall that N is the number of vertices of G.

Definition 6.1 (violating edges and good partitions): *We say that an edge* $(u, v) \in \mathrm{E(G)}$ *is a* violating *edge with respect to a partition* $(\mathrm{V_1}, \mathrm{V_2})$ *of* $\mathrm{V(G)}$ *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 6.2 *The* Bipartite Testing Algorithm *is a property testing algorithm for the class of bipartite graphs whose query complexity and running time are* $O\left(\frac{\log^2(1/(\epsilon\delta))}{\epsilon^4}\right)$*. 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 (additional) time* $O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon} \cdot N\right)$*,*

Proof: 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 $\mathrm{G} \notin \mathcal{B}$. 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 will 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 $(\mathrm{V_1}, \mathrm{V_2})$ of V(G), defined by a partition $(\mathrm{U_1}, \mathrm{U_2})$ of U, is ϵ-bad, then with very high probability there is *no* partition $(\mathrm{S_1}, \mathrm{S_2})$ of S such that $(\mathrm{U_1} \cup \mathrm{S_1}, \mathrm{U_2} \cup \mathrm{S_2})$ is a perfect partition of X. We next make the above notions more precise.

Given a partition $(\mathrm{U_1}, \mathrm{U_2})$ of U we define the following partition $(\mathrm{V_1}, \mathrm{V_2})$ of V(G): let $\mathrm{V_1} \stackrel{\mathrm{def}}{=} \Gamma(\mathrm{U_2})$, and $\mathrm{V_2} \stackrel{\mathrm{def}}{=} \mathrm{V(G)} \setminus \Gamma(\mathrm{U_2})$. That is, $\mathrm{V_1}$ is the set of neighbors of $\mathrm{U_2}$, and $\mathrm{V_2}$ contains all neighbors of $\mathrm{U_1}$ (which are not neighbors of $\mathrm{U_2}$), 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 actually most "influential" vertices in V(G)) have a neighbor in U.

Definition 6.3 (influential vertices and covering sets): *We say that a vertex $v \in \mathrm{V(G)}$ 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).*

Note that in the above definition, we did not require of U that *every* influential vertex have a neighbor in U, but rather that this hold for almost all influential vertices. This slackness (which appears in various form in our other algorithms as well) is what allows our algorithm to use a sample size that is independent of the size of the graph.

Claim 6.4 *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: Let $t = \left(\frac{6 \cdot \log(12/(\delta\epsilon))}{\epsilon}\right)$. 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

$$\left(1 - \frac{\epsilon}{6}\right)^t \;\le\; \exp\left(-t \cdot \frac{\epsilon}{6}\right) \;=\; \frac{\epsilon \cdot \delta}{12}\,. \tag{2}$$

Hence, the expected number of influential vertices that do not have a neighbor in a random set U is $\frac{\delta \cdot \epsilon}{12} \cdot N$, and by Markov's inequality (see Appendix B), 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 $\Gamma(\mathrm{U}_i)$, for $i = 1, 2$. The reason being that the total number of edges incident to vertices not in $\Gamma(\mathrm{U})$ is small. This motivates the following definition, where a useful partition of U implies that G has a good partition (see Lemma 6.8).

Definition 6.5 (useful partitions): *Let $\mathrm{U} \subset \mathrm{V(G)}$. A partition $(\mathrm{U}_1, \mathrm{U}_2)$ of* U *is called* ϵ-useful *(or just* useful*) if*

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

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 $(\mathrm{U}_1, \mathrm{U}_2)$ 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 $(v, v') \in \mathrm{S} \times \mathrm{S}$ between neighbors of vertices in, say, U_1. Let $u \in \mathrm{U}_1$ (resp., $u' \in \mathrm{U}_1$) be a neighbor of v (resp., v'). In case $u = u'$ there is a triangle in $\mathrm{G_X}$ (which means that the test would reject). In case $u \neq u'$, due to the edge (v, v'), in any perfect partition of $\mathrm{X} = \mathrm{U} \cup \mathrm{S}$, u and u' must belong to different sides of the partition (and so $(\mathrm{U}_1, \mathrm{U}_2)$ cannot be used in the partition of $\mathrm{X} = \mathrm{U} \cup \mathrm{S}$).

Claim 6.6 *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 $m = O\left(\frac{t + \log(1/\delta)}{\epsilon}\right)$,*

$$\mathrm{Pr_S}\left[\forall v, v' \in \mathrm{S},\; i \in \{1,2\} \;:\; (v, v') \notin \mathrm{E}(\Gamma(\mathrm{U}_i), \Gamma(\mathrm{U}_i))\right] \;<\; \delta \cdot 2^{-(t+1)}\,.$$

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 $\mathrm{X} = \mathrm{U} \cup \mathrm{S}$ (respecting $(\mathrm{U}_1, \mathrm{U}_2)$) is a perfect partition of the subgraph induced by X.

Proof: If $(\mathrm{U}_1, \mathrm{U}_2)$ is an ϵ-unuseful partition, then by Eq. (3):

$$\mathrm{Pr}_{v,v'}[\exists i \in \{1,2\} \text{ s.t. } (v,v') \in \mathrm{E}(\Gamma(\mathrm{U}_i), \Gamma(\mathrm{U}_i))] \geq \frac{\epsilon}{3}\,. \tag{4}$$

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

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

■

Since there are 2^t possible partitions of U, using Claim 6.6, we have

Corollary 6.7 *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* $\mathrm{X} = \mathrm{U} \cup \mathrm{S}$, *where* S *is chosen uniformly. (In such a case* 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 this partition induces a good partition of the entire graph. More precisely,

Lemma 6.8 *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:* $\mathrm{V}_1 \stackrel{\mathrm{def}}{=} \Gamma(\mathrm{U}_2)$, $\mathrm{V}_2 \stackrel{\mathrm{def}}{=} \mathrm{V}(\mathrm{G}) \setminus \Gamma(\mathrm{U}_2)$.

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: Since U is a covering set, 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 $(v, v') \in \mathrm{E}(\mathrm{V}_1, \mathrm{V}_1)$. Since $\mathrm{V}_1 = \Gamma(\mathrm{U}_2)$, these edge have both end-points in $\Gamma(\mathrm{U}_2)$.
  - Edges of the form $(v, v') \in \mathrm{E}(\mathrm{V}_2, \mathrm{V}_2)$. By definition of V_2, both v and v' are not in $\Gamma(\mathrm{U}_2)$. However, since $v, v' \in \Gamma(\mathrm{U})$ 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 6.8 with Claim 6.4 and Corollary 6.7, we complete the proof of Theorem 6.2 as follows. If G is accepted with probability greater than δ, then, by Claim 6.4, the probability that G is accepted *and* U is a covering set is greater than $\delta/2$. Thus, there exists a covering set $\mathrm{U} \in \mathrm{V(G)}$, 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 6.7 that (the covering set) U must have a useful partition, and we can apply Lemma 6.8 to show that G must be ϵ-close to bipartite.

Finally, let G be a bipartite graph, and let $(\mathrm{V}_1, \mathrm{V}_2)$ 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 $(\mathrm{U}_1, \mathrm{U}_2)$ of U, such that $\mathrm{U}_1 \subseteq \mathrm{V}_1$ and $\mathrm{U}_2 \subseteq \mathrm{V}_2$. Thus, necessarily, $(\mathrm{U}_1, \mathrm{U}_2)$ is a useful partition of U which by Lemma 6.8 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 $\mathrm{U} \cup \mathrm{S}$, of the form $(\mathrm{U}_1 \cup \mathrm{S}_1, \mathrm{U}_2 \cup \mathrm{S}_2)$. On the other hand, by Claim 6.6, for any set U, with probability at least $1 - \delta/2$ over the choice of S, there will be no perfect partition of $\mathrm{U} \cup \mathrm{S}$ 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 $\mathrm{U} \cup \mathrm{S}$ 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 6.8) in time $O(|\mathrm{U}| \cdot N) = O(\epsilon^{-1}\log(1/(\epsilon\delta))) \cdot N$.
■ (THEOREM 6.2) ■

IMPROVING THE QUERY COMPLEXITY. We can save a factor of $1/\epsilon$ in the query complexity and in the running time of the testing algorithm. This is done simply by observing that we do not need to perform 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 $\frac{m}{2} = O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^2}\right)$ pairs are edges, as well as query all $m \cdot t = O\left(\frac{\log^2(1/(\epsilon\delta))}{\epsilon^3}\right)$ pairs (u, v) where $u \in \mathrm{U}$ and $v \in \mathrm{S}$. Note that the proof of Theorem 6.2 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 6.6).

IMPOSSIBILITY OF TESTING WITHOUT QUERIES. A natural question that may arise is whether 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 (of pairs of vertices) must have very large sample complexity. More precisely:

Proposition 6.9 *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, $\mathrm{V(G)} = \mathrm{V}_1 \dot\cup \mathrm{V}_2$, $|\mathrm{V}_1| = |\mathrm{V}_2| = N/2$, and $\mathrm{E(G)} = \{(v_1, v_2) : v_1 \in \mathrm{V}_i, v_2 \in V_j,\ i \neq j\}$. $\mathcal{G}^2$ is the class of graphs which consist of two disjoint cliques of size $N/2$. That is, $\mathrm{V(G)} = \mathrm{V}_1 \dot\cup \mathrm{V}_2$, $|\mathrm{V}_1| = |\mathrm{V}_2| = N/2$, and $\mathrm{E(G)} = \{(v, v') : v, v' \in V_1 \text{ or } v, v' \in V_2\}$. Clearly, all graphs in $\mathcal{G}^2$ are $\frac{1}{4}$-far from bipartite. Note that all graphs in both classes have the same edge density, since every vertex has degree N. Intuitively, we want to show 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 greater than $\sqrt{N/2}$. ■

6.2 Testing k-Colorability ($k > 2$)

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 subsection: 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 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 (improved) Bipartite Tester (i.e., by a factor of $\widetilde{O}(k^4/\epsilon^3)$).

k-Colorability Testing Algorithm

1. Choose uniformly set of $O\left(\frac{k^2(\log(k/\delta)}{\epsilon^3}\right)$ vertices, denoted 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.[14]

Definition 6.10 (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 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.*

[14] 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 $(V_1 \stackrel{\text{def}}{=} \pi^{-1}(1), \ldots, V_k \stackrel{\text{def}}{=} \pi^{-1}(k))$. We shall use the mapping notation π, and the explicit partition notation $(V_1, \ldots, V_k)$, interchangeably.

Theorem 6.11 *The* k-Colorability Testing Algorithm *is a property testing algorithm for the class of k-Colorable graphs whose query complexity and running time are*

$$O\left(\frac{k^4 \cdot \log^2(k/\delta)}{\epsilon^6}\right) \quad \text{and} \quad \exp\left(O\left(\frac{k^2 \cdot \log^2(k/\delta)}{\epsilon^3}\right)\right)$$

respectively. 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 (additional) time $O\left(\frac{\log(k/\delta)}{\epsilon^2}\right) \cdot N$.*

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 $1-\delta$. Again, 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 $\mathrm{G} \in \mathcal{G}_k$, 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 $\mathrm{G} \notin \mathcal{G}_k$, 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 $m = O(\ell \cdot \epsilon^{-1} \log(k/\delta))$ and $\ell = \lceil 4k/\epsilon \rceil$. 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 $\mathrm{X} = \mathrm{U} \cup \mathrm{S}$, 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 $\mathrm{U}' \subset \mathrm{U}$ (which is determined later), 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 in the same components of the partition of U'. For every $\mathrm{A} \subseteq [k]$, the A-cluster, denoted C_A, contains all vertices in V(G) which have neighbors in U'_i for every $i \in \mathrm{A}$ (and do not have neighbors 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 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 $i \in \mathrm{A}$. As a special case, $\mathrm{C}_\emptyset$ is the set of vertices that do not have any neighbors in U' (and hence can be put in any component of the partition). In the other extreme, $\mathrm{C}_{[k]}$ is the set of vertices that in any extension of the partition of U' will cause violations. For each i, the vertices in $\mathrm{C}_{[k]\setminus\{i\}}$ are *forced* to be put in V_i, and thus are easy to handle. In the bipartite case we focused on the clusters $\mathrm{C}_{\{1\}}$ and $\mathrm{C}_{\{2\}}$, where vertices in $\mathrm{C}_{\{i\}}$ were forced to the side opposite to i. (The cluster $\mathrm{C}_\emptyset$ was explicitly shown to be unimportant and the cluster $\mathrm{C}_{1,2}$ was dealt with implicitly.) In the case of k-Coloring the situation is more complex. In particular, the clusters C_A where $|\mathrm{A}| < k-1$ do not force a placement of vertices.

Definition 6.12 (clusters): *Let* U' *be a set of vertices, and let* π' *be a perfect k-partition of* U'. *Define* $\mathrm{U}'_i \stackrel{\text{def}}{=} \{v \in \mathrm{U} : \pi'(v) = i\}$. *For each subset* $\mathrm{A} \subseteq [k]$ *we define the* A-cluster with respect to π' *as follows:*

$$\mathrm{C}_\mathrm{A} \stackrel{\text{def}}{=} \left(\bigcap_{i \in \mathrm{A}} \Gamma(\mathrm{U}'_i)\right) \setminus \left(\bigcup_{i \notin \mathrm{A}} \Gamma(\mathrm{U}'_i)\right) . \tag{5}$$

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

Definition 6.13 (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* $v \in \Gamma(u) \setminus \mathrm{C}_{[k]}$*, where* $\mathrm{C}_{[k]}$ *is the* $[k]$*-cluster with respect to* π'*.*

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

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 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 in a certain component 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. A vertex $v \in \mathrm{C_A}$ is *restricting* if for every $i \in \bar{\mathrm{A}}$, adding v to U'_i (and thus to U') will cause many of its neighbors to move to a cluster corresponding to a bigger subset. That is, v's neighbors in the B-cluster (with respect to $(\mathrm{U}'_1, \ldots, \mathrm{U}'_k)$) move to the $(\mathrm{B} \cup \{i\})$-cluster (with respect to $(\mathrm{U}'_1, \ldots, \mathrm{U}'_i \cup \{v\}, \ldots, \mathrm{U}'_k)$). More precisely,

Definition 6.14 (restricting vertex): *A* pair (v, i), *where* $v \in \mathrm{C_A}$, $\mathrm{A} \neq [k]$ *and* $i \in \bar{\mathrm{A}}$ *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* $\cup_{\mathrm{B}:i \notin \mathrm{B}} \mathrm{C_B}$*. Otherwise,* (v, i) *is* non-restricting*. A* vertex $v \in \mathrm{C_A}$, $\mathrm{A} \neq [k]$, *is* restricting *with respect to* π' *if for every* $i \in \bar{\mathrm{A}}$*, the pair* (v, i) *is restricting. Otherwise,* v *is* non-restricting*. As always, the clusters are with respect to* π'*.*

Given a perfect k-partition of U, we construct $\mathrm{U}' \subset \mathrm{U}$ in steps starting with the empty set. At step j we add to U' a vertex $u \in \mathrm{U}^j$ (recall that $\mathrm{U} = \mathrm{U}^1 \dot\cup \ldots \dot\cup \mathrm{U}^\ell$), 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 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 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 $\frac{\epsilon}{8} \cdot N$ restricting vertices with respect to π' then halt and output U'.

- If there are at least $\frac{\epsilon}{8} \cdot N$ restricting vertices but there is no restricting vertex in U^j, then halt and output an error symbol.
- Otherwise (there is a restricting vertex in U^j), add the first (by any fixed order) restricting vertex to U' (and proceed to the next iteration).

Claim 6.15 *For every* U *and every perfect k-partition of* U, *after at most $\ell = \lceil 4k/\epsilon \rceil$ iterations, the Restriction Procedure halts and outputs either* U' *or* error.

Proof: If the procedure has not halted (and output an error symbol), then in each iteration a restricting vertex is added to U' causing at least $\frac{\epsilon}{4} \cdot N$ vertices in V(G) to move to a cluster corresponding to a bigger subset. Since each vertex can be moved at most k times (before it belongs to $\mathrm{C}_{[k]}$), 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 an error symbol. 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 6.16 (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 *symbol).*

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 $\frac{\epsilon}{8} \cdot N$ restricting vertices with respect to the current partition of U', then U^j will include at least one such restricting vertex.

Lemma 6.17 *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 $\frac{\epsilon}{8} \cdot N$ restricting vertices with respect to the partition π' of the current U', then the probability that in a uniformly chosen sample of size m $(= |\mathrm{U}^j|)$ there will be no restricting vertex with respect to π', is at most $(1 - \frac{\epsilon}{8})^m$. By our choice of $m = O(\ell \cdot \epsilon^{-1} \log(k/\delta))$, 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^{th} iteration is at most k^{j-1}.

We shall prove the above claim inductively. Let the set U' at the start of iteration j (before adding a new restricting vertex to it) be denoted by $\mathrm{U}'(j-1)$, and let its partition be denoted by π'_{j-1}. For the base case, $j = 1$, the set $\mathrm{U}'(0)$ is empty and the claim trivially holds. Assuming the claim holds for $j-1$, we now prove it for j. In the j^{th} iteration, for each of the possible pairs $(\mathrm{U}'(j-1), \pi'_{j-1})$, such that there exist at least $\frac{\epsilon}{8} \cdot N$ restricting vertices with respect to π'_{j-1}, the vertex $u_j \in \mathrm{U}^j$ which is the *first* restricting vertex in U^j, is uniquely defined.[15] Hence, for each such pair $(\mathrm{U}'(j-1), \pi'_{j-1})$, there is a single possible extension $\mathrm{U}'(j)$ of $\mathrm{U}'(j-1)$, namely, $\mathrm{U}'(j) = \mathrm{U}'(j-1) \cup \{u_j\}$. The new partition, π'_j which extends π'_{j-1} can be one of at most k possibilities (depending only on $\pi'_j(u_j)$). ■

Definition 6.18 (closed partitions): *Let* U' *be a set and π' a k-partition of it. We call* (U', π') closed *if there are less than $\frac{\epsilon}{8} \cdot N$ restricting vertices with respect to π'.*

[15]It may be the case that no such restricting vertex exists in U^j, but the probability for this event has been bounded above.

Clearly, if the Restriction Procedure outputs a set U' then this set together with 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 $\mathrm{C_A}$, $\mathrm{A} \neq [k]$, has the following property. There exists at least one index $i \in \bar{\mathrm{A}}$, such that (v, i) is non-restricting. It follows from Definition 6.14 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.[16] However, even if v is non-restricting there might be indices $i \in \bar{\mathrm{A}}$ 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 6.19 (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*

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

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

Lemma 6.20 *Let (U', π') be an arbitrary closed pair and F_v's be as in Definition 6.19. Then:*

1. $|\{v : (v \notin C_{[k]}) \wedge (\mathrm{F}_v = [k])\}| \leq \frac{\epsilon}{8}N$.
2. *Let π be any k-partition of $\mathrm{V(G)} \setminus \{v : \mathrm{F}_v = [k]\}$ such that $\pi(v) \notin \mathrm{F}_v$, 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 $\frac{\epsilon}{2} \cdot N^2$.*

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 $\mathrm{C}_{[k]}$. We deal with vertices in $\mathrm{C}_{[k]}$ at a later stage.

Proof: The first item follows from the closeness of (U', π'). Namely, if $\mathrm{F}_v = [k]$ and $v \notin \mathrm{C}_{[k]}$ then by the second item of Definition 6.19 it follows that for every $i \in \bar{\mathrm{A}}$, vertex v has at least $\frac{\epsilon}{4} \cdot N$ neighbors in clusters $\mathrm{C_B}$ such that $i \notin \mathrm{B}$. But in this case, it is a restricting vertex with respect to π'. By Definition 6.18 as applied to (U', π'), there are at most $\frac{\epsilon}{8} \cdot N$ such vertices.

For the second item, consider a vertex v such that $\pi(v) = i$ and so $v \in \mathrm{C_A}$ where $i \notin \mathrm{F}_v \supseteq \mathrm{A}$. All edges (v, u) and (u, v) such that $u \in \mathrm{C_B}$ and $i \in \mathrm{B}$ cannot be violating edges since $i \in \mathrm{F}_u$ (by the first item in Definition 6.19). As for edges (v, u) and (u, v) where $u \in \mathrm{C_B}$ and $i \notin \mathrm{B}$, vertex v can have at most $\frac{\epsilon}{2} \cdot N$ such edges (according to the second item in Definition 6.19). The total of violating edges is hence at most $\frac{\epsilon}{2} \cdot N^2$. ■

We next 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 $\mathrm{C}_{[k]}$ is small. This implies that all the vertices in $\mathrm{C}_{[k]}$ (which were left out of the partition in the previous lemma) can be placed in any component of the partition without contributing too many violating edges (which are incident to them).

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

Definition 6.21 (useful k-partitions): *We say that a pair* (U', π') *is* **ϵ-useful** *if* $|C_{[k]}| < \frac{\epsilon}{8}N$. *Otherwise it is* **$\epsilon$-unuseful**.

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

Claim 6.22 *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* $1 - \frac{\delta}{2}k^{-\ell}$*, there exists no perfect* k*-partition of* $\mathrm{U}' \cup \mathrm{S}$ *which extends* π'.

Proof: By definition, for $\mathrm{C}_{[k]}$ defined based on (U', π'), we have $|\mathrm{C}_{[k]}| \geq \frac{\epsilon}{8} \cdot N$, and so a uniformly chosen S contains a vertex in $\mathrm{C}_{[k]}$ with the claimed probability, in which any extension of π' to S is non-perfect. ■

By the same argument applied in the proof of Lemma 6.17, 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 6.23 *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 main pair of the proof of Theorem 6.11. If G is accepted with probability greater than δ, then by Lemma 6.17, 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 6.23), there must be a useful closed pair (U', π') (where $\mathrm{U}' \subset \mathrm{U}$). If we now partition V(G) as described in Lemma 6.20, where vertices with forbidden set $[k]$ are placed arbitrarily, then from the two items of Lemma 6.20 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.

Similar 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 determine a partition of V(G). Details follow.

Efficient Construction of an ϵ-Good k-Partition of V(G). For the efficient implementation, we assume the testing algorithm is run with distance parameter $\epsilon/2$ and confidence parameter $\delta/2$. The main point we need to address is the question of efficiently implementing the Restricting Procedure (i.e., constructing U' given U and a perfect k-partition of 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 $\frac{\epsilon}{8} \cdot N$ 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 restricting vertices, we do the following. We uniformly select ℓ sets of vertices, $\mathrm{W}^1, \ldots, \mathrm{W}^{\ell}$, each of size $\Theta\left(\frac{k \log(k/(\epsilon\delta))}{\epsilon^2}\right)$. Consider the j^{th} iteration of the restricting procedure, where we search for a restricting vertex in U^j (with respect to the current U' and π'). Using W^j we approximate, for each $u \in \mathrm{U}^j$, and each $i \in \bar{\mathrm{A}}$ (where $u \in \mathrm{C_A}$), how many neighbors does u have in B-clusters such that $i \notin \mathrm{B}$. To do so we simply perform queries on all pairs of vertices in $\mathrm{U}' \times \mathrm{W}^j$, and all pairs in $\mathrm{U}^j \times \mathrm{W}^j$. For each $u \in \mathrm{U}^j$, let $\mathrm{R}(u,i)$ be the set of neighbors of u in W_j that belong to B-clusters such that $i \notin \mathrm{B}$. If for some $u \in \mathrm{U}^j$, $|\mathrm{R}(u,i)|/|\mathrm{W}^j| \geq 3\epsilon/8$ for every $i \in \bar{\mathrm{A}}$, then we conclude that u is a restricting vertex, and we select the first such vertex in U^j to add to U'. By our choice of the size of each W^j and an additive Chernoff bound, with probability at least $1-\delta/4$, for every j, if U^j contains a restricting vertex then it will be detected. Furthermore, any vertex in U^j that is considered as restricting in fact has at least $\frac{\epsilon}{8} \cdot N$ neighbors (in $\mathrm{V(G)}$) that belong to B-clusters such that $i \notin \mathrm{B}$, for every $i \in \bar{\mathrm{A}}$. Note that the first such vertex is uniquely determined by the set W^j, and that the time for implementing the restriction procedure is negligible with respect to the running time of the testing algorithm.

Once U' is constructed, we cluster all vertices in $\mathrm{V(G)}$ according to U' and π'. To do so we simply perform all queries between vertices in $\mathrm{V(G)}$ and U'. As for implementing the definition of forbidden sets, here each vertex $v \in \mathrm{V(G)}$ samples its neighbors to determine F_v. Namely, for each vertex v we select a sample of $\Theta\left(\frac{\log(k/(\epsilon\delta))}{\epsilon^2}\right)$ vertices, and approximate, for each $i \in \bar{\mathrm{A}}$ (where $v \in \mathrm{C_A}$), the number of neighbors that v has in B-clusters such that $i \notin \mathrm{B}$. The expected number of vertices v for which these approximations differ significantly from the expected value is $O(\epsilon\delta \cdot N)$, and hence with probability at least $1-\delta/4$, there are $\frac{\epsilon}{4} \cdot N$ vertices for which these approximations are in fact far from correct. Adding the contribution of these vertices to the number of violating edges accounted by Lemma 6.20 and Claim 6.22, we get an ϵ-good partition. The running time is hence governed by the implementation of the forbidden sets, and is $O\left(\frac{\log(k/(\epsilon\delta))}{\epsilon^2}\right) \cdot N$. ■ (Theorem 6.11) ■

7 Testing Max-Clique

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

Theorem 7.1 *There exists a property testing algorithm, $\mathcal{A}$, for the class $\mathcal{C}_\rho$ whose query complexity and running time are*

$$O\left(\frac{\log^2(1/(\epsilon\delta))\rho^2}{\epsilon^6}\right) \quad \textit{and} \quad \exp\left(O\left(\frac{\log(1/(\epsilon\delta))\rho}{\epsilon^2}\right)\right)$$

respectively. In particular, $\mathcal{A}$ uniformly selects $O\left(\frac{\log(1/(\epsilon\delta))\rho}{\epsilon^4}\right)$ vertices in G *and queries the oracle only on the existence of edges between (some of) these vertices. In case $\mathrm{G} \in \mathcal{C}_\rho$, one can also retrieve in time $O\left(\frac{\log(1/(\epsilon\delta))\rho}{\epsilon^2}\right) \cdot N$ a set of $\rho \cdot N$ vertices in* G *which is almost a clique (in the sense that it lacks at most $\epsilon \cdot N^2$ edges to being a clique).*

Theorem 7.1 is proven by presenting a seemingly unnatural algorithm/tester (see below). However, as a corollary, we observe that the "natural" algorithm, which uniformly selects $\mathrm{poly}(\epsilon^{-1}\log(1/\delta))$ many vertices and accepts iff they induce a subgraph with a clique of density $\rho - \frac{\epsilon}{2}$, is a valid $\mathcal{C}_\rho$-tester as well.

Corollary 7.2 *Let $q(\cdot,\cdot)$ be the query complexity of algorithm $\mathcal{A}$ guaranteed by Theorem 7.1 (i.e., $q(\epsilon,\delta) = \mathrm{poly}(\epsilon^{-1}\log(1/\delta))$), and let $\rho, \epsilon, \delta > 0$. Let* R *be a uniformly selected set of $m \stackrel{\mathrm{def}}{=} 20 \cdot q(\epsilon/2, \delta/5)$ vertices in $V(\mathrm{G})$, and $\mathrm{G_R}$ be the subgraph (of* G*) induced by* R*. Then,*

- *if* $\mathrm{G} \in \mathcal{C}_\rho$ *then* $\Pr_{\mathrm{R}}[\omega(\mathrm{G_R}) \leq (\rho - \epsilon/2) \cdot m] < \delta$.
- *if* $\mathrm{dist}(\mathrm{G}, \mathcal{C}_\rho) > \epsilon$ *then* $\Pr_{\mathrm{R}}[\omega(\mathrm{G_R}) \geq (\rho - \epsilon/2) \cdot m] < \delta$.

Proof: Our presentation presupposes that $\mathcal{A}$ is given oracle access to a graph whose vertices 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 $\mathrm{G_R}$ (as done below), we provide it with oracle access to a graph in which each vertex of $\mathrm{G_R}$ is duplicated $\frac{|\mathrm{V(G)}|}{|\mathrm{R}|}$ times and edges are duplicated in the natural manner.

With the above technicality being settled, let us present the underlying ideas of the proof. The first item follows easily from the fact that $q(\epsilon, \delta) = \Omega(\epsilon^{-2}\log(1/\delta))$ (as such a sample is likely to hit enough vertices of the clique). The issue is the second item. The basic idea is that if $\mathrm{G_R}$ has a clique of size $\rho' \stackrel{\mathrm{def}}{=} \rho - \frac{\epsilon}{2}$ and we were to run $\mathcal{A}$ on it (with density parameter ρ'), then $\mathcal{A}$ would accept with high probability. On the other hand, when $\mathcal{A}$ is invoked on G (with density parameter ρ' and distance parameter $\epsilon/2$), algorithm $\mathcal{A}$ rejects with high probability. Loosely speaking, these two facts stand in contradiction since the samples that $\mathcal{A}$ sees in the two cases are statistically close. Actually, the above would be true if we were to set $m = O(q(\epsilon/2, \delta/5)^2)$ (as done in preliminary versions of this paper), as in both cases $\mathcal{A}$ would be likely to see a uniformly distributed set of $q(\epsilon/2, \delta/5)$ distinct vertices. This however does not happen with $m = O(q(\epsilon/2, \delta/5))$, since when sampling $\mathrm{G_R}$ algorithm $\mathcal{A}$ is likely to see several occurrences of some vertices (whereas this is unlikely when sampling G). Thus, we start by presenting an interface between the graph and algorithm $\mathcal{A}$. Loosely speaking, the interface obtains a sample 10 times bigger than what is required by $\mathcal{A}$, and passes to $\mathcal{A}$ a uniformly chosen subset of the required size consisting of distinct vertices.

Interface to $\mathcal{A}$: We use the fact that $q(\epsilon/2, \delta/5) > 2\log(5/\delta)$ (otherwise we would have needed to increase the size of m). Let $q \stackrel{\mathrm{def}}{=} q(\epsilon/2, \delta/5)$, and suppose that $q < N/2$ (as otherwise we can just scan the entire graph). Given a sample of $10q$ vertices (with possible repetitions), the interface passes on the first q distinct vertices, and aborts if such elements do no exists. We adopt the convention by which whenever the interface aborts, the (modified) algorithm accepts.

Comment: Actually, we need to further modify the above interface so that collisions may occur with small probability, as would be the case when sampling independently q elements from V(G). This is easy to do (e.g., one always passes the first vertex, and for $i = 2, ..., q$ and $j = 1, ..., i-1$, one repeats the j^{th} element in the i^{th} location with probability $1/N$ and otherwise passes the i^{th} element). In the rest of the proof we ignore this modification of the interface.

Claim 1: Suppose that $q \leq m/2$, and that S is a set of m distinct elements. Then, the probability that the interface aborts when given a sample of $10q$ elements uniformly and independently selected in S is smaller than $\delta/5$.

Proof: It suffices to upper bound the probability that a random multiset of $10q$ elements selected in a set of size $2q$ contains less than q distinct elements. Let ζ_i represent the event that either the $i-1$ first multiset elements contain at least q distinct elements or the i^{th} element in the multiset is different from each of the previous ones. Clearly, $\Pr[\zeta_i = 1 | \sum_{j<i} \zeta_j < q] \geq 1/2$, whereas $\Pr[\zeta_i = 1 | \sum_{j<i} \zeta_j \geq q] = 1$. Note that $\Pr[\sum_{i=1}^{10q} \zeta_i < q]$ represents the probability of aborting.

$$\begin{aligned}
\Pr\left[\sum_{i=1}^{10q} \zeta_i < q\right] &< \sum_{s=1}^{q-1} \binom{10q}{s} \cdot 0.5^{10q-s} \\
&< 2^{q-1} \cdot \exp\left(-2 \cdot (0.5 - 0.1)^2 \cdot 10q\right) \\
&< 2^{-q/2}
\end{aligned}$$

which is at most $\delta/5$ as required. ■

Claim 1 will be applied both with respect to the set V(G), and with respect to the subset R as in the statement of the corollary. The next claim refers to the latter set R, and is quite obvious.

Claim 2: Suppose that $q \leq m/2$, and suppose that the interface is given a uniformly selected set $\mathrm{R} \subset \mathrm{V(G)}$ of size m. Then conditioned on not having aborted, the interface passes on a set of q vertices which is uniformly distributed among all such subsets of V(G).

We now turn to the actual claims of the corollary. As stated above, the first item is obvious (given that $q(\epsilon/2, \delta/5) = \Omega(\epsilon^{-2}\log(1/\delta))$). Our aim is to prove the second item. Suppose that $\mathrm{dist}(\mathrm{G}, \mathcal{C}_\rho) > \epsilon$, and consider what happens if we were to run $\mathcal{A}$, through the interface, on $\mathrm{G_R}$ with density parameter $\rho' = \rho - \epsilon/2$, distance parameter $\epsilon' = \epsilon/2$ and confidence parameter $\delta' = \delta/5$. In such a case, algorithm $\mathcal{A}$ requires a sample of size $q = q(\epsilon', \delta')$, which is supplied by taking a random sample of $10q$ vertices of R and passing them through the interface to $\mathcal{A}$. By Claim 1, the interface aborts with probability less than $\delta/5$. By Claim 2, for a uniformly selected R, conditioned on the abort event not happening, algorithm $\mathcal{A}$ obtains a sample which is identically distributed to the sample it obtains when testing G under the same set of parameters, denoted $\pi' \stackrel{\text{def}}{=} (\rho', \epsilon', \delta')$. Denoting by $\mathcal{B}$ the composition of the interface with $\mathcal{A}$, we conclude that

$$\left|\Pr_{\mathrm{R}}[\mathcal{B}^{\mathrm{G_R}}(\pi') = 1] - \Pr[\mathcal{B}^{\mathrm{G}}(\pi') = 1]\right| < \frac{\delta}{5} \tag{6}$$

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 $\mathrm{G}' \in \mathcal{C}_{\rho'} = \mathcal{C}_{\rho-(\epsilon/2)}$). Using this fact, Eq. (6) and the fact that the interface increases the success probability of $\mathcal{A}$ by at most $\frac{\delta}{5}$, we have

$$\begin{aligned}
\Pr_{\mathrm{R}}[\mathcal{B}^{\mathrm{G_R}}(\pi') = 1] &< \Pr[\mathcal{B}^{\mathrm{G}}(\pi') = 1] + \frac{\delta}{5} \\
&< \Pr[\mathcal{A}^{\mathrm{G}}(\pi') = 1] + 2 \cdot \frac{\delta}{5} \\
&< \frac{\delta}{5} + \frac{2\delta}{5}
\end{aligned}$$

On the other hand, using the fact that $\mathcal{B}$'s accepting probability is lower bounded by that of $\mathcal{A}$, we have,

$$\begin{aligned}
\Pr_{\mathrm{R}}[\mathcal{B}^{\mathrm{G_R}}(\pi') = 1] &\geq \Pr_{\mathrm{R}}[\omega(\mathrm{G_R}) > \rho' \cdot m] \cdot \min_{\mathrm{G}' : \omega(\mathrm{G}') > \rho' \cdot m} \{\Pr[\mathcal{B}^{\mathrm{G}'}(\pi') = 1]\} \\
&> \Pr_{\mathrm{R}}[\omega(\mathrm{G_R}) > \rho' \cdot m] \cdot \left(1 - \frac{\delta}{5}\right)
\end{aligned}$$

We conclude that

$$\begin{aligned}
\Pr_{\mathrm{R}}[\omega(\mathrm{G_R}) > \rho' \cdot m] &< \frac{\Pr_{\mathrm{R}}[\mathcal{B}^{\mathrm{G_R}}(\pi') = 1]}{1 - (\delta/5)} \\
&< \frac{2\delta}{5} + \frac{3\delta}{5}
\end{aligned}$$

as required. This completes the proof of Corollary 7.2. ■

We start by presenting and analyzing an algorithm that, given a graph G that has a clique of size ρN, finds a set of vertices of size ρN that is close to being a clique Namely, for any given ϵ

and δ, with probability at least $1-\delta$, the algorithm finds a set of vertices of size ρN, such that there are at most ϵN^2 pairs of vertices in this set that are not connected by an edge. Our testing algorithm builds on this algorithm, which we call the Approximate-Clique Finding algorithm. The Approximate-Clique Finding algorithm (described in Figure 2) runs in time proportional to N^2. We later show how an approximate clique can be found in time linear in N as asserted in Theorem 7.1. Throughout the analysis we assume that $\epsilon < \rho^2$ (as otherwise $\mathrm{dist}(\mathrm{G}, \mathcal{C}_\rho) \leq \rho^2 \leq \epsilon$ for every graph G).

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 $N - |\mathrm{V}(\mathrm{G}')|$ isolated vertices. In all cases $|\mathrm{V}(\mathrm{G}')| \leq N$. With slight abuse of notation, for a set $\mathrm{X} \subseteq \mathrm{V}(\mathrm{G})$, 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.

7.1 The Approximate-Clique Finder

A MENTAL EXPERIMENT. To motivate our algorithm we start with the following mental experiment. Assume that G has a clique C of size ρN. Suppose we had an oracle that would tell us for every given vertex v whether v is a neighbor of *every* vertex in C. By querying the oracle on each vertex in the graph, we could determine the set of vertices, denoted T(C), that neighbor every vertex in C. Note that since C is a clique, C is a subset of T(C). Unfortunately there might be many other vertices in T(C) that do not belong to C. However, assume that we order the vertices in T(C) according to their degree in the subgraph induced by T(C) (breaking ties arbitrarily), and let C$'$ be first ρN vertices according to this order. Then we claim that C$'$ is a clique (though it might be different from C). To see this, observe that by definition of T(C), each vertex in C neighbors every vertex T(C) (except itself). Thus, each vertex in C has degree $2(|\mathrm{T}(\mathrm{C})|-1)$ in the subgraph induced by T(C), which is the maximum possible. (Recall that according to our convention, the 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.) Since $|\mathrm{C}| = \rho N$, every vertex in C$'$ must have degree $2(|\mathrm{T}(\mathrm{C})|-1)$ as well (because the vertices in C are all candidates for the set C$'$ whose size is ρN as well). In other words, every vertex in C$'$ neighbors every (other) vertex in T(C), and in particular it neighbors every other vertex in C$'$, making C$'$ a clique.

Suppose next that instead of having an oracle to C, we were given a uniformly chosen set U$'$ in C of sufficient size (i.e., of size $\Theta(\epsilon^{-2}\cdot\log(1/(\epsilon\delta)))$). Let T(U$'$) be the set of vertices that neighbor every vertex in U$'$. Then, with high probability over the choice of U$'$, almost every vertex in T(U$'$) neighbors almost all vertices in C. Similarly to the above, we could order the vertices in T(U$'$) according to their degree in the graph induced by T(U$'$), and take the first ρN vertices. It can be shown (and we prove something slightly stronger later) that the resulting set is close to being a clique.

THE ACTUAL ALGORITHM. Since a uniformly chosen set in C is not provided to the algorithm, it instead "guesses" such a set. More precisely, it uniformly selects a set U from all graph vertices, and it considers all its subsets U$'$ of size $\frac{\rho}{2}|\mathrm{U}|$. Since with high probability $|\mathrm{U}\cap\mathrm{C}| \geq \frac{\rho}{2}|\mathrm{U}|$, there exists a subset U$'$ contained in C. Furthermore, with high probability, for this U$'$, almost all vertices in T(U$'$) (the set of vertices that neighbor every vertex in U$'$), neighbor almost every vertex in C. The next idea is to *approximate* the degree of each $v \in \mathrm{T}(\mathrm{U}')$ in the subgraph induced by T(U$'$), instead of computing it exactly. While this is not necessary for the efficiency of the Approximate-

Clique Finding algorithm (as it runs in time quadratic in N anyhow), it will be useful to apply this modification here so as to simplify the analysis of the tester (which is presented later).

To approximate the degree of vertices in $\mathrm{T}(\mathrm{U}')$, we uniformly select an additional set, W, and let $\mathrm{W}(\mathrm{U}') \subseteq \mathrm{W}$ contain all vertices in W that neighbor every vertex in U'. Thus, $\mathrm{W}(\mathrm{U}')$ is effectively sampling from $\mathrm{T}(\mathrm{U}')$. We now order the vertices in $\mathrm{T}(\mathrm{U}')$ according to the number of neighbors they have in $\mathrm{W}(\mathrm{U}')$, and take the first ρN vertices according to this order (if $|\mathrm{T}(\mathrm{U}')| < \rho N$ then necessarily $\mathrm{U}' \not\subseteq \mathrm{C}$ and we don't need to consider this U'). Thus for every subset U' we obtain a set of ρN vertices, and we output the one that misses the fewest edges to being a clique. (We note that U and W together play the role that the subset U of the sampled vertices plays in our other algorithms. Namely, U and W together are used to determined partitions of $\mathrm{V}(\mathrm{G})$.)

Approximate-Clique Finding Algorithm

Let $t = \Theta\left(\frac{\log(1/(\epsilon\delta))\cdot\rho}{\epsilon^2}\right)$, *and* $r = \Theta(\frac{\log(1/(\epsilon\delta))\cdot\rho}{\epsilon^2})$.

1. *Uniformly select two independent samples,* U, W *of sizes* t *and* r*, respectively.*
2. *For each* $\mathrm{U}' \subset \mathrm{U}$ *of cardinality* $\frac{\rho}{2}\cdot t$*, perform the following steps:*
   (a) *Let* $\mathrm{T}(\mathrm{U}') \subseteq \mathrm{V}(\mathrm{G})$ *be the subset of vertices in* G *which neighbor all vertices in* U' *(i.e.,* $\mathrm{T}(\mathrm{U}') \stackrel{\rm def}{=} \{v \in \mathrm{V} : \Gamma(v) \supseteq \mathrm{U}'\} = \mathrm{V} \cap (\bigcap_{u\in \mathrm{U}'}\Gamma(u))$*).*
   If $|\mathrm{T}(\mathrm{U}')| < \rho N$ *then continue with the next subset* U'*.*
   (b) *Let* $\mathrm{W}(\mathrm{U}') \subseteq \mathrm{W}$ *be the subset of vertices in* W *which neighbor all vertices in* U' *(i.e.,* $\mathrm{W}(\mathrm{U}') \stackrel{\rm def}{=} \{v \in \mathrm{W} : \Gamma(v) \supseteq \mathrm{U}'\} = \mathrm{W} \cap (\bigcap_{u\in \mathrm{U}'}\Gamma(u))$*).*
   (c) *For each* $v \in \mathrm{T}(\mathrm{U}')$*, compute* $\hat{d}(v) \stackrel{\rm def}{=} 2\cdot|\Gamma(v)\cap \mathrm{W}(\mathrm{U}')|$*. Let* $\mathrm{C}(\mathrm{U}') \subseteq \mathrm{T}(\mathrm{U}')$ *be the set of* ρN *vertices of the highest* $\hat{d}(\cdot)$ *value in* $\mathrm{T}(\mathrm{U}')$*. (Ties are broken by lexicographic order).*
3. *Among all sets* $\mathrm{C}(\mathrm{U}')$*, let* $\widetilde{\mathrm{C}}$ *be the one for which* $\mathrm{dist}(\mathrm{C}(\mathrm{U}'), C_N^\rho)$ *is minimizes.*
   Output $\widetilde{\mathrm{C}}$*.*

Figure 2: Approximate-Clique Finding Algorithm

The resulting Approximate-Clique Finding algorithm is given in Figure 2. In order to establish the correctness of the algorithm, we first show that with high probability, the set U has certain desired properties.

As before, let C be a clique of size ρN in G. For $\epsilon_1 \in [0,1]$, we say that a set $\mathrm{U}' \subset \mathrm{C} \subset \mathrm{V}(\mathrm{G})$ is **ϵ_1-clique-representative** with respect to C, if for all but $\epsilon_1 N$ of the vertices, $v \in \mathrm{V}(\mathrm{G})$,

$$\text{if } \ |\Gamma(v)\cap \mathrm{C}| < (\rho-\epsilon_1)N \text{ then } \ \Gamma(v)\cap \mathrm{U}' \neq \mathrm{U}' \tag{7}$$

That is, for most vertices v for which $\Gamma(v)\cap \mathrm{U}' = \mathrm{U}'$, it holds that $|\Gamma(v)\cap \mathrm{C}| \geq (\rho-\epsilon_1)N$. Note that for every $v \in \mathrm{C}$ the above condition holds for all $\epsilon_1 \geq 0$ (since $\Gamma(v) \supseteq \mathrm{C} \supset \mathrm{U}'$).

Claim 7.3 *Let* $t = \Omega(\frac{\log(1/(\epsilon_1\delta))}{\epsilon_1\rho})$*, where* $\epsilon_1 \in [0,1]$*. 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_1*-clique-representative subset of size* $\frac{\rho}{2}t$*.*

Proof: Using a multiplicative Chernoff Bound (see Appendix B) 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 $t' \stackrel{\rm def}{=} \frac{\rho}{2}t$. Clearly, for each $v \in \mathrm{C}$ Equation (7.3) holds. For each $v \in \mathrm{V}(\mathrm{G})\setminus\mathrm{C}$

$$\Pr_{\mathrm{U}'}[\text{Eq. (7) does not hold for } v] \;=\; (1-\epsilon_1)^{t'} \;<\; \frac{\epsilon_1\delta}{4} \tag{8}$$

where the last inequality is due to $t' = \frac{\rho}{2}t$ and the hypothesis regarding t. Thus, the expected number of vertices which violate Eq. (7) is bounded by $\frac{\delta}{4}\cdot\epsilon_1 N$. Applying Markov's Inequality (see Appendix B) we conclude that with probability at least $1-\delta/4$ there are at most $\epsilon_1 N$ vertices which violate Eq. (7). The lemma follows. ■

We next show that for an ϵ_1-clique-representative U′, if we take a subset of T(U′) of size ρN which approximately contains the vertices that have highest degree in the subgraph induces by T(U′), then we obtain an approximate clique.

Lemma 7.4 *Let $\epsilon_2 < \epsilon/\rho$ and $\epsilon_1 = \epsilon_2^2$. Let U' be ϵ_1-clique-representative* (with respect to C) *and $\mathrm{T} = \mathrm{T}(\mathrm{U}') \stackrel{\rm def}{=} \cap_{u\in\mathrm{U}'}\Gamma(u)$. Let α be such that αN is the degree of the $(\rho-\epsilon_2)\cdot N^{\rm th}$ vertex of highest degree in* T. *(We stress that we consider degrees in the subgraph of* G *induced by* T.*) Let $\widetilde{\mathrm{C}} \subseteq \mathrm{T}$ be a set of size $\rho\cdot N$ that contains at least $(\rho-3\epsilon_2)\cdot N$ vertices of degree at least $(\alpha-2\epsilon_2)\cdot N$ (in the subgraph induced by* T*). Then, $\widetilde{\mathrm{C}}$ satisfies $\mathrm{dist}(\widetilde{\mathrm{C}}, C_N^{\rho}) = 8\epsilon_2\rho$.*

The somewhat complicated formulation of the lemma is meant to fit its application in the analysis of the Approximate-Clique Finding Algorithm. Specifically, it addresses the fact that the algorithm uses approximations to the degrees of vertices in T, and that these approximations are slightly inaccurate for most vertices, and very inaccurate for few vertices.

Proof: Clearly $\mathrm{T}\supseteq\mathrm{C}$ (as U′ is a subset of the clique C). Let $\mathrm{H} \stackrel{\rm def}{=} \{v\in \mathrm{V(G)}\setminus\mathrm{C} : |\Gamma(v)\cap\mathrm{C}| \geq (\rho-\epsilon_1)\cdot N\}$ be the set of vertices outside of C having many neighbors in C. Let $\mathrm{R} \stackrel{\rm def}{=} \mathrm{T}\setminus(\mathrm{C}\cup\mathrm{H})$ (i.e., the rest of T). Since U′ is ϵ_1-clique-representative, it follows that $|\mathrm{R}| < \epsilon_1 N$ (since a vertex not in $\mathrm{C}\cup\mathrm{H}$ may enter T only if it is adjacent to all vertices in U′ whereas it is adjacent to less than $(\rho-\epsilon_1)N$ vertices in C). Let the degree of vertex v in the subgraph induced by T be denoted by $\deg_{\mathrm{T}}(v)$. (Recall that according to our convention, the 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 have,

$$\begin{aligned}
\sum_{v\in\mathrm{C}}\deg_{\mathrm{T}}(v) &\geq 2\cdot|\{(u,v): u,v\in\mathrm{C}\}| + 2\cdot|\{(u,v): u\in\mathrm{C}, v\in\mathrm{H}\}| \\
&\geq 2\cdot(|\mathrm{C}|^2 + |\mathrm{H}|\cdot(\rho-\epsilon_1)\cdot N) \\
&> 2\rho N\cdot\left(\rho N + |\mathrm{H}| - \frac{\epsilon_1}{\rho}N\right) \qquad (9)
\end{aligned}$$

Thus, the average value of $\deg_{\mathrm{T}}(v)$ for $v\in\mathrm{C}$ is at least $2\cdot\left(\rho N + |\mathrm{H}| - \frac{\epsilon_1}{\rho}N\right)$. On the other hand, the maximum value of $\deg_{\mathrm{T}}(v)$ for $v\in\mathrm{C}$ is bounded above by $2|\mathrm{T}| = 2(|\mathrm{C}|+|\mathrm{H}|+|\mathrm{R}|) \leq 2(\rho N + |\mathrm{H}| + \epsilon_1 N)$. Therefore, the difference between the maximum value and the average value of $\deg_{\mathrm{T}}(v)$ for $v\in\mathrm{C}$ is $2\cdot(\epsilon_1+\epsilon_1/\rho)\cdot N$. By a simple counting argument (which is essentially a variant of Markov's Inequality) we have that for every k,

$$\left|\left\{v\in\mathrm{C} : \deg_{\mathrm{T}}(v) < 2\left(\rho N + |\mathrm{H}| - \frac{\epsilon_1}{\rho}N\right) - k\left(2\epsilon_1\cdot(1+\rho^{-1})\cdot N\right)\right\}\right| < \frac{|\mathrm{C}|}{k}$$

Setting $k=\rho/\epsilon_2$, we get $|\mathrm{C}|/k = \epsilon_2 N$, and $k\cdot 2\epsilon_1(1+1/\rho) \leq 4\epsilon_2$. Thus, at least $(\rho-\epsilon_2)\cdot N$ vertices in C have degree (in the subgraph induced by T) of at least $2\rho N + 2|\mathrm{H}| - \frac{2\epsilon_1}{\rho} - 4\epsilon_2 N$. Since $\frac{2\epsilon_1}{\rho} = \frac{2\epsilon_2^2}{\rho}$, $\epsilon_2 < \frac{\epsilon}{\rho}$, and $\epsilon < \rho^2$, we have that α, as defined in the lemma, satisfies

$$\alpha \;\geq\; 2\rho + \frac{2|\mathrm{H}|}{N} - 6\epsilon_2 \;=\; 2\rho + \frac{2(|\mathrm{T}|-(|\mathrm{R}|+|\mathrm{C}|))}{N} - 6\epsilon_2 \;\geq\; \frac{2|\mathrm{T}|}{N} - 8\epsilon_2 \qquad (10)$$

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 $\deg_{\mathrm{T}}(v) \geq (\alpha - 2\epsilon_2) \cdot N$, we have by the lemma's hypothesis $|\mathrm{Z}| \geq (\rho - 3\epsilon_2) \cdot N$. By Eq. (10) we also have, for each $v \in \mathrm{Z}$,

$$\begin{aligned} \deg_{\widetilde{\mathrm{C}}}(v) &\geq \deg_{\mathrm{T}}(v) - 2 \cdot |\mathrm{T} \setminus \widetilde{\mathrm{C}}| \\ &\geq [(2|\mathrm{T}| - 8\epsilon_2 N) - 2\epsilon_2 N] - 2\,[|\mathrm{T}| - \rho \cdot N] \\ &= 2(\rho - 5\epsilon_2) \cdot N \end{aligned}$$

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

$$\begin{aligned} \sum_{v \in \widetilde{\mathrm{C}}} \deg_{\widetilde{\mathrm{C}}}(v) &\geq \sum_{v \in \mathrm{Z}} \deg_{\widetilde{\mathrm{C}}}(v) \\ &\geq 2 \cdot |\mathrm{Z}| \cdot ((\rho - 5\epsilon_2) \cdot N) \\ &\geq 2 \cdot (\rho - 3\epsilon_2) \cdot (\rho - 5\epsilon_2) \cdot N^2 \\ &> 2 \cdot (\rho^2 - 8\epsilon_2 \rho) \cdot N^2 \\ &= 2|\widetilde{\mathrm{C}}|^2 - 16\epsilon_2 \rho \cdot N^2 \end{aligned}$$

It follows that $\widetilde{\mathrm{C}}$ is $8\epsilon_2\rho$-close to being a clique and so the subgraph induced by it satisfies the claim of the lemma. ■

For a fixed U' (of cardinality $\frac{\rho}{2} \cdot t$), let $\mathrm{T} = \mathrm{T}(\mathrm{U}')$ ($\stackrel{\text{def}}{=} \cap_{u \in \mathrm{U}'} \Gamma(u)$). We first prove that $\deg_{\mathrm{W}(\mathrm{U}')}(\cdot)$ $(= \hat{d}(\cdot))$ provides a good estimate of $\deg_{\mathrm{T}}(\cdot)$ (where recall that $\mathrm{W}'(\mathrm{U}) = \mathrm{W} \cap \mathrm{T}$). For $\epsilon_2 \in [0,1]$, we say that a set W is **ϵ_2-representative** (with respect to T), if for all but $\epsilon_2 N$ of the vertices, $v \in \mathrm{V}(\mathrm{G})$,

$$\left| \frac{\deg_{\mathrm{W} \cap \mathrm{T}}(v)}{r} - \frac{\deg_{\mathrm{T}}(v)}{N} \right| < \epsilon_2 \tag{11}$$

where for set of vertices Q, $\deg_Q(v)$ is the degree of v in the subgraph of G induced by Q.

Claim 7.5 *Let $r = \Omega(\frac{\log(1/\epsilon_2\delta)}{\epsilon_2^2 \rho})$, where $\epsilon_2 \in [0,1]$. 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 $1 - \frac{\delta}{4}$, the set* W *is ϵ_2-representative of* T.

Proof: By applying a multiplicative Chernoff Bound, we get

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

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

$$\mathrm{Pr}_{\mathrm{W}'}\left[\left|\frac{\deg_{\mathrm{W}'}(v)}{r} - \frac{\deg_{\mathrm{T}}(v)}{N}\right| \geq \epsilon_2\right] \;=\; \exp(-\Omega(\epsilon_2^2 \rho r)) \;<\; \frac{\epsilon_2 \delta}{8}$$

Applying Markov's Inequality, the claim follows. ■

As a corollary to Lemma 7.4 we have,

Corollary 7.6 *Let* C *be a ρN-Clique in* G*, let $\epsilon_2 < \epsilon/\rho$ and $\epsilon_1 = \epsilon_2^2$. Suppose that $\mathrm{U}' \subset \mathrm{C}$ is ϵ_1-clique-representative with respect to* C*, and that* W *is ϵ_2-representative with respect to $\mathrm{T}(\mathrm{U}')$. Then, the set of ρN vertices of highest $\hat{d}(\cdot)$ value (in $\mathrm{T}(\mathrm{U}') \supseteq \mathrm{C}$) is $8\epsilon_2\rho$-close to being a ρN-Clique. Recall that $\hat{d}(v) \stackrel{\text{def}}{=} 2 \cdot |\Gamma(v) \cap \mathrm{W}(\mathrm{U}')| = d_{\mathrm{W}(\mathrm{U}')}(v)$, and that $\mathrm{W}(\mathrm{U}') = \mathrm{W} \cap \mathrm{T}(\mathrm{U}')$.*

Proof (of Corollary 7.6). 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 7.4 (i.e., $\deg_{\mathrm{T}}(v) \geq \alpha N$ for at least $(\rho - \epsilon_2)N$ vertices in T). By the hypothesis that W is ϵ_2-representative of T, it follows that at least $(\rho - 2\epsilon_2)N$ of the vertices v of T satisfy

$$\frac{1}{r}\hat{d}(v) \;\geq\; \frac{1}{N}\deg_{\mathrm{T}}(v) - \epsilon_2 \;\geq\; \alpha - \epsilon_2$$

Since $\widetilde{\mathrm{C}}$ contains vertices with highest $\hat{d}(\cdot)$ value, it must contain at least $(\rho - 2\epsilon_2)N$ vertices of $\hat{d}(\cdot)$ value at least $(\alpha - \epsilon_2)r$. Using the hypothesis regarding W, we conclude that $\widetilde{\mathrm{C}}$ contains at least $(\rho - 3\epsilon_2)N$ vertices of $\deg_T(\cdot)$ value at least $(\alpha - 2\epsilon_2)N$. Using the hypothesis that U′ is ϵ_1-clique-representative with respect to C, we may now invoke Lemma 7.4 and the corollary follows. ■

The correctness of the Approximate-Clique Finding Algorithm follows from Claims 7.3 and 7.5, and Corollary 7.6, where $\epsilon_2 = \epsilon/(8\rho)$. Note that W need be ϵ_2-representative only with respect to T(U′) where U′ is ϵ_1-clique-representative with respect to C.

7.2 The Clique-Degree Tester

Given the correctness of the Approximate-Clique Finding algorithm, if we could sample from each C(U′) to tests whether it is close to being a clique, then we would obtain a testing algorithm for $\mathcal{C}_\rho$. Namely, if $\mathrm{G} \in \mathcal{C}_\rho$, then from the previous subsection we know that with high probability over the choice of U and W, one of the sets C(U′) is close to being a clique, and if G is far from $\mathcal{C}_\rho$, then every set of size ρN, and in particular every C(U′), is far from being a clique. What we would like to do is test each C(U′) without actually first constructing it, that is without ordering all (the $\Omega(\rho N)$) vertices in T(U′). To this end, we uniformly choose an additional set of vertices, S, and essentially run the Approximate-Clique Finding algorithm on S. Namely, for every subset U′ of U (of size $\frac{\rho}{2}|\mathrm{U}|$) we let S(U′) be the subset of vertices in S that neighbor every vertex in U′ (so that $\mathrm{S}(\mathrm{U}') = \mathrm{S} \cap \mathrm{T}(\mathrm{U}')$). We then order the vertices in S(U′) according to the number of neighbors they have in W(U′). Finally we check whether for some U′ the first $\rho|\mathrm{S}|$ vertices in S(U′) (according the the above order) are close to being a clique. The resulting testing algorithm, called the Clique-Degree Tester, is described in Figure 3.

The correctness of the Clique-Degree Tester follows by two observations: (1) with high probability there exist an iteration where the sets U and W are as required in Corollary 7.6; and (2) the set S is a “good” sample of T(U′). We start by formulating the second observation in the lemma below. We note that when the lemma is applied, X is set to be T(U′), and the order on X is as determined by $\hat{d}(\cdot)$ (which is computed based on the number of neighbors each vertex has in W(U′) — see Figure 3).

Lemma 7.7 *Let* X *be any subset of* V*, and assume* $|\mathrm{X}| \geq (\rho - \epsilon/40)N$*. Consider a fixed ordering* $x_1, \ldots, x_{|X|}$ *of the vertices in* X*, and let* X′ *be the first* $\min(\rho N, |\mathrm{X}|)$ *vertices in* X *according to the above ordering. Let* $\mathrm{S} = \{s_1, ..., s_m\}$ *be a uniformly selected set in* V *of size* $m = \Omega(\frac{t+\log(1/\delta)}{\epsilon^2})$*, and let* $\mathrm{S}' \subseteq \mathrm{S}$ *be the first* $\min(\lfloor \rho m \rfloor, |\mathrm{S} \cap \mathrm{X}|)$ *vertices in* $\mathrm{S} \cap \mathrm{X}$ *according to the order defined on* X*. Then*

$$\Pr_{\mathrm{S}}\left[\left|\frac{|\{(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{S}', \mathrm{S}')\}|}{m/2} - \frac{|\mathrm{E}(\mathrm{X}', \mathrm{X}')|}{N^2}\right| > \frac{\epsilon}{3}\right] \;<\; \frac{\delta}{8} \cdot 2^{-t}$$

Before proving Lemma 7.7 we state a simple claim that follows directly from an additive Chernoff bound.

Clique-Degree Tester

Let $t = \Theta\left(\frac{\log(1/(\epsilon\delta))\cdot\rho}{\epsilon^2}\right)$, $r = \Theta(\frac{\log(1/(\epsilon\delta))\cdot\rho}{\epsilon^2})$, *and* $m = \Theta(\frac{t+\log(1/\delta)}{\epsilon^2})$.

1. *Uniformly select three independent samples,* U, W *and* $\mathrm{S} = \{s_1, ..., s_m\}$, *of sizes* t, r *and* m, *respectively.*
2. *For each* $\mathrm{U}' \subset \mathrm{U}$ *of cardinality* $\frac{\rho}{2}\cdot t$, *perform the following steps:*
   (a) *Let* $\mathrm{S}(\mathrm{U}') \subseteq \mathrm{S}$ *be the subset of vertices which neighbor all vertices in* U' *(i.e.,* $\mathrm{S}(\mathrm{U}') \stackrel{\rm def}{=} \{v \in \mathrm{S} : \Gamma(v) \supseteq \mathrm{U}'\} = \mathrm{S} \cap (\bigcap_{u\in\mathrm{U}'}\Gamma(u))$*).*
   (b) *Let* $\mathrm{W}(\mathrm{U}') \subseteq \mathrm{W}$ *be the subset of vertices which neighbor all vertices in* U'
   (i.e., $\mathrm{W}(\mathrm{U}') \stackrel{\rm def}{=} \{v \in \mathrm{W} : \Gamma(v) \supseteq \mathrm{U}'\} = \mathrm{W} \cap (\bigcap_{u\in\mathrm{U}'}\Gamma(u))$*).*
   (c) *For each* $v \in \mathrm{S}(\mathrm{U}')$, *compute* $\hat{d}(v) \stackrel{\rm def}{=} 2\cdot|\Gamma(v)\cap\mathrm{W}(\mathrm{U}')|$. *Let* $\widehat{\mathrm{C}}(\mathrm{U}') \subseteq \mathrm{S}(\mathrm{U}')$ *be the set of* $\lfloor\rho m\rfloor$ *vertices of the highest* $\hat{d}(\cdot)$ *value in* $\mathrm{S}(\mathrm{U}')$. *(Ties are broken by lexicographic order, and in case* $|\mathrm{S}(\mathrm{U}')| < \lfloor\rho m\rfloor$ *we let* $\widehat{\mathrm{C}}(\mathrm{U}') \stackrel{\rm def}{=} \mathrm{S}(\mathrm{U}')$*.)*
   (d) *If the following two conditions hold then* accept *and halt.*
   Condition (a): $|\widehat{\mathrm{C}}(\mathrm{U}')| \geq (\rho - \epsilon/80)m$.
   Condition (b): $|\{(s_{2k-1}, s_{2k}) \in (\widehat{\mathrm{C}}(\mathrm{U}') \times \widehat{\mathrm{C}}(\mathrm{U}')) \setminus \mathrm{E}(\mathrm{G})\}| \leq \frac{2\epsilon}{3}\cdot\frac{m}{2}$.
3. *If none of these iterations made the algorithm accept* G *then it halts and* rejects G.

Figure 3: Clique-Degree Tester

Claim 7.8 *Let* $\mathrm{S} = \{s_1, ..., s_m\}$ *be a uniformly selected set of vertices of size* $m \geq \frac{t+\log(32/\delta)}{2\epsilon_3^2}$, *where* $\epsilon_3 \in [0, 1]$. *Then for any fixed set of vertices* X, $\mathrm{Pr}_{\mathrm{S}}\left[\left|\frac{|\mathrm{S}\cap\mathrm{X}|}{m} - \frac{|\mathrm{X}|}{N}\right| > \epsilon_3\right] < \frac{\delta}{16}\cdot 2^{-t}$.

Proof of Lemma 7.7. Let $\epsilon_3 \stackrel{\rm def}{=} \epsilon/40$, and let X'' be the first $(\rho-\epsilon_3)N$ vertices in X. (Throughout the proof, whenever we refer to a number of vertices such that the number is not an integer, we mean the floor of that number.) By Claim 7.8,

$$\mathrm{Pr}_{\mathrm{S}}\left[\left|\frac{|\mathrm{S}\cap\mathrm{X}''|}{m} - (\rho-\epsilon_3)\right| > \epsilon_3\right] < \frac{\delta}{16}\cdot 2^{-t}$$

Thus, assume from now on that S is such that:

1. $|\mathrm{S}\cap\mathrm{X}''| \leq \rho m$, from which it follows (by definition of S') that $\mathrm{S}'\cap\mathrm{X}'' = \mathrm{S}\cap\mathrm{X}''$;
2. $|\mathrm{S}\cap\mathrm{X}''| \geq (\rho - 2\epsilon_3)m$, from which it follows that $|\mathrm{S}'\cap\mathrm{X}''| \geq (\rho-2\epsilon_3)m$.

Next observe that

$$\begin{aligned}\{(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{S}', \mathrm{S}')\} &= \{(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{S}'\cap\mathrm{X}'', \mathrm{S}'\cap\mathrm{X}'')\} \\ &\bigcup \{(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{S}'\setminus\mathrm{X}'', \mathrm{S}')\}\end{aligned} \tag{12}$$

(Recall that for two sets of vertices A and B, we let $\mathrm{E}(\mathrm{A},\mathrm{B})$ denote the set of edges with one end point in A and the other in B). By Item (1) above, and a Chernoff bound to get that

$$\begin{aligned}&\mathrm{Pr}_{\mathrm{S}}\left[\left|\frac{|\{(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{S}'\cap\mathrm{X}'', \mathrm{S}'\cap\mathrm{X}'')\}|}{m/2} - \frac{|\mathrm{E}(\mathrm{X}'', \mathrm{X}'')|}{N^2}\right| > \frac{\epsilon}{9}\right] \\ &= \mathrm{Pr}_{\mathrm{S}}\left[\left|\frac{|\{(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{S}\cap\mathrm{X}'', \mathrm{S}\cap\mathrm{X}'')\}|}{m/2} - \frac{|\mathrm{E}(\mathrm{X}'', \mathrm{X}'')|}{N^2}\right| > \frac{\epsilon}{9}\right] < \frac{\delta}{16}\cdot 2^{-t}\end{aligned} \tag{13}$$

By definition of X$'$ and X$''$, we have that

$$\frac{|\mathrm{E}(\mathrm{X}',\mathrm{X}')|}{N^2} - \frac{|\mathrm{E}(\mathrm{X}'',\mathrm{X}'')|}{N^2} \;\leq\; 2\epsilon_3\rho \;<\; \frac{\epsilon}{9} \tag{14}$$

By Item (2) above, we know that $|\mathrm{S}' \setminus \mathrm{X}''| \leq 2\epsilon_3 m$, and so

$$\begin{aligned} &\frac{|\{(s_{2k-1},s_{2k}) \in \mathrm{E}(\mathrm{S}'\setminus\mathrm{X}'',\mathrm{S}')\}|}{m/2} \\ &\qquad\leq\; \frac{|\{s_k \in \mathrm{S}'\setminus\mathrm{X}''\}|}{m/2} \;\leq\; 4\epsilon_3 \;<\; \frac{\epsilon}{9} \end{aligned} \tag{15}$$

Summing up the probabilities of errors and substituting the bounds of Equations (14)–(15) in Equation (13), the lemma follows. ■

Corollary 7.9 *Let $\mathcal{A}$ be the algorithm of Figure 3.*

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

The main part of Theorem 7.1 follows from Corollary 7.9. The construction of an approximate clique in time linear in N is discussed following the proof of the corollary.

Proof (of Corollary 7.9). Let $\epsilon_2 = \epsilon/(24\rho)$ and let $\epsilon_1 = \epsilon_2^2$. In proving Part (1), we let C be an arbitrary ρN-Clique in G. By Claim 7.3 (and our choice of ϵ_1 and t), with probability at least $1-\frac{\delta}{2}$, the set U contains an ϵ_1-clique-representative (with respect to 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 7.5 (and our choice of ϵ_2 and r), with probability at least $1-\frac{\delta}{4}$, the set W is ϵ_2-representative of T(U$'$). Let $\widetilde{\mathrm{C}}$ denote the set of ρN vertices of highest $\hat{d}(\cdot)$ value in T(U$'$). By Corollary 7.6 and $8\epsilon_2\rho = \epsilon/3$, the set $\widetilde{\mathrm{C}}$ is $\frac{\epsilon}{3}$-close to being a ρN-Clique. Applying Claim 7.8 to T(U$'$) with $\epsilon_3 = \frac{\epsilon}{80}$, Condition (a) of Step (4) holds with probability greater than $1-\frac{\delta}{8}$. Applying Lemma 7.7 (with X = T(U$'$) and the order on X determined by $\hat{d}(\cdot)$), Condition (b) of Step (4) also holds with probability greater than $1-\frac{\delta}{8}$ (as the fraction of missing edges is at most $\frac{\epsilon}{3}$ more than the $\frac{\epsilon}{3}$ loss of $\widetilde{\mathrm{C}}$). 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 C(U$'$), of $\min(\rho N, |\mathrm{T}(\mathrm{U}')|)$ vertices of highest $\hat{d}(\cdot)$ value in T = T(U$'$). If $|\mathrm{C}(\mathrm{U}')| < (\rho - \epsilon/40)N$ then necessarily T = C(U$'$). Applying Claim 7.8 to T, Condition (a) of Step (4) is violated with probability greater than $1 - 2^{-t}\cdot\delta$. Otherwise, we apply Lemma 7.7 (again (with X = T(U$'$) and the order on X determined by $\hat{d}(\cdot)$). Since $\mathrm{dist}(\mathrm{C}(\mathrm{U}'), C_N^\rho) \geq \epsilon$, 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 $2^{-t}\cdot\delta$. 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. ■

A Linear (in N) Time Algorithm for Finding an Approximate Clique. Given a graph G having a clique of size ρN, we can find an approximate clique (with high probability) as follows. We first run the Clique-Degree Tester (which with high probability accepts G), and record the sets U$'$ and W(U') which gave rise to the set $\widehat{\mathrm{C}}(\mathrm{U}') \subseteq \mathrm{S}(\mathrm{U}')$ that was close to being a clique. We then determine the set T(U$'$) of vertices that neighbor every vertex in U$'$ and order them according to $\hat{d}(\cdot)$ (i.e. according to the number of neighbors they have in W(U$'$). Finally, we take the first ρN

vertices according to this order. Thus the running time of this algorithm is the running time of the Clique-Degree Tester plus

$$O(|\mathrm{U}'| \cdot N + |\mathrm{W}| \cdot N) \;=\; O(\epsilon^{-2}\rho \log(1/(\epsilon\delta)) \cdot N)$$

as desired. The correctness of this algorithm follows by combining the arguments used above in the proof of Corollary 7.9. this completes the proof of Theorem 7.1. ■

8 Cut Problems

In this section we present algorithms for testing ρ-Cut and ρ-Bisection. In both cases we start by describing an algorithm that actually finds a partition that approximately maximizes the number of edges crossing the cut. In the case of the Bisection algorithm, the two sides of the partition are of equal size. We then show how these algorithms can be modified so as to *approximate* the size of the maximum cut. These modifications directly yield testing algorithms for the respective properties. Finally, we show how to achieve improved partitioning algorithms by first running the approximation algorithms. In the Bisection Subsection we also describe how the algorithms can be easily modified to deal with the case in which one seeks a bisection that *minimizes* the number of edges crossing the cut.

8.1 Testing Max-Cut

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

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

Let $\mu(\mathrm{G})$ denote the edge density of the largest cut in G. Namely, it is the maximum of $\mu(\mathrm{V}_1, \mathrm{V}_2)$ taken over all partitions $(\mathrm{V}_1, \mathrm{V}_2)$ of $\mathrm{V}(\mathrm{G})$. The main results of this subsection are summarized below.

Theorem 8.1

1. *There exists an algorithm that on input ϵ and δ and oracle access to a graph* G, *with probability at least $1 - \delta$, outputs a value $\widehat{\mu}$ such that $|\widehat{\mu} - \mu(\mathrm{G})| \leq \epsilon$. The algorithm has query complexity and running time*

$$O\left(\frac{\log^2(1/(\epsilon\delta))}{\epsilon^7}\right) \qquad \textit{and} \qquad \exp\left(O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^3}\right)\right)$$

respectively.

2. *There exists an algorithm that on input ϵ and δ, and oracle access to* G, *runs in time*

$$\exp\left(O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^3}\right)\right) + O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^2}\right) \cdot N$$

and with probability at least $1 - \delta$ outputs a partition $(\mathrm{V}_1, \mathrm{V}_2)$ of $\mathrm{V}(\mathrm{G})$ such that $\mu(\mathrm{V}_1, \mathrm{V}_2) \geq \mu(\mathrm{G}) - \epsilon$.

For any given ρ, let $\mathcal{MC}_\rho \stackrel{\rm def}{=} \{\mathrm{G} : \mu(\mathrm{G}) \geq \rho\}$ be the class of graphs with cuts of density ρ. Note that for $\rho > \frac{1}{2}$, the class $\mathcal{MC}_\rho$ is empty, and so the problem of testing ρ-Cut for $\rho > \frac{1}{2}$ is trivial. For $\rho < \frac{1}{2}$ we get the following as a corollary to Item (1) of Theorem 8.1.

Corollary 8.2 *For every constant $0 \leq \rho < \frac{1}{2}$, there exists a property testing algorithm for the class $\mathcal{MC}_\rho$ whose query complexity and running time are*

$$O\left(\frac{\log^2(1/(\epsilon\delta))}{\epsilon^7}\right) \qquad \text{and} \qquad \exp\left(O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^3}\right)\right)$$

respectively.

Although very appealing, an approximation of $\mu(\mathrm{G})$ does not *directly* translate to a tester for the class $\mathcal{MC}_\rho$, for any ρ. The following proof provides the slightly more subtle connection. In particular, the complexity of the property testing algorithm asserted in Corollary 8.2 increases as a function of the inverse of $\frac{1}{2} - \rho$ (which is assumed to be a constant). When ρ is arbitrarily close to $\frac{1}{2}$ then we need to run a variant of the Bisection testing algorithm described in Subsection 8.2.4.

Proof: Let $\gamma \stackrel{\rm def}{=} \sqrt{1-2\rho}$. The testing algorithm runs the approximation algorithm referred to in Item (1) of Theorem 8.1 with approximation parameter $\epsilon' \stackrel{\rm def}{=} \frac{\gamma\cdot\epsilon}{3}$ and confidence parameter δ (where ϵ and δ are the distance parameter and confidence parameter, respectively, of the testing algorithm). It accepts G if and only if $\widehat{\mu} \geq \rho - \epsilon'$.

If $\mu(\mathrm{G}) \geq \rho$, then by Item (1) of Theorem 8.1, G is accepted with probability $1-\delta$, as required. Conversely, if the graph is accepted with probability greater than δ, then $\mu(\mathrm{G}) \geq \rho - 2\epsilon'$. We claim that this implies that G is ϵ-close to some graph G$'$ in the class $\mathcal{MC}_\rho$. Details follow.

Let $(\mathrm{V}_1, \mathrm{V}_2)$ be a partition of V(G) such that $\mu(\mathrm{V}_1, \mathrm{V}_2) \geq \rho - 2\epsilon'$. Then necessarily, $2|\mathrm{V}_1|\cdot|\mathrm{V}_2| \geq (\rho - 2\epsilon')N^2$. If $2|\mathrm{V}_1|\cdot|\mathrm{V}_2| \geq \rho N^2$, then to obtain G$'$ we can simply add edges between vertices in V_1 and vertices in V_2 until $\mu(\mathrm{V}_1, \mathrm{V}_2) = \rho$. In this case, $\mathrm{dist}(\mathrm{G}, \mathrm{G}') \leq 2\epsilon' < \epsilon$. Otherwise, we cannot obtain G$'$ by simply adding edges between V_1 and V_2, as the total possible number of edges between V_1 and V_2 is less than ρN^2. Instead, we first move a sufficient number of vertices from the larger set among V_1 and V_2 to the smaller set so as to "make room" for the needed number of added edges. Assume without loss of generality that $|\mathrm{V}_1| < |\mathrm{V}_2|$, and consider a partition $(\mathrm{V}_1', \mathrm{V}_2')$ such that $\mathrm{V}_1' \supset \mathrm{V}_1$, and $|\mathrm{V}_1'|$ has the minimum value such that $2|\mathrm{V}_1'|\cdot|\mathrm{V}_2'| \geq \rho N^2$. It is not hard to verify (by solving two quadratic equations) that $|\mathrm{V}_1'| - |\mathrm{V}_1| \leq \frac{\epsilon}{\gamma}\cdot N$, and so

$$\mu(\mathrm{V}_1', \mathrm{V}_2') \;\geq\; \mu(\mathrm{V}_1, \mathrm{V}_2) - \frac{\epsilon'}{\gamma} \;\geq\; \rho - \epsilon$$

We can now proceed as in the first case by adding edges between V_1' and V_2' until we obtain a cut of the desired density. ■

A more natural property tester follows as in the case of ρ-Clique:

Corollary 8.3 *Let ρ be any non-negative constant smaller than $\frac{1}{2}$. Let $m = \mathrm{poly}(\epsilon^{-1}\log(1/\delta))$ 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* $\mathrm{G} \in \mathcal{MC}_\rho$ *then* $\mathrm{Pr}_\mathrm{R}[\mu(\mathrm{G}_\mathrm{R}) > \rho - \frac{\epsilon}{2}] > 1 - \delta$.
- *if* $\mathrm{dist}(\mathrm{G}, \mathcal{MC}_\rho) > \epsilon$ *then* $\mathrm{Pr}_\mathrm{R}[\mu(\mathrm{G}_\mathrm{R}) \leq \rho - \frac{\epsilon}{2}] > 1 - \delta$.

Our algorithms can be easily generalized to approximate and test Max-k-way-Cut for $k > 2$ (see Subsection 8.1.4). Furthermore, since maximizing the density of cut edges effectively minimizes the density of edges inside the different components 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 $\mu(\mathrm{G})-\frac{3}{4}\epsilon$. This algorithm runs in time $\exp\left(\mathrm{poly}\left(\frac{\log(1/\delta)}{\epsilon}\right)\right)\cdot N^2$ and is the basis for the approximation algorithm of Item (1) of Theorem 8.1. The algorithm claimed in Item (2) follows by combining the two algorithms. The extension to k-way cuts is presented in Subsection 8.1.4.

8.1.1 A Preliminary Graph Partitioning Algorithm

Let $\ell = \lceil \frac{4}{\epsilon} \rceil$, 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 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 that defines a maximum cut). Let $v \in \mathrm{H}_1$ and assume 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,

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

Furthermore,

$$\mu\left(\mathrm{H}_1 \setminus \{v\}, \mathrm{H}_2 \cup \{v\}\right) \;-\; \mu(\mathrm{H}_1, \mathrm{H}_2) \;=\; \frac{2\cdot(|\Gamma(v) \cap \mathrm{H}_1| - |\Gamma(v) \cap \mathrm{H}_2|)}{N^2}\,. \tag{16}$$

THE PLAN. Taking this observation one step further, we next show how we can define a new partition of V based on some (little) information concerning $(\mathrm{H}_1, \mathrm{H}_2)$, where we move $\Theta(\epsilon N)$ vertices between the components 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). The oracle-aided algorithm will work in $O(1/\epsilon)$ stages. It will be viewed as starting from a partition corresponding to a maximum cut and moving $O(\epsilon N)$ vertices in each stage. The total decrease in the size of the cut will hence be bounded by $O((1/\epsilon)\cdot\epsilon^2 N) = O(\epsilon N)$. 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 $\frac{N}{\ell}$, (where we assume for simplicity that ℓ divides N), let $\mathrm{W} \stackrel{\text{def}}{=} \mathrm{V} \setminus \mathrm{X}$, and let $(\mathrm{W}_1, \mathrm{W}_2)$ be the partition of W induced by $(\mathrm{H}_1, \mathrm{H}_2)$.

That is, $W_1 \stackrel{\text{def}}{=} H_1 \cap W$, and $W_2 \stackrel{\text{def}}{=} H_2 \cap W$. Recall that (H_1, H_2) is some fixed partition of $V(G)$, and that in particular, we may consider a partition which defines a maximum cut. Assume we knew for every vertex $x \in X$ how many neighbors it has on each side of the partition (W_1, W_2). In such a case, define X_{UB} to be the set of *unbalanced* vertices in X with respect to (W_1, 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 $X_B = X \setminus X_{UB}$ to be the set of *balanced* vertices with respect to (W_1, W_2).

Assume we partition X into (X_1, 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: $(H'_1, H'_2) = (W_1 \cup X_1, W_2 \cup X_2)$, which differs from (H_1, H_2) only in the placement of vertices in X. Then the difference between $\mu(H'_1, H'_2)$ and $\mu(H_1, H_2)$, is only due to the change in the number of edges between vertices in X and vertices in W, 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 X_{UB} and vertices in W could not have decreased. By definition of X_B, the arbitrary placement of these vertices decreased the number of cut edges between X_B and W by at most $|X_B| \cdot 2 \cdot \frac{1}{8}\epsilon N \leq \frac{\epsilon}{4\ell} N^2$, and the number of cut edges between pairs of vertices in X decreased by at most $|X|^2 = \frac{1}{\ell^2} N^2 \leq \frac{\epsilon}{4\ell} N^2$.

Now, let X be V^1 (i.e. the first N/ℓ vertices in lexicographical order), let (H_1, H_2) define a maximum cut, and let the partition resulting from the process defined above be denoted by (H_1^1, H_2^1). Assume we continue iteratively, where in stage i we perform the above partitioning process for V^i, given the partition (H_1^{i-1}, 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 $V \setminus V^i$ induced by (H_1^{i-1}, H_2^{i-1}). Then we can apply the same argument used above to each pair of consecutive partitions (H_1^i, H_2^i) and (H_1^{i+1}, H_2^{i+1}), and get that $\mu\left(H_1^\ell, H_2^\ell\right)$ is smaller than $\mu(H_1, H_2) = \mu(G)$ by no more than and additive factor of $\ell \cdot \frac{\epsilon}{2\ell} = \frac{\epsilon}{2}$.

Graph Partitioning Algorithm (for Max-Cut)

1. *Choose* $\ell = \lceil \frac{4}{\epsilon} \rceil$ *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* $\overline{U} = \langle U^1, \ldots, U^\ell \rangle$.
2. *For each sequence of partitions* $\pi(\overline{U}) = \left\langle (U_1^1, U_2^1), \ldots, (U_1^\ell, U_2^\ell) \right\rangle$ *(where for each* i, (U_1^i, U_2^i) *is a partition of* U^i*) 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* $\left|\Gamma(v) \cap U_1^i\right| \geq \left|\Gamma(v) \cap U_2^i\right|$ *then put* v *in* V_2^i.
      ii. *Otherwise put* v *in* V_1^i.
   (b) *Let* $V_1^{\pi(\overline{U})} = \bigcup_{i=1}^{\ell} V_1^i$, *and let* $V_2^{\pi(\overline{U})} = \bigcup_{i=1}^{\ell} V_2^i$.
3. *Among all partitions* $\left(V_1^{\pi(\overline{U})}, V_2^{\pi(\overline{U})}\right)$, *created in Step (2), let* $\left(V_1^{\widetilde{\pi(\overline{U})}}, V_2^{\widetilde{\pi(\overline{U})}}\right)$ *be the one which defines the largest cut, and output it.*

Figure 4: Graph Partitioning Algorithm for Max-Cut

The Actual Algorithm. The graph partitioning algorithm, depicted in Figure 4, approximately implements the iterative procedure described above, starting from a partition (H_1^0, H_2^0) which defines a maximum cut. Clearly, we do not have a clue as to what (H_1^0, 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 (W_1^0, W_2^0) of $W^0 \stackrel{\text{def}}{=} V \setminus V^1$ induced by this partition. However, we can approximate the number of neighbors v has on each side of (W_1^0, W_2^0) by sampling. Namely, if we uniformly choose a set of vertices U^1 of size $t = \text{poly}\left(\frac{\log(1/\delta)}{\epsilon}\right)$ in W^0, then (as we later prove formally), with high probability over the choice of U^1 *there exists* a partition (U_1^1, U_2^1) of U^1, which is *representative* with respect to (W_1^0, 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 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 as in (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 4. In the following lemma we formalize the intuition given previously as to why the partitioning algorithm works.

Lemma 8.4 *Let* (H_1, H_2) *be a fixed partition of* $V(G)$*. Then with probability at least* $1 - \delta/2$ *over the choice of* $\overline{U} = \langle U^1 \ldots U^\ell \rangle$*, there exists a sequence of partitions* $\pi(\overline{U})$*, such that* $\mu(V_1^{\pi(\overline{U})}, V_2^{\pi(\overline{U})}) \geq \mu(H_1, H_2) - \frac{3}{4}\epsilon$*.*

Proof: For a given sequence of partitions $\pi(\overline{U})$, we consider the following $\ell + 1$ *hybrid* partitions. The hybrid (H_1^0, H_2^0) is simply (H_1, H_2). The i^{th} hybrid partition, (H_1^i, H_2^i), has the vertices in $V^{i+1}, \ldots, V^\ell$ partitioned as in (H_1, H_2) and the vertices in $V^1, \ldots, V^i$ as placed by the algorithm. More precisely, 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 $j \in \{1, 2\}$,

$$V_j^i \stackrel{\text{def}}{=} V_j^{\pi(\overline{U})} \cap W^i, \quad W_j^{i-1} \stackrel{\text{def}}{=} H_j^{i-1} \cap W^{i-1}, \text{ and } W^{i-1} \stackrel{\text{def}}{=} V \setminus V^i$$

Note, that in particular, (H_1^ℓ, H_2^ℓ) is the partition $\left(V_1^{\pi(\overline{U})}, V_2^{\pi(\overline{U})}\right)$. 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 $U^1, \ldots, U^i$ and their partitions, *but not* by the choice nor the partitions of $U^{i+1}, \ldots, U^\ell$.

We shall show that for every $1 \leq i \leq \ell$, 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

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

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 a fraction of $\frac{1}{8}\epsilon$ of the vertices v in V^i the following holds:

$$\text{For each } j \in \{1,2\}, \qquad \frac{\left|\Gamma(v) \cap \mathrm{U}_j^i\right|}{t} = \frac{\left|\Gamma(v) \cap \mathrm{W}_j^{i-1}\right|}{N} \pm \frac{\epsilon}{32} \tag{17}$$

(Recall that $a = b \pm c$ is a shorthand for $b - c \leq a \leq b + c$.) 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

$$\text{for } j, j' \in \{1,2\},\ \ j \neq j' \qquad \left|\Gamma(v) \cap \mathrm{W}_j^{i-1}\right| \geq \left|\Gamma(v) \cap \mathrm{W}_{j'}^{i-1}\right| + \frac{1}{8}\epsilon N \ . \tag{18}$$

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 (17) is satisfied, then $\left|\Gamma(v) \cap \mathrm{U}_j^i\right| \geq \left|\Gamma(v) \cap \mathrm{U}_{j'}^i\right| + \frac{1}{16}\epsilon t$. 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 (17) 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 V^i and vertices in W^{i-1}, and between pairs of vertices in V^i. In particular:

1. The number of cut edges between unbalanced vertices in V^i for which Equation (17) is satisfied and vertices in W^{i-1} can not decrease.

2. The number of cut edges between unbalanced vertices in V^i for which Equation (17) is *not* satisfied and vertices in W^{i-1} decreases by at most $\frac{\epsilon}{8} \cdot |\mathrm{V}^i| \cdot 2N \leq \frac{\epsilon}{4\ell} N^2$.

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

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

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 $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$ and V^i). We first fix a vertex $v \in \mathrm{V}^i$. Let $\mathrm{U}^i = \{u_1, \ldots, u_t\}$. Recall that U^i is chosen uniformly in $\mathrm{W}^{i-1} \stackrel{\text{def}}{=} \mathrm{V} \setminus \mathrm{V}^i$. For $j \in \{1,2\}$, and for $1 \le k \le t$, define a 0/1 random variable, ξ_j^k, which is 1 if u_k is a neighbor of v and $u_k \in \mathrm{W}_j^{i-1}$, 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 $(= \mathrm{U}^i \cap \mathrm{W}_j^{i-1})$ and the probability that $\xi_j^k = 1$ is $\frac{1}{N}\left|\Gamma(v) \cap \mathrm{W}_j^{i-1}\right|$. By an additive Chernoff bound (see Appendix B), and our choice of t, for each $j \in \{1, \ldots, k\}$,

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

By Markov's inequality (see Appendix B), 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 (17) holds (for that j), and thus with probability at least $1 - \frac{\delta}{2\ell}$, U^i is good as required. ■

Applying Lemma 8.4 to a maximum cut of G, we get

Corollary 8.5 *With probability at least* $1 - \frac{\delta}{2}$ *over the choice of* $\overline{\mathrm{U}}$ *we have,* $\mu(\widetilde{\mathrm{V}}_1^{\pi(\overline{\mathrm{U}})}, \widetilde{\mathrm{V}}_2^{\pi(\overline{\mathrm{U}})}) \ge \mu(\mathrm{G}) - \frac{3}{4}\epsilon$, *where* $(\widetilde{\mathrm{V}}_1^{\pi(\overline{\mathrm{U}})}, \widetilde{\mathrm{V}}_2^{\pi(\overline{\mathrm{U}})})$ *is as defined in Step (3) of the Graph Partitioning Algorithm (Figure 4).*

8.1.2 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 vertices 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 sequences of partitions $\pi(\overline{\mathrm{U}})$). Instead, we view $\mathrm{S} = \{s_1, \ldots, s_m\}$ as a multiset of $m/2$ (ordered) pairs of vertices, (i.e., $\{(s_1, s_2), \ldots, (s_{m-1}, s_m)\}$) and we choose the cut that maximizes the number of such pairs that are edges in the cut. This is done for technical reasons since it ensures a certain independence in the probabilistic analysis. The exact code is given in Figure 5.

Lemma 8.6 *For any fixed* $\overline{\mathrm{U}}$, *with probability at least* $1 - \delta/2$ *over the choice of* S, $\widehat{\mu}\left(\widetilde{\mathrm{S}}_1^{\pi(\overline{\mathrm{U}})}, \widetilde{\mathrm{S}}_2^{\pi(\overline{\mathrm{U}})}\right) = \mu\left(\widetilde{\mathrm{V}}_1^{\pi(\overline{\mathrm{U}})}, \widetilde{\mathrm{V}}_2^{\pi(\overline{\mathrm{U}})}\right) \pm \frac{1}{4}\epsilon$, *where* $\left(\widetilde{\mathrm{S}}_1^{\pi(\overline{\mathrm{U}})}, \widetilde{\mathrm{S}}_2^{\pi(\overline{\mathrm{U}})}\right)$ *and* $\widehat{\mu}(\cdot,\cdot)$ *are as defined in step 4 of the Max-Cut approximation algorithm.*

Proof: Consider first a particular sequence of partitions, $\pi(\overline{\mathrm{U}})$. The key observation is that for every $s \in \mathrm{S}$, and for $j \in \{1,2\}$, $s \in \mathrm{S}_j^{\pi(\overline{\mathrm{U}})}$ if and only if $s \in \mathrm{V}_j^{\pi(\overline{\mathrm{U}})}$. Thus for each sequence of partitions $\pi(\overline{\mathrm{U}})$ we are effectively sampling from $\left(\mathrm{V}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{V}_2^{\pi(\overline{\mathrm{U}})}\right)$. 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 $1 \le k \le m/2$, let ξ_k be a 0/1 random variable which is 1 if $(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{G})$, and for $j \ne j'$, $s_{2k-1} \in \mathrm{S}_j^{\pi(\overline{\mathrm{U}})}$ and $s_{2k} \in \mathrm{S}_{j'}^{\pi(\overline{\mathrm{U}})}$. Then, by definition, $\widehat{\mu}\left(\mathrm{S}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{S}_2^{\pi(\overline{\mathrm{U}})}\right) = \frac{2}{m}\sum_k^{m/2} \xi_k$,

Max-Cut Approximation Algorithm

1. *As Step (1) of Figure 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* $\mathrm{S}^i \stackrel{\rm def}{=} \mathrm{V}^i \cap \mathrm{S}$.

3. *Analogously to Step (2) of Figure 4, for each of the sequences of partitions* $\pi(\overline{\mathrm{U}}) = \left\langle (\mathrm{U}_1^1, \mathrm{U}_2^1), \ldots, (\mathrm{U}_1^\ell, \mathrm{U}_2^\ell) \right\rangle$, *partition each* S^i *into two disjoint sets* S_1^i *and* S_2^i, *and let* $\mathrm{S}_j^{\pi(\overline{\mathrm{U}})} = \bigcup_{i=1}^{\ell} \mathrm{S}_j^i$ *(for* $j = 1, 2$*).*

4. *For each partition* $\left(\mathrm{S}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{S}_2^{\pi(\overline{\mathrm{U}})}\right)$, *compute the fraction of cut edges between pairs of vertices* (s_{2k-1}, s_{2k}). *More precisely, define*

$$\widehat{\mu}\left(\mathrm{S}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{S}_2^{\pi(\overline{\mathrm{U}})}\right) \stackrel{\rm def}{=} \frac{\left|\left\{(s_{2k-1}, s_{2k}) \in \mathrm{E}\left((\mathrm{S}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{S}_2^{\pi(\overline{\mathrm{U}})}\right) \cup \mathrm{E}\left(\mathrm{S}_2^{\pi(\overline{\mathrm{U}})}, \mathrm{S}_1^{\pi(\overline{\mathrm{U}})}\right)\right\}\right|}{m/2}$$

Let $\left(\mathrm{S}_1^{\widetilde{\pi}(\overline{\mathrm{U}})}, \mathrm{S}_2^{\widetilde{\pi}(\overline{\mathrm{U}})}\right)$ *be a partition for which this fraction is maximized, and output* $\widehat{\mu}\left(\mathrm{S}_1^{\widetilde{\pi}(\overline{\mathrm{U}})}, \mathrm{S}_2^{\widetilde{\pi}(\overline{\mathrm{U}})}\right)$.

Figure 5: Max-Cut Approximation Algorithm

and the probability that $\xi_k = 1$ is $\mu\left(\mathrm{V}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{V}_2^{\pi(\overline{\mathrm{U}})}\right)$. Hence, by an additive Chernoff bound and our choice of m,

$$\begin{aligned} \Pr_{\mathrm{S}}\left[\left|\widehat{\mu}\left(\mathrm{S}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{S}_2^{\pi(\overline{\mathrm{U}})}\right) - \mu\left(\mathrm{V}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{V}_2^{\pi(\overline{\mathrm{U}})}\right)\right| > \frac{1}{8}\epsilon\right] &= \exp(-\Omega(\epsilon^2 m)) \\ &= O\left(\delta \cdot 2^{-\ell \cdot t}\right) \end{aligned} \tag{19}$$

Since there are $2^{\ell \cdot t}$ sequences of partitions of $\overline{\mathrm{U}}$, with probability at least $1 - \delta/2$, for every sequence of partitions $\pi(\overline{\mathrm{U}})$, $\widehat{\mu}\left(\mathrm{S}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{S}_2^{\pi(\overline{\mathrm{U}})}\right) = \mu\left(\mathrm{V}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{V}_2^{\pi(\overline{\mathrm{U}})}\right) \pm \frac{1}{8}\epsilon$, and hence $\widehat{\mu}\left(\mathrm{S}_1^{\widetilde{\pi}(\overline{\mathrm{U}})}, \mathrm{S}_2^{\widetilde{\pi}(\overline{\mathrm{U}})}\right) = \mu\left(\mathrm{V}_1^{\widetilde{\pi}(\overline{\mathrm{U}})}, \mathrm{V}_2^{\widetilde{\pi}(\overline{\mathrm{U}})}\right) \pm \frac{1}{4}\epsilon$, ■

Combining Corollary 8.5 and Lemma 8.6, Part (1) of Theorem 8.1 follows. Part (2) of the theorem is proved below.

8.1.3 An Improved Graph Partitioning Algorithm

The improved (in terms of running time) graph partitioning algorithm starts by invoking the Max-Cut approximation algorithm of Figure 5, and recording the sequence of sets $\overline{\mathrm{U}}$ uniformly selected in Step (1) and the sequence of partitions $\widetilde{\pi}(\overline{\mathrm{U}})$ selected in Step (4). Using this specific sequence $\widetilde{\pi}(\overline{\mathrm{U}})$, the algorithm executes a *single* iteration of Step (2) of the Graph Partitioning Algorithm of Figure 4 and obtains the partition $\left(\mathrm{V}_1^{\widetilde{\pi}(\overline{\mathrm{U}})}, \mathrm{V}_2^{\widetilde{\pi}(\overline{\mathrm{U}})}\right)$. Since this improved algorithm only partitions all vertices according to $\widetilde{\pi}(\overline{\mathrm{U}})$ and does not even compute the size of the resulting cut (as the original algorithm did), its running time is linear in N instead of quadratic. More precisely, it is $O(t \cdot N) = O(\log(1/(\epsilon\delta)/\epsilon^2)) \cdot N$. As for its correctness, by Lemma 8.4, we have that with probability at least $1 - \delta/2$ over the choice of $\overline{\mathrm{U}}$, there exists a sequence of partitions $\pi(\overline{\mathrm{U}})$, such that $\mu(\mathrm{V}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{V}_2^{\pi(\overline{\mathrm{U}})}) \ge \mu(\mathrm{G}) - \frac{3}{4}\epsilon$. From the proof of Lemma 8.6 we have that for a fixed $\overline{\mathrm{U}}$, with probability at least $1 - \delta/2$ over the choice of S, $\widehat{\mu}\left(\mathrm{S}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{S}_2^{\pi(\overline{\mathrm{U}})}\right)$ is within $\frac{\epsilon}{8}$ from $\mu\left(\mathrm{V}_1^{\pi(\overline{\mathrm{U}})}, \mathrm{V}_2^{\pi(\overline{\mathrm{U}})}\right)$

for *every* sequence of partitions $\pi(\overline{\mathrm{U}})$. It follows that with probability at least $1-\delta$, the recorded partition $\widetilde{\pi}(\overline{\mathrm{U}})$ is such that $\mu\left(\mathrm{V}_1^{\widetilde{\pi}(\overline{\mathrm{U}})}, \mathrm{V}_2^{\widetilde{\pi}(\overline{\mathrm{U}})}\right) \geq \mu(\mathrm{G})-\epsilon$, as required.

8.1.4 Generalization to k-way Cuts

For a k-way partition $(\mathrm{V}_1,\ldots,\mathrm{V}_k)$ of $\mathrm{V}(\mathrm{G})$, we denote by $\mu_k(\mathrm{V}_1,\ldots,\mathrm{V}_k)$ the edge density of the cut defined by $(\mathrm{V}_1,\ldots,\mathrm{V}_k)$. Namely,

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

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

Theorem 8.7

1. *There exists an algorithm that on input k, ϵ and δ, and oracle access to a graph* G*, with probability at least $1-\delta$, outputs a value $\widehat{\mu}_k$ such that $|\widehat{\mu}_k - \mu_k(\mathrm{G})| \leq \epsilon$. The algorithm has query complexity an running time*

$$O\left(\frac{\log^2(k/(\epsilon\delta))}{\epsilon^7}\right) \qquad \textit{and} \qquad \exp\left(O\left(\frac{\log^2(k/(\epsilon\delta))}{\epsilon^3}\right)\right)$$

respectively.

2. *There exists an algorithm that on input k, ϵ, and δ, and oracle access to* G*, runs in time*

$$\exp\left(O\left(\frac{\log^2(k/(\epsilon\delta))}{\epsilon^3}\right)\right) + O\left(\frac{\log(k/(\epsilon\delta))}{\epsilon^2}\right)\cdot N$$

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* $\mu_k(\mathrm{V}_1,\ldots,\mathrm{V}_k) \geq \mu_k(\mathrm{G})-\epsilon$.

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, $(\mathrm{U}_1^i,\ldots,\mathrm{U}_k^i)$, we partition V^i (into k disjoint sets) as follows. For each vertex $v\in\mathrm{V}^i$, and for $1\leq j\leq k$, let $d_j^i(v) = \left|\Gamma(v)\cap \mathrm{U}_j^i\right|$, and let $j_{\min} = \mathrm{argmin}_j\{d_j^i(v)\}$. Then we put v in $\mathrm{V}_{j_{\min}}^i$.

The following (minor) changes suffice for the adapting the analysis to the modified algorithm. Let $(\mathrm{H}_1^0,\ldots,\mathrm{H}_k^0)$ 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 $(\mathrm{H}_1^{i-1},\ldots,\mathrm{H}_k^{i-1})$ 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 $\mathrm{U}_j^i \stackrel{\mathrm{def}}{=} \mathrm{W}_j^{i-1}\cap\mathrm{U}^i$ where $\mathrm{W}_j^{i-1} \stackrel{\mathrm{def}}{=} \mathrm{H}_j^{i-1}\setminus\mathrm{V}^i$. We shall say that U^i is *good* with respect to $(\mathrm{W}_1^{i-1},\ldots,\mathrm{W}_k^{i-1})$ and V^i, if for all but $\frac{1}{8}\epsilon$ of the vertices v in V^i,

$$\text{For each } j\in\{1,\ldots,k\}, \qquad \frac{\left|\Gamma(v)\cap\mathrm{U}_j^i\right|}{t} = \frac{|\Gamma(v)\cap\mathrm{W}_j^{i-1}|}{N} \pm \frac{\epsilon}{32} \tag{20}$$

It follows that in order to ensure that each U^i be good (with respect to $(\mathrm{W}_1^{i-1},\ldots,\mathrm{W}_k^{i-1})$ 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 *balanced* and *unbalanced* vertices is generalized as follows. Consider a partition $(\mathrm{W}_1^{i-1},\ldots,\mathrm{W}_k^{i-1})$ of V. For vertex v and $j \in \{1,\ldots,k\}$, let $\Gamma_j(v) \stackrel{\rm def}{=} \Gamma(v) \cap \mathrm{W}_j^{i-1}$ be the subset of v's neighbors that belong to W_j^{i-1}. Let $\min(v) \stackrel{\rm def}{=} \min_j\{|\Gamma_j(v)|\}$ be the minimum size among these neighbor sets, and let

$$\mathrm{J}(v) \stackrel{\rm def}{=} \left\{ j \in \{1,\ldots,k\} : \ |\Gamma_j(v)| \leq \min(v) + \frac{\epsilon}{16}N \right\}$$

be the *balanced* (index) set of v. Note that in particular, for $k = 2$, either $|\mathrm{J}(v)| = 1$, so that v is *unbalanced*, or $|\mathrm{J}(v)| = k$ $(= 2)$, so that v is *balanced*. Consider the case in which Equation (20) holds for v. In such a case, v is placed in a component j' of the partition such that $j' \in \mathrm{J}(v)$. Namely, it is placed in a component in which its number of neighbors is not much far from the minimum $\min(v)$. The key point is that if v is moved from component j (in $(\mathrm{H}_1^{i-1},\ldots,\mathrm{H}_k^{i-1})$ to component j' (in $(\mathrm{H}_1^{i},\ldots,\mathrm{H}_k^{i})$), all edges that it has with vertices in components $j'' \neq j, j'$ remain cut edges, and only the number of cut-edges between v and vertices in components j and j' might change. Thus, when $j' \in \mathrm{J}(v)$, the number of cut edges between v 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, and cut edges that are incident to vertices for which Equation (20) does *not* hold. 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).

8.2 Testing Bisection

In this subsection 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 Subsection 8.1, let

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

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*.[17] For sake of the exposition (due to the similarity to Max-Cut), 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 8.2.4). The main result of this subsection is

Theorem 8.8

1. *There exists an algorithm that on input ϵ and δ and oracle access to a graph* G*, with probability at least $1-\delta$, outputs a value $\widehat{\mu}^{(\frac{1}{2})}$ such that $|\widehat{\mu}^{(\frac{1}{2})} - \mu^{(\frac{1}{2})}(\mathrm{G})| \leq \epsilon$. The algorithm has query complexity and running time*

$$O\left(\frac{\log^2(1/(\epsilon\delta))}{\epsilon^8}\right) \qquad \textit{and} \qquad \exp\left(O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^2}\right)\right)$$

respectively.

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

2. *There exists an algorithm that on input ϵ and δ, and oracle access to a graph* G*, 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 $\mu(\mathrm{V}_1,\mathrm{V}_2) \geq \mu^{(\frac{1}{2})}(\mathrm{G}) - \epsilon$.

Item (1) yields a property tester for the class $\mathcal{MC}_\rho^{(\frac{1}{2})} \stackrel{\mathrm{def}}{=} \{\mathrm{G} : \mu^{(\frac{1}{2})}(\mathrm{G}) \geq \rho\}$, for every $0 \leq \rho \leq 1$, with query complexity $O\left(\frac{\log^2(1/(\epsilon\delta))}{\epsilon^8}\right)$ and running time $\exp\left(O\left(\frac{\log(1/(\epsilon\delta))}{\epsilon^3}\right)\right)$. If $\rho > \frac{1}{2}$ then the tester rejects G since for $\rho > \frac{1}{2}$ the class $\mathcal{MC}_\rho^{(\frac{1}{2})}$ is empty. Otherwise the tester runs the approximation algorithm referred to in Item (1) with approximation parameter $\frac{\epsilon}{2}$ and confidence parameter δ (where ϵ and δ are the distance and confidence parameters, respectively, of the testing algorithm). The tester accepts G if an only if $\widehat{\mu}^{(\frac{1}{2})} \geq \rho - \frac{\epsilon}{2}$. If $\mathrm{G} \in \mathcal{MC}_\rho^{(\frac{1}{2})}$ (i.e., $\mu^{(\frac{1}{2})}(\mathrm{G}) \geq \rho$), then by Item (1) it is accepted with probability at least $1-\delta$. Conversely, if G is accepted with probability greater than δ, then $\mu^{(\frac{1}{2})}(\mathrm{G}) \geq \rho - \epsilon$. That is, there exists an equal partition $(\mathrm{V}_1,\mathrm{V}_2)$ of G such that $\mu(\mathrm{V}_1,\mathrm{V}_2) \geq \rho - \epsilon$. Therefore, $\mathrm{dist}(\mathrm{G},\mathcal{MC}_\rho^{(\frac{1}{2})}) \leq \epsilon$ since by adding at most ϵN^2 edges between V_1 and V_2 we can obtain a graph $\mathrm{G}' \in \mathcal{MC}_\rho^{(\frac{1}{2})}$.

A more natural property tester follows as in previous cases:

Corollary 8.9 *Let $m = \mathrm{poly}(\epsilon^{-1}\log(1/\delta))$ and let* R *be a uniformly selected set of m vertices in* $\mathrm{V}(\mathrm{G})$*. Let* G_R *be the subgraph (of* G*) induced by* R*. Then,*

- *if* $\mathrm{G} \in \mathcal{MC}_\rho^{(\frac{1}{2})}$ *then* $\Pr_\mathrm{R}[\mu^{(\frac{1}{2})}(\mathrm{G}_\mathrm{R}) > \rho - \frac{\epsilon}{2}] > 1-\delta$.
- *if* $\mathrm{dist}(\mathrm{G},\mathcal{MC}_\rho^{(\frac{1}{2})}) > \epsilon$ *then* $\Pr_\mathrm{R}[\mu^{(\frac{1}{2})}(\mathrm{G}_\mathrm{R}) \leq \rho - \frac{\epsilon}{2}] > 1-\delta$.

Our proof of Theorem 8.8 follows the outline of the proof of Theorem 8.1 (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 case of Max-Cut, it is always beneficial (and possible) to relocate a vertex so that it is on the side opposite 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. 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 $(\mathrm{H}_1,\mathrm{H}_2)$, 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 $\mathrm{W}_1 \stackrel{\mathrm{def}}{=} \mathrm{H}_1 \setminus \mathrm{X}$ and how many in $\mathrm{W}_2 \stackrel{\mathrm{def}}{=} \mathrm{H}_2 \setminus \mathrm{X}$. Further assume that we know $|\mathrm{H}_1 \cap \mathrm{X}|$ and $|\mathrm{H}_2 \cap \mathrm{X}|$. Let us relate with each vertex in X an *unbalance* value, which 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 there are $|\mathrm{H}_1 \cap \mathrm{X}|$ vertices on side 1 (and $|\mathrm{H}_2 \cap \mathrm{X}|$ on side 2) and 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 resulting partition is clearly a bisection and the size of the cut decreases by at most $|\mathrm{X}|^2 = O(\epsilon^2 N^2)$. The latter is due to the fact that for any other partition of X with $|\mathrm{H}_1 \cap \mathrm{X}|$ vertices on one side (and the rest on the other), there cannot be more cut edges between vertices in X and vertices in $\mathrm{V} \setminus \mathrm{X}$ 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.

8.2.1 A High Level Description of the Bisection Algorithm

As was done in Subsection 8.1, 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 for Max-Cut, the (oracle aided) graph bisection algorithm proceeds in $\ell = O(1/\epsilon)$ iterations where in the i^{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 $\mathcal{H}^0 = (\mathrm{H}_1^0, \mathrm{H}_2^0)$, which is a maximum bisection, the i^{th} hybrid partition $\mathcal{H}^i = (\mathrm{H}_1^i, \mathrm{H}_1^i)$ is a hybrid of the partition $(\mathrm{V}_1^i, \mathrm{V}_2^i)$ of V^i (constructed in the i^{th} iteration), and the partition $\mathcal{W}^{i-1} = (\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$ of $\mathrm{W}^{i-1} \stackrel{\text{def}}{=} \mathrm{V} \setminus \mathrm{V}^i$ induced by the $i-1$ hybrid partition, $\mathcal{H}^{i-1} = (\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1})$. 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 belonging to V^i) on each side of the new hybrid. That is, for each $i \in \{1, \ldots, \ell\}$, let

$$\beta^i \stackrel{\text{def}}{=} \frac{|\mathrm{V}^i \cap \mathrm{H}_1^{i-1}|}{N/\ell} \tag{21}$$

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 be each hybrid partition, 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:

$$\mathrm{ub}(v) \stackrel{\text{def}}{=} \frac{|\Gamma(v) \cap \mathrm{W}_2^{i-1}| - |\Gamma(v) \cap \mathrm{W}_1^{i-1}|}{N}\,. \tag{22}$$

Let $L \stackrel{\text{def}}{=} N/\ell$ (where for simplicity we assume ℓ divides N), 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 $(\mathrm{V}_1^i, \mathrm{V}_2^i)$ of V^i: $\mathrm{V}_1^i \stackrel{\text{def}}{=} \{v_1, \ldots, v_{\beta^i L}\}$, and $\mathrm{V}_2^i \stackrel{\text{def}}{=} \mathrm{V}^i \setminus \mathrm{V}_1^i$. Let $(\mathrm{H}_1^i, \mathrm{H}_2^i) \stackrel{\text{def}}{=} (\mathrm{W}_1^{i-1} \cup \mathrm{V}_1^i, \mathrm{W}_2^{i-1} \cup \mathrm{V}_2^i)$. Then the number of cut edges in $(\mathrm{H}_1^i, \mathrm{H}_2^i)$ between vertices in V^i and vertices in W^{i-1} 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 $|\mathrm{V}^i|^2 = (N/\ell)^2 = O(\frac{\epsilon}{\ell} N^2)$.

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 an $O(\epsilon)$ fraction of the vertices in V^i. Similarly to the Max-Cut partitioning algorithm, we use sample sets U^i to obtain approximations $\widehat{\mathrm{ub}}(v)$ to $\mathrm{ub}(v)$. Namely, for $v \in \mathrm{V}^i$ and for a partition $(\mathrm{U}_1^i, \mathrm{U}_2^i)$ of U^i (where we consider all such partitions), we let

$$\widehat{\mathrm{ub}}(v) \stackrel{\text{def}}{=} \frac{|\Gamma(v) \cap \mathrm{U}_1^i| - |\Gamma(v) \cap \mathrm{U}_2^i|}{t} \tag{23}$$

where t is the size of each U^i. The approximations $\widehat{\beta}^i = \beta^i \pm O(\epsilon)$ are obtained by simply trying all integer multiples of $\epsilon/8$ which sum up to $1/2$.[18] Each possible setting of $\widehat{\beta}^1, \ldots, \widehat{\beta}^\ell$ and sequence of partitions of $U^1, \ldots, U^\ell$ 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 $\widehat{\beta}^i$'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 8.10, if we put the vertices $\{\hat{v}_1, \ldots, \hat{v}_{\lfloor \widehat{\beta}^i L \rfloor})\}$ on side 1, and the vertices $\{\hat{v}_{\lfloor \widehat{\beta}^i L \rfloor + 1}, \ldots, \hat{v}_L\}$ 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 6, and its formal analysis is provided in Lemma 8.10.

Graph Bisection Algorithm

1. *Choose* $\ell = \lceil \frac{4}{\epsilon} \rceil$ *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* $V \setminus V^i$.
2. *For each sequence of partitions* $\Pi = \langle (U_1^1, U_2^1), \ldots, (U_1^\ell, U_2^\ell) \rangle$ *and for each of the* ℓ*-tuples* $\widehat{\beta} = \left(\widehat{\beta}^1, \ldots, \widehat{\beta}^\ell\right)$, *where each* $\widehat{\beta}^i \in [0,1]$ *is an integer multiple of* $\frac{\epsilon}{8}$, *and* $\sum_i \widehat{\beta}^i = \frac{1}{2}$, *construct a bisection*[a] $\left(V_1^{\Pi,\widehat{\beta}}, V_2^{\Pi,\widehat{\beta}}\right)$ *as follows:*
   (a) *For* $i = 1 \ldots \ell$ *do:*
      i. *For each* $v \in V^i$ *let* $\widehat{\mathrm{ub}}(v) \stackrel{\mathrm{def}}{=} \frac{1}{t}\left(|\Gamma(v) \cap U_2^i| - |\Gamma(v) \cap U_1^i|\right)$;
      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. $V_1^i \leftarrow \{\hat{v}_1, \ldots, \hat{v}_{\lfloor \widehat{\beta}^i L \rfloor}\}$, *and* $V_2^i \leftarrow \{\hat{v}_{\lfloor \widehat{\beta}^i L \rfloor + 1}, \ldots, \hat{v}_L\}$
   (b) *Let* $V_1^{\Pi,\widehat{\beta}} = \bigcup_{i=1}^\ell V_1^i$, *and let* $V_2^{\Pi,\widehat{\beta}} = \bigcup_{i=1}^\ell V_2^i$.
3. *Among all bisections,* $\left(V_1^{\Pi,\widehat{\beta}}, V_2^{\Pi,\widehat{\beta}}\right)$, *let* $\left(V_1^{\widetilde{\Pi},\widetilde{\beta}}, V_2^{\widetilde{\Pi},\widetilde{\beta}}\right)$ *be the one with maximum number of crossing edges.*

[a] Due to questions of integrability (i.e. $\widehat{\beta}^i L$ not being an integer) we might not get an exact bisection. However this can easily be treated by moving (at most ℓ) vertices between sides once the partition is constructed.

Figure 6: Graph Bisection algorithm

Lemma 8.10 *Let* (H_1, H_2) *be a fixed bisection of* $V(G)$. *Then, with probability at least* $1 - \delta/2$ *over the choice of* $\overline{U}$, *there exists a sequence of partitions* Π *of* $\overline{U}$, *and an* ℓ*-tuple* $\widehat{\beta}$, *such that* $\mu(V_1^{\Pi,\widehat{\beta}}, V_2^{\Pi,\widehat{\beta}}) \geq \mu(H_1, H_2) - \frac{3}{4}\epsilon$.

Proof: Let $\widehat{\beta}$ be such that for every i, $\widehat{\beta}^i = \beta^i \pm \epsilon/16$, where $\beta^i \stackrel{\mathrm{def}}{=} |H_1 \cap V^i|$. The existence of such $\widehat{\beta}^i$ follows from the resolution of the values taken by $\widehat{\beta}^i$ and the fact that all possibilities (to within

[18] This is always possible in case $1/\epsilon$ is an integer. Otherwise, we can try all integer multiples of $\epsilon'/8$ 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.

this resolution) were tried. For a fixed $\overline{\mathrm{U}}$, a fixed sequence of partitions Π of $\overline{\mathrm{U}}$, and the above choice of $\widehat{\beta}$, let the i^{th} hybrid partition $(\mathrm{H}_1^i, \mathrm{H}_2^i)$ determined by Π and $\widehat{\beta}$ be defined as follows. For $i = 0$, the partition $(\mathrm{H}_1^0, \mathrm{H}_2^0)$ equals $(\mathrm{H}_1, \mathrm{H}_2)$. For $i > 0$ and $j \in \{1, 2\}$, let $\mathrm{W}_j^{i-1} \stackrel{\text{def}}{=} \mathrm{H}_j^{i-1} \setminus \mathrm{V}^i$. As done in the Graph Bisection Algorithm (see Figure 6), let $\hat{v}_1, \ldots, \hat{v}_L$ be an ordering of the vertices in V^i according to their $\widehat{\mathrm{ub}}(\cdot)$ values. Let $\mathrm{V}_1^i \stackrel{\text{def}}{=} \{\hat{v}_1, \ldots, \hat{v}_{\lfloor \widehat{\beta}^i L \rfloor}\}$, and $\mathrm{V}_2^i \stackrel{\text{def}}{=} \{\hat{v}_{\lfloor \widehat{\beta}^i L \rfloor + 1}, \ldots, \hat{v}_L\}$. Then $\mathrm{H}_j^i \stackrel{\text{def}}{=} \mathrm{W}_j^{i-1} \cup \mathrm{V}_j^i$.

Similarly to what was observed in Lemma 8.4, for $\widehat{\beta}$ fixed as above, an i^{th} hybrid partition is actually determined by the partitions of $\mathrm{U}^1, \ldots, \mathrm{U}^i$ (which determine the $\widehat{\mathrm{ub}}(\cdot)$ values as in Equation (23)), 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

$$\mu\left(\mathrm{H}_1^i, \mathrm{H}_2^i\right) \geq \mu\left(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1}\right) - \frac{3\epsilon}{4\ell}. \tag{24}$$

By induction on i we have that the ℓ^{th} hybrid partition (which is necessarily a bisection since $\sum_i \widehat{\beta}^i = 1/2$) has a cut whose size is at most $\frac{3\epsilon}{4} N^2$ smaller than the 0^{th} 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 8.4 except that here we make the stronger quantitative requirement that for all but $\frac{\epsilon}{32}$ of the vertices v in V^i

$$\text{For each } j \in \{1, 2\}, \qquad \frac{\left|\Gamma(v) \cap \mathrm{U}_j^i\right|}{t} = \frac{\left|\Gamma(v) \cap \mathrm{W}_j^{i-1}\right|}{N} \pm \frac{\epsilon}{64} \tag{25}$$

As was shown in the proof of Lemma 8.4, 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, we shall bound the difference between the size of the cut determined by the i^{th} hybrid partition and that determined by the $(i-1)$ hybrid partition via two auxiliary hybrid partitions. 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 $(\mathrm{W}_1^{i-1}, \mathrm{W}_2^{i-1})$. Consider first the "ideal" partition $(\mathrm{V}_{1,\mathrm{id}}^i, \mathrm{V}_{2,\mathrm{id}}^i)$ of V^i, where $\mathrm{V}_{1,\mathrm{id}}^i \stackrel{\text{def}}{=} \{v_1, \ldots, v_{\beta^i L}\}$, and $\mathrm{V}_{2,\mathrm{id}}^i \stackrel{\text{def}}{=} \mathrm{V}^i \setminus \mathrm{V}_{1,\mathrm{id}}^i$. (Namely, this partition uses both the correct unbalance values and the correct β^i). Let $(\mathrm{H}_{1,\mathrm{id}}^i, \mathrm{H}_{2,\mathrm{id}}^i)$ be the corresponding ideal hybrid partition (namely, $\mathrm{H}_{j,\mathrm{id}}^i \stackrel{\text{def}}{=} \mathrm{W}_j^{i-1} \cup \mathrm{V}_{j,\mathrm{id}}^i$). As was noted previously, the size of the cut determined by this ideal hybrid partition is at most $|\mathrm{V}^i|^2 = \frac{\epsilon}{4\ell} N^2$ smaller than the cut determined by the previous hybrid partition $(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1})$.

Next consider the following "almost-ideal" partition $(\mathrm{V}_{1,\mathrm{id}'}^i, \mathrm{V}_{2,\mathrm{id}'}^i)$ of V^i, where $\mathrm{V}_{1,\mathrm{id}'}^i \stackrel{\text{def}}{=} \{v_1, \ldots, v_{\lfloor \widehat{\beta}^i L \rfloor}\}$, and $\mathrm{V}_{2,\mathrm{id}'}^i \stackrel{\text{def}}{=} \mathrm{V}^i \setminus \mathrm{V}_{1,\mathrm{id}'}^i$. Let $(\mathrm{H}_{1,\mathrm{id}'}^i, \mathrm{H}_{2,\mathrm{id}'}^i)$ be the corresponding almost-ideal hybrid partition (namely, $\mathrm{H}_{j,\mathrm{id}'}^i \stackrel{\text{def}}{=} \mathrm{W}_j^{i-1} \cup \mathrm{V}_{j,\mathrm{id}'}^i$). Since $|\widehat{\beta}^i - \beta^i| \leq \frac{\epsilon}{16}$, then the only difference between $(\mathrm{H}_{1,\mathrm{id}'}^i, \mathrm{H}_{2,\mathrm{id}'}^i)$ and $(\mathrm{H}_{1,\mathrm{id}}^i, \mathrm{H}_{2,\mathrm{id}}^i)$ is the placement of at most $\frac{\epsilon}{16} L$ vertices. By the above, and applying what we know about the ideal hybrid partition we have,

$$\mu(\mathrm{H}_{1,\mathrm{id}'}^i, \mathrm{H}_{2,\mathrm{id}'}^i) \;\geq\; \mu(\mathrm{H}_{1,\mathrm{id}}^i, \mathrm{H}_{2,\mathrm{id}}^i) - \frac{\epsilon}{16} L \cdot 2N \;\geq\; \mu(\mathrm{H}_1^{i-1}, \mathrm{H}_2^{i-1}) - \frac{3\epsilon}{8\ell} N^2 \tag{26}$$

In what follows we bound the difference between the size of the cut determined by the i^{th} hybrid partition (in which V^i is partitioned by the algorithm), and the size of the cut determined by the almost-ideal hybrid partition. This difference is due to the algorithm's use of approximate unbalance values (i.e., the values $\widehat{\text{ub}}(\cdot)$ used by the algorithm induce a different order on V^i than the "correct" order used in the almost-ideal partition). Let Y^i be the set of *misplaced* vertices in V^i which are put on a different side in (H^i_1, H^i_2) than in $(H^i_{1,\text{id}'}, H^i_{2,\text{id}'})$. Namely, $Y^i \stackrel{\text{def}}{=} \{V^i_{1,\text{id}'} \cap V^i_2\} \bigcup \{V^i_{2,\text{id}'} \cap V^i_1\}$.

Claim: There exists a value $y \in [-1, 1]$ such that for all but at most $\frac{\epsilon}{16}L$ of the vertices v in Y^i, $\text{ub}(v) = y \pm \frac{\epsilon}{16}$.

We prove the claim momentarily, and first derive a bound on $\mu(H^i_{1,\text{id}'}, H^i_{2,\text{id}'}) - \mu(H^i_1, H^i_2)$ based on the claim. By definition of the almost-ideal hybrid partition, we know that in the actual algorithm we put exactly the same number of vertices from V^i on each side of the partition as in the almost-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 $\frac{\epsilon}{8}$, the decrease in the number of cut edges between these two vertices and vertices in W^{i-1} is at most $\frac{\epsilon}{4}N$. The contribution of all such pairs is at most $\frac{L}{2} \cdot \frac{\epsilon}{4}N = \frac{\epsilon}{8\ell}N^2$. 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 $\frac{\epsilon}{16}L \cdot 2N = \frac{\epsilon}{8\ell}N^2$. Thus,

$$\mu(H^i_1, H^i_2) \;\; \geq \;\; \mu(H^i_{1,\text{id}'}, H^i_{2,\text{id}'}) - 2 \cdot \frac{\epsilon}{8\ell}N^2 \tag{27}$$

Combining Equation (26) and Equation (27), we have

$$\mu(H^i_1, H^i_2) \;\; \geq \;\; \mu(H^{i-1}_1, H^{i-1}_2) - \frac{3\epsilon}{8\ell}N^2 - \frac{\epsilon}{4\ell}N^2 \;\; = \;\; \mu(H^{i-1}_1, H^{i-1}_2) - \frac{5\epsilon}{8\ell}N^2 \tag{28}$$

and the lemma follows.

Proof of Claim: Consider a grouping of the vertices in V^i into $2/\epsilon'$ *unbalance bins* according to their (correct) unbalance value, where $\epsilon' = \epsilon/32$. For $k = -\frac{1}{\epsilon'}, \ldots, \frac{1}{\epsilon'} - 1$, the k^{th} bin, denoted B_k, is defined as follows:

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

Let g be the index of the bin which $v_{\lfloor \widehat{\beta}^i L \rfloor}$ 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 $k = -\frac{1}{\epsilon'}, \ldots, \frac{1}{\epsilon'} - 1$,

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

By our assumption on the representativeness of (U^i_1, U^i_2), at most $\frac{\epsilon}{32}$ 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 $\widehat{\text{ub}}(v)$, and let $\hat{g}$ be the index of the bin which $\hat{v}_{\widehat{\beta}^i L}$ belongs to. We consider two cases. Case 1: $\hat{g} = g \pm 1$. In this case all but at most $\frac{\epsilon}{16}L$ of the misplaced vertices have unbalance value ranging between $g \cdot \epsilon' N - \frac{\epsilon}{32}N$ and $(g+1) \cdot \epsilon' \cdot N + \frac{\epsilon}{32}N = g \cdot \epsilon' \cdot N + \frac{\epsilon}{16}N$, as required. Case 2: $\hat{g} \geq g + 2$ (the case $\hat{g} \leq g - 2$ is analogous). In such a case, necessarily, all but $\frac{\epsilon}{32}L$ of the vertices $v_1, \ldots, v_{\lfloor \widehat{\beta}^i L \rfloor}$ (which belong to bins $B_1, \ldots, B_g$), are put on side 1 (as they should). But since we put exactly $\lfloor \widehat{\beta}^i L \rfloor$ vertices on side 1, the total number of misplaced vertices is bounded by $2 \cdot \frac{\epsilon}{32}L$, and the claim follows. ■

Applying Lemma 8.10 to a maximum cut of G, we get

Corollary 8.11 *With probability at least $1-\frac{\delta}{2}$ over the choice of $\overline{\mathrm{U}}$, there exists a sequence of partitions Π of $\overline{\mathrm{U}}$, and an ℓ-tuple $\widehat{\beta}$ such that*

$$\mu\left(\mathrm{V}_1^{\Pi,\widehat{\beta}},\mathrm{V}_2^{\Pi,\widehat{\beta}}\right) \geq \mu^{(\frac{1}{2})}(\mathrm{G}) - \frac{3}{4}\epsilon$$

Thus, with probability at least $1-\frac{\delta}{2}$, the Graph Bisection Algorithm (described in Figure 6) outputs a bisection $\left(\mathrm{V}_1^{\widetilde{\Pi},\widetilde{\beta}},\mathrm{V}_2^{\widetilde{\Pi},\widetilde{\beta}}\right)$ such that $\mu\left(\mathrm{V}_1^{\widetilde{\Pi},\widetilde{\beta}},\mathrm{V}_2^{\widetilde{\Pi},\widetilde{\beta}}\right) \geq \mu^{(\frac{1}{2})}(\mathrm{G}) - \frac{3}{4}\epsilon$.

Bisection Approximation Algorithm

1. *As Step (1) of Figure 6.*
2. *Uniformly choose a set $\mathrm{S} = \{s_1,\ldots,s_m\}$ of size $m = \Theta\left(\frac{\ell^2 t + \ell^2\cdot\log(1/(\epsilon\delta)))}{\epsilon^2}\right)$. For $1 \leq i \leq \ell$, let $\mathrm{S}^i \stackrel{\rm def}{=} \mathrm{V}^i \cap \mathrm{S}$.*
3. *Analogously to Step (2) of Figure 6, for each of the sequence of partitions Π and for each of the ℓ-tuples $\widehat{\beta} = \left(\widehat{\beta}^1,\ldots,\widehat{\beta}^\ell\right)$, (where each $\widehat{\beta}^i \in [0,1]$ is an integer multiple of $\frac{\epsilon}{16\ell}$, and $\sum_i \widehat{\beta}^i = 1/2$), construct a partition $\left(\mathrm{S}_1^{\Pi,\widehat{\beta}},\mathrm{S}_2^{\Pi,\widehat{\beta}}\right)$ of S. Specifically, in the $i^{\rm th}$ iteration of Substep (a), S_1^i is assigned the $\lfloor\widehat{\beta}^i|\mathrm{S}^i|\rfloor$ vertices with the biggest $\widehat{\mathrm{ub}}(\cdot)$ value.*
4. *For each resulting partition $\left(\mathrm{S}_1^{\Pi,\widehat{\beta}},\mathrm{S}_2^{\Pi,\widehat{\beta}}\right)$, compute the fraction of cut edges between pairs of vertices (s_{2k-1},s_{2k}). More precisely, define*

$$\widehat{\mu}\left(\mathrm{S}_1^{\Pi,\widehat{\beta}},\mathrm{S}_2^{\Pi,\widehat{\beta}}\right) \stackrel{\rm def}{=} \frac{\left|\left\{(s_{2k-1},s_{2k}) \in \mathrm{E}\left(\mathrm{S}_1^{\Pi,\widehat{\beta}},\mathrm{S}_2^{\Pi,\widehat{\beta}}\right) \cup \mathrm{E}\left(\mathrm{S}_2^{\Pi,\widehat{\beta}},\mathrm{S}_1^{\Pi,\widehat{\beta}}\right)\right\}\right|}{m/2}.$$

   Let $\left(\mathrm{S}_1^{\widetilde{\Pi},\widetilde{\beta}},\mathrm{S}_2^{\widetilde{\Pi},\widetilde{\beta}}\right)$ be a partition of S for which this fraction is maximized, and output $\widehat{\mu}^{(\frac{1}{2})}(\mathrm{G}) = \widehat{\mu}\left(\mathrm{S}_1^{\widetilde{\Pi},\widetilde{\beta}},\mathrm{S}_2^{\widetilde{\Pi},\widetilde{\beta})}\right)$.

Figure 7: Bisection Approximation Algorithm

8.2.2 The Bisection Approximation Algorithm

Similarly to the Max-Cut approximation algorithm, the Bisection Approximation Algorithm (described in Figure 7) performs the same steps as the algorithm described in Figure 6, 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 similar to that of the Max-Cut approximation algorithm (Lemma 8.6) except for the following detail. Here we need to take into account that it is *not* necessarily the case that for a given $\overline{\mathrm{U}}$, a sequence of partitions Π of $\overline{\mathrm{U}}$ and $\widehat{\beta}$, for each $s \in \mathrm{S}$, $s \in \mathrm{S}_j^{\Pi,\widehat{\beta}}$, if and only if $s \in \mathrm{V}_j^{\Pi,\widehat{\beta}}$. This unfortunate phenomena is due to the possibility that for some vertices $v \in \mathrm{S}^i$, vertex v appears before (resp., after) the $\widehat{\beta}^i N^{\rm th}$ vertex in the ordering of V^i, but after (resp., before) the $\widehat{\beta}^i m^{\rm th}$ vertex in the ordering of S^i. To deal with this we prove the following lemma using arguments analogous to Lemma 7.7 (which deals with an analogous phenomena in the analysis of the ρ-Clique tester).

Lemma 8.12 *For a fixed* $\overline{\mathrm{U}}$, Π, *and* $\widehat{\beta}$, *let* $(\mathrm{V}_1, \mathrm{V}_2) = (\mathrm{V}_1^{\Pi,\widehat{\beta}}, \mathrm{V}_2^{\Pi,\widehat{\beta}})$ *be as defined in the Graph Bisection Algorithm. Let* S *be a uniformly chosen sample of size* $m = \Omega\left(\frac{\ell(\ell t+\log(1/(\epsilon\delta)))}{\epsilon^2}\right)$ *such that that for each* i, $|\mathrm{S}^i| \geq m/(2\ell)$, *and let* $(\mathrm{S}_1, \mathrm{S}_2) = (\mathrm{S}_1^{\Pi,\widehat{\beta}}, \mathrm{S}_2^{\Pi,\widehat{\beta}})$ *and* $\widehat{\mu}(\mathrm{S}_1, \mathrm{S}_2)$ *be as defined in the Bisection Approximation Algorithm. Then*

$$\mathrm{Pr}_{\mathrm{S}}\left[|\widehat{\mu}(\mathrm{S}_1,\mathrm{S}_2) - \mu(\mathrm{V}_1,\mathrm{V}_2)| > \frac{\epsilon}{4}\right] < \frac{\delta}{2}\cdot 2^{-\ell\cdot t}\cdot\left(\frac{\epsilon}{16}\right)^{\ell}$$

Proof: Let $\delta' \stackrel{\rm def}{=} \frac{\delta}{2}\cdot 2^{-\ell\cdot t}\cdot\left(\frac{\epsilon}{16}\right)^{\ell}$. For each $i \in \{1,\ldots,\ell\}$, let V_1^i and V_2^i be as defined in the Graph Bisection Algorithm, and let S_1^i and S_2^i be as defined in the Bisection Approximation Algorithm (for the fixed $\overline{\mathrm{U}}$, Π and $\widehat{\beta}$ considered in the lemma). Let $\epsilon_1 \stackrel{\rm def}{=} \epsilon/40$, and let $\overline{\mathrm{V}}_1^i$ be the first $\lfloor(\widehat{\beta}^i - \epsilon_1)L\rfloor$ vertices in V^i, and $\overline{\mathrm{V}}_2^i$ the last $\lceil(1-\widehat{\beta}^i-\epsilon_1)L\rceil$ vertices in V^i. For each i, let $m_i \stackrel{\rm def}{=} |\mathrm{S}^i|$, where by the lemma's hypothesis, $m_i \geq m/(2\ell)$. By Claim 7.8, for each i,

$$\mathrm{Pr}_{\mathrm{S}}\left[\left|\frac{|\mathrm{S}^i\cap\overline{\mathrm{V}}_1^i|}{m_i} - (\widehat{\beta}^i-\epsilon_1)\right| > \epsilon_1\right] < \frac{\delta'}{4\ell}$$

and similarly,

$$\mathrm{Pr}_{\mathrm{S}}\left[\left|\frac{|\mathrm{S}^i\cap\overline{\mathrm{V}}_2^i|}{m_i} - (1-\widehat{\beta}^i-\epsilon_1)\right| > \epsilon_1\right] < \frac{\delta'}{4\ell}$$

Note that the effects of rounding quantities such as $\widehat{\beta}^i\cdot L$ (and $\widehat{\beta}^i\cdot m_i$) are negligible since they effect the placement of at most one vertex from each V^i (respectively, S^i), and since $1/m_i$ (and certainly $1/L$) are much smaller than ϵ_1 we may ignore these effects. Thus, putting aside an error probability of $\frac{\delta}{2}$, assume from now on that for each i:

1. $|\mathrm{S}^i\cap\overline{\mathrm{V}}_1^i| \leq \widehat{\beta}^i\cdot m_i$, and $|\mathrm{S}^i\cap\overline{\mathrm{V}}_2^i| \leq (1-\widehat{\beta}^i)\cdot m_i$ from which it follows (by definition of S_1^i and S_2^i) that for each $j\in\{1,2\}$, $\mathrm{S}_j^i\cap\overline{\mathrm{V}}_j^i = \mathrm{S}^i\cap\overline{\mathrm{V}}_j^i$;

2. $|\mathrm{S}^i\cap\overline{\mathrm{V}}_1^i| \geq (\widehat{\beta}^i-2\epsilon_1)\cdot m_i$, and $|\mathrm{S}^i\cap\overline{\mathrm{V}}_2^i| \geq (1-\widehat{\beta}^i-2\epsilon_1)\cdot m_i$. Combining this with Item (1) it follows that $|\mathrm{S}_1^i\cap\overline{\mathrm{V}}_1^i| \geq (\widehat{\beta}^i-2\epsilon_1)\cdot m_i$ and $|\mathrm{S}_2^i\cap\overline{\mathrm{V}}_2^i| \geq (1-\widehat{\beta}^i-2\epsilon_1)\cdot m_i$.

Let $\overline{\mathrm{V}}_1$ be the union of the $\overline{\mathrm{V}}_1^i$'s and let $\overline{\mathrm{V}}_2$ be the union of the $\overline{\mathrm{V}}_2^i$'s. Then

$$\begin{aligned}\{(s_{2k-1},s_{2k})\in\mathrm{E}(\mathrm{S}_1,\mathrm{S}_2)\} \quad &= \quad \{(s_{2k-1},s_{2k})\in\mathrm{E}(\mathrm{S}_1\cap\overline{\mathrm{V}}_1,\mathrm{S}_2\cap\overline{\mathrm{V}}_2)\} \\ &\bigcup \quad \{(s_{2k-1},s_{2k})\in(\mathrm{E}(\mathrm{S}_1\setminus\overline{\mathrm{V}}_1,\mathrm{S}_2)\cup\mathrm{E}(\mathrm{S}_1,\mathrm{S}_2\setminus\overline{\mathrm{V}}_2))\}\end{aligned} \tag{29}$$

By Item (1) above and an additive Chernoff bound we get that

$$\mathrm{Pr}_{\mathrm{S}}\left[\left|\frac{\left|\{(s_{2k-1},s_{2k})\in\mathrm{E}(\mathrm{S}_1\cap\overline{\mathrm{V}}_1,\mathrm{S}_2\cap\overline{\mathrm{V}}_2)\}\right|}{m/2} - \frac{\left|\mathrm{E}(\overline{\mathrm{V}}_1,\overline{\mathrm{V}}_2)\right|}{N^2}\right| \;>\; \frac{\epsilon}{10}\right] \;<\; \frac{\delta'}{2} \tag{30}$$

By definition of $\overline{\mathrm{V}}_1$ and $\overline{\mathrm{V}}_2$, we have that

$$\frac{|\mathrm{E}(\mathrm{V}_1,\mathrm{V}_2)|}{N^2} - \frac{|\mathrm{E}(\overline{\mathrm{V}}_1,\overline{\mathrm{V}}_2)|}{N^2} \;\leq\; 2\epsilon_1 \;=\; \frac{\epsilon}{20} \tag{31}$$

By Item (2) above, we know that for each $i \in \{1, \ldots, \ell\}$ and $j \in \{1, 2\}$, $|S_j^i \setminus \overline{V}_j^i| \leq 2\epsilon_1 \cdot m_i$, and so

$$\frac{\left|\{(s_{2k-1}, s_{2k}) \in (E(S_1 \setminus \overline{V}_1, S_2) \cup E(S_1, S_2 \setminus \overline{V}_2))\right|}{m/2}$$
$$\leq \quad \frac{\left|\{k : \ s_{2k-1} \in S_1 \setminus \overline{V}_1 \text{ or } s_{2k} \in S_2 \setminus \overline{V}_2)\right|}{m/2} \quad \leq \quad 4\epsilon_1 \quad = \quad \frac{\epsilon}{10} \tag{32}$$

Summing up the probabilities of errors and combining the bounds of Equations (30)–(32), the lemma follows. ■

Part (1) of Theorem 8.8 follows by combining Corollary 8.11 and Lemma 8.12. We only need to observe that: (1) by a multiplicative Chernoff Bound, with very high probability in fact $|S^i| \geq m/(2\ell)$ for each i; and (2) the number of sequences of partitions Π is $2^{\ell t}$ and the number of settings of $\widehat{\beta}$ is less than $(16/\epsilon)^{\ell}$).

8.2.3 The Improved Graph-Bisection Algorithm

Similarly to the improved graph-partitioning algorithm for Max-Cut, The improved graph-bisection algorithm (whose running time is as stated in Theorem 8.8, Part (2)) starts by invoking the Bisection approximation algorithm of Figure 7, and recording the sequence of sets $\overline{U}$ uniformly selected in Step (1), and the sequence of partitions $\widetilde{\Pi}$ and the ℓ-tuple $\widetilde{\beta}$, selected in Step (4). Using these specific $\widetilde{\Pi}$ and $\widetilde{\beta}$, the algorithm executes a single iteration of Step (2) of the Graph Bisection Algorithm in Figure 6. Since it does not check the resulting partition it saves a multiplicative factor of N in its running time. More precisely, it has running time $O(t \cdot N) = O(\log(1/(\epsilon\delta)/\epsilon^2)) \cdot N$ (on top of the running time of the testing algorithm).

As for its correctness, by Corollary 8.11, we have that with probability at least $1 - \delta/2$ over the choice of $\overline{U}$, there exists Π and $\widehat{\beta}$ such that $\mu(V_1^{\Pi,\widehat{\beta}}, V_2^{\Pi,\widehat{\beta}}) \geq \mu(G) - \frac{3}{4}\epsilon$. From Lemma 8.12 we have that for a fixed $\overline{U}$, with probability at least $1 - \delta/2$ over the choice of S, $\widehat{\mu}\left(S_1^{\Pi,\widehat{\beta}}, S_2^{\Pi,\widehat{\beta}}\right)$ is within $\frac{\epsilon}{4}$ from $\mu\left(V_1^{\Pi,\widehat{\beta}}, V_2^{\Pi,\widehat{\beta}}\right)$ for every sequence of partitions Π of $\overline{U}$, and every $\widehat{\beta}$. It follows that with probability at least $1 - \delta$ over the choice of $\overline{U}$ and S, the recorded $\widetilde{\Pi}$ and $\widetilde{\beta}$ (from the Bisection Approximation Algorithm) are such that $\mu\left(V_1^{\widetilde{\Pi},\widetilde{\beta}}, V_2^{\widetilde{\pi},\widetilde{\beta}}\right) \geq \mu(G) - \epsilon$, as required.

8.2.4 Variations

Bisection Minimization. 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 $\widehat{\beta}^i$ 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 $\widehat{\text{ub}}(v)$

(where $\widehat{\mathrm{ub}}(v)$ is as defined in the maximization algorithms), and put the first $\widehat{\beta}^i$ vertices on side 1 and the rest on side 2.

Other Restrictions on the Partition. We can also easily generalize the algorithms to construct (resp., approximate the size of) partitions with other predetermined proportion of vertices on each side. A key observation is that the main steps of our algorithms are oblivious of the bisection requirement other than in asking that $\sum_i \widehat{\beta}^i = \frac{1}{2}$. In fact, the main steps can produce partitions with maximum (resp., minimum) number of crossing edges per each proportion of vertices on each side (up to some resolution). Therefore all we need to do is modify the restriction on the sum of the $\widehat{\beta}^i$'s to allow either a different fixed proportion or a range of proportions. Thus, for example, we can approximate quantities such as $\mathrm{opt}_{|\mathrm{V}_1|=N/3}\{\mu(\mathrm{V}_1, \mathrm{V(G)} \setminus \mathrm{V}_1)\}$, or $\mathrm{opt}_{\frac{N}{3} \le |\mathrm{V}_1| \le 2\frac{N}{3}}\{\mu(\mathrm{V}_1, \mathrm{V(G)} \setminus \mathrm{V}_1)\}$, where $\mathrm{opt} \in \{\max, \min\}$.

Testing algorithms for properties corresponding to the above optimization problems essentially follow from the approximation algorithms. In particular, consider first properties corresponding to minimization problems. When the property is defined as having a cut of density *at most* ρ (subject to certain constraints on the partition defining the cut) then the corresponding class is never empty (in particular, the empty graph belongs to the class). Furthermore, for any α, if a graph G has a partition $(\mathrm{V}_1, \mathrm{V}_2)$ such that $|\mathrm{V}_1| = \alpha N$ and $\mu(\mathrm{V}_1, \mathrm{V}_2) \le \rho + \epsilon$, then we can always remove at most ϵN^2 edges to obtain $\mu(\mathrm{V}_1, \mathrm{V}_2) = \rho$. Therefore, in the case of cut-minimization problems of the type discussed in this subsection, the corresponding testing algorithms follow directly from the approximation algorithms.

The situation is slightly more involved when dealing with maximization problems. For $0 \le \alpha \le \frac{1}{2}$ and a graph G, let $\mu^{(\alpha)}(\mathrm{G})$ denote the maximum edge density among all cuts $(\mathrm{V}_1, \mathrm{V(G)} \setminus \mathrm{V}_1)$ such that $|\mathrm{V}_1| = \alpha N$. Let $\mathcal{MC}_\rho^{(\alpha)} \stackrel{\mathrm{def}}{=} \{\mathrm{G} : \mu^{(\alpha)}(\mathrm{G}) \ge \rho\}$. To test whether a graph G belongs to the class $\mathcal{MC}_\rho^{(\alpha)}$, we first check whether the class is empty, in which case we reject G. Namely, if $2\alpha(1-\alpha) < \rho$ then the class must be empty, since the maximum number of edges connecting vertices in a set of size αN to vertices in a set of size $(1-\alpha)N$ is $2\alpha(1-\alpha)N^2$. If $2\alpha(1-\alpha) \ge \rho$ (and so the class is not empty), the testing algorithm follows from the approximation algorithm mentioned above, analogously to the way the ρ-Bisection algorithm follows from the Bisection approximation algorithm (see discussion following Theorem 8.8).

When the sizes of sides of the partition are not required to be fixed but rather are allowed to be within a certain range, the testing algorithm is a little less straightforward. For $0 \le \alpha_1 < \alpha_2 \le \frac{1}{2}$ and a graph G, let $\mu^{(\alpha_1,\alpha_2)}(\mathrm{G})$ denote the maximum edge density among all cuts $(\mathrm{V}_1, \mathrm{V(G)} \setminus \mathrm{V}_1)$ such that $\alpha_1 N \le |\mathrm{V}_1| \le \alpha_2 N$. Let $\mathcal{MC}_\rho^{(\alpha_1,\alpha_2)} \stackrel{\mathrm{def}}{=} \{\mathrm{G} : \mu^{(\alpha_1,\alpha_2)}(\mathrm{G}) \ge \rho\}$. Note that the class of graphs having a cut of size at least ρ (i.e., $\mathcal{MC}_\rho = \mathcal{MC}_\rho^{(0,\frac{1}{2})}$), which was considered in Subsection 8.1, is indeed a special case. If $2\alpha_2(1-\alpha_2) < \rho$ then the class $\mathcal{MC}_\rho^{(\alpha_1,\alpha_2)}$ is empty. Otherwise, let $\alpha_1' \ge \alpha_1$ be the minimum value such that $2\alpha_1'(1-\alpha_1') \ge \rho$. We run the algorithm for approximating $\mu^{(\alpha_1,\alpha_2)}(\mathrm{G})$ as described above with approximation parameter $\frac{\epsilon}{2}$ (where ϵ is the distance parameter of the testing algorithm), while requiring that $\alpha_1' \le \sum_i \widehat{\beta}^i \le \alpha_2$. We accept the graph if only if the approximate value obtained is at least $\rho - \frac{\epsilon}{2}$.

The following example best illustrates why we introduce the restriction that $\sum_i \widehat{\beta}^i \ge \alpha_1'$ instead of just using $\sum_i \widehat{\beta}^i \ge \alpha_1$. Consider the case in which $\alpha_1 = 0$, $\alpha_2 = \frac{1}{2}$, and $\rho = \frac{1}{2}$ (and so we are simply asking whether the graph G has any cut of density $\frac{1}{2}$). Suppose that G is a complete bipartite graph between a set of vertices V_1, and a set of vertices V_2, such that $|\mathrm{V}_1| = (\frac{1}{2} - \sqrt{\epsilon/2})$ and $|\mathrm{V}_2| = (\frac{1}{2} + \sqrt{\epsilon/2})$. Thus, $\mu^{(0,\frac{1}{2})}(\mathrm{G}) = \rho - \epsilon$, and so with fairly high probability the approximation algorithm would output an approximate value that is close to ρ. However, G is $\sqrt{\epsilon}$-far from having the desired property, and should be rejected. Note that the difficulty is not with the approximation algorithm but rather with the relation between approximation and testing. In this case, setting$\alpha_1' = \alpha_2 = \frac{1}{2}$ (as suggested above, since $2\alpha_1' \cdot (1 - \alpha_1') \geq \frac{1}{2}$ implies $\alpha_1' = \frac{1}{2}$), we restrict our algorithm to consider only partitions for which $\sum_i \widehat{\beta}^i \geq \alpha_1'$, and so the algorithm will detect that the graph should be rejected.

9 The General Partition Problem

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

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 $\varrho_{j,j'}^{\rm LB} \leq \varrho_{j,j'}^{\rm UB}$ $(\forall j, j')$. (LB stands for Lower Bound, and UB stands for Upper Bound.) 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 conditions being satisfied.

$$\forall j \qquad \rho_j^{\rm LB} \cdot N \;\leq\; |{\rm V}_j| \;\leq\; \rho_j^{\rm UB} \cdot N \tag{33}$$

and

$$\forall j, j' \qquad \varrho_{j,j'}^{\rm LB} \cdot N^2 \;\leq\; |{\rm E}({\rm V}_j, {\rm V}_{j'})| \;\leq\; \varrho_{j,j'}^{\rm UB} \cdot N^2 \tag{34}$$

where recall that ${\rm E}({\rm V}_j, {\rm V}_{j'})$ 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. (33) places lower and upper bounds on the relative sizes of the various components of the partition; whereas Eq. (34) imposes lower and upper bounds on the density of edges among the various pairs of components.

Remark. (A tedious one.) 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 $3M$ vertices may be in $\mathcal{GP}_\Phi$. In such a case, the tester must reject any graph with $3M+1$ vertices (as the class of graphs with $3M+1$ vertices having the property defined by the parameters in empty), whereas it must accept some $3M$-vertex graphs. Consequently, such a tester 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.

In this section we describe a testing algorithm for the class $\mathcal{GP}_\Phi$ (for any given set of parameters $\Phi = \{\rho_j^{\rm UB}, \rho_j^{\rm LB}\} \cup \{\varrho_{j,j'}^{\rm UB}, \varrho_{j,j'}^{\rm LB}\}$). Similarly to the testing algorithms described in Sections 7 and 8, the testing algorithm of this section 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 (33) and (34), the partitioning algorithm constructs a partition $({\rm V}_1, \ldots, {\rm V}_k)$ of G for which the following hold with probability at least $1-\delta$:

$$\forall j, \;\; (\rho_j^{\rm LB} - \epsilon) \cdot N \;\leq\; |{\rm V}_j| \;\leq\; (\rho_j^{\rm UB} + \epsilon) \cdot N, \tag{35}$$

and

$$\forall j, j', \;\; (\varrho_{j,j'}^{\rm LB} - \epsilon) \cdot N^2 \;\leq\; |{\rm E}({\rm V}_j, {\rm V}_{j'})| \;\leq\; (\varrho_{j,j'}^{\rm UB} + \epsilon) \cdot N^2\,, \tag{36}$$

A partition obeying (35) and (36) is called an ϵ-*approximation* for the partitioning problem defined by the set of parameters Φ.

As already indicated in the special case of ρ-Cut, and in the generalization of the Bisections testing algorithm (Subsection 8.2.4), the relationship between having an ϵ-approximation for the general partitioning problem, and being ϵ-close to the class of graphs having the property is not completely straightforward. In particular, a graph may have a partition that is an ϵ-approximation for the partitioning problem defined by the set of parameters Φ, but is $\Omega(\sqrt{\epsilon})$-far from the class $\mathcal{GP}_\Phi$. We shall deal with this difficulty when designing the testing algorithm. Jumping ahead we mentioned that instead of checking whether the tested graph has a partition that is an ϵ-approximation for the partitioning problem (or an $f(\epsilon, k)$-approximation, for some function f of ϵ and k), we directly check whether the graph is ϵ-close to $\mathcal{GP}_\Phi$.

As stated above, all properties considered in previous subsections can be casted as special cases of the general partition problem. For example, k-Colorability is expressed by setting $\varrho^{\mathrm{UB}}_{j,j} = 0$ for every j (and placing no other constraints which means setting $\rho^{\mathrm{LB}}_j = 0$, $\rho^{\mathrm{UB}}_j = 1$, and similarly setting the $\varrho^{\mathrm{UB}}_{j,j'}$'s and $\varrho^{\mathrm{LB}}_{j,j'}$'s for $j' \neq j$). In case we are interested in maximizing or minimizing a parameter (e.g. maximizing $\mathrm{E}(\mathrm{V}_1, \mathrm{V}_2)$ 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.[19] However, as can be seen from our theorems below (and the table in Figure 1), 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 k-Coloring, Max-Clique, and Max-Cut, were presented in previous sections. While we cannot exploit the problem-specific properties as done in the previous sections, we nonetheless apply some of the ideas used in the above algorithms.

Theorem 9.1 *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 and running time*

$$\log^2\left(\frac{1}{\epsilon\delta}\right) \cdot \left(\frac{O(k^2)}{\epsilon}\right)^{2k+8} \quad \text{and} \quad \exp\left(\log\left(\frac{1}{\epsilon\delta}\right) \cdot \left(\frac{O(k^2)}{\epsilon}\right)^{k+1}\right)$$

respectively.

We note that in the running time of the algorithm, we ignore a factor that is polynomial in the length of the description of Φ, as we view this length as a fixed constant and not a parameter to the problem. Furthermore, in any reasonable application, it is much smaller than all other factors. As in previous sections, we also obtain an analogous graph partitioning algorithm.

Theorem 9.2 *There exists a graph partitioning algorithm that on input Φ, ϵ, and δ, and oracle access to a graph* G*, runs in time*

$$\exp\left(\log(1/(\epsilon\delta)) \cdot \left(\frac{O(1)}{\epsilon}\right)^{k+1}\right) + O\left(\frac{\log(k/(\epsilon\delta))}{\epsilon^2}\right) \cdot N$$

and if G *has a k-way partition satisfying Equations (33) and (34) then with probability at least $1-\delta$ the graph partitioning algorithm outputs a partition which satisfies Equations (35) and (36).*

[19] 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.

We start by describing a less efficient graph partitioning algorithm with running time $\exp\left(\log(1/(\epsilon\delta)\cdot\left(\frac{O(1)}{\epsilon}\right)^{k+1}\right)\cdot N^2$. Based on this algorithm we shall obtain the tester postulated in Theorem 9.1 and finally derive the (more efficient) graph partitioning algorithm (postulated in Theorem 9.2).

9.1 High Level Description of the Partitioning Algorithm

The algorithm is based on the following observation, which generalizes an observation applied in the Bisection algorithm (Subsection 8.2). 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 (33) and (34). Let X be a set of vertices of small size (i.e., of size $O(\epsilon N)$) and suppose that all but $O(\epsilon|\mathrm{X}|)$ of the vertices in X have approximately the same (i.e., $\pm O(\epsilon N)$) number of neighbors in each individual $\mathrm{H}_j \setminus \mathrm{X}$. Namely, for each $j \in \{1, \ldots, k\}$, there exists a value β_j such that for all but $O(\epsilon|\mathrm{X}|)$ of the vertices v in X, we have

$$\frac{|\Gamma(v) \cap (\mathrm{H}_j \setminus \mathrm{X})|}{N} \;=\; \beta_j \;\pm\; O(\epsilon) \tag{37}$$

(Recall that $a = b \pm c$ is a shorthand for $b - c \leq a \leq b + c$.) The observation is that if we arbitrarily redistribute the vertices of X among the k components (i.e., H_j's) while maintaining the *number of vertices* in each component, then the *number of edges* between every pair of components is approximately maintained.

More precisely, let $(\mathrm{X}_1, \ldots, \mathrm{X}_k)$ be an arbitrary partition of X so that $|\mathrm{X}_j| = |\mathrm{X} \cap \mathrm{H}_j| \pm O(\epsilon|\mathrm{X}|)$, and let $\mathrm{H}'_j = (\mathrm{H}_j \setminus \mathrm{X}) \cup \mathrm{X}_j$. Let $\mathcal{H}' \stackrel{\mathrm{def}}{=} (\mathrm{H}'_1, \ldots, \mathrm{H}'_k)$. Then by our assumption on the X_j's, we have $\Big||\mathrm{H}_j| - |\mathrm{H}'_j|\Big| = O(\epsilon|\mathrm{X}|)$ for every j. Furthermore, by our assumption concerning the "neighbor-profile" of vertices in X (Equation (37)), for every j, j',

$$\Big||\mathrm{E}(\mathrm{H}_j, \mathrm{H}_{j'})| - |\mathrm{E}(\mathrm{H}'_j, \mathrm{H}'_{j'})|\Big| \;=\; O(|\mathrm{X}|^2 + |\mathrm{X}| \cdot \epsilon N) \;=\; O(|\mathrm{X}| \cdot \epsilon N) \tag{38}$$

where the second equality is due to the size of X. The first equality, namely the bound on the difference of the number of edges, is proved in detail in Lemma 9.3. The first crucial observation is that edges with both endpoints *not* in X are in the same component in $\mathcal{H}'$ as they were in $\mathcal{H}$, and thus are not effected by the redistribution of X. As for edges with at least one endpoint in X, there are $O(|\mathrm{X}|^2)$ edges with both endpoints in X (accounted for in the first term of Equation (38)), and the changes in the number of edges with exactly one endpoint in X (due to Equation (37) and $|\mathrm{X}_j| = |\mathrm{X}_j \cap \mathrm{H}_j| \pm O(\epsilon|\mathrm{X}|)$) can be bounded by $O(|\mathrm{X}| \cdot \epsilon N)$.

So far we have dealt with a single set of vertices such that all vertices in the set have approximately the same "neighborhood profile" with respect to a given partition. In general we wish to handle the case where vertices have arbitrary neighborhood profile with respect to the partition. The idea is to pack vertices into *clusters* according to their neighborhood profile. Specifically, 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). We first *cluster* the vertices in Y according to the approximate 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 by the above discussion, which may be viewed as concerning a single cluster, for every j,

$$\Big||\mathrm{H}_j| - |\mathrm{H}'_j|\Big| \;=\; \sum_{\text{clusters } \mathrm{X}\subseteq\mathrm{Y}} O(\epsilon|\mathrm{X}|) \;=\; O(\epsilon|\mathrm{Y}|) \;=\; O(\epsilon^2 N) \tag{39}$$

and for every j, j',

$$\Big||\mathrm{E}(\mathrm{H}_j,\mathrm{H}_{j'})| - |\mathrm{E}(\mathrm{H}'_j,\mathrm{H}'_{j'})|\Big| \;=\; \sum_{\text{clusters } \mathrm{X}\subseteq\mathrm{Y}} O(|\mathrm{X}|\cdot\epsilon N) \;=\; O(\epsilon^2 N^2) \tag{40}$$

Similarly to the analysis of the graph partitioning algorithms for Max-Cut and Bisection, we shall use the above observation to define a sequence of $O(1/\epsilon)$ hybrid partitions. It will follow that for the final hybrid, the differences in vertex and edge densities as compared to the initial one (which obeys Equations (33) and (34)) is at most ϵ.

The Oracle Aided Procedure. In view of the above observation, we are almost ready to describe the partitioning algorithm. Similarly to the graph-partitioning algorithms for Max-Cut and Bisection, 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, using as before a *fixed* partition into ℓ equal-size sets $\mathrm{V}^1,\ldots,\mathrm{V}^\ell$, where $\ell = \frac{4}{\epsilon}$. In the i^{th} iteration the algorithm partitions the set V^i into $(\mathrm{V}^i_1,\ldots,\mathrm{V}^k_i)$. Let $\mathcal{H}^0 = (\mathrm{H}^0_1,\ldots,\mathrm{H}^0_k)$ be a k-way partition which obeys Equations (33) and (34), and for each $i > 0$, let $\mathcal{H}^i = (\mathrm{H}^i_1,\ldots,\mathrm{H}^i_k)$ be the i^{th} *hybrid partition*, where $\mathrm{H}^i_j \stackrel{\text{def}}{=} (\mathrm{H}^{i-1}_j \setminus \mathrm{V}^i) \cup \mathrm{V}^i_j$. Let $\mathcal{W}^{i-1} = (\mathrm{W}^{i-1}_1,\ldots,\mathrm{W}^{i-1}_k)$ be the partition induced on $\mathrm{V}\setminus\mathrm{V}^i$ by $\mathcal{H}^{i-1}$. That is, for each j, $\mathrm{W}^{i-1}_j \stackrel{\text{def}}{=} \mathrm{H}^{i-1}_j \setminus \mathrm{V}^i$. For any given vertex $v \in \mathrm{V}^i$ and for every $j \in \{1,\ldots,k\}$, let $\gamma_j(v) \stackrel{\text{def}}{=} \frac{|\Gamma(v)\cap\mathrm{W}^{i-1}_j|}{N}$. The k-tuple $(\gamma_1(v),\ldots,\gamma_k(v))$ is called the *neighborhood profile* of v, referred to in the above discussion.

We assume that the algorithm has an oracle that for each i, given a vertex $v \in \mathrm{V}^i$ and $j \in \{1,\ldots,k\}$, returns a value $\hat{\gamma}_j(v)$ so that for all but $O(\epsilon L)$ of the vertices v in V^i it holds that for every j, $\hat{\gamma}_j(v) = \gamma_j(v) \pm \epsilon/32$. Using this oracle, the algorithm clusters the vertices in V^i according to the number of neighbors they have in each component of $\mathcal{W}^{i-1}$ *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

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

We refer to the $\vec{\alpha}$'s as the *cluster names*, since each $\vec{\alpha}$ uniquely defines a different cluster of V^i.

Assume further that the algorithm also had access to an oracle which for every cluster $\mathrm{V}^{i,\vec{\alpha}}$ and for each j, returns an approximation, up to an error of $\frac{\epsilon}{16}$, of the fraction of vertices in $\mathrm{V}^{i,\vec{\alpha}}$ which belong to H^{i-1}_j. Let this approximate fraction be denoted $\beta^{i,\vec{\alpha}}_j$. That is,

$$\beta^{i,\vec{\alpha}}_j = \frac{|\mathrm{V}^{i,\vec{\alpha}} \cap \mathrm{H}^{i-1}_j|}{|\mathrm{V}^{i,\vec{\alpha}}|} \pm \frac{\epsilon}{16}$$

We refer to $\langle\beta^{i,\vec{\alpha}}_1,\ldots,\beta^{i,\vec{\alpha}}_k\rangle$ as the *quantitative partition* of $\mathrm{V}^{i,\vec{\alpha}}$, since it only determines how *many* vertices from $\mathrm{V}^{i,\vec{\alpha}}$ should be in each component of the partition (and does not specify *which* vertices should be in each component). However, by our observation concerning redistribution of vertices belonging to the same cluster, the quantitative partition of $\mathrm{V}^{i,\vec{\alpha}}$ is all that matters, and we may as well partition $V^{i,\vec{\alpha}}$ in an arbitrary way as long as the quantitative partition is satisfied. Let $\left(V_1^{i,\vec{\alpha}},\ldots,V_k^{i,\vec{\alpha}}\right)$ be such a partition; that is, $\lfloor \beta_j^{i,\vec{\alpha}}|V^{i,\vec{\alpha}}|\rfloor \leq |V_j^{i,\vec{\alpha}}| \leq \lceil \beta_j^{i,\vec{\alpha}}|V^{i,\vec{\alpha}}|\rceil$, for every j.

Let $(V_1^i,\ldots,V_k^i)$ be defined by $V_j^i \stackrel{\rm def}{=} \bigcup_{\vec{\alpha}} V_j^{i,\vec{\alpha}}$, for each j. By our previous discussion we know that for each i, the changes in vertex and edge densities between the $i-1$ and $i^{\rm th}$ hybrid partitions is $O(\epsilon^2)$ (see Equation (39) and (40)). Combining this with our hypothesis concerning $\mathcal{H}^0$ (i.e., that $\mathcal{H}^0$ obeys Equations (33) and (34)) we conclude that $\mathcal{H}^\ell = \{\cup_i V_1^i,\ldots,\cup_i V_k^i\}$ is an $O(\epsilon)$-approximation of the partitioning problem (as defined by Equations (33) and (34)).

Simulating the Oracles. We next 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. Instead, we try all possibilities. Recall that i takes on $\ell = O(1/\epsilon)$ values, and for each i, there are $O(1/\epsilon)^k$ possible values of $\vec{\alpha}$ (i.e., clusters). 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 simply try all possible

$$
\begin{aligned}
O\left(\left(\frac{1}{\epsilon}\right)^k\right)^{O\left((1/\epsilon)^k\cdot\ell\right)} &< \exp\left(O\left(\left(\frac{1}{\epsilon}\right)^{k+1}\cdot k\log(1/\epsilon)\right)\right) \\
&< \exp\left(\left(\frac{1}{\epsilon}\right)^{k+1}\cdot \log(1/\epsilon)\right) \qquad (41)
\end{aligned}
$$

(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.

In order to approximately simulate the oracles for $\gamma_j(v)$, we apply the same technique used in previous sections. Namely, we uniformly select ℓ sets, $U^1,\ldots,U^\ell$, where $U^i \subset V\setminus V^i$ (each of size $t = \Theta(\epsilon^{-2}\log(k/(\epsilon\delta)))$, and for each i we consider all k-way partitions $(U_1^i,\ldots,U_k^i)$, of U^i. For each possible sequence of partitions (i.e., one partition per U^i), when we partition V^i, for each $v \in V^i$ we use the approximation $\hat{\gamma}_j(v) \stackrel{\rm def}{=} |\Gamma(v)\cap U_j^i|/t$ for $\gamma_j(v)$. As we prove in detail in Lemma 9.3, with high probability over the choice of $U^1,\ldots,U^\ell$, there exist *representative* partitions of the U^i, such that $\hat{\gamma}_j(v)$ is in fact a good approximation of $\gamma_j(v)$ for almost all vertices.

9.2 The Preliminary Partitioning Algorithm

A detailed description of the graph partitioning algorithm is given in Figure 8, and its formal analysis is provided in Lemma 9.3.

Lemma 9.3 *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 $\overline{U}$, there exists a partition $\Pi = (U^1,\ldots,U^\ell)$ of $\overline{U}$, and a setting of $\vec{\beta}$, such that*

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

and

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

Graph Partitioning Algorithm for $\mathcal{GP}_\Phi$

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* $\ell = \lceil \frac{4}{\epsilon} \rceil$ *sets* $\mathrm{U}^1, \ldots, \mathrm{U}^\ell$ *each of size* $t = \Theta\left(\frac{1}{\epsilon^2} \log \frac{k}{\epsilon\delta}\right)$, *where* U^i *is chosen uniformly in* $\mathrm{V} \setminus \mathrm{V}^i$. *Let* $\overline{\mathrm{U}} = \langle \mathrm{U}^1, \ldots, \mathrm{U}^\ell \rangle$.
2. *For every setting of* $\vec{\beta} = \left\langle \vec{\beta}^1, \ldots, \vec{\beta}^\ell \right\rangle$, *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 sequence of partitions* $\Pi = \langle (\mathrm{U}_1^1, \mathrm{U}_k^1), \ldots (\mathrm{U}_1^\ell, \mathrm{U}_k^\ell) \rangle$, *construct a partition* $\mathcal{V}^{\Pi,\vec{\beta}} = \left(\mathrm{V}_1^{\Pi,\vec{\beta}}, \ldots, \mathrm{V}_k^{\Pi,\vec{\beta}}\right)$ *as follows:*
   - (a) *For* $i = 1 \ldots \ell$ *do:*
     - i. *For each* $v \in \mathrm{V}^i$, *and for* $j \in \{1, \ldots, k\}$, *let* $\hat{\gamma}_j(v) \stackrel{\mathrm{def}}{=} \frac{1}{t} \cdot \left|\Gamma(v) \cap \mathrm{U}_j^i\right|$.
     - ii. *For each* $\vec{\alpha} \in \Lambda$ *let the* $\vec{\alpha}$*-cluster of* V^i, *denoted* $\mathrm{V}^{i,\vec{\alpha}}$, *be defined as follows:*
       $\mathrm{V}^{i,\vec{\alpha}} \stackrel{\mathrm{def}}{=} \left\{ v \in V^i : \forall j,\ \alpha_j - \epsilon/32 < \hat{\gamma}_j(v) \leq \alpha_j + \epsilon/32 \right\}$.
     - iii. *For each cluster* $\mathrm{V}^{i,\vec{\alpha}}$, $\vec{\alpha} \in \Lambda$, *let* $\mathrm{V}_1^{i,\vec{\alpha}}$ *be the first* $\beta_1^{i,\vec{\alpha}} \cdot |\mathrm{V}^{i,\vec{\alpha}}|$ *vertices in* $\mathrm{V}^{i,\vec{\alpha}}$, *let* $\mathrm{V}_2^{i,\vec{\alpha}}$ *be the next* $\beta_2^{i,\vec{\alpha}} \cdot |\mathrm{V}^{i,\vec{\alpha}}|$ *vertices in* $\mathrm{V}^{i,\vec{\alpha}}$, *and in general, let* $\mathrm{V}_j^{i,\vec{\alpha}}$ *be the first* $\beta_j^{i,\vec{\alpha}} \cdot |\mathrm{V}^{i,\vec{\alpha}}|$ *vertices in* $\mathrm{V}^{i,\vec{\alpha}} \setminus \bigcup_{j'<j} \mathrm{V}_{j'}^{i,\vec{\alpha}}$ *(where in case the resulting numbers are not integers, they are rounded, alternating between rounding up and down).*
       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* $\mathrm{V}_j^i = \bigcup_{\alpha \in \Lambda} \mathrm{V}_j^{i,\vec{\alpha}}$.
   - (b) *For each* $j \in \{1, \ldots, k\}$, *let* $\mathrm{V}_j = \bigcup_{i=1}^{\ell} \mathrm{V}_j^i$.
     Since each V_j *actually depends on* Π *and* $\vec{\beta}$ *we denote it by* $\mathrm{V}_j^{\Pi,\vec{\beta}}$.
   - (c) *If the partition* $\mathcal{V}^{\Pi,\vec{\beta}}$ *obeys Equations (35) and (36) then* Output $\mathcal{V}^{\Pi,\vec{\beta}}$.

Figure 8: Graph Partitioning Algorithm

Proof: For a fixed partition Π of $\overline{\mathrm{U}}$ and a fixed setting $\vec{\beta} = \left\langle \vec{\beta}^1, \ldots, \vec{\beta}^\ell \right\rangle$, we consider the following $\ell + 1$ *hybrid* partitions: The hybrid $\mathcal{H}^0 = (\mathrm{H}_1^0, \ldots, \mathrm{H}_k^0)$ is simply $\mathcal{H}$; The hybrid partition $\mathcal{H}^i = (\mathrm{H}_1^i, \ldots, \mathrm{H}_k^i)$ is defined as follows:

$$\mathrm{H}_j^i \stackrel{\mathrm{def}}{=} \mathrm{W}_j^{i-1} \cup \mathrm{V}_j^i$$

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 $\mathrm{W}_j^{i-1} \stackrel{\mathrm{def}}{=} \mathrm{H}_j^{i-1} \setminus \mathrm{V}^i$, and V_j^i's are determined as in Figure 8. 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

$$\forall j, \quad \left| \frac{\left|\mathrm{H}_j^i\right| - |\mathrm{H}_j^{i-1}|}{N} \right| \leq \frac{\epsilon}{16\ell}\,, \tag{42}$$

and

$$\forall j, j', \quad \left| \frac{\left|\mathrm{E}\left(\mathrm{H}_j^i, \mathrm{H}_{j'}^i\right)\right| - \left|\mathrm{E}\left(\mathrm{H}_j^{i-1}, \mathrm{H}_{j'}^{i-1}\right)\right|}{N^2} \right| \leq \frac{\epsilon}{\ell}\,. \tag{43}$$

The lemma follows by 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, $\mathrm{U}_j^i \stackrel{\mathrm{def}}{=} \mathrm{W}_j^{i-1} \cap \mathrm{U}^i$. 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 at most an $\frac{\epsilon}{8}$ fraction of the vertices v in V^i,

$$\text{For each } j \in \{1, \ldots, k\} \quad \frac{\left|\Gamma(v) \cap \mathrm{U}_j^i\right|}{t} = \frac{|\Gamma(v) \cap \mathrm{W}_j^{i-1}|}{N} \pm \frac{\epsilon}{32} \tag{44}$$

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. (42) and (43) 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 8, Step (2.a.ii), where $\hat{\gamma}_j(v)$ (for every $v \in \mathrm{V}^i$ and j) is defined based on V_j^i 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, $\beta_j^{i,\vec{\alpha}} = \frac{|\mathrm{H}_j^{i-1} \cap \mathrm{V}^{i,\vec{\alpha}}|}{|\mathrm{V}^{i,\vec{\alpha}}|} \pm \frac{\epsilon}{16}$. Since each $\beta_j^{i,\vec{\alpha}}$ takes on all values which are multiples of $\frac{\epsilon}{8}$, there is such a setting of $\vec{\beta}^i$'s. It follows that when the vertices in V^i are partitioning using this (approximately correct) $\vec{\beta}^i$, then Equation (42) holds. That is,

$$\begin{aligned}
\left| |\mathrm{H}_j^i| - |\mathrm{H}_j^{i-1}| \right| &= \left| \sum_{\vec{\alpha}} \left|\mathrm{H}_j^{i-1} \cap \mathrm{V}^{i,\vec{\alpha}}\right| - \sum_{\vec{\alpha}} \beta_j^{i,\vec{\alpha}} \cdot \left|\mathrm{V}^{i,\vec{\alpha}}\right| \right| \\
&\leq \sum_{\vec{\alpha}} \frac{\epsilon}{16} \cdot |\mathrm{V}^{i,\vec{\alpha}}| \\
&= \frac{\epsilon}{16} \cdot \frac{N}{L}
\end{aligned}$$

We now show that Equation (43) holds as well. Towards this end we fix arbitrary j, j' and consider the contribution of three types of edges to the left hand side of Equation (43):

1. The contribution of edges with both endpoints NOT in V^i. Since $v \in H_j^i$ iff $v \in H_j^{i-1}$, for every $v \notin V^i$, such edges do not contribute to the difference (i.e., to the left hand side of Equation (43)).

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

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 (44) does NOT hold. Since $(U_1^i, \ldots, U_k^i)$ is representative with respect to $(W_1^{i-1}, \ldots, W_k^{i-1})$ and V^i, there are at most $\frac{\epsilon}{8\ell}N$ such vertices. Thus, the contribution of these edges to the left hand side of Equation (43) 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 (44) 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}. Specifically, for every v, v' which belong to the same cluster $V^{i,\vec{\alpha}}$ and for each j, we have

$$\begin{aligned} |\gamma_j(v) - \gamma_j(v')| &\leq |\hat{\gamma}_j(v) - \hat{\gamma}_j(v')| + |\hat{\gamma}_j(v) - \gamma_j(v)| + |\hat{\gamma}_j(v') - \gamma_j(v')| \\ &\leq \frac{\epsilon}{16} + 2 \cdot \frac{\epsilon}{32} \\ &= \frac{\epsilon}{8} \end{aligned}$$

   The second approximation error is due to the fact that the fractional partition is specified with bounded precision (i.e., $\beta_j^{i,\vec{\alpha}} = \frac{|V_j^{i-1} \cap V^{i,\vec{\alpha}}|}{|V^{i,\vec{\alpha}}|} \pm \epsilon/16$), and so at most $\frac{\epsilon}{16\ell} \cdot N$ vertices might be misplaced, contributing at most $\frac{\epsilon}{8\ell} \cdot N^2$ edges. Thus, the contribution of both cases to the left hand side of Equation (43) is bounded above by $\frac{3\epsilon}{8\ell}N^2$.

Summing all cases we get a contribution bounded above by $(\frac{1}{4} + \frac{1}{4} + \frac{3}{8}) \cdot \frac{\epsilon}{\ell}N^2 < \frac{\epsilon}{\ell}N^2$.

BOUNDING THE PROBABILITY THAT U^i IS NOT GOOD. We first fix a vertex $v \in V^i$. Let $U^i = \{u_1, \ldots, u_t\}$. For $j \in \{1 \ldots, k\}$, and for $1 \leq s \leq t$, define a 0/1 random variable, ξ_j^s, which is 1 if u_s is a neighbor of v and $u_s \in W_j^{i-1}$, 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 $\frac{1}{N}|\Gamma(v) \cap W_j^{i-1}|$. By an additive Chernoff bound (see Appendix B) and our choice of t, for each $j \in \{1, \ldots, k\}$,

$$\Pr_{U^i}\left[\left|\frac{|\Gamma(v) \cap U_j^i|}{t} - \frac{|\Gamma(v) \cap W_j^{i-1}|}{N}\right| > \frac{\epsilon}{32}\right] = \exp(-\Omega(\epsilon^2 t)) = \frac{\epsilon\delta}{8k\ell}\,.$$

By Markov's inequality (see Appendix B), 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 (44) holds (for that j), and thus with probability at least $1 - \frac{\delta}{\ell}$, U^i is good as required. This concludes the proof of Lemma 9.3. ■

By considering a partition $\mathcal{H}$ for which Equations 33 and 34 hold, Lemma 9.3 implies that if $G \in \mathcal{GP}_\Phi$, then with high probability the Graph Partitioning Algorithm described in Figure 8 will find an ϵ-approximation for the partitioning problem defined by Φ. As noted previously, the number of sequences of partitions considered is bounded by $k^{\ell t} = \exp(\ell t \log k) = \exp\left(O\left(\frac{\log k}{\epsilon^3}\log(k/\epsilon\delta)\right)\right)$, and the number of settings of $\vec{\beta}$ is $\exp\left(O\left((1/\epsilon)^{k+1}\cdot\log(1/\epsilon)\right)\right)$ (see Equation (41). Hence, the number of iterations (in Step (2)), is $\exp\left(O\left((1/\epsilon)^{k+1}\cdot\log(1/(\epsilon\delta)\right)\right)$. In each iteration, the running time is governed by Step (2.c), that takes $O(N^2)$ time. As was mentioned previously, we later show how to produce such a partitioning in a more efficient way by first running the testing algorithm for $\mathcal{GP}_\Phi$.

9.3 The Testing Algorithm

The testing algorithm for $\mathcal{GP}_\Phi$ (described in Figure 9) 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 (35) and (36) hold. As noted previously, a graph may have a partition that satisfies Equations (35) and (36), but still be more than ϵ-far from the class $\mathcal{GP}_\Phi$. By checking whether the sampled densities are close to an admissible set we are verifying directly whether the tested graph is close to the class $\mathcal{GP}_\Phi$.

In the following definition, for each j, a_j corresponds to the fraction of vertices in the jth component of the partition (so in particular the a_j's must sum to 1), and for each j, j', $b_{j,j'}$ corresponds to the fraction of edges (among all vertex pairs) between components j and j'. The first two items in the definition merely state conditions on these a_j's and $b_{j,j'}$'s that must hold in any graph. The last item refers to the desired bounds with respect to Φ itself.

Admissible Set of Densities: A set of (non-negative) reals, $\{a_j\}\cup\{b_{j,j'}\}$, is called **admissible** with respect to Φ ($=\{\rho_j^{\text{LB}},\rho_j^{\text{UB}}\}\cup\{\varrho_{j,j'}^{\text{LB}},\varrho_{j,j'}^{\text{UB}}\}$) if it satisfies the following inequalities

$$\sum_{j=1}^{k} a_j = 1 \quad\text{and}\quad \sum_{j,j'=1}^{k} b_{j,j'} \le 1 \tag{45}$$

$$b_{j,j} \le a_j^2 \quad (\forall j\ ) \quad\text{and}\quad b_{j,j'} \le 2\cdot a_j\cdot a_{j'} \quad (\forall j \ne j'\ ) \tag{46}$$

and

$$\rho_j^{\text{LB}} \le a_j \le \rho_j^{\text{UB}} \quad (\forall j\ ) \quad\text{and}\quad \varrho_{j,j'}^{\text{LB}} \le b_{j,j'} \le \varrho_{j,j'}^{\text{UB}} \quad (\forall j,j'\ ) \tag{47}$$

In Step (4) of Figure 9 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 where $\epsilon' \stackrel{\text{def}}{=} \frac{\epsilon}{3k^2}$. That is, given the $\rho_j^{\Pi,\vec{\beta}}$'s and $\varrho_{j,j'}^{\Pi,\vec{\beta}}$'s, we ask whether there are non-negative a_j's and $b_{j,j'}$'s which, in addition to the above admissibility conditions, also satisfy

$$|a_j - \rho_j^{\Pi,\vec{\beta}}| \le 2\epsilon' \quad (\forall j) \quad\text{and}\quad |b_{j,j'} - \varrho_{j,j'}^{\Pi,\vec{\beta}}| \le 2\epsilon' \quad (\forall j,j')$$

We note that the problem of whether a set of densities is $2\epsilon'$-close to an admissible set for Φ can be 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 C). Note that this time-bound is dominated by the number of partitions examined in Step (4) (of Figure 9).

Proof of Theorem 9.1. Recall that $\epsilon' = \frac{\epsilon}{3k^2}$. We first note that if $G \in \mathcal{GP}_\Phi$, then by Lemma 9.3, with probability $1-\frac{\delta}{2}$ over the choice of $\overline{U}$, 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 (that is, to the set of densities defined by a partition $(H_1,\ldots,H_k)$

Testing Algorithm for $\mathcal{GP}_\Phi$

Let $\epsilon' \stackrel{\rm def}{=} \frac{\epsilon}{3k^2}$.

1. *Choose* $\ell = \frac{4}{\epsilon'}$ *sets* $\mathrm{U}^1, \ldots, \mathrm{U}^\ell$ *each of size* $t = \Theta\left(\left(\frac{1}{\epsilon'}\right)^2 \log \frac{k}{\epsilon'\delta}\right)$, *where* U^i *is chosen uniformly in* $\mathrm{V} \setminus \mathrm{V}^i$. *Let* $\overline{\mathrm{U}} = \langle \mathrm{U}^1, \ldots, \mathrm{U}^\ell \rangle$.
2. *Uniformly choose a set* $\mathrm{S} = \{s_1, \ldots, s_m\}$ *of size* $m = \Theta\left(\left(\frac{O(1)}{\epsilon'}\right)^{2k+5} \cdot \log(1/(\epsilon'\delta))\right)$.

   For $1 \le i \le \ell$, *let* $\mathrm{S}^i \stackrel{\rm def}{=} \mathrm{V}^i \cap \mathrm{S}$.
3. *For every setting of* $\vec{\beta} = \left\langle \vec{\beta}^1, \ldots, \vec{\beta}^\ell \right\rangle$ *and for each of the sequence of partitions* $\Pi = \left\langle (\mathrm{U}_1^1, \mathrm{U}_k^1), \ldots (\mathrm{U}_1^\ell, \mathrm{U}_k^\ell) \right\rangle$ *(as in Figure 8),*
   construct a partition $\mathcal{S}^{\Pi,\vec{\beta}} = \left(\mathrm{S}_1^{\Pi,\vec{\beta}}, \ldots, \mathrm{S}_k^{\Pi,\vec{\beta}}\right)$ *as follows:*

   (a) *For* $i = 1 \ldots \ell$ *do:*

      i. *For each* $v \in \mathrm{S}^i$, *and for* $j \in \{1, \ldots, k\}$, *let* $\hat{\gamma}_j(v) \stackrel{\rm def}{=} \frac{1}{t}\left(\left|\Gamma(v) \cap \mathrm{U}_j^i\right|\right)$.

      ii. *For each* $\vec{\alpha} \in \Lambda$ *let the* $\vec{\alpha}$*-cluster of* S^i, *denoted* $\mathrm{S}^{i,\vec{\alpha}}$ *be defined as follows:*
      $\mathrm{S}^{i,\vec{\alpha}} \stackrel{\rm def}{=} \left\{v \in S^i : \forall j,\ \alpha_j - \epsilon'/32 < \hat{\gamma}_j(v) \le \alpha_j + \epsilon'/32\right\}$.

      iii. *For each cluster* $\mathrm{S}^{i,\vec{\alpha}}$, $\vec{\alpha} \in \Lambda$, *let* $\mathrm{S}_1^{i,\vec{\alpha}}$ *be the first* $\beta_1^{i,\vec{\alpha}} \cdot |\mathrm{S}^{i,\vec{\alpha}}|$ *vertices in* $\mathrm{S}^{i,\vec{\alpha}}$, *let* $\mathrm{S}_2^{i,\vec{\alpha}}$ *be the next* $\beta_2^{i,\vec{\alpha}} \cdot |\mathrm{S}^{i,\vec{\alpha}}|$ *vertices in* $\mathrm{S}^{i,\vec{\alpha}}$ *and in general, for* $j < k$ *let* $\mathrm{S}_j^{i,\vec{\alpha}}$ *be the first* $\beta_j^{i,\vec{\alpha}} \cdot |\mathrm{S}^{i,\vec{\alpha}}|$ *vertices in* $\mathrm{S}^{i,\vec{\alpha}} \setminus \bigcup_{j'<j} \mathrm{S}_{j'}^{i,\vec{\alpha}}$, *and let* $\mathrm{S}_k^{i,\vec{\alpha}}$ *be the remaining vertices in* $\mathrm{S}^{i,\vec{\alpha}}$ *(where in case the resulting numbers are not integers, they are rounded, alternating between rounding up and down).*

      iv. *For each* $j \in \{1, \ldots, k\}$, *let* $\mathrm{S}_j^i = \bigcup_{\alpha \in \Lambda} \mathrm{S}_j^{i,\vec{\alpha}}$.

   (b) *For each* $j \in \{1, \ldots, k\}$, *let* $\mathrm{S}_j = \bigcup_{i=1}^{\ell} \mathrm{S}_j^i$.
4. *For each partition,* $\mathcal{S}^{\Pi,\vec{\beta}}$, *let* $\rho_j^{\Pi,\vec{\beta}} \stackrel{\rm def}{=} \frac{1}{m} \cdot |\mathcal{S}^{\Pi,\vec{\beta}}|$ $(\forall j)$
   and $\varrho_{j,j'}^{\Pi,\vec{\beta}} \stackrel{\rm def}{=} \frac{1}{m/2} \cdot \left|\{(s_{2r-1}, s_{2r}) \in \mathrm{E}(\mathrm{S}_j^{\Pi,\vec{\beta}}, S_{j'}^{\Pi,\vec{\beta}} \cup \mathrm{E}(\mathrm{S}_{j'}^{\Pi,\vec{\beta}}, S_j^{\Pi,\vec{\beta}}\}\right|$ $(\forall j, j')$.
   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.* Φ *then output* Accept, *otherwise output* Reject.

Figure 9: Testing Algorithm for $\mathcal{GP}_\Phi$

for which Equations (33) and (34) hold). 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.[20]

Suppose, on the 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 in two steps as follows. Step 1: We move up to $3k\epsilon' \cdot N$ vertices (according to the guaranteed admissible vertex densities), so that two conditions hold. First, the number of vertices in each component of the partition is between the required lower and upper bounds. Second, for each j, j', the number of vertex pairs, one in component j and one in component j' is at least as required by the edge-density lower bound $\varrho^{\text{LB}}_{j,j'}$. Step 2: We omit/add up to $3k^2\epsilon' \cdot N^2 = \epsilon N^2$ edges (according to the required edge-densities upper and lower bounds). Hence, we obtain a graph in $\mathcal{GP}_\Phi$ in contradiction to the hypothesis. The first part of Theorem 9.1 follows from the lemma below, which analogously to Lemmas 7.7 and 8.12 deals with the performance of the partitions of S as an "approximation" to the corresponding partitions of V.

For a fixed $\overline{\mathrm{U}}$ (which we assume is good, as defined in Lemma 9.3), 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 partitions $\mathcal{S}^{\Pi,\vec{\beta}}$'s that are generated by the Testing Algorithm when using the same $\overline{\mathrm{U}}$. Let n_p, n_a and n_b be upper bounds on the number of sequences of partitions Π, number of clusters (one per each i and each setting of $\vec{\alpha}$), and number of settings of $\vec{\beta}$, respectively. Thus $n_p = k^{\ell t}$, $n_a = \ell \cdot \left(\frac{16}{\epsilon'}\right)^k = \left(\frac{O(1)}{\epsilon'}\right)^{k+1}$, and $n_b = \left(\frac{8}{\epsilon'}\right)^{k \cdot n_a} = \exp\left(\log(1/\epsilon') \cdot \left(\frac{O(1)}{\epsilon'}\right)^{k+1}\right)$,

Lemma 9.4 *For a fixed $\overline{\mathrm{U}}$, Π, and $\vec{\beta}$, let $\mathcal{V}^{\Pi,\vec{\beta}}$ be as defined in the Graph Partitioning algorithm, when executed with approximation parameter $\epsilon' = \epsilon/3k^2$. Let S be a uniformly chosen sample of size*

$$m = \Omega\left(\frac{n_a}{\epsilon'} \cdot \frac{\log(k \cdot n_p \cdot n_a \cdot n_b)/\delta)}{(\epsilon')^2}\right)$$

and let $\mathcal{S}^{\Pi,\vec{\beta}}$, $\{\rho^{\Pi,\vec{\beta}}_j\}$ and $\{\varrho^{\vec{\beta}}_{j,j'}\}$ be as defined by the Testing algorithm. Then

$$\Pr_{\mathrm{S}}\left[\exists j : \left|\rho^{\Pi,\vec{\beta}}_j - \frac{|\mathrm{V}^{\Pi,\vec{\beta}}_j|}{N}\right| > \epsilon'\right] < \frac{\delta}{4 \cdot n_p \cdot n_b}$$

and

$$\Pr_{\mathrm{S}}\left[\exists j, j' : \varrho^{\Pi,\vec{\beta}}_{j,j'} - \frac{\left|\mathrm{E}\left(\mathrm{V}^{\Pi,\vec{\beta}}_j, \mathrm{V}^{\Pi,\vec{\beta}}_{j'}\right)\right|}{N^2} > \epsilon'\right] < \frac{\delta}{4 \cdot n_p \cdot n_b}$$

Proof: Let $\delta' \stackrel{\text{def}}{=} \frac{\delta}{4 \cdot n_p \cdot n_b}$. and let $\epsilon_1 \stackrel{\text{def}}{=} \epsilon'/24$. For each $i \in \{1, \ldots, \ell\}$ and $\vec{\alpha}$, let $\mathrm{V}^{i,\vec{\alpha}}$ and $\mathrm{V}^{i,\vec{\alpha}}_j$ be as defined in the Graph Partitioning Algorithm, and let $\mathrm{S}^{i,\vec{\alpha}}$ and $\mathrm{S}^{i,\vec{\alpha}}_j$ be as defined in the Testing algorithm (for the fixed $\overline{\mathrm{U}}$, Π and $\vec{\beta}$ considered in the lemma). By an additive Chernoff bound, for each cluster,

$$\Pr_{\mathrm{S}}\left[\left|\frac{|\mathrm{S}^{i,\vec{\alpha}}|}{m} - \frac{|\mathrm{V}^{i,\vec{\alpha}}|}{N}\right| > \frac{\epsilon'}{2}\right] \;<\; \exp\left(-\Omega(m \cdot (\epsilon')^2)\right) \;<\; \frac{\delta'}{4n_a} \tag{48}$$

[20] Here is where the notion of admissible solution plays an important role. As noted in the beginning of this section, it would not suffice to check whether no k-way partition of V is an $O(\epsilon/k^2)$-good approximation (of the partition problem). This is true since the existence of a good approximation does not ensure that the distance of G to the class is of the same order as the approximation.

Let V_j be the union of the $\mathrm{V}_j^{i,\vec{\alpha}}$'s and let S_j be the union of the $\mathrm{S}_j^{i,\vec{\alpha}}$'s (as defined by the algorithm). Recall that for each j, $\lfloor \vec{\beta}^{i,\vec{\alpha}} \cdot |\mathrm{V}^{i,\vec{\alpha}}| \rfloor \leq |\mathrm{V}_j^{i,\vec{\alpha}}| \leq \lceil \vec{\beta}^{i,\vec{\alpha}} \cdot |\mathrm{V}^{i,\vec{\alpha}}| \rceil$, and similarly, $\lfloor \vec{\beta}^{i,\vec{\alpha}} \cdot |\mathrm{S}^{i,\vec{\alpha}}| \rfloor \leq |\mathrm{S}_j^{i,\vec{\alpha}}| \leq \lceil \vec{\beta}^{i,\vec{\alpha}} \cdot |\mathrm{V}^{i,\vec{\alpha}}| \rceil$. Thus the effects of rounding can be taken into account by assuming an arbitrary placement of $2 \cdot n_a$ vertices. Equation (48) together with this bound on the rounding effects imply that for every j, with probability at least $1 - \frac{\delta'}{4}$,

$$\frac{|\mathrm{S}_j|}{m} \;=\; \sum_{i,\vec{\alpha}} \beta_j^{i,\vec{\alpha}} \cdot \frac{|\mathrm{S}^{i,\alpha}|}{m} \pm \frac{n_a}{m} \;=\; \sum_{i,\vec{\alpha}} \beta_j^{i,\vec{\alpha}} \cdot \left(\frac{|\mathrm{V}^{i,\alpha}|}{N} \pm \frac{\epsilon'}{2} \right) \pm \frac{2n_a}{m} \;=\; \frac{|\mathrm{V}_j|}{N} \pm \epsilon'$$

so we have proved the first part of the lemma.

We say that a cluster $\mathrm{V}^{i,\vec{\alpha}}$ is *significant*, if it contains at least $\epsilon_1 \cdot N/n_a$ vertices. Otherwise it is insignificant. Note that the total number of vertices belonging to insignificant clusters is at most $\epsilon_1 \cdot N$. For each significant cluster and each j, let $\overline{\mathrm{V}}_j^{i,\vec{\alpha}}$ be the same as $\mathrm{V}_j^{i,\vec{\alpha}}$ *except* that it does not contain the first and the last $\epsilon_1 \cdot |\mathrm{V}_j^{i,\vec{\alpha}}|$ vertices in the cluster (where first and last are with respect to the lexicographic order on vertices). For each i and $\vec{\alpha}$, let $m_{i,\vec{\alpha}} \stackrel{\mathrm{def}}{=} |\mathrm{S}^{i,\vec{\alpha}}|$. By a multiplicative Chernoff bound, with probability at least $1 - \delta'/12$, for every significant cluster,

$$m_{i,\vec{\alpha}} \;\geq\; \frac{1}{2} \cdot m \cdot \frac{|\mathrm{V}^{i,\vec{\alpha}}|}{N} \;\geq\; \frac{1}{2} \cdot m \cdot \frac{\epsilon_1}{n_a} \;=\; \Omega\left(\frac{\log(k \cdot n_b/\delta')}{(\epsilon')^2} \right)$$

Therefore, by an additive Chernoff bound, for each significant cluster, and each j,

$$\Pr_{\mathrm{S}} \left[\left| \frac{\left| \mathrm{S}_j^{i,\vec{\alpha}} \cap \overline{\mathrm{V}}_j^{i,\vec{\alpha}} \right|}{m_{i,\alpha}} - (\vec{\beta}_j^{i} - 2\epsilon_1) \right| > 2\epsilon_1 \right] < \frac{\delta'}{12 \cdot k \cdot n_a} \tag{49}$$

Thus, putting aside an error probability of $\frac{\delta'}{12}$, assume from now on that in fact for every significant cluster and every j,

1. $|\mathrm{S}_j^{i,\vec{\alpha}} \cap \overline{\mathrm{V}}_j^{i,\vec{\alpha}}| \leq \vec{\beta}_j^{i,\alpha} \cdot m_{i,\vec{\alpha}}$ from which it follows (by definition of $\mathrm{S}_j^{i,\vec{\alpha}}$) that for every j, $\mathrm{S}_j^{i,\vec{\alpha}} \cap \overline{\mathrm{V}}_j^{i,\vec{\alpha}} = \mathrm{S}^{i,\vec{\alpha}} \cap \overline{\mathrm{V}}_j^{i,\vec{\alpha}}$;

2. $|\mathrm{S}_j^{i,\vec{\alpha}} \cap \overline{\mathrm{V}}_j^{i,\vec{\alpha}}| \geq (\vec{\beta}_j^{i,\vec{\alpha}} - 4\epsilon_1) \cdot m_{i,\vec{\alpha}}$, from which it follows that $|\mathrm{S}_j^{i,\vec{\alpha}} \cap \overline{\mathrm{V}}_j^{i,\vec{\alpha}}| \geq (\vec{\beta}_j^{i,\vec{\alpha}} - 4\epsilon_1) \cdot m_{i,\vec{\alpha}}$.

Next observe that for each j, j',

$$\begin{aligned} \{(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{S}_j, \mathrm{S}_{j'})\} \quad &= \quad \{(s_{2k-1}, s_{2k}) \in \mathrm{E}(\mathrm{S}_j \cap \overline{\mathrm{V}}_j, \mathrm{S}_{j'} \cap \overline{\mathrm{V}}_{j'})\} \\ &\bigcup \quad \{(s_{2k-1}, s_{2k}) \in (\mathrm{E}(\mathrm{S}_j \setminus \overline{\mathrm{V}}_j, \mathrm{S}_{j'}) \cup \mathrm{E}(\mathrm{S}_j, \mathrm{S}_{j'} \setminus \overline{\mathrm{V}}_{j'}))\} \end{aligned} \tag{50}$$

By Item (1) above and an additive Chernoff bound we get that for each j, j'

$$\Pr_{\mathrm{S}} \left[\left| \frac{\left| \{(s_{2k-1}, s_{2i}) \in \mathrm{E}(\mathrm{S}_j \cap \overline{\mathrm{V}}_j, \mathrm{S}_{j'} \cap \overline{\mathrm{V}}_{j'})\} \right|}{m/2} - \frac{\left| \mathrm{E}(\overline{\mathrm{V}}_j, \overline{\mathrm{V}}_{j'}) \right|}{N^2} \right| \;>\; \frac{\epsilon'}{4} \right] \;<\; \frac{\delta'}{12k^2} \tag{51}$$

By definition of $\overline{\mathrm{V}}_j$ and $\overline{\mathrm{V}}_{j'}$, we have that

$$\frac{|\mathrm{E}(\mathrm{V}_j, \mathrm{V}_{j'})|}{N^2} - \frac{|\mathrm{E}(\overline{\mathrm{V}}_j, \overline{\mathrm{V}}_{j'})|}{N^2} \;\leq\; 4\epsilon_1 \;<\; \frac{\epsilon'}{4} \tag{52}$$

By Item (2) above, we know that for every *significant* cluster and every j, $|\mathrm{S}_j^{i,\vec{\alpha}} \setminus \overline{\mathrm{V}}_j^{i,\vec{\alpha}}| \leq 4\epsilon_1 m_{i,\vec{\alpha}}$. By a multiplicative Chernoff bound, with probability at least $1-\frac{\delta'}{12}$, the sum of $m_{i,\vec{\alpha}}$ taken over all *insignificant* clusters is at most $2\epsilon_1 \cdot m$. Thus,

$$\sum_{i,\vec{\alpha}} \left|\mathrm{S}_j^{i,\vec{\alpha}} \setminus \mathrm{V}_j^{i,\vec{\alpha}}\right| \;\leq\; r\epsilon_1 \cdot \sum_{i,\vec{\alpha}} m_{i,\vec{\alpha}} + 2\epsilon_1 \cdot m \;=\; 6\epsilon_1 m$$

Therefore,

$$\begin{aligned} & \frac{\left|\{(s_{2k-1}, s_{2k}) \in (\mathrm{E}(\mathrm{S}_j \setminus \overline{\mathrm{V}}_j, \mathrm{S}_{j'}) \cup \mathrm{E}(\mathrm{S}_j, \mathrm{S}_{j'} \setminus \overline{\mathrm{V}}_{j'}))\right|}{m/2} \\ \leq\; & \frac{\left|\{k:\; s_{2k-1} \in \mathrm{S}_j \setminus \overline{\mathrm{V}}_j \text{ or } s_{2k} \in \mathrm{S}_{j'} \setminus \overline{\mathrm{V}}_{j'})\right|}{m/2} \;\leq\; 12\epsilon_1 \;=\; \frac{\epsilon'}{2} \end{aligned} \tag{53}$$

Summing up all the probabilities of errors and combining the bounds of Equations (51)–(53), the lemma follows. ■

Proof of Theorem 9.2. In order to get a partitioning algorithm whose running time is as stated in the theorem, we first run the Testing Algorithm and find a choice of Π and $\vec{\beta}$ (in time independent of N) for which Step (4) of the Testing Algorithm accepts. Next we use this choice to partition all of V by running Step (2) of the Partition Algorithm only on this choice.. As we have shown in the proof of Theorem 9.1, 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 cause the test to reject. 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. ■

10 Extensions, Limitations and Beyond

10.1 Extensions and Limitations of the Above Algorithms

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 6.9 shows that (edge) 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 Max-Cut (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 a directed version of 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. For any two disjoint sets of vertices, W_1 and W_2, let $\mu(\mathrm{W}_1, \mathrm{W}_2)$ denote density of the edges crossing from W_1 to W_2 (i.e., the number of edges crossing divided by N^2). 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 $|\mathrm{E}(v, \mathrm{W}_2)| > |\mathrm{E}(\mathrm{W}_1, v)|$, 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 $\varrho^{\text{LB}}_{j,j'}$ and $\varrho^{\text{UB}}_{j,j'}$ to differ from $\varrho^{\text{LB}}_{j',j}$ and $\varrho^{\text{UB}}_{j',j}$, 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, 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 $\Psi : \mathrm{V}(\mathrm{G})^2 \mapsto [0,1]$ a *product distribution* if there exists a distribution on vertices $\psi : \mathrm{V}(\mathrm{G}) \mapsto [0,1]$ so that $\Psi(u,v) = \psi(u) \cdot \psi(v)$. 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 Ψ, we sample V(G) according to the distribution ψ. Note that the algorithm need not "know" ψ; 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 Ψ). 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 $\mathrm{V}(\mathrm{G}')$ is almost the same as distribution Ψ 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 Ψ 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 Ψ) 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 $\{1, \ldots, N\}$ 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. Furthermore, 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).

ON THE POSSIBILITY OF WORKING IN $\text{poly}(1/\epsilon)$ TIME. The algorithm for Bipartite Testing works in $\text{poly}(1/\epsilon)$-time (see Theorem 6.2), whereas all the other testers presented in this paper work in time exponential in $\text{poly}(1/\epsilon)$. 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 $\text{poly}(1/\epsilon)$ 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 $n \stackrel{\text{def}}{=} |\mathrm{V(G)}|$ 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 $\epsilon = 1/\text{poly}\log|\mathrm{V(G')}|$.

ON ONE-SIDED FAILURE PROBABILITY. The testers for Bipartite and k-Colorability always accept graphs which have the property. In contrast, all other testers presented in this paper may fail to accept a yes-instance (yet, with probability at most δ). We claim that non-zero failure probability, in these cases, is inevitable. Consider for example 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 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.[21] We conclude that there is a fundamental difference between testing k-Colorability and testing ρ-Clique.

IMPOSSIBILITY OF LEARNING WITH QUERIES. All the above classes of (graph-)functions are not learnable, not even under uniform distribution and when allowing (membership) 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 (functions) in each class. 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 (i.e., be labeled 1 or 0

[21] 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.)

by the corresponding function).

Extension to Weighted Graphs (with bounded weights). Let $G = (V, E)$ be a (simple undirected) graph, and $\omega : E \mapsto [0, B]$ be a weight function assigning each edge $e \in E$ 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 $f_{G,\omega} : V \times V \to [0, B]$, where $f_{G,\omega}(v_1, v_2) = 0$ if $(v_1, v_2) \notin E$ 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 $f_G(v_1, v_2)$. 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 $f_{G,\omega}$ and $f_{G',\omega}$, to be $\frac{1}{N^2}\sum_{v_1,v_2 \in V} |f_{G,\omega}(v_1, v_2) - f_{G',\omega}(v_1, v_2)|$. 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 $G = (V, E, \omega)$ has a cut of weight at least ρ, if there exists a partition (V_1, V_2) of V such that $\frac{1}{N^2}\sum_{v_1 \in V_1, v_2 \in V_2} 2f_G(v_1, v_2) \geq \rho$. 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.[22] 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-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 distance parameter 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.

10.2 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 be quite challenging.

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 10.1 (testing almost trivial classes): *Let $\alpha > 0$ be a constant and $\mathcal{C}$ a class of graphs*

[22] 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 (W_1, W_2) if $\frac{1}{N}|\sum_{w_1 \in W_1} f_G(v, w_1) - \sum_{w_2 \in W_2} f_G(v, w_2)|$ is non-negligible. Similarly to the unweighted case we shall use partitions (U_1, 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.

so that for every graph, G,

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

Then, $\mathcal{C}$ can be tested by using at most $\text{poly}(\epsilon^{-1}\log(1/\delta))$ *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 $\kappa(\cdot)$-vertex Dominating Set*, for a given $\kappa(\cdot)$ (add $\leq |\mathrm{V}(\mathrm{G})| - \kappa(|\mathrm{V}(\mathrm{G})|)$ 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).[23] 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 $\kappa(\cdot)$-vertex Dominating Set* and *Graphs not having Perfect Matching* (e.g., by removing edges to make one vertex isolated).

Proof: Let $\Delta(N) \stackrel{\text{def}}{=} \max_{\mathrm{G}:|\mathrm{V}(\mathrm{G})|=N}\{\text{dist}(\mathrm{G},\mathcal{C})\}$ and suppose that $\Delta(N)$ or a good upper bound on it is known. (One may always use $\Delta(N) = N^{-\alpha}$.) Given oracle access to an N-vertex graph G (and a distance parameter ϵ), if $\epsilon > \Delta(N)$ then the tester always accepts. Otherwise, the tester inspects all $N^2 = \text{poly}(1/\epsilon)$ 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 (i.e., the graph is sufficiently sparse).

Proposition 10.2 (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* $\text{poly}(\epsilon^{-1}\log(1/\delta))$ *labeled random examples, where ϵ and δ are the distance and confidence parameters.*

Classes which satisfy the hypothesis of the proposition include: *Trees*, *Forests* and *Planar Graphs* (all with $|\mathrm{E}(\mathrm{G})| = O(|\mathrm{V}(\mathrm{G}))|)$.

Proof: Let $\rho(N) \stackrel{\text{def}}{=} \max_{\mathrm{G}\in\mathcal{C}:|\mathrm{V}(\mathrm{G})|=N}\{|\mathrm{E}(\mathrm{G})|/N^2\}$ and suppose that $\rho(N)$ or a good upper bound on it is known. (One may always use $\rho(N) = N^{-\alpha}$.) Given oracle access to an N-vertex graph G (and parameters ϵ, δ), if $\epsilon > 3\rho(N)$ then the tester takes a sample of $t \stackrel{\text{def}}{=} O(\epsilon^{-2}\log(1/\delta))$ labeled random examples and accepts iff it has seen at most $(\rho(N) + (\epsilon/3))\cdot t$ edges. Otherwise, the tester inspects all $N^2 = \text{poly}(1/\epsilon)$ 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 the graphs in $\mathcal{C}$ have edge density at most $\rho(N)$, whereas graphs that are ϵ-away from $\mathcal{C}$ have edge density at least $\epsilon - \rho(N)$ which for $\epsilon > 3\rho(N)$ is more than $\rho(N) + 2\epsilon/3$. 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. ■

[23] 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.

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

Proof: Given oracle access to an N-vertex graph G (and parameters ϵ, δ), the test takes a sample, S, of $O(\epsilon^{-1}\log(1/\delta))$ many vertices and for each $v \in \mathrm{S}$ makes $O(\epsilon^{-2}\log(1/(\epsilon\delta)))$ queries of the form "is $(v, w) \in \mathrm{E(G)}$", where $w \in \mathrm{V(G)}$ 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 $\epsilon' = \epsilon/13$, 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 D) which asserts that a graph G′ in which the difference between the maximum and minimum degree is bounded above by $\epsilon''|\mathrm{V(G')}|$ 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 10.4 (testing min-cut): *The class of graphs* G *with minimum cut at least* $\kappa = \kappa(|\mathrm{V(G)}|)$ *can be tested by using at most* $\mathrm{poly}(\epsilon^{-1}\log(1/\delta))$ *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 $\kappa = \kappa(N)$. That is, to test that the minimum cut is at least κ we sample sufficiently many vertices, approximate their degree according to the sample, and accept iff all estimated degrees are above, say, $\kappa - \frac{\epsilon}{2}N$. The analysis utilizes the observation that at most $O(N \log N)$ edges must be added to an N-vertex graph of minimum degree d in order to make it have min-cut at least d. 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 d. Thus, all that is needed is to guarantee d edge-disjoint paths between each pair of such subsets. This can be done easily, by designating d special vertices in the graph, and randomly connecting 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 d-matching. Thus, we obtain d edge-disjoint paths between each pair of neighborhoods, which implies that the augmented graph is d-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 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 probably 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 4.1, we show that there are graph properties requiring inspection of a constant fraction of all possible vertex-pairs.

Proposition 10.5 *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 4.1, 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 10.6 *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 constant distance parameter $\epsilon < 1/2$ and confidence parameter $\delta < 1/2$, and when allowing the algorithm to make queries and use unlimited computing time.*

This proposition subsumes Proposition 10.5.
Proof: We adapt the proof of Proposition 10.5, by considering, for each N, all graphs which arise for particular "pseudorandom" sequences. Specifically, we consider $\binom{N}{2}$-long sequences taken from a $\frac{1}{2} \cdot 2^{-t}$-biased sample space (cf., [NN93] or [AGHP92]), where $t \stackrel{\text{def}}{=} \frac{1}{100}N^2$. Efficiently constructible sample spaces of size $(2^t \cdot N)^5$ 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, and so there are there are $\exp\left(\frac{1}{20}N^2\right)$ such graphs and each graph is specified by a sequence of length $O(t + \log N) = \text{poly}(N)$. 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 from the uniform distribution by at most $\frac{1}{2}$ (i.e., using the notation of Proposition 4.1,

$\delta_{\mathrm{S}}(\mathcal{G}) < \frac{1}{2}$ for every S of size t). 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). ■

Acknowledgments

We wish to thank Noga Alon, Michel Goemans, Ravi Kannan, David Karger and Madhu Sudan for useful discussions. We are also grateful to two anonymous referees for their helpful comments.

References

[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. Håstad, 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.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. Technical Report TR–682, LCS/MIT, 1996. Extended abstract in *28th STOC*, pp. 639-648, 1996.

[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.

[EKK+98] F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, 1998.

[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.

[GR97a] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 406–415, 1997.

[GR97b] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. Manuscript, 1997.

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

[Hås96a] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the Thirty-Seventh Annual Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[Hås96b] J. Håstad. Testing of the long code and hardness for clique. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 11–19, 1996.

[Hås97] J. Håstad. Getting optimal in-approximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 1–10, 1997.

[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.

[KR98] M. Kearns and D. Ron. Testing problems with sub-learning sample complexity. Manuscript, 1998.

[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.

[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.

[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. Szemerédi. Regular partitions of graphs. In *Proceedings, Colloque Inter. CNRS*, pages 399–401, 1978.

[Tre97] L. Trevisan. Recycling queries in pcps and in linearity tests. Manuscript, 1997.

[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.

A Recurring Notation

For an integer b, we let $[b] \stackrel{\text{def}}{=} \{1, \ldots, b\}$.
For any three rational number, α, β and γ, we let $\alpha = \beta \pm \gamma$ stand for $\beta - \gamma \leq \alpha \leq \beta + \gamma$.
A graph is typically denoted $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ and N typically denotes the number of vertices in G. Furthermore, typically $\mathrm{V} = \{1, 2, ..., N\}$.

B Useful Inequalities

Below are several important inequalities that are used throughout the paper.

Theorem B.1 (Markov's Inequality) *Let X be a random variable assuming only non-negative values. Then for all $k \in \Re^+$,*

$$\Pr[X \geq k \cdot \mathrm{Exp}[X]] \leq \frac{1}{k}$$

Theorem B.2 (Chernoff Bounds) *Let $X_1, X_2, ...X_m$ be m independent random variables where $X_i \in [0,1]$. Let $p \stackrel{\text{def}}{=} \frac{1}{m}\sum_i \mathrm{Exp}[X_i]$. Then, for every $\gamma \in [0,1]$, the following bounds hold:*

- *(Additive Form)*

$$\Pr\left[\frac{1}{m} \cdot \sum_{i=1}^{m} X_i > p + \gamma\right] < \exp\left(-2\gamma^2 m\right)$$

  and

$$\Pr\left[\frac{1}{m} \cdot \sum_{i=1}^{m} X_i < p - \gamma\right] < \exp\left(-2\gamma^2 m\right)$$

- *(Multiplicative Form)*

$$\Pr\left[\frac{1}{m} \cdot \sum_{i=1}^{m} X_i > (1+\gamma)p\right] < \exp\left(-\gamma^2 pm/3\right)$$

  and

$$\Pr\left[\frac{1}{m} \cdot \sum_{i=1}^{m} X_i < (1-\gamma)p\right] < \exp\left(-\gamma^2 pm/2\right)$$

C Determining Closeness to an Admissible 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 $\rho_1^{\mathrm{LB}}, ..., \rho_k^{\mathrm{LB}}$, $\rho_1^{\mathrm{UB}}, ..., \rho_k^{\mathrm{UB}}$, $\varrho_{1,1}^{\mathrm{LB}}, ..., \varrho_{k,k}^{\mathrm{LB}}$, $\varrho_{1,1}^{\mathrm{UB}}, ..., \varrho_{k,k}^{\mathrm{UB}}$, and ϵ'.
- densities: $\rho_1, ..., \rho_k$ and $\varrho_{1,1}, ..., \varrho_{k,k}$.

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

$$\sum_{i=1}^{k} x_i = 1 \quad \text{and} \quad \sum_{i,j=1}^{k} y_{i,j} \leq 1 \tag{54}$$

$$\rho_i^{\text{LB}} \le x_i \le \rho_i^{\text{UB}} \quad (\forall i\,) \quad \text{and} \quad \varrho_{i,j}^{\text{LB}} \le y_{i,j} \le \varrho_{i,j}^{\text{UB}} \quad (\forall i,j\,) \tag{55}$$

$$y_{i,i} \le x_i^2 \;\; (\forall i\,) \quad \text{and} \quad y_{i,j} \le 2\cdot x_i \cdot x_j \quad (\forall i \ne j\,) \tag{56}$$

$$|x_i - \rho_i| \le 2\epsilon' \quad \text{and} \quad |y_{i,j} - \varrho_{i,j}| \le 2\epsilon' \quad (\forall i,j\,). \tag{57}$$

We first observe that the corresponding lower and upper bounds in Eq. (55) and Eq. (57) 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. (55) and Eq. (57). This yields the following system of inequalities, where $L_i = \max\{\rho_i^{\text{LB}}, \rho_i - 2\epsilon'\}$, $\mathrm{U}_i = \min\{\rho_i^{\text{UB}}, \rho_i + 2\epsilon'\}$, $L_{i,i} = \max\{\varrho_{i,i}^{\text{LB}}, \varrho_{i,i} - 2\epsilon'\}$, and $L_{i,j} = \frac{1}{2}\cdot\max\{\varrho_{i,j}^{\text{LB}}, \varrho_{i,j} - 2\epsilon'\}$ $(i \ne j)$:

$$\sum_{i=1}^{k} x_i = 1 \tag{58}$$

$$L_i \le x_i \le U_i \quad (\forall i\,) \tag{59}$$

$$x_i \cdot x_j \ge L_{i,j} \quad (\forall i,j\,) \tag{60}$$

We first observe that Eq. (58)–(60) constitute a Convex Program; furthermore, its feasibility region, in case it is not empty, is a t-dimensional convex set, where $t \le k-1$. 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 $2^{-\text{poly}(k)\cdot L}$ (and is contained in $[0,1]^t$). Thus, the feasibility problem can be solved by exhaustive search in $\exp(\text{poly}(k)\cdot L)$-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 $2^{-\text{poly}(k)\cdot L}$.

REMARK: The feasibility problem can be solvable by the Ellipsoid Method (cf., [GLS88]) in $\text{poly}(k\cdot L)$-time, but this saving has little affect on our application.

D 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 D.1 *Let* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ *be a graph on* N *vertices with maximum degree* D *and minimum degree* d*, where* $D - d \le \epsilon N$*. 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 $D \le \frac{1}{2}(1+\epsilon)N$. 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 degree D' 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. Clearly, $D_t \leq D - t$, and $D_t - d_t \leq \epsilon N$. 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 $D_t \leq 0.5N - 2$. We let $H \stackrel{\text{def}}{=} \mathrm{G}_t$, $D' \stackrel{\text{def}}{=} D_t$, $d' \stackrel{\text{def}}{=} d_t$, 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. Following is a description of a typical step:

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, we 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 (u, w) be such an edge. Then we omit (u, w) from the graph, add the two edges (v, u) and (v, w) to the graph, 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. Note that, as before, if some vertex in either Y or Z has a positive s value the 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. ■