# Fast Distributed Algorithms for Testing Graph Properties

Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman(✉), and Yadu Vasudev

Department of Computer Science, Technion – Israel Institute of Technology, Haifa, Israel
{ckeren,eldar,gregory}@cs.technion.ac.il, yaduvasudev@gmail.com

**Abstract.** We provide a thorough study of *distributed property testing* – producing algorithms for the approximation problems of property testing in the CONGEST model. In particular, for the so-called *dense* graph testing model we emulate sequential tests for nearly all graph properties having 1-sided tests, while in the *general* and *sparse* models we obtain faster tests for triangle-freeness, cycle-freeness and bipartiteness, respectively. In addition, we show a logarithmic lower bound for testing bipartiteness and cycle-freeness, which holds even in the LOCAL model.

In most cases, aided by parallelism, the distributed algorithms have a much shorter running time as compared to their counterparts from the sequential querying model of traditional property testing. The simplest property testing algorithms allow a relatively smooth transitioning to the distributed model. For the more complex tasks we develop new machinery that may be of independent interest.

## 1 Introduction

The performance of many distributed algorithms naturally depends on properties of the underlying network graph. Therefore, an inherent goal is to check whether the graph, or some given subgraph, has certain properties. However, in some cases this is known to be hard, such as in the CONGEST model [29]. In this model, computation proceeds in synchronous rounds, in each of which every vertex can send an $O(\log n)$-bit message to each of its neighbors. Lower bounds for the number of rounds of type $\tilde{\Omega}(\sqrt{n} + D)$ are known for *verifying* many global graph properties, where $n$ is the number of vertices in the network, $D$ is its diameter and $\tilde{\Omega}$ hides polylogarithmic factors (see, e.g. Das-Sarma et al. [34]).

To overcome such difficulties, we adopt the relaxation used in graph property testing, as first defined in [17,19], to the distributed setting. That is, rather than aiming for an exact answer to the question of whether the graph $G$ satisfies a certain property $P$, we settle for distinguishing the case of satisfying $P$ from the case of being $\epsilon$-*far* from it, for an appropriate measure of being far.

Apart from its theoretical interest, this relaxation is motivated by the common scenario of having distributed algorithms for some tasks that perform better

Supported in part by the Israel Science Foundation (grant 1696/14).

given a certain property of the network topology, or given that the graph *almost* satisfies that property. For example, Hirvonen et al. [23] show an algorithm for finding a large cut in triangle-free graphs (with additional constraints), and for finding an $(1-\epsilon)$-approximation if at most an $\epsilon$ fraction of all edges are part of a triangle. Similarly, Pettie and Su [30] provide fast algorithms for coloring triangle-free graphs.

We construct fast distributed algorithms for testing various graph properties. An important byproduct of this study is a toolbox that we believe will be useful in other settings as well.

## 1.1 Our Contributions

We provide a rigorous study of property testing methods in the realm of distributed computing under the CONGEST model. We construct *1-sided error distributed $\epsilon$-tests*, in which if the graph satisfies the property then all vertices output `accept`, and if it is $\epsilon$-far from satisfying the property then at least one vertex outputs `reject` with probability at least 2/3. Using the standard amplification method of invoking such a test $O(\log n)$ times and having a vertex output `reject` if there is at least one invocation in which it should output `reject`, gives rejection with higher probability at the price of a multiplicative $O(\log n)$ factor for the number of rounds.

The definition of a graph being $\epsilon$-far from satisfying a property is roughly one of the following (see the preliminaries section in the full version, [8], for precise definitions): (1) Changing any $\epsilon n^2$ entries in the adjacency matrix does not give a graph that satisfies the property (dense model), or (2) changing any $\epsilon \cdot \max\{n, m\}$ entries in the adjacency matrix does not give a graph that satisfies the property, where $m$ is the number of edges (general model). A particular case here is when the degrees are bounded by some constant $d$, and any resulting graph must comply with this restriction as well (sparse model).

In a *sequential $\epsilon$-test*, access to the input is provided by queries, whose type depends on the model. In the dense model these are asking whether two vertices $v, u$ are neighbors, and in the general and sparse models these can be either asking what the degree of a vertex $v$ is, or asking what the $i$-th neighbor of $v$ is (the ordering of neighbors is arbitrary). While a sequential $\epsilon$-test can touch only a small handful of vertices with its queries, in a distributed test the lack of ability to communicate over large distances is offset by having all $n$ vertices operating in parallel.

Our first contribution is a general scheme for a near-complete emulation in the distributed context of $\epsilon$-tests originating from the dense graph model (Sect. 2). This makes use of the fact that in the dense model all (sequential) testing algorithms can be made *non-adaptive*, which roughly means that queries do not depend on responses to previous queries (see the preliminaries section in the full version for definition). In fact, such tests can be made to have a very simple structure, allowing the vertices in the distributed model to "band together" for an emulation of the test. There is only one additional technical condition (which we define in Sect. 2), since in the distributed model we cannot handle properties whose counter-examples can be "split" to disjoint graphs. For example, the distributed model cannot hope to handle the property of the graph having no disjoint union of two triangles, a property for which there exists a test in the dense model.

**Theorem 1.** *Any $\epsilon$-test in the dense graph model for a non-disjointed property that makes $q$ queries can be converted to an $O(q^2)$-round distributed $\epsilon$-test.*

We next move away from the dense graph model to the sparse and general models, that are sometimes considered to be more realistic. In the general model, there exists no test for the property of containing no triangle that makes a number of queries independent of the number of graph vertices [2]. Here the distributed model can do better, because the reason for this deficiency is addressed by having all vertices operate concurrently. In Sect. 3 we adapt the interim lemmas used in the best testing algorithm constructed in [2], and construct a distributed algorithm whose number of rounds is independent of $n$.

**Theorem 2.** *There is a distributed $\epsilon$-test in the general graph model for triangle-freeness, that requires $O(\epsilon^{-2})$ rounds.*

The sparse and general models inherently require *adaptive* property testing algorithms, since there is no other way to trace a path from a given vertex forward, or follow its neighborhood. Testing triangle-freeness sequentially uses adaptivity only to a small degree. However, other problems in the sparse and general models, such as the one we explore next, have a high degree of adaptivity built into their sequential algorithms, and we need to take special care for emulating it in the distributed setting.

In the sparse model (degrees bounded by a constant $d$), we adapt ideas from the bipartiteness testing algorithm of [18], in which we search for odd-length cycles. Here again the performance of a distributed algorithm surpasses that of the test (a number of rounds polylogarithmic in $n$ vs. a number of queries which is $\Omega(\sqrt{n})$ – a lower bound that is given in [19]). The following is proved in Sect. 4.

**Theorem 3.** *There is a distributed $\epsilon$-test in the bounded degree graph model for the property of being bipartite, that requires $O(poly(\epsilon^{-1}\log(n\epsilon^{-1})))$ rounds.*

In the course of proving Theorem 3 we develop a method that we consider to be of independent interest[1]. The algorithm performs $2n$ random walks concurrently (two starting from each vertex). The parallel execution of random walks despite the congestion restriction is achieved by making sure that the walks have a uniform stationary distribution, and then showing that congestion is "close to average", which for the uniform stationary distribution is constant.

In Sect. 5 we show a fast test for cycle-freeness. This makes use of a combinatorial lemma that we prove, about cycles that remain in the graph after removing edges independently with probability $\epsilon/2$. The following summarizes our result for testing cycle-freeness.

[1] This was recently independently and concurrently devised in [16] for a different use.

**Theorem 4.** *There is a distributed $\epsilon$-test in the general graph model for cycle-freeness, that requires $O(\log n/\epsilon)$ rounds.*

We also prove lower bounds for testing bipartiteness and cycle-freeness that match the upper bound for the latter property. Roughly speaking, these are obtained by using the probabilistic method with alterations to construct graphs which are far from being bipartite or cycle-free, but all of their cycles are of length that is at least logarithmic. This technique bears some similarity to the classic result by Erdös [12], which showed the existence of graphs with large girth and large chromatic number. The following are given in Sect. 6.

**Theorem 5.** *Any distributed $1/100$-test in the bounded degree or general graph model for the property of being bipartite requires $\Omega(\log n)$ rounds of communication.*

**Theorem 6.** *Any distributed $1/100$-test in the bounded degree graph or general model for cycle-freeness requires $\Omega(\log n)$ rounds of communication.*

*Roadmap:* The paper is organized as follows. The remainder of this section consists of related work and historical background on property testing. The emulation of sequential tests for the dense model is given in Sect. 2. In Sect. 3 we give our distributed test for triangle-freeness. In Sect. 4 we provide a distributed test for bipartiteness, along with our new method of executing many random walks, and in Sect. 5 we give our test for cycle-freeness. Section 6 gives our logarithmic lower bounds for testing bipartiteness and cycle-freeness. We conclude with a short discussion in Sect. 7.

## 1.2 Related Work

The only previous work that directly relates to our distributed setting is due to Brakerski and Patt-Shamir [7]. They show a *tolerant* property testing algorithm for finding large (linear in size) *near-cliques* in the graph. An $\epsilon$-near clique is a set of vertices for which all but an $\epsilon$-fraction of the pairs of vertices have an edge between them. The algorithm is tolerant, in the sense that it finds a linear near-clique if there exists a linear $\epsilon^3$-near clique. That is, the testing algorithm considers two thresholds of being close to having the property (in this case – containing a linear size clique). We are unaware of any other work on property testing in this distributed setting.

Testing in a different distributed setting was considered in Arfaoui et al. [4]. They study testing for cycle-freeness, in a setting where each vertex may collect information of its entire neighborhood up to some distance, and send a short string of bits to a central authority who decides whether the graph is cycle-free.

Related to having information being sent to, or received by, a central authority, is the concept of proof-labelling schemes, introduced by Korman et al. [26] (for extensions see, e.g., Baruch et al. [5]). In this setting, each vertex is given some external label, and by exchanging labels the vertices need to decide whether a given property of the graph holds. This is different from our setting in which no information other than vertex IDs is available. Another setting that is related to proof-labelling schemes, but differs from our model, is the prover-verifier model of Foerster et al. [14].

Sequential property testing has the goal of computing without processing the entire input. The wider family of *local computation algorithms* (LCA) is known to have connections with distributed computing, as shown by Parnas and Ron [28] and later used by others. A recent study by Göös et al. [22] proves that under some conditions, the fact that a centralized algorithm can query distant vertices does not help with speeding up computation. However, they consider the LOCAL model, and their results apply to certain properties that are not influenced by distances.

Finding induced subgraphs is a crucial task and has been studied in several different distributed models (see, e.g., [9–11,25]). Notice that for *finding* subgraphs, having *many* instances of the desired subgraph can help speedup the computation, as in [10]. This is in contrast to algorithms that perform faster if there are *no* or only *few* instances, as explained above, which is why we test for, e.g., the property of being *triangle-free*, rather for the property of *containing* triangles. (Notice that these are not the same, and in fact every graph with $3/\epsilon$ or more vertices is $\epsilon$-close to having a triangle.)

Parallelizing many random walks was addressed in [1], where the question of graph covering via random walks is discussed. It is shown there that for certain families of graphs there is a substantial speedup in the time it takes for $k$ walks starting from the same vertex to cover the graph, as compared to a single walk. No edge congestion constraints are taken into account. In [35], it is shown how to perform, under congestion, a single random walk of length $L$ in $\tilde{O}(\sqrt{LD})$ rounds, and $k$ random walks in $\tilde{O}(\sqrt{kLD}+k)$ rounds, where $D$ is the diameter of the graph. Our method has no dependence on the diameter, allowing us to perform a multitude of *short walks* much faster.

## 1.3 Historical Overview

The first papers to consider the question of property testing were [6] and [33]. The original motivations for defining property testing were its connection to some Computerized Learning models, and the ability to leverage some properties to construct Probabilistically Checkable Proofs (PCPs – this is related to property testing through the areas of Locally Testable Codes and Locally Decodable Codes, LTCs and LDCs). Other motivations since then have entered the fray, and foremost among them are sublinear-time algorithms, and other big-data considerations. Since virtually no property can be decidable without reading the entire input, property testing introduces a notion of the allowable approximation to the original problem. In general, the algorithm has to distinguish inputs satisfying the property, from inputs that are *$\epsilon$-far* from it. For more information on the general scheme of "classical" property testing, consult the surveys [13,20,31].

The older of the graph testing models discussed here is the dense model, as defined in the seminal work of Goldreich, Goldwasser and Ron [17]. The dense graph model has historically kick-started combinatorial property testing in earnest, but it has some shortcomings. Its main one is the distance function, which makes sense only if we consider graphs having many edges (hence the name "dense model") – any graph with $o(n^2)$ edges is indistinguishable in this model from an empty graph.

The stricter and at times more plausible distance function is one which is relative to the actual number of edges, rather than the maximum $\binom{n}{2}$. The general model was defined in [2], while the sparse model was defined already in [19]. The main difference between the sparse and the general graph models is that in the former there is also a guaranteed upper bound $d$ on the degrees of the vertices, which is given to the algorithm in advance (the query complexity may then depend on $d$, either explicitly, or more commonly implicitly by considering $d$ to be a constant).

## 2 Distributed Emulation of Sequential Tests in the Dense Model

We begin by showing that under a certain assumption of being *non-disjointed*, which we define below, a property $P$ that has a sequential test in the dense model that requires $q$ queries can be tested in the distributed setting within $O(q^2)$ rounds. We prove this by constructing an emulation that translates sequential tests to distributed ones. We first introduce a definition of a *witness* graph and then adapt [21, Theorem 2.2], restricted to 1-sided error tests, to our terminology.

**Definition 1.** *Let $P$ be a property of graphs with $n$ vertices. Let $G'$ be a graph with $k < n$ vertices. We say that $G'$ is a* witness against P, *if it is not an induced subgraph of any graph that satisfies $P$.*

Notice that if $G'$ has an induced subgraph $H$ that is a witness against $P$, then by the above definition $G'$ is also a witness against $P$. The work of [21] transforms tests of graphs in the dense model to a canonical form where the query scheme is based on vertex selection. This is useful in particular for the distributed model, where the computational work is essentially based in the vertices. We require the following special case for 1-sided error tests.

**Lemma 1** ([21, Theorem 2.2]). *Let $P$ be a property of graphs with $n$ vertices. If there exists a 1-sided error $\epsilon$-test for $P$ with query complexity $q(n, \epsilon)$, then there exists a 1-sided error $\epsilon$-test for $P$ that uniformly selects a set of $q' = 2q(n, \epsilon)$ vertices, and accepts iff the induced subgraph is not a witness against $P$.*

Our emulation leverages Lemma 1 under an assumption on the property $P$.

**Definition 2.** *We say that $P$ is a* non-disjointed *property if for every graph $G$ that does not satisfy $P$ and an induced subgraph $G'$ of $G$ such that $G'$ is a witness against $P$, $G'$ has some connected component which is also a witness against $P$. We call such components* witness components.

We are now ready to formally state our main theorem for this section.

**Theorem 1.** *Any $\epsilon$-test in the dense graph model for a non-disjointed property that makes $q$ queries can be converted to an $O(q^2)$-round distributed $\epsilon$-test.*

We claim that not satisfying a non-disjointed property cannot rely on subgraphs that are not connected, which is exactly what we need to forbid in a distributed setting. Formally, the property $P$ is a non-disjointed property if and only if all minimal witnesses that are induced subgraphs of $G$ are connected. Here *minimal* refers to the standard terminology, which means that no proper induced subgraph is a witness against $P$.

Next, we give the distributed test (its pseudo-code form appears in the full version). The test has an outer loop in which each vertex picks itself with probability $5q/n$, collects its neighborhood of a certain size of edges between *picked* vertices in an inner loop, and rejects if it identifies a witness against $P$. The outer loop repeats two times because not only does the sequential test have an error probability, but also with some small probability we may randomly pick too many or not enough vertices in order to emulate it. Repeating the main loop twice reduces the error probability back to below 1/3. In the inner loop, each vertex collects its neighborhood of picked vertices and checks if its connected component is a witness against $P$. To limit communications this is done only for components of picked vertices that are sufficiently small: if a vertex detects that it is part of a component with too many edges then it accepts and does not participate until the next iteration of the outer loop.

To analyze the algorithm, we begin by proving (see full version) that there is a constant probability of at least 2/3 for the number of picked vertices to be sufficient and not too large, namely, between $q$ and $10q$. Now, we can use the guarantees of the sequential test to obtain the guarantees of our algorithm.

**Lemma 2.** *Let $P$ be a non-disjointed graph property. If $G$ satisfies $P$ then all vertices output* `accept` *in the emulation algorithm. If $G$ is $\epsilon$-far from satisfying $P$, then with probability at least 2/3 there exists a vertex that outputs* `reject`.

We now address the round complexity. Each vertex only sends and receives information from its $q$-neighborhood about edges between the chosen vertices. If too many vertices are chosen we detect this and accept. Otherwise we only communicate the chosen vertices and their edges, which requires $O(q^2)$ rounds using standard *pipelining*[2]. Together with Lemma 2, this proves Theorem 1.

**Applications: $k$-colorability and perfect graphs.** We provide some examples of usage of Theorem 1. A result by Alon and Shapira [3] states that all graph properties closed under induced subgraphs are testable in a number of queries that depends only on $\epsilon^{-1}$. We note that, except for certain specific properties for which there are ad-hoc proofs (such as $k$-colorability), the dependence is usually a tower function in $\epsilon^{-1}$ or worse (asymptotically larger).

[2] Pipelining means that each vertex has a buffer for each edge, which holds the information (edges between chosen vertices, in our case) it needs to send over that edge. The vertex sends the pieces of information one after the other.

From this, together with Lemma 1 and Theorem 1, we deduce that if $P$ is a non-disjointed property closed under induced subgraphs, then it is testable, for every fixed $\epsilon$, in a constant number of rounds. Our emulation implies a distributed 1-sided error $\epsilon$-test for *k-colorability* that requires $O(\text{poly}(k\epsilon^{-1}))$ rounds, and a distributed 1-sided error $\epsilon$-test for being a *perfect graph*[3] whose running time depends only on $\epsilon$ (see full version for complete details).

## 3 Distributed Test for Triangle-Freeness

In this section we show a distributed $\epsilon$-test for triangle-freeness. Notice that since triangle-freeness is a non-disjointed property, Theorem 1 gives a distributed $\epsilon$-test for triangle-freeness under the dense model with a number of rounds that is $O(q^2)$, where $q$ is the number of queries required for a sequential $\epsilon$-test for triangle-freeness. However, for triangle-freeness, the known number of queries is a tower function in $\log(1/\epsilon)$ [15].

Here we leverage the inherent parallelism that we can obtain when checking the neighbors of a vertex, and show a test for triangle-freeness that requires only $O(\epsilon^{-2})$ rounds. Importantly, our algorithm works not only for the dense graph model, but for the general graph model (where distances are relative to the actual number of edges), which subsumes it. In the sequential setting, a test for triangle-freeness in the general model requires a number of queries that is some constant power of $n$ by [2]. Our proof, which appears in the full version, actually follows the groundwork laid in [2] for the general graph model – their algorithm picks a vertex and checks two of its neighbors for being connected, while we perform the check for all vertices in parallel.

**Theorem 2.** *There is a distributed $\epsilon$-test in the general graph model for triangle-freeness, that requires $O(\epsilon^{-2})$ rounds.*

## 4 Distributed Bipartiteness Test for Bounded Degree Graphs

In this section we show a distributed $\epsilon$-test for being bipartite for graphs with degrees bounded by $d$. Our test builds upon the sequential test of [18] and, as in the case of triangle freeness, takes advantage of the ability to parallelize queries. While the number of queries of the sequential test is $\Omega(\sqrt{n})$ [19], the number of rounds in the distributed test is only *polylogarithmic* in $n$ and polynomial in $\epsilon^{-1}$. As in [18], we assume that $d$ is a constant, and omit it from our expressions (it is implicit in the $O$ notation for $L$ below).

Let us first outline the algorithm of [18], since our distributed test borrows from its framework and our analysis is in part derived from it. The sequential test basically tries to detect odd cycles. It consists of $T$ iterations, in each of

[3] A graph $G$ is said to be *perfect* if for every induced subgraph $G'$ of $G$, the chromatic number of $G'$ equals the size of the largest clique in $G'$.

which a vertex $s$ is selected uniformly at random and $K$ random walks of length $L$ are performed starting from the source $s$. If, in any iteration with a chosen source $s$, there is a vertex $v$ which is reached by an even prefix of a random walk and an odd prefix of a random walk (possibly the same walk), then the algorithm rejects, as this indicates the existence of an odd cycle. Otherwise, the algorithm accepts. To obtain an $\epsilon$-test the parameters are chosen to be $T = O(\epsilon^{-1})$, $K = O(\epsilon^{-4}\sqrt{n}\log^{1/2}(n\epsilon^{-1}))$, and $L = O(\epsilon^{-8}\log^6 n)$.

The main approach of our distributed test is similar, except that a key ingredient is that we can afford to perform much fewer random walks from every vertex, namely $O(\text{poly}(\epsilon^{-1}\log n\epsilon^{-1}))$. This is because we can run random walks in parallel originating from all vertices at once. However, a crucial challenge that we need to address is that several random walks may collide on an edge, violating its congestion bound. To address this issue, our central observation is that *lazy* random walks (chosen to have a uniform stationary distribution) provide for a very low probability of having too many of these collisions at once. The main part of the analysis is in showing that with high probability there will never be too many walks concurrently in the same vertex, so we can comply with the congestion bound. We begin by formally defining the lazy random walks we use.

**Definition 3.** *A* lazy *random walk over a graph $G$ with degree bound $d$ is a random walk, that is, a (memory-less) sequence of random variables $Y_1, Y_2, \ldots$ taking values from the vertex set $V$, where the transition probability $Pr[Y_k = v | Y_{k-1} = u]$ is $\frac{1}{2d}$ if $uv$ is an edge of $G$, $\frac{1-deg(u)}{2d}$ if $u = v$, and $0$ otherwise.*

The stationary distribution for the lazy random walk of Definition 3 is uniform [32, Section 8]. Next, we describe a procedure to handle one iteration of moving the random walks (Algorithm 1). Our distributed test for bipartiteness (its pseudo-code form is in the full version) initiates only 2 lazy random walks from every vertex concurrently, and searches for odd cycles that can be detected if an even prefix and an odd prefix of 2 such random walks collide at some vertex.

It is quite immediate that Algorithm 1 takes $O(\xi)$ rounds (the value of $\xi$ is given below). Our main result here is that using $L$ iterations of Algorithm 1 indeed provides a distributed $\epsilon$-test for bipartiteness.

**Theorem 3.** *There is a distributed $\epsilon$-test in the bounded degree graph model for the property of being bipartite, that requires $O(poly(\epsilon^{-1}\log(n\epsilon^{-1})))$ rounds.*

The number of rounds is immediate from the algorithm – it is dominated by the $L$ calls to Algorithm 1, making a total of $O(\xi L)$ rounds, which is indeed $O(\text{poly}(\epsilon^{-1}\log(n\epsilon^{-1})))$. To prove the rest of Theorem 3 we need some notation, and a lemma from [18] that bounds from below the probabilities for detecting odd cycles if $G$ is $\epsilon$-far from being bipartite.

Given a source $s$, if there is a vertex $v$ which is reached by an even prefix of a random walk $w_i$ from $s$ and an odd prefix of a random walk $w_j$ from $s$, we say that walks $w_i$ and $w_j$ *detect a violation*. Let $p_s(k, \ell)$ be the probability that, out of $k$ random walks of length $\ell$ starting from $s$, there are two that detect a violation. Using this notation, $p_s(K, L)$ is the probability that the sequential

**Algorithm 1.** Move random walks once with input $\xi$

```
Variables: $W_v$ walks residing in $v$ (multiset), $H_v$ history of walks through $v$
Input: $\xi$, the maximum congestion per vertex allowed
# each walk is characterized by $(i, u)$ where $i$ is the number of actual
  moves and $u$ is the origin vertex
1 for each vertex $v$ simultaneously
2     if $|W_v| \leq \xi$ then # give up if exceeded the maximum allowed
3         for every $(i, u)$ in $W_v$ do
4             draw next destination $w$ (according to the lazy walk scheme)
5             if $w \neq v$ then # walk exits $v$
6                 send $(i + 1, u)$ to $w$
7                 remove $(i, u)$ from $W_v$
8     wait until the maximum time for all other vertices to process up to $\xi$ walks
9     add the walks received by $v$ to $W_v$ and $H_v$ # walks entering $v$
```

algorithm outlined in the beginning rejects in an iteration in which $s$ is chosen. Since we are only interested in walks of length $L$, we denote $p_s(k) = p_s(k, L)$. A good vertex is a vertex for which this probability is bounded as follows.

We say a vertex $s$ is called *good* if $p_s(K) \geq 1/10$. In [18] it was proved that if $G$ is $\epsilon$-far from being bipartite then at least an $\epsilon/16$-fraction of the vertices are good. In contrast to [18], we do not perform $K$ random walks from every vertex in each iteration, but rather only 2. Hence, what we need for our analysis is a bound on $p_s(2)$. To this end, we use $K$ as a parameter, and express $p_s(2)$ in terms of $K$ and $p_s(K)$, by showing that for every vertex $s$, $p_s(2) \geq 2p_s(K)/K(K-1)$.

Using this relationship between $p_s(2)$, $K$ and $p_s(K)$, we prove that our algorithm is an $\epsilon$-test. First we prove this for the random walks themselves, ignoring the possibility that Algorithm 1 will skip moving random walks due to its condition in Line 2.

**Lemma 3.** *If $G$ is $\epsilon$-far from being bipartite, and we perform $\eta$ iterations of starting 2 random walks of length $L$ from every vertex, the probability that no violation is detected is bounded by 1/4.*

As explained earlier, the main hurdle on the road to prove Theorem 3 is in proving that the allowed congestion will not be exceeded. We prove the following general claim about the probability for $k$ lazy random walks of length $\ell$ from each vertex to exceed a maximum *congestion factor* of $\xi$ walks allowed in each vertex at the beginning of each iteration. Here, an iteration is a sequence of rounds in which all walks are advanced by one step (whether or not they actually switch vertices).

**Lemma 4.** *With probability at least $1 - 1/n$, running $k$ lazy random walks of length $\ell$ originating from every vertex will not exceed the maximum* congestion factor *of $\xi = \gamma + k = 3(2\ln n + \ln \ell) + k$ walks allowed in each vertex at the beginning of each iteration, if $\gamma > k$.*

If $G$ is bipartite then all vertices output `accept` in our bipartiteness test, because there are no odd cycles and thus no violation detecting walks. If $G$ is $\epsilon$-far from bipartite, we use Lemma 3, in conjunction with Lemma 4 with parameters $k = 2$, $\ell = L$ and $\gamma = 3(2 \ln n + \ln L)$ as used by our bipartiteness test. By a union bound the probability to accept $G$ will be bounded by $1/4+1/n < 1/3$ (assuming $n > 12$), providing for the required bound on the rejection probability. This, with the communication complexity analysis of our distributed bipartiteness test, gives Theorem 3.

## 5 Distributed Test for Cycle-Freeness

In this section, we give a distributed algorithm to test if a graph $G$ with $m$ edges is cycle-free or if at least $\epsilon m$ edges have to be removed to make it so. Intuitively, in order to search for cycles, one can run a breadth-first search (BFS) and have a vertex output `reject` if two different paths reach it. The downside of this exact solution is that its running time depends on the diameter of the graph. To overcome this, a basic approach would be to run a BFS from each vertex of the graph, but for shorter distances. However, running multiple BFSs simultaneously is expensive, due to the congestion on the edges. Instead, we use a simple prioritization rule that drops BFS constructions with lower priority, which makes sure that one BFS remains alive.[4]

Our technique consists of three parts. First, we make the graph $G$ sparser, by removing each of its edges independently with probability $\epsilon/2$. We denote the sampled graph by $G'$ and prove that if $G$ is far from being cycle-free then so is $G'$, and in particular, $G'$ contains a cycle.

Then, we run a partial BFS over $G'$ from each vertex, while prioritizing by ids: each vertex keeps only the BFS that originates in the vertex with the largest id and drops the rest of the BFSs. The length of this procedure is according to a threshold $T = 20 \log n/\epsilon$. This gives detection of a cycle that is contained in a component of $G'$ with a low diameter of up to $T$, if such a cycle exists, since a surviving BFS covers the component. Such a cycle is also a cycle in $G$. If no such cycle exists in $G'$, then $G'$ has some component with diameter larger than $T$. For large components, we take each surviving BFS that reached some vertex $v$ at a certain distance $\ell$, and from $v$ we run a new partial BFS in the *original* graph $G$. These BFSs are again prioritized, this time according to the distance $\ell$. Our main tool is proving that with high probability, if there is a shortest path in $G'$ of length $T/2$ between two vertices, then there is a cycle in $G$ between them of length at most $T$. This allows our BFSs on $G$ to find such a cycle. We start with the following combinatorial lemma that shows the above claim.

[4] A more involved analysis of multiple prioritized BFS executions was used in [24], allowing all BFS executions to fully finish in a short time without too much delay due to congestion. Since we require a much weaker guarantee, we can avoid the strong full-fledged prioritization algorithm of [24] and settle for a simple rule that keeps one BFS tree alive. Also, the multiple BFS construction of [27] does not fit our demands as it may not reach all desired vertices within the required distance, in case there are many vertices that are closer.

**Lemma 5.** *Given a graph $G$, let $G'$ be obtained by independently deleting each edge in $G$ with probability $\frac{\epsilon}{2}$. Then, with probability at least $1 - \frac{1}{n^3}$, every vertex $v \in G'$ that has a vertex $w \in G'$ at a distance $\frac{10 \log n}{\epsilon}$, has a closed path passing through it in $G$, that contains a simple cycle, of length at most $\frac{20 \log n}{\epsilon}$.*

Next, we prove that indeed there is a high probability, of at least $1-e^{-\epsilon^2 m/32}$, that $G'$ contains a cycle if $G$ is $\epsilon$-far from being cycle-free.

In the full version we provide pseudocode for both our prioritized multiple BFS and an $\epsilon$-test for cycle freeness.

**Theorem 4.** *Our algorithm is a distributed $\epsilon$-test in the general graph model for the property of being cycle-free, that requires $O(\log n/\epsilon)$ rounds.*

## 6 Lower Bounds

In this section, we prove that any distributed algorithm for $\epsilon$-testing bipartiteness or cycle-freeness in bounded-degree graphs requires $\Omega(\log n)$ rounds of communication. This applies even to the less restricted LOCAL model, which does not limit the size of the messages. We construct bounded-degree graphs that are $\epsilon$-far from being bipartite, such that all cycles are of length $\Omega(\log n)$. We argue that any distributed algorithm that runs in $O(\log n)$ rounds does not detect a witness for non-bipartiteness. We also show that the same construction proves that every distributed algorithm for $\epsilon$-testing cycle-freeness requires $\Omega(\log n)$ rounds of communication. Formally, we prove the following theorem.

**Theorem 5.** *Any distributed $1/100$-test in the bounded degree or general graph model for the property of being bipartite requires $\Omega(\log n)$ rounds of communication.*

To prove Theorem 5, we show the existence of a graph $G'$ that is far from being bipartite, but all of its cycles are at least of logarithmic length. Since in $T$ rounds of a distributed algorithm, the output of every vertex cannot depend on vertices that are at distance greater than $T$ from it, no vertex can detect a cycle in $G'$ in less than $O(\log n)$ rounds, which proves Theorem 5. To prove the existence of $G'$ we use the probabilistic method with alterations, and prove the following.

**Lemma 6.** *Let $G$ be a random graph on $n$ vertices where each edge is present with probability $1000/n$. Let $G'$ be obtained by removing all edges incident with vertices of degree greater than $2000$, and one edge from each cycle of length at most $\log n/\log 1000$. Then with probability at least $1/2 - e^{-100} - e^{-n}$, $G'$ is $1/100$-far from being bipartite.*

Since a graph that is $\epsilon$-far from being bipartite is also $\epsilon$-far from being cycle-free, we immediately obtain the same lower bound for testing cycle-freeness:

**Theorem 6.** *Any distributed $1/100$-test in the bounded degree graph or general model for cycle-freeness requires $\Omega(\log n)$ rounds of communication.*

## 7 Discussion

This paper provides a thorough study of distributed property testing. It provides an emulation technique for the dense graph model and constructs fast distributed algorithms for testing triangle-freeness, cycle-freeness and bipartiteness. We also present lower bounds for both bipartiteness and cycle-freeness.

This work raises many important open questions, the immediate of which is to devise fast distributed testing algorithms for additional problems. One example is testing freeness of other small subgraphs. More ambitious goals are to handle dynamic graphs, and to find more general connections between testability in the sequential model and the distributed model. Finally, there is fertile ground for obtaining additional lower bounds in this setting, in order to fully understand the complexity of distributed property testing.

## References

1. Alon, N., Avin, C., Koucký, M., Kozma, G., Lotker, Z., Tuttle, M.R.: Many random walks are faster than one. Comb. Probab. Comput. **20**(4), 481–502 (2011)
2. Alon, N., Kaufman, T., Krivelevich, M., Ron, D.: Testing triangle-freeness in general graphs. SIAM J. Discrete Math. **22**(2), 786–819 (2008)
3. Alon, N., Shapira, A.: A characterization of the (natural) graph properties testable with one-sided error. SIAM J. Comput. **37**(6), 1703–1727 (2008)
4. Arfaoui, H., Fraigniaud, P., Ilcinkas, D., Mathieu, F.: Distributedly testing cycle-freeness. In: Kratsch, D., Todinca, I. (eds.) WG 2014. LNCS, vol. 8747, pp. 15–28. Springer, Heidelberg (2014)
5. Baruch, M., Fraigniaud, P., Patt-Shamir, B.: Randomized proof-labeling schemes. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, (PODC), pp. 315–324 (2015)
6. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. J. Comput. Syst. Sci. **47**(3), 549–595 (1993)
7. Brakerski, Z., Patt-Shamir, B.: Distributed discovery of large near-cliques. Distrib. Comput. **24**(2), 79–89 (2011)
8. Censor-Hillel, K., Fischer, E., Schwartzman, G., Vasudev, Y.: Fast distributed algorithms for testing graph properties. CoRR abs/1602.03718 (2016)
9. Censor-Hillel, K., Kaski, P., Korhonen, J.H., Lenzen, C., Paz, A., Suomela, J.: Algebraic methods in the congested clique. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, (PODC), pp. 143–152 (2015)
10. Dolev, D., Lenzen, C., Peled, S.: "Tri, tri again": finding triangles and small subgraphs in a distributed setting. In: Aguilera, M.K. (ed.) DISC 2012. LNCS, vol. 7611, pp. 195–209. Springer, Heidelberg (2012)
11. Drucker, A., Kuhn, F., Oshman, R.: The communication complexity of distributed task allocation. In: Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pp. 67–76 (2012)
12. Erdös, P.: Graph theory and probability. J. Math. **11**, 34G38 (1959)
13. Fischer, E.: The art of uninformed decisions: a primer to property testing. Current Trends Theor. Comput. Sci. Challenge New Century I **2**, 229–264 (2004)
14. Foerster, K.T., Luedi, T., Seidel, J., Wattenhofer, R.: Local checkability, no strings attached. In: Proceedings of the 17th International Conference on Distributed Computing and Networking (ICDCN), pp. 21: 1–21: 10 (2016)

15. Fox, J.: A new proof of the graph removal lemma. CoRR abs/1006.1300 (2010)
16. Ghaffari, M., Kuhn, F., Su, H.H.: Manuscript (2016)
17. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. J. ACM **45**(4), 653–750 (1998)
18. Goldreich, O., Ron, D.: A sublinear bipartiteness tester for bounded degree graphs. Combinatorica **19**(3), 335–373 (1999)
19. Goldreich, O., Ron, D.: Property testing in bounded degree graphs. Algorithmica **32**(2), 302–343 (2002)
20. Goldreich, O., Ron, D.: Algorithmic aspects of property testing in the dense graphs model. In: Goldreich, O. (ed.) Property Testing. LNCS, vol. 6390, pp. 295–305. Springer, Heidelberg (2010)
21. Goldreich, O., Trevisan, L.: Three theorems regarding testing graph properties. Random Struct. Algorithms **23**(1), 23–57 (2003)
22. Göös, M., Hirvonen, J., Levi, R., Medina, M., Suomela, J.: Non-local probes do not help with graph problems. CoRR abs/1512.05411 (2015)
23. Hirvonen, J., Rybicki, J., Schmid, S., Suomela, J.: Large cuts with local algorithms on triangle-free graphs. CoRR abs/1402.2543 (2014)
24. Holzer, S., Wattenhofer, R.: Optimal distributed all pairs shortest paths and applications. In: Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing, pp. 355–364. ACM (2012)
25. Kari, J., Matamala, M., Rapaport, I., Salo, V.: Solving the induced subgraphproblem in the randomized multiparty simultaneous messages model. In: Proceedings of the 22nd International Colloquium on Structural Informationand Communication Complexity (SIROCCO), pp. 370–384 (2015)
26. Korman, A., Kutten, S., Peleg, D.: Proof labeling schemes. Distrib. Comput. **22**(4), 215–233 (2010)
27. Lenzen, C., Peleg, D.: Efficient distributed source detection with limited bandwidth. In: Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pp. 375–382 (2013)
28. Parnas, M., Ron, D.: Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. Theor. Comput. Sci. **381**(1–3), 183–196 (2007)
29. Peleg, D.: Distributed computing: a locality-sensitive approach. Soc. Ind. Appl. Math. **157**, 2153–2169 (2000)
30. Pettie, S., Su, H.: Distributed coloring algorithms for triangle-free graphs. Inf. Comput. **243**, 263–280 (2015)
31. Ron, D.: Property testing: a learning theory perspective. Found. Trends Mach. Learn. **1**(3), 307–402 (2008)
32. Ron, D.: Algorithmic and analysis techniques in property testing. Found. Trends Theor. Comput. Sci. **5**(2), 73–205 (2009)
33. Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. SIAM J. Comput. **25**(2), 252–271 (1996)
34. Sarma, A.D., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. SIAM J. Comput. **41**(5), 1235–1265 (2012)
35. Sarma, A.D., Nanongkai, D., Pandurangan, G., Tetali, P.: Distributed random walks. J. ACM **60**(1), 2 (2013)