Testing C_k-Freeness in Bounded-Arboricity Graphs

Talya Eden
Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

Reut Levi
Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

Dana Ron
School of Electrical Engineering, Tel Aviv University, Israel

Abstract

We study the problem of testing C_k-freeness (k-cycle-freeness) for fixed constant $k > 3$ in graphs with bounded arboricity (but unbounded degrees). In particular, we are interested in one-sided error algorithms, so that they must detect a copy of C_k with high constant probability when the graph is ϵ-far from C_k-free.

We next state our results for constant arboricity and constant ϵ with a focus on the dependence on the number of graph vertices, n. The query complexity of all our algorithms grows polynomially with $1/\epsilon$.

1. As opposed to the case of $k = 3$, where the complexity of testing C_3-freeness grows with the arboricity of the graph but not with the size of the graph (*Levi, ICALP 2021*)[1] this is no longer the case already for $k = 4$. We show that $\Omega(n^{1/4})$ queries are necessary for testing C_4-freeness, and that $\widetilde{O}(n^{1/4})$ are sufficient. The same bounds hold for C_5.
2. For every fixed $k \geq 6$, any one-sided error algorithm for testing C_k-freeness must perform $\Omega(n^{1/3})$ queries.
3. For $k = 6$ we give a testing algorithm whose query complexity is $\widetilde{O}(n^{1/2})$.
4. For any fixed k, the query complexity of testing C_k-freeness is upper bounded by $O(n^{1-1/\lfloor k/2 \rfloor})$.

The last upper bound builds on another result in which we show that for any fixed subgraph F, the query complexity of testing F-freeness is upper bounded by $O(n^{1-1/\ell(F)})$, where $\ell(F)$ is a parameter of F that is always upper bounded by the number of vertices in F (and in particular is $k/2$ in C_k for even k).

We extend some of our results to bounded (non-constant) arboricity, where in particular, we obtain sublinear upper bounds for all k.

Our $\Omega(n^{1/4})$ lower bound for testing C_4-freeness in constant arboricity graphs provides a negative answer to an open problem posed by (Goldreich, 2021).

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Property Testing, Cycle-Freeness, Bounded Arboricity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.60

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2404.18126`

Funding *Reut Levi*: ISF grant number 1867/20
Dana Ron: ISF grant number 1146/18

[1] As presented in (*Levi, ICALP 2021*), the complexity of the algorithm depends on $\log \log n$, but this dependence can be replaced with at most a polylogarithmic dependence on the arboricity.

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 60; pp. 60:1–60:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Detecting small subgraphs with specific structures (referred to as *finding network motifs*) is a basic algorithmic task, with a variety of applications in biology, sociology and network science (see e.g. [21, 8, 31, 11, 7, 28, 5, 18, 6, 32, 23]). Of special interest is the natural case of subgraphs that are cycles of a fixed size k, which we denote by C_k. When the algorithm receives the entire graph as input, then by the well known result of Alon, Yuster and Zwick [4], this task can be solved in time $\widetilde{O}(n^\omega)$ where n is the number of graph vertices and ω is the exponent of matrix multiplication.[2] But what if we seek a sublinear-time (randomized) algorithm that does not read the entire graph? Namely, the algorithm is given query access to the graph[3] and should find a C_k when the graph is not C_k-free. This is clearly not possible if the graph contains only a single copy of C_k. However, is it possible to detect such a copy in sublinear-time when the graph is relatively far from being C_k-free? By "relatively far" we mean that it is necessary to remove a non-negligible fraction, denoted ϵ, of its edges in order to obtain an C_k-free graph. A closely related formulation of the question is whether we can design a one-sided error algorithm for testing C_k-freeness.[4]

If the maximum degree in the graph is upper bounded by a parameter $d_{\max}$, then the C_k-freeness testing problem can easily be solved by performing a number of queries that grows polynomially with $d_{\max}$ and exponentially with $\Theta(k)$ [20]. In particular, when $d_{\max} = O(1)$, then there is no dependence on the size of the graph G. We are however interested in considering graphs with varying degrees, so that, in particular, the maximum degree may be much larger than the average degree, and possibly as large as $\Theta(n)$.

For the special and interesting case where $k = 3$, i.e., the cycle is a triangle, Alon, Kaufman, Krivelevich and Ron [3] gave several upper and lower bounds on the query complexity of testing triangle-freeness as a function of the average degree d of the graph (in addition to the dependence on n and ϵ). While the upper and lower bounds are not tight in general, they are tight for $d = O(1)$, where the complexity is $\Theta(\sqrt{n})$ (for constant ϵ). The lower bound in this case is essentially based on "hiding" a small clique.

Since the aforementioned lower bound relies on the existence of a small dense subgraph, a natural question, studied by Levi [26], is whether it is possible to obtain improved (and possibly tight) results when the arboricity of the graph, denoted $arb(G)$, is bounded.[5] Focusing on the result under the assumption that $m \geq n$ (i.e., $d = \Omega(1)$) Levi showed that $\widetilde{O}(arb(G))$ queries are sufficient for testing triangle-freeness (the dependence on $1/\epsilon$ is polynomial), and that $\Omega(arb(G))$ queries are necessary.[6] In particular, when $arb(G)$ is a constant, the complexity is polynomial in $1/\epsilon$ and does not depend on the size of the graph.

In this work we seek to understand the complexity of testing C_k-freeness, in particular with one-sided error, for fixed $k > 3$. Our main focus is on constant arboricity graphs and some of our results extend to bounded arboricity graphs, as well as to F-freeness for any

[2] The dependence on k is exponential, but k is considered a constant.

[3] The types of queries typically considered are neighbor queries ("what is the ith neighbor of a vertex v?"), degree queries ("what is the degree of a vertex v?"), and pair queries ("is there an edge between a pair of vertices v and u?").

[4] The problems are equivalent if the algorithm is not given access to degree queries, otherwise the algorithm might find evidence to the existence of a C_k without actually detecting one. We note that all our algorithms do detect copies of C_k when they reject.

[5] The arboricity of a graph G is the minimum number of forests required to cover its edges, and is equal (up to a factor of 2) to the maximum average degree of any subgraph of G.

[6] To be precise, this lower bound holds for $m \geq (arb(G))^3$ – if $m < (\arg(G))^3$ then the lower bound is $\Omega(m^{1/3})$. See also Footnote 1 regarding the upper bound.

general subgraph F (of constant size). We note that the problem of testing cycle-freeness without requiring the cycle to be of specific length, is different from our problem. We further discuss this in Section 1.3. In the next subsection we state our findings.

1.1 Our results

Since our main focus is on graphs with constant arboricity, we first state our results in this setting, and later discuss our extensions to graphs with non-constant arboricity. Throughout this paper we assume that $m = \Omega(n)$ since even obtaining a single edge in the graph requires $\Omega(n/m)$ queries. Our algorithms use degree and neighbor queries and our lower bounds also allow pair queries (see Footnote 3). For simplicity, we state our results for constant ϵ. All our algorithms have a polynomial dependence on $1/\epsilon$.

Our first finding is that, as opposed to the case of $k = 3$, where the complexity of testing C_3-freeness grows with the arboricity of the graph but not with the size of the graph,[7] this is no longer the case for $k = 4$ (and larger k). In particular:

▶ **Theorem 1.** *The query complexity of one-sided error testing of C_4-freeness in constant-arboricity graphs over n vertices is $\widetilde{\Theta}(n^{1/4})$. The same bound holds for testing C_5-freeness.*

Theorem 1 (together with the upper bound in [20]) answers negatively the following open problem raised by Goldreich.

Open problem (number 3.2 in [19]): From bounded degree to bounded arboricity.
Suppose that property Π is testable within complexity $Q(n, \epsilon)$ in the bounded-degree graph model. Provide an upper bound on the complexity of testing Π in the general graph model under the promise that the tested graph has constant arboricity. For example, can the latter complexity be linear in $Q(n, \epsilon)$ while permitting extra poly$(logn)$ or $1/\epsilon$ factors?

The $\Omega(n^{1/4})$ lower bound for testing C_4-freeness, answers this question negatively. Indeed, testing C_4-freeness in d-bounded degree graphs can be done with $poly(d, \epsilon)$ queries [20], while our lower bound suggest that even in constant arboricity graphs, a polynomial dependence on n is necessary.

When $k \geq 6$, we show that it is no longer possible to obtain a complexity of $\widetilde{O}(n^{1/4})$ as is the case for $k = 4, 5$.

▶ **Theorem 2.** *Let $k \geq 6$. Any one-sided error tester for the property of C_k-freeness in graphs of constant arboricity over n vertices must perform $\Omega(n^{1/3})$ queries.*

While for C_6 we were not able to match the lower bound of $\Omega(n^{1/3})$, we were able to obtain a sublinear-time algorithm, as stated next.

▶ **Theorem 3.** *There exists a one-sided error algorithm for testing C_6-freeness in graphs of constant arboricity over n vertices whose query complexity is $\widetilde{O}(n^{1/2})$.*

For general (fixed) k we prove the following upper bound.

▶ **Theorem 4.** *There exists a one-sided error algorithm for testing C_k-freeness in graphs of constant arboricity over n vertices whose query complexity is $O(n^{1-1/\lfloor k/2 \rfloor})$.*

We also prove a more general result for testing F-freeness for any constant size subgraph F. Below, $\ell(F)$ is as defined in Definition 10, and is always upper bounded by the number of vertices in F.

[7] We note that this is true also for other k-cliques for $k > 3$.

▶ **Theorem 5.** *There exists a one-sided error algorithm for testing F-freeness in graphs of constant arboricity over n vertices whose query complexity is $O(n^{1-1/\ell(F)})$.*

1.1.1 Extensions for general arboricity

We state our results for general arboricity graphs assuming that the algorithm is given an upper bound α on the arboricity of the graph (in the lower bounds the algorithm may be assumed to know the arboricity). Alternatively, if the algorithm receives as an input the number of edges, m, (as in previous results for C_k-freeness [3, 26]) instead of an upper bound on the arboricity, then we can estimate a notion known [26] as the "effective arboricity" of the graph, and depend on it instead of α. This is potentially beneficial since the effective arboricity can be much smaller than the actual arboricity of the graph, and it does not affect the asymptotic running times of our algorithms in terms of the dependence on the size of the graph and α. For further details see Section 2.

For C_4-freeness we give both an upper bound and a lower bound for general arboricity graphs. In particular, we show that a linear dependence on α is sufficient and a $\sqrt{\alpha}$-dependence is necessary (both for one-sided error algorithms) as stated next.

▶ **Theorem 6.** *There exists a one-sided error algorithm for testing C_4-freeness in graphs of arboricity at most α over n vertices whose query complexity is $\tilde{O}\left(\min\{n^{1/4}\alpha, \alpha + n^{3/4}\}\right)$.*[8]

▶ **Theorem 7.** *Testing C_4-freeness with one-sided error in graphs over n vertices with arboricity $c_1 \log n < \alpha < n^{1/2}/c_1'$ for sufficiently large constants c_1 and c_1' requires $\Omega(n^{1/4}\alpha^{1/2})$ queries.*[9]

For general constant size subgraphs F (and in particular C_k) our upper bound also has at most a linear dependence on α (recall that $\ell(F)$ is defined in Definition 10).

▶ **Theorem 8.** *There exists a one-sided error tester for F-freeness whose query complexity is $O\left(k^{2+1/\ell(F)} \cdot m^{1-1/\ell(F)} \cdot \alpha^{1/\ell(F)}\right)$.*

▶ **Corollary 9.** *There exists a one-sided error tester for C_k-freeness whose query complexity for even k is $O\left(k^{2+(2/k))} \cdot m^{1-2/k} \cdot \alpha^{2/k}\right)$, and for odd k is $O\left(k^{2+2/(k+1)} \cdot m^{1-2/(k+1)} \cdot \alpha^{2/(k+1)}\right)$.*

We comment that our lower bound of $\Omega(n^{1/3})$ for one-sided error algorithms, $k \geq 6$ and constant arboricity (stated in Theorem 2) also applies to graphs with non-constant arboricity (by adding a C_k-free subgraph with higher arboricity).[10]

We also note that it is possible to extend our algorithms for C_5 and C_6 freeness so as to get a polynomial (but not linear) dependence on α. However, these extensions do not introduce new techniques (and are most probably not optimal), so we do not present them here.

---

[8] More precisely, for values $\alpha < \log n$, the complexity is $O(n^{1/4}\alpha^{1/2}\log^{1/2} n/\epsilon^3)$, for values $\log n < \alpha < \sqrt{n}$, the complexity is $O(n^{1/4}\alpha/\epsilon^3)$, and for values $\alpha > n^{1/2}$, it is $O((\alpha + n^{3/4})/\epsilon^3)$.

[9] Note that the two-sided error lower bound of $\Omega(n^{1/4})$ for constant arboricity graphs (as stated in Theorem 1) also holds for graphs with higher arboricity α, and in particular, $\alpha = O(\log n)$. This is the case since we can simply add a small subgraph with arboricity α and no C_4s to the lower bound construction.

[10] For an odd k, it suffices to add a dense bipartite graph, and for even k, by the Erdős girth conjecture [16], one can add a subgraph with arboricity $n^{2/k}$.

1.2 A high-level discussion of our algorithms and lower bounds

Before discussing each of our results in more detail, we highlight some common themes. The starting point of all our algorithms is that if a graph is ϵ-far from being C_k-free (for a constant k), then it contains $\Omega(\epsilon m)$ edge-disjoint cycles.[11] We next use the bounded arboricity of the graph. Specifically, if a graph has arboricity at most α, then the number of edges between pairs of vertices that both have degree greater than $\theta_0 = \Theta(\alpha/\epsilon)$, is at most $O(\epsilon m)$.

Hence, there is a set of edge-disjoint C_ks, which we denote by $\mathcal{C}$, such that $|\mathcal{C}| = \Omega(\epsilon m)$, and no C_k in $\mathcal{C}$ contains any edge between two vertices with degree greater than θ_0. In other words, for every k-cycle ρ in $\mathcal{C}$, and for every vertex v with degree greater than θ_0 in ρ, the two neighbors of v in ρ have degree at most θ_0. In particular, when α is a constant, the two neighbors have degree $O(1/\epsilon)$.

At this point our algorithms diverge, but there are two common aspects when $k = 4, 5, 6$, which we would like to highlight. The first is that for the sake of "catching" one of the C_ks in $\mathcal{C}$, it will be useful to consider a subset, $\mathcal{C}'$, in which every vertex v that participates in one of the edge-disjoint C_ks in $\mathcal{C}'$ actually participate in $\Omega(\epsilon \cdot d(v))$ C_ks in $\mathcal{C}'$. The existence of such a subset follows by applying (as a mental experiment) a simple iterative process that removes C_ks with vertices that do not obey this constraint.

To illustrate why it is useful to have such a set $\mathcal{C}'$, consider the case of $k = 4$, and assume that a relatively large fraction of the C_4s in $\mathcal{C}'$ contain, in addition to the two vertices of degree at most $\theta_0 = O(\alpha/\epsilon)$, at least one other vertex that has degree at most $\theta_1 = O(n^{1/2}/\epsilon)$. In this case we can obtain such a vertex v with high probability (as discussed below), and then sample roughly $\sqrt{d(v)/\epsilon} = O(\sqrt{\theta_1/\epsilon}) = O(n^{1/4}/\epsilon)$ of its neighbors, so that the following holds. By (a slight variant of) the birthday paradox, with high constant probability we hit two of its neighbors, u and u', that reside on the same C_4 in $\mathcal{C}'$ (and hence have degree at most θ_0). By querying all the neighbors of u and u', we obtain this C_4.

However, what if for most of the C_4's in $\mathcal{C}'$ there are two vertices with degree significantly larger than $\sqrt{n}$ (that are "one opposite the other" on the C_4s)? Roughly speaking, in this case we exploit the fact that the number of such high-degree vertices is bounded, and we show how to detect a C_4 by performing random walks of length 2. A related issue arises in the case of $k = 6$, when there are three very high degree vertices on most C_6s in $\mathcal{C}'$. In this case we show how to essentially reduce the problem to testing triangle-freeness in a certain auxiliary graph. More precisely, the auxiliary graph is a multi-graph to which we have access only to certain types of queries, so that we cannot apply the algorithm of [3]. However, we can still show how to obtain a triangle in this graph, and hence a C_6 in the original graph. Interestingly, our general lower bound of $\Omega(n^{1/3})$ for C_k-freeness, $k \geq 6$ builds on the lower bound for testing triangle-freeness of [3].

In the following subsections we assume for the sake of the exposition that ϵ is a constant.

1.2.1 The results for C_4-freeness (and C_5-freeness)

We discuss our results for C_4-freeness in graphs with general arboricity. The results for C_5-freeness in constant arboricity graphs are obtained using very similar techniques.

[11] To verify this, let G be a graph that is ϵ-far from being C_k-free for a fixed constant k. Consider any maximal set S of edge-disjoint k-cycles. Since by removing all $k \cdot |S|$ edges on these cycles, the graph can be made cycle-free, $|S| \geq \epsilon m/k = \Omega(\epsilon m)$.

The algorithm

Our algorithm for testing C_4-freeness, Test-C_4-freeness, which has query complexity $\widetilde{O}(n^{1/4}\alpha)$, is governed by two thresholds: $\theta_0 = \Theta(\alpha)$, and $\theta_1 = \Theta(n^{1/2})$. For the sake of the current high-level presentation, we assume that[12] $\alpha \le n^{1/2}$, so that $\theta_0 \le \theta_1$.

The algorithm first samples $O(1)$ edges approximately uniformly by invoking a procedure Select-an-Edge,[13] and then randomly selects one of their endpoints. For each vertex v selected, it queries its degree, $d(v)$. If $d(v) \le \theta_1$, then the algorithm selects $O(\sqrt{d(v)})$ random neighbors of v, and for each selected neighbor u such that $d(u) \le \theta_0$, it queries all the neighbors of u. If $d(v) > \theta_1$, then the algorithm performs $\widetilde{O}(n^{1/4}\alpha^{1/2})$ random walks of length two from v. If a C_4 is observed in any one of these steps, then the algorithm rejects, otherwise it accepts.

The analysis of the algorithm

By the above description, the algorithm will only reject a graph if it detects a C_4, implying that it never errs on C_4-free graphs. Hence, consider a graph G that is far from being C_4-free. As discussed at the start of Section 1.2, the setting of $\theta_0 = \Theta(\alpha)$ (together with the fact that G is $\Omega(1)$-far from being C_4-free) implies the following. There exists a set, denoted $\mathcal{C}$, of $\Omega(m)$ *edge-disjoint* C_4s in G, such that no C_4 in $\mathcal{C}$ contains an edge between two vertices that both have degree greater than θ_0. Thus, for each C_4 in $\mathcal{C}$, there are at most two vertices with degree greater than θ_0, and they do not neighbor each other.

Considering the second aforementioned degree threshold θ_1 (and recalling that $\theta_1 \ge \theta_0$), we partition $\mathcal{C}$ into two subsets. The first, $\mathcal{C}_1$, consists of those C_4s in $\mathcal{C}$ that contain at most one vertex with degree greater than θ_1, and the second, $\mathcal{C}_2$, of the remaining C_4s in $\mathcal{C}$, which contain exactly two vertices with degree greater than θ_1. Since $\mathcal{C} = \mathcal{C}_1 \uplus \mathcal{C}_2$, at least one of these subsets is of size $\Omega(m)$.

C_4s with at most one high-degree vertex. Suppose first that $|\mathcal{C}_1| = \Omega(m)$. Observe that since each 4-cycle $\rho \in \mathcal{C}_1$ contains at least two vertices with degree at most θ_0 and at most one vertex with degree greater than θ_1, it must contain at least one vertex, with degree at most θ_1 whose neighbors on the C_4 both have degree at most θ_0. For an illustration, see the LHS of Figure 1. Furthermore, we show that there exists a subset of $\mathcal{C}_1$, which we denote by $\mathcal{C}'_1$, such that $|\mathcal{C}'_1| = \Omega(m)$, and every vertex v that participates in one of the C_4s in $\mathcal{C}'_1$, actually participates in $\Omega(d(v))$ edges-disjoint C_4s in $\mathcal{C}_1$. It follows that in this case, when the algorithm selects a random edge (almost uniformly), with high constant probability it will obtain an edge with (at least) one endpoint v having the above properties. Conditioned on the selection of such a vertex v, the algorithm selects $\Theta(\sqrt{d(v)})$ random neighbors of v. By the birthday paradox, with high constant probability, among these neighbors there will be a pair of vertices that reside, together with v, on a common C_4 in $\mathcal{C}'_1$. Once their (at most θ_0) neighbors are queried, this C_4 is revealed.

C_4s with two high-degree vertex. We now turn to the case in which $|\mathcal{C}_2| = \Omega(m)$. Here too we can show that there exists a subset of $\mathcal{C}_2$, denoted $\mathcal{C}'_2$, such that $|\mathcal{C}'_2| = \Omega(m)$, and every vertex v that participates in one of the C_4s in $\mathcal{C}'_2$ actually participates in $\Omega(d(v))$ edges-disjoint C_4s in $\mathcal{C}_2$.

[12] Indeed, graphs with arboricity greater than $n^{1/2}$ necessarily contain at least one C_4, but since we are interested in a one-sided error algorithm, and α is only known to be an upper bound on α, the algorithm cannot reject if it is provided with $\alpha > n^{1/2}$.

[13] This is a fairly standard and simple procedure, where we use the fact that graph has bounded arboricity, so that most of its edges have at least one endpoint with degree θ_0.

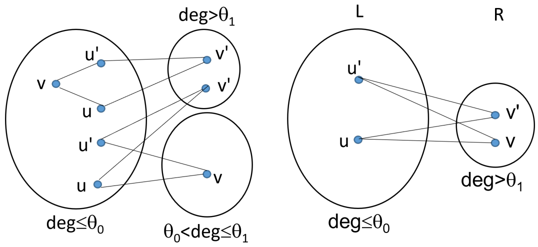

Figure 1 An illustration for some of the cases considered in the analysis of the algorithm for C_4-freeness. On the left side are two examples in which there is a single vertex v' with degree greater than θ_1, so that there is a vertex v with degree at most θ_1 with two neighbors whose degree is at most θ_0. On the right is an illustration when there are two such vertices with degree greater than θ_1.

Recall that by the definitions of $\mathcal{C}$ and $\mathcal{C}_2$ and since $\mathcal{C}_2' \subseteq \mathcal{C}_2 \subseteq \mathcal{C}$, the following holds. For each 4-cycle ρ in $\mathcal{C}_2'$, since it is in $\mathcal{C}_2$, there are two vertices whose degree is greater than θ_1. Therefore, by the definition of $\mathcal{C}$, they are both adjacent on ρ to two vertices whose degree is at most θ_0. Hence, if we consider the subgraph induced by the vertices and edges of the C_4s in $\mathcal{C}_2'$, it is a bipartite graph, where on one side, denoted L, all vertices have degree at most θ_0, and on the other side, denoted R, all vertices have degree greater than θ_1. Furthermore, by the definition of $\mathcal{C}_2'$, for each vertex in R, a constant fraction of its neighbors (in the original graph G) belong to L, and for each vertex in L, a constant fraction of its neighbors belong to R. For an illustration, see the RHS of Figure 1.

Hence, if we select an edge almost uniformly and pick one of its endpoints with equal probability, with high constant probability we obtain a vertex $v \in R$. Conditioned on this event, since $d(v) > \theta_1$, the algorithm will perform $\widetilde{O}(n^{1/4}\alpha^{1/2})$ random walks of length two from v, and with high constant probability, a constant fraction of these walks will be of the form (v, u, v') where $u \in L$ and $v' \in R$. If for some v' we get two walks, (v, u, v') and (v, u', v') for $u \neq u'$, then a C_4 is detected.

Observe that since all vertices in R have degree greater than $\theta_1 = \Theta(n^{1/2})$, we have that $|R| \leq 2m/\theta_1 = O(n^{1/2}\alpha)$. This can be used to show that the expected number of pairs of walks that induce a C_4 is greater than 1. In order to show that we actually get such a pair with high constant probability, we perform a more careful analysis to bound the variance.

A (two-sided error) lower bound for testing C_4-freeness in constant arboricity graphs

To obtain this lower bound of $\Omega(n^{1/4})$, we define two distributions over graphs. In the support of the first distribution, $\mathcal{D}_0$, all graphs are C_4-free, and in the support of the second distribution, $\mathcal{D}_1$, all graphs are $\Omega(1)$-far from being C_4-free. Furthermore, $\mathcal{D}_0$ is uniform over all graphs isomorphic to a specific graph G_0, and $\mathcal{D}_1$ is uniform over all graphs isomorphic to a specific graph G_1.

We next describe a slightly simplified version of the two graphs (which cannot be used to prove the lower bound, but gives the essence of the proof). Both graphs are bipartite graphs, where one side, Y, contains $\Theta(\sqrt{n})$ vertices, and the other side, X, contains $\Theta(n)$ vertices, In G_0, each vertex in X has a unique pair of neighbors in Y (so there are no C_4s). On the other hand, in G_1, each vertex x in X has a "twin", x', where x and x' have the same pair of neighbors in Y (thus creating $\Omega(n)$ edge-disjoint C_4. See Figure 2. Observe that the arboricity of both graphs is 2 as for any subset of vertices S, the number of edges within S is at most $|S \cap X| \cdot 2$ so the average degree in the subgraph induced by S is at most 2.

In order to prove that no (possibly adaptive) algorithm can distinguish between a graph selected according to $\mathcal{D}_0$ and a graph selected according to $\mathcal{D}_1$, we define two processes, $\mathcal{P}_0$ and $\mathcal{P}_1$, which answer the queries of a testing algorithm while selecting a graph from $\mathcal{D}_0$ (respectively, $\mathcal{D}_1$) "on the fly". The lower bound of $\Omega(n^{1/4})$ follows from the fact that when performing fewer than $n^{1/4}/c$ queries (where c is a sufficiently large constant), for both distributions, with high constant probability, each new neighbor query is answered by a uniformly selected vertex id.

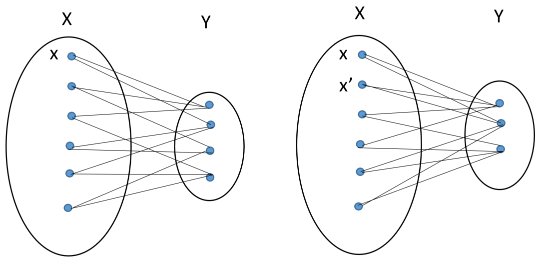

Figure 2 An illustration for the lower bound construction. The graph on the left is C_4-free while the graph on the right contains $\Omega(m)$ edge-disjoint C_4s and is hence $\Omega(1)$-far from being C_4-free.

A one-sided error lower bound for testing C_4-freeness in graphs with arboricity α

We next discuss the lower bound of $\Omega(n^{1/4}\alpha^{1/2})$ for graphs with arboricity $\alpha = \Omega(\log n)$ and one-sided error algorithms.

Here we define a single distribution $\mathcal{D}$ which is uniform over a family of graphs with arboricity α such that almost all graphs in this family are $\Omega(1)$-far from C_4-free.

Roughly speaking, the graphs in the support of $\mathcal{D}$ are random bipartite graphs, where one side, Y, is of size $\Theta(\sqrt{n}\alpha)$ and the other side, X, is of size $\Theta(n)$. Every vertex in X has α neighbors in Y, and every vertex in Y has $\Theta(\sqrt{n})$ neighbors in X. We need to show that if we select such a graph randomly, then on one hand it will be $\Omega(1)$-far from C_4-free, and on the other hand, in order to detect a C_4, any algorithm must perform $\Omega(n^{1/4}\alpha^{1/2})$ queries.

We next discuss the high-level idea as to why the resulting graphs are (with high constant probability) far from being C_4-free. Consider a fixed edge (x, y) in the bipartite graph, where $x \in X, y \in Y$. The number of C_4s this edge participates in is determined by the number of edges between the sets of neighbors of x and y, respectively $\Gamma(x)$ and $\Gamma(y)$. Recall that x has $\Theta(\alpha)$ neighbors and y has $\Theta(\sqrt{n})$ neighbors. Since overall there are $|X| \cdot |Y| = \Theta(n^{3/2}\alpha)$ potential pairs in the bipartite graph, and $\Theta(n\alpha)$ edges, each pair in $X \times Y$ is an edge with probability $\Theta(1/\sqrt{n})$. Hence, the expected number of edges between $\Gamma(x)$ and $\Gamma(y)$ is $|\Gamma(x)| \cdot |\Gamma(y)| \cdot (1/\sqrt{n}) = \Theta(\alpha)$. By analyzing the variance between pairs of edges, we furthermore show that with high constant probability, most edges do not participate in too many C_4s. Combining the two insights, it follows that with high constant probability, the graph is indeed far form being C_4-free.

In order to prove that any algorithm that performs at most $n^{1/4}\alpha^{1/2}/c$ queries (for a sufficiently large constant c), will not detect a C_4 with high constant probability, we actually prove that it will not detect *any* cycle. Roughly speaking, we show that by the randomness of the construction, since $|Y| = \Theta(\sqrt{n}\alpha)$, and the algorithm performs $O(\sqrt{|Y|})$ queries, each new neighbor query is answered by a uniformly distributed vertex that has not yet been observed. Therefore, the algorithm essentially views a forest.

A central challenge that we need to overcome is that we do not want to allow parallel edges, where the above construction might lead to their existence. One possibility is to first define the distribution over graphs with parallel edges and then to remove them. The benefit is that due to the higher degree of independence in the construction, it is somewhat easier to formally prove that the graphs obtained (with parallel edges) are with high probability $\Omega(1)$-far from C_4-free, and this remains the case when we remove parallel edges.

However, this creates a difficulty when we turn to argue that no (one-sided error) algorithm can detect a C_4 unless it makes $\Omega(n^{1/4}\alpha^{1/2})$ queries. The difficulty is due to the fact that in the formal proof we need to deal with dependencies that arise due to varying degrees (which occur because parallel edges are removed). While intuitively, varying degree should not actually "help" the algorithm, this intuition is difficult to formalize. Hence, we have chosen to define the distribution, from the start, over graphs that do not have parallel edges. This choice creates some technical challenges of its own (in particular in the argument that the graphs obtained are $\Omega(1)$-far from C_4-free), but we are able to overcome them. For more details see the full version.

1.2.2 The algorithm for C_6-freeness

Recall that for C_6 we have a (one-sided error) testing algorithm whose query complexity is $\widetilde{O}(n^{1/2})$. In addition to assuming (for the sake of the exposition) that ϵ is a constant, we also ignore polylogarithmic factors in n. Similarly to the algorithm for testing C_4-freeness, the algorithm for testing C_6-freeness in constant arboricity graphs is governed by two thresholds. The first, θ_0, is of the order of the arboricity, so that it is a constant (recall that we assume that ϵ is a constant). The second, θ_2, is of the order of $\sqrt{n}$.

The algorithm repeats the following process several times. It selects a vertex v uniformly at random, and if $d(v) \leq \theta_0$, it performs a *restricted* BFS starting from v to depth 4. Specifically:

1. Whenever a vertex u is reached such that $d(u) \leq \theta_0$, all its neighbors are queried.
2. Whenever a vertex u is reached such that $d(u) > \theta_0$ and u is reached from a vertex u' such that $d(u') \leq \theta_0$, there are two sub-cases. If $d(u) \leq \theta_1$, then all of u's neighbors are queried. Otherwise, θ_1 neighbors of u are selected uniformly at random.
3. Whenever a vertex u is reached from a vertex u' such that both $d(u) > \theta_0$ and $d(u') > \theta_0$, the BFS does not continue from u.

The algorithm rejects if and only if it observes a C_6.

Consider a graph that is far from being C_6-free, so that it contains a set of $\Omega(m) = \Omega(n)$ edge-disjoint C_6s. Furthermore, it contains such a set, denoted $\mathcal{C}$ for which every C_6 in $\mathcal{C}$ contains at most three vertices with degree greater than θ_0, and furthermore, these vertices are *not* adjacent on the C_6. We partition $\mathcal{C}$ into three subsets: $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, depending on the number of vertices with degree greater than θ_0 that it contains.

If either $|\mathcal{C}_1| = \Omega(m)$, or $|\mathcal{C}_2| = \Omega(m)$, then it is not hard to show that the algorithm will detect a C_6 with high constant probability. The more interesting part of the proof is handling the case in which only $|\mathcal{C}_3| = \Omega(m)$.

In this case we define an auxiliary multi-graph, denoted G', over the set of vertices that participate in C_6s belonging to $\mathcal{C}_3$, and have degree greater than θ_0 (in G). We denote this set of vertices by M, and the set of vertices with degree at most θ_0 that participate in these C_6s, by L.

Assume for simplicity that each vertex in L has degree exactly 2 (i.e., it participates in a single C_6). For each pair of vertices in M, we put in G' a set of parallel edges, whose size equals the number of length-2 paths between them in G that pass through vertices in L. Hence, for each C_6 in $\mathcal{C}$, we have a triangle in G', where these triangles are edge-disjoint, and we denote their set by $\mathcal{T}$. See Figure 3.

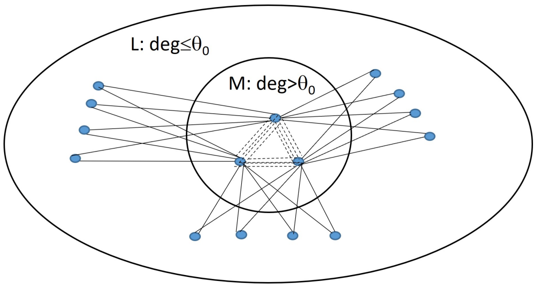

Figure 3 An illustration of the auxiliary (multi-)graph G' in the C_6-freeness testing algorithm. The dashed lines represent edges in G', each one corresponding to a length-2 path in G that passes through a vertex with degree at most θ_0.

Observe that selecting a vertex uniformly at random from L and querying its two neighbors in M corresponds to selecting an edge uniformly at random in G'. If we add an additional simplifying assumption by which (in G), vertices belonging to M only neighbor vertices belonging to L, then our algorithm on G essentially translates to picking a random edge in G'. Then depending on the degree of the endpoints, either querying all their neighbors in G' or θ_1 random neighbors.

Let H denote the subset of vertices in M whose degree in G is greater than θ_1. If relatively many triangles in $\mathcal{T}$ contain at most one vertex in H, then we are done, since these triangles contain an edge for which both endpoints have degree at most θ_1. Hence, it remains to address the case in which almost all triangles in $\mathcal{T}$ have two or three vertices in H.

Roughly speaking, in this case we show that the existence of many edge-disjoint, but not vertex-disjoint, triangles in G' that contain such high-degree vertices implies the existence of "many more" triangles that may be caught by our algorithm. As an illustrative extreme (but easy) special case, assume that in G' there are only three vertices. Then the existence of some number t of edge-disjoint triangles between them, actually implies the existence of t^3 (non edge-disjoint) triangles.

1.2.3 The general lower bound for C_k-freeness, $k \geq 6$

We establish our general lower bound of $\Omega(n^{1/3})$ for one-sided error testing of C_k-freeness when $k \geq 6$ by building on a lower bound for testing triangle-freeness that appears in [3, Lemma 2]. This lower bound for testing triangle-freeness is based on the difficulty of detecting a triangle in graphs selected uniformly from a family $\mathcal{G}_{n'}$ of graphs in which almost all graphs are $\Omega(1)$-far from being triangle-free. All graphs in the family are d-regular tri-partite graphs over n' vertices and the lower bound on the number of queries necessary to detect a triangle (with constant probability), is $\Omega(\min\{d, n'/d\})$. By setting $d = \sqrt{n'}$, the lower bound is $\Omega(\sqrt{n'})$.

We show that, for any constant $k \geq 6$, if we had a one-sided error testing algorithm $\mathcal{A}$ for C_k-freeness of graphs with n vertices and constant arboricity using at most $n^{1/3}/c$ queries (for a constant c), then we would be able to detect triangles in graphs selected uniformly from $\mathcal{G}_{n'}$ using at most $\sqrt{n'}/c'$ queries (for a constant c').

To this end we define an algorithm $\mathcal{A}$ that, given query access to a graph $G' \in \mathcal{G}_{n'}$, implicitly defines a graph G for which the following holds. First, the number of vertices in G is $n = \Theta((n')^{3/2})$, and the number of edges is $m = \Theta(m')$, where m' is the number of edges in G' (so that $m' = \Theta((n')^{3/2})$). Second, G has arboricity 2. Third, the distance of G to C_k-freeness is of the same order as the distance of G' to triangle-freeness. Fourth, there is a one-to-one correspondence between triangles in G' and C_ks in G. The basic idea is to replace edges in the tri-partite graph G' with paths of length $k/3$. See Figure 4

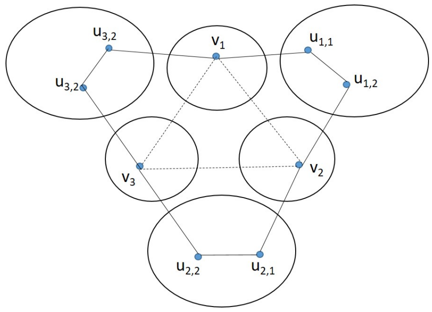

Figure 4 An illustration for the lower bound construction for C_k-freeness in constant arboricity graphs when $k = 9$. The three circles in the middle and the dashed lines represent a graph $G' \in \mathcal{G}_{n'}$. The outer circles represent the additional vertices in G. Since $k = 9$ in this example, each edge in G' is replaced by a path of length 3 in G.

Assuming there existed a testing algorithm $\mathcal{A}$ as stated above, the algorithm $\mathcal{A}'$ would use it to try and find a C_k in G (and hence a triangle in G'). In order to be able to run $\mathcal{A}$ on G, the algorithm $\mathcal{A}'$ must be able to answer queries of $\mathcal{A}$ to G by performing queries to G'. We show how this can be done with a constant multiplicative overhead. Hence, the lower bound of $\Omega(\sqrt{n'})$ for testing triangle-freeness (when the degree is $\Theta(\sqrt{n'})$) translated into a lower bound of $\Omega(n^{1/3})$ for testing C_k-freeness.

1.2.4 The general upper bound for C_k-freeness

Recall that our starting point is that if G is $\Omega(1)$-far from being C_k-free, then it contains a set $\mathcal{C}$ of $\Omega(m)$ edge-disjoint C_k's that do not contain any edge between vertices that both have degree greater than $\theta_0 = \Theta(\alpha)$. We refer to vertices with degree at most θ_0 as *light* vertices, and to those with degree greater than θ_0 as *heavy*. Hence, each C_k in $\mathcal{C}$ has at least $\lceil k/2 \rceil$ light vertices, and each heavy vertex on it neighbors two light vertices.

We present two different algorithms, where each of them is suitable for a different setting. The basic idea of both algorithms is to take a large enough sample of vertices and edges so that the subgraph determined by the sampled light vertices and their incident edges, as well as the sampled edges, contains a copy of C_k. The query complexity of each algorithm is stated following its high-level description.

The first algorithm

Our first algorithm simply samples vertices uniformly, independently at random, and then performs queries that reveal the neighbors of all light vertices in the sample. To analyze what is the sufficient sample size for this algorithm, consider the following generalization of the birthday paradox for k-way collisions. Assume we sample elements under the uniform distribution over $[n]$. Then we obtain a k-way collision after taking $\Theta(n^{1-1/k})$ samples.

Similarly, suppose we sample vertices uniformly from a graph that is composed only of n/k vertex-disjoint copies of C_k. Then, after sampling $\Theta(n^{1-1/k})$ vertices, we will hit all the vertices of at least one of the copies (with high constant probability). Conditioned on this event, if we reveal the neighborhood of all the vertices in the sample, then we obtain a C_k.

The next observation is that, in fact, we only need to hit a *vertex cover* of a copy of a C_k (as opposed to all its vertices). In particular we would like to hit such a cover that contains only light vertices, which we refer to as a *light vertex cover*. For constant α, this yields an improved dependence on k in the exponent, i.e., $O(n^{1-1/\lfloor k/2 \rfloor})$ sampled vertices suffice.

When taking into account the dependence on α (so that it is not necessarily true that $m = O(n)$) and incorporating this in the analysis, we prove that the query complexity is upper bounded by $O(m \cdot (\alpha/m)^{2/k})$ for even k and $O(m \cdot (\alpha/m)^{2/(k+1)})$ for odd k (up to a polynomial dependence on k). Since $\alpha \leq \sqrt{m}$ it follows that the above bounds are at most $O(m^{1-1/k})$ and $O(m^{1-1/(k+1)})$, respectively.

The second algorithm

Our second algorithm is designed for the case in which k is odd and $m = \Omega(\alpha^{(k+3)/2})$. In particular it is preferable when α is constant. We observe that when k is odd, for each C_k in $\mathcal{C}$, there is an edge in which both endpoints are light vertices. Therefore, if we sample edges (almost) uniformly from the graph (using a variant of the procedure Select-an-Edge), then we are likely to hit one of these edges. This additional step reduces the number of vertices we need to hit in each copy by 2, which results in improved complexity for some range of the parameters. In particular, the query complexity of this algorithm is $O(m \cdot (\alpha^2/m)^{2/(k-1)})$. Specifically, when α is a constant, the query complexity of this algorithm (which works for odd k) is $O(m^{1-2/(k-1)})$ (instead of $O(m^{1-2/(k+1)})$).

General subgraph F

Our first algorithm also works for any constant-size subgraph, F, where the upper bound on the sample size is of the form $m^{1-1/\ell(F)}$ where $\ell(F)$ depends on the structure of F, as defined next.

▶ **Definition 10.** *For a graph $F = (V_F, E_F)$ let $\mathcal{VC}(F)$ denote the set of all vertex covers of F. For a vertex cover Z of F we denote by $\mathcal{VC}'(Z)$ the set of vertex covers of F that are subsets of Z. We define $\ell(F) = \max_{Z \in \mathcal{VC}(F)} \left\{ min_{B \in \mathcal{VC}'(Z)} (|B|) \right\}$.*

Observe that by Definition 10, we have that $\ell(F)$ is lower bounded by the size of a minimum vertex cover of F and is upper bounded by $k = |V_F|$.

The high-level idea is that if we want to find a copy of F, it suffices to hit a light vertex cover of this copy and then query all neighbors of the sampled light vertices.

1.3 Related work

In this subsection we shortly discuss several related works, in addition to the two aforementioned works regarding testing C_3-freeness [3, 26].

Testing subgraph-freeness for fixed, constant size subgraphs in the dense-graphs model can be done using a number of queries that depends only on $1/\epsilon$ (where the dependence is a tower of height poly$(1/\epsilon)$), as shown by Alon, Fischer, Krivelevich and Szegedy [2]. Alon [1] proved that a super polynomial dependence on $1/\epsilon$ is necessary, unless the subgraph F is bipartite. Goldreich and Ron addressed the problem in the bounded-degree model [20], and gave a simple algorithm that depends polynomially on $1/\epsilon$ and the maximum degree in the graph, and exponentially on the diameter of F.

A special case of graphs that have bounded arboricity is the family of graphs that exclude a fixed minor (a.k.a. minor-free graphs). Newman and Sohler [29] showed that for this family of graphs, in the bounded-degree model, all properties can be tested with no dependence on the size of the graph G. Moreover, it was recently shown [25, 27] that any property which is monotone and additive[14] (and in particular F-freeness where F is a connected graph) can be tested using a number of queries that is only polynomial in $1/\epsilon$ and d, where d is the degree bound (and $O(d^{\rho(\epsilon)})$ in general $(\epsilon, \rho(\epsilon))$-hyperfinite graphs[15]). For minor-free graphs with unbounded degrees, Czumaj and Sohler [10] showed that a property is testable with one sided error and a number of queries that does not depend on the size of the graph if and only if it can be reduced to testing for a finite family of finite forbidden subgraphs.[16] The correctness of their algorithm relies on the fact that the arboricity of minor-free graphs remains constant even after contractions of edges (which is not the case for general constant-arboricity graphs).

In general graphs, it was shown that k-path freeness [22] and more generally T-freeness where T is a tree of order k [17], can be tested with time and query complexity that depend only on k, assuming the edges of the graph can be accessed uniformly at random. Testing cycle-freeness (where a no instance is a graph that is far from being a forest) was studied in the bounded-degree model in [20], where a two-sided error algorithm was given whose query complexity is polynomial in $1/\epsilon$ and the degree bound. Czumaj et. al [9] showed that the complexity of this problem for one-sided error algorithms in the bounded-degree model is $\widetilde{\Theta}(\sqrt{n})$ (for constant ϵ – their algorithm has a polynomial dependence on $1/\epsilon$), and the algorithm can be adapted to the general-graphs model.

Other sublinear-time graph algorithms for counting and sampling (rather than detecting) subgraphs that give improved results when the graph G has bounded arboricity include [14, 12, 15, 13].

1.4 Organization

We start in Section 2 with some preliminaries. In Section 3 we give the upper bound for testing C_4-freeness. All missing details and proofs appear in the full version of the paper.

2 Preliminaries

Unless stated explicitly otherwise, the graphs we consider are simple, so that in particular they do not contain any parallel edges. We denote the number of vertices in the graph by n and the number of edges by m. Every vertex v in the graph has a unique id, denoted $id(v)$, and its degree is denoted by $d(v)$.

We work in what is known as the *general graph model* [30, 24]. In particular, under this model, the *distance* of a graph G to C_k-freeness, denoted $dist(G, C_k\text{-free})$, is the minimum fraction of edges that should be removed from G in order to obtain a C_k-free graph. As for the allowed queries, a neighbor query to the ith neighbor of a vertex v is denoted by $nbr(v, i)$, and to its degree by $deg(v)$. A pair query between two vertices v_1 and v_2 is denoted by $pair(v_1, v_2)$. Given query access to a graph G and a parameter ϵ, a one-sided error testing

[14] A property is monotone if it closed under removal of edges and vertices. A property is additive if it is closed under the disjoint union of graphs.

[15] Let ρ be a function from $\mathbb{R}_+$ to $\mathbb{R}_+$. A graph $G = (V, E)$ is $(\epsilon, \rho(\epsilon))$-hyperfinite if for every $\epsilon > 0$ it is possible to remove $\epsilon|V|$ edges of the graph such that the remaining graph has connected components of size at most $\rho(\epsilon)$.

[16] They consider a model in which they can perform only random neighbor queries.

algorithm for C_k-freeness should accept G if it is C_k-free, and should reject G with probability at least 2/3 if $dist(G, C_k\text{-free}) > \epsilon$. If the algorithm may also reject C_k-free graphs with probability at most 1/3, then it has two-sided error.

As noted in the introduction, we assume our algorithms for graphs whose arboricity is not promised to be constant, are given an upper bound α on the arboricity $arb(G)$ of the tested graph G, and their complexity depends on this upper bound. Alternatively, if the algorithm is provided with the number of edges, m, then it may run a procedure from [26] to obtain a value α^* that with high constant probability satisfies the following: (1) $\alpha^* \leq 2arb(G)$; (2) The number of edges between vertices whose degree is at least $\alpha^*/(c\epsilon)$ for a constant c is at most $(1-\epsilon/c')m$ (for another, sufficiently large, constant c'). Up to polylogarithmic factors in n, the query complexity and running time of the procedure are $O(arb(G)/\epsilon^3)$ with high probability (assuming the average degree is $\Omega(1)$).

Throughout this work we assume, whenever needed, that ϵ is upper bounded by some sufficiently small constant (or else it can be set to that constant).

We also make use of the following claim – whose proof is given in the full version.

▷ Claim 11. For an integer s let $\{\chi_{i,j}\}_{(i,j)\in\Phi(s)}$ be Bernoulli random variables where $\Pr[\chi_{i,j} = 1] = \mu$ for every $(i,j) \in \Phi(s)$. Suppose that the following conditions hold for some $c_1 > 0$ and $c_2 > 4$.

1. For every $(i_1, j_1) \in \Phi(s)$ and $(i_2, j_2) \in \Phi(s)$ such that the four indices are distinct, χ_{i_1,j_1} and χ_{i_2,j_2} are independent.
2. For every $(i_1, j_1) \in \Phi(s)$ and $(i_2, j_2) \in \Phi(s)$ such that exactly two of the four indices are the same, $\Pr[\chi_{i_1,j_1} = \chi_{i_2,j_2} = 1] \leq c_1 \cdot \mu^{3/2}$.
3. $s \geq c_2/\sqrt{\mu}$.

Then $\Pr\left[\sum_{(i,j)\in\Phi(s)} \chi_{i,j} = 0\right] \leq \frac{1+c_1}{c_2}$.

3 An upper bound of $\widetilde{O}(n^{1/4}\alpha)$ for testing C_4-freeness

In this section we prove the more general (arboricity-dependent) form of the upper bound for testing C_4-freeness which is stated as Theorem 6 in the introduction.

Recall that the assumption on α is that it is an upper bound on the arboricity $arb(G)$. While it is known that for graphs with $arb(G) > \sqrt{n}$ there exists a C_4, we cannot simply reject if we get $\alpha > n^{1/2}$ since it might be that $arb(G) < \sqrt{n}$ (and we want one-sided error). However, in the case that $\alpha > n^{1/2}$, the $n^{1/4}\alpha$ term is replaced by $n^{3/4}$ (and the additive α term is due to the edge sampling).

The algorithm referred to in Theorem 6 is described next.

Algorithm 1 Test-C_4-freeness(n, ϵ, α).

1. Let $\theta_0 = 4\alpha/\epsilon$, $\theta_1 = c_1 \cdot \sqrt{n}/\epsilon$ (where c_1 will be determined subsequently) and $\theta_{min} = \min\{\theta_0, \theta_1\}$ (it is useful to read the algorithm while having in mind that $\theta_0 \leq \theta_1$ (i.e., $\alpha = O(\sqrt{n})$) so that $\theta_{min} = \theta_0$).
2. Repeat the following $t = \Theta(1/\epsilon)$ times:
   a. Select an edge e by calling the procedure Select-an-Edge(α, ϵ), which appears below. If it does not return an edge, then continue to the next iteration.
   b. Select an endpoint v of e by flipping a fair coin.
   c. If $d(v) \leq \theta_1$, then select $s_1 = \Theta(\sqrt{d(v)/\epsilon})$ $(= O(n^{1/4}/\epsilon))$ random neighbors of v, and for each neighbor u such that $d(u) \leq \theta_{min}$ query all the neighbors of u.
   d. Otherwise $(d(v) > \theta_1)$, perform $s_2 = \Theta(\sqrt{(n\alpha/\theta_1)\log n/\epsilon^2})$ $(= \tilde{O}(n^{1/4}\alpha^{1/2}/\epsilon^2))$ random walks of length 2 starting from v.
   e. If a C_4 is detected, then return it, "Reject" and terminate.
3. Return "Accept".

We note that the algorithm can be unified/simplified so that it only performs random walks of length-2, where the number of walks is $\Theta(n^{1/4}\alpha/\epsilon^2)$, but then the analysis becomes slightly more complicated.

Algorithm 2 Select-an-edge(ϵ, α).

1. Repeat the following $\Theta(\alpha/\epsilon)$ times:
   a. Select a vertex u uniformly at random.
   b. If $d(u) \leq \theta_0$ for $\theta_0 = 4\alpha/\epsilon$, then with probability $d(u)/\theta_0$ select an edge incident to u uniformly at random and return it.
2. If no edge was selected, then return "Fail".

We start by stating a claim concerning the procedure Select-an-Edge where its proof is deferred to the full version.We then state and prove two additional claims that will be used in the proof of Theorem 6.

▷ Claim 12. With probability at least $2/3$ the procedure Select-an-Edge returns an edge. Conditioned on it returning an edge, each edge incident to a vertex with degree at most θ_0 is returned with probability at least $1/(2m')$ and at most $1/m'$, where m' is the number of edges incident to vertices with degree at most θ_0.

▷ Claim 13. Let v be a vertex and let $C(v, \theta_{min})$ be a set of edge-disjoint C_4's containing v such that the neighbors of v on these C_4s all have degree at most θ_{min}, where θ_{min} is as defined in the algorithm.[17] Suppose that $|C(v, \theta_{min})| \geq 1$ and let $\epsilon' = |C(v, \theta_{min})|/d(v)$. If we select $s = 16\sqrt{d(v)/\epsilon'}$ random neighbors of v, and for each selected neighbor u such that $d(u) \leq \theta_{min}$ we query all the neighbors of u, then the probability that we obtain a C_4 is at least $9/10$.

Proof. Let $E'(v)$ denote the set of edges incident to v that participate in the set $C(v, \theta_{min})$. By the premise of the claim, $|E'(v)|/d(v) = 2|C(v, \theta_{min})|/d(v) = 2\epsilon'$. Let s' be the number of neighbors of v that are incident to edges in $E'(v)$ among the s selected random neighbors of v. It holds that $\mathbb{E}[s'] = 2\epsilon' \cdot s$, and by the multiplicative Chernoff bound, $s' \geq \epsilon' \cdot s$ with probability at least $1 - e^{-\epsilon' \cdot s/4}$. We first show that this probability is at least $19/20$, and then condition on this event. By the setting of $s = 16\sqrt{d(v)/\epsilon'}$, it holds that $\epsilon' \cdot s = 16\sqrt{\epsilon' \cdot d(v)}$, and by the setting of $\epsilon' = |C(v, \theta)|/d(v)$, we get $\epsilon' \cdot s \geq 16\sqrt{|C(v, \theta_{min})|} \geq 16$. Therefore, with probability at least $19/20$, $s' > \epsilon' \cdot s = 16\sqrt{\epsilon' \cdot d(v)}$. We condition on this event and consider only those s' selected neighbors of v that are endpoints of $E'(v)$.

For each 4-cycle $\rho \in C(v, \theta_{min})$, let $u_1(\rho)$ and $u_2(\rho)$ be the two neighbors of v on this C_4 (so that they are endpoints of edges in $E'(v)$). Since the C_4s in $C(v, \theta_{min})$ are edge-disjoint, these vertices are distinct. Observe that the s' selected neighbors of v are uniformly distributed in $\bigcup_{\rho \in C(v, \theta_{min})} \{u_1(\rho), u_2(\rho)\}$, and that $s' \geq 16 \cdot \sqrt{|C(v, \theta_{min})|}$. Hence, by the "birthday paradox", with high constant probability, the sample of neighbors of v contains two vertices, $u_1(\rho)$, and $u_2(\rho)$ for some $\rho \in C(v, \theta_{min})$. Conditioned on this event, once the (at most θ_{min}) neighbors of $u_1(\rho)$ and $u_2(\rho)$ are queried, the four-cycle ρ is observed. ◁

▷ Claim 14. Let G be a graph over n vertices and m edges, and let $\theta_1, \epsilon', \epsilon''$ be parameters. Suppose that G contains a bipartite subgraph $G' = (L, R, E(G'))$ such that every vertex in R has degree at least θ_1 in G. Let v be a vertex in R such that v has at least $\epsilon' \cdot d(v)$

[17] Actually, we do not rely on the setting of θ_{min}, so this claim holds for any threshold value.

neighbors in L where each of these neighbors, u, has at least $\epsilon'' \cdot \max\{d(u), \frac{m}{n}\}$ neighbors in R. If $\theta_1 \geq 2\sqrt{n/(\epsilon' \cdot \epsilon'')}$ and we take $s_2 \geq \frac{32}{\epsilon' \cdot \epsilon''} \cdot \sqrt{2\log n \cdot |R|}$ random walks of length 2 from v for a sufficiently large constant c', then with probability at least $9/10$ we shall detect a C_4 in G.

Proof. For a pair of vertices v and $v' \neq v$ in R, let $\ell_2(v, v')$ be the number of length-2 paths between v and v', and let $\ell_2(v, R) = \sum_{v' \in R} \ell_2(v, v')$. Consider taking two random length-2 walks from v, and let $\mathcal{E}_1$ be the event that both of them end at vertices in R. Let $\mathcal{E}_2$ be the event that these two paths are distinct and end at the same vertex. Then for each single vertex $v' \in R$, conditioned on $\mathcal{E}_1$, the probability that the two walks end at v' is exactly $\frac{\ell_2(v,v')}{\ell_2(v,R)} \cdot \frac{\ell_2(v,v')-1}{\ell_2(v,R)}$. Therefore,

$$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] = \sum_{v' \in R} \frac{\ell_2(v, v')}{\ell_2(v, R)} \cdot \frac{\ell_2(v, v') - 1}{\ell_2(v, R)} = \frac{1}{(\ell_2(v, R))^2} \cdot \sum_{v' \in R} (\ell_2(v, v'))^2 - \frac{1}{\ell_2(v, R)} . \quad (1)$$

We would like to lower bound the above probability. For the first term on the right-hand-side, by applying the Cauchy-Schwartz inequality we get that

$$\frac{1}{(\ell_2(v, R))^2} \cdot \sum_{v' \in R} (\ell_2(v, v'))^2 \geq \frac{1}{(\ell_2(v, R))^2} \cdot |R| \cdot \left(\frac{\ell_2(v, R))}{|R|}\right)^2 = \frac{1}{|R|} . \quad (2)$$

By combining Equations (1) and (2) we get that $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq \frac{1}{|R|} - \frac{1}{\ell_2(v,R)}$. Since each vertex in R has degree at least θ_1, we have that $|R| \leq \frac{2m}{\theta_1}$. By the premise of the claim regarding v and its neighbors, v has $\epsilon' d(v) \geq \epsilon' \cdot \theta_1$ neighbors in L, and each of them has at least $\epsilon'' \cdot (m/n)$ neighbors in R. Therefore,

$$\ell_2(v, R) \geq \epsilon' \cdot \theta_1 \cdot \epsilon'' \cdot \frac{m}{n} \geq \frac{\epsilon' \cdot \epsilon'' \cdot \theta_1^2 \cdot |R|}{2n} \geq 2|R|, \quad (3)$$

where the last inequality is by the premise $\theta_1 \geq 2\sqrt{n/(\epsilon' \cdot \epsilon'')}$. Therefore, $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq \frac{1}{2|R|}$. So far we have shown that when taking two distinct random walks from v, and conditioned on them both ending at R (the event $\mathcal{E}_1$), the two paths collide on the end vertex (and hence result in a C_4) with probability at least $1/2|R|$. We shall now prove, that when taking s length-2 random walks from v, sufficiently many of them indeed end at R, and that with high probability, at least two of them collide, resulting in a C_4.

Consider first the event $\mathcal{E}_1$. By the premise of the claim, v has at least $\epsilon' \cdot d(v)$ neighbors in L, and each u of them has at least $\epsilon'' \max\{d(u), m/n\} \geq \epsilon'' d(u)$ neighbors in R. Therefore, the probability that a single random walk from v ends at R is at least $\epsilon' \cdot \epsilon''$. Hence, if we take $s \geq \frac{32}{\epsilon' \cdot \epsilon''}\sqrt{2\log n \cdot |R|}$ length-2 random walks from v, and let s' denote the number of walks that end at a vertex in R, we have that $\mathbb{E}[s'] = 32 \cdot \sqrt{2\log n \cdot |R|}$, and that with probability at least $9/10$, we have $s' \geq 16 \cdot \sqrt{2\log n \cdot |R|}$. We henceforth condition on this event.

Let $\chi'_{i,j}$ denote the event that the ith and jth random walks among the ones that end at R collide on the ending vertex (and thus result in a C_4). By the above discussion, we have that for a specific pair $i \neq j$, $\Pr[\chi'_{i,j} = 1] \geq 1/2|R|$. We now lower bound the probability that at least one pair of random walks from the s' that end in R detects a C_4, i.e. lower bound $\sum_{i,j \in [s']} \chi_{i,j}$, using Claim 11. For that end we also need to upper bound the variance of the sum.

Partition the vertices in R according to $\ell_2(v, v')$, where $R_x(v) = \{v' : 2^{x-1} < \ell_2(v, v') \leq 2^x\}$ for $x = 0, \ldots \log L \leq \log n$. Since $\sum_{v' \in R} \frac{\ell_2(v,v')}{\ell_2(v,R)} \cdot \frac{\ell_2(v,v')-1}{\ell_2(v,R)} > \frac{1}{2|R|}$, there exists at least one setting of x for which $\sum_{v' \in R_x} \frac{\ell_2(v,v')}{\ell_2(v,R)} \cdot \frac{\ell_2(v,v')-1}{\ell_2(v,R)} \geq \frac{1}{2|R|\log n}$. We denote this setting by x^* and observe that $x^* > 0$ (since for every $v' \in R_0$, $\ell_2(v, v') - 1 = 0$).

For every $i, j \in [s']$, $i < j$, we define a Bernoulli random variable $\chi_{i,j}$ that is 1 if and only if the ith and the jth random walks from v (among the s' considered) end at the same $v' \in R_{x^*}$ and pass through a different vertex in L. We next show that we can apply Claim 11 (with s in that claim set to s') to get an upper bound on the probability that $\sum_{i,j\in[s'],i<j} \chi_{i,j} = 0$ (which is an upper bound on the probability that we do not detect a C_4).

By the definition of the random variables, for every $i_1 \neq i_2, j_1 \neq j_2$, it holds that $\chi_{i_1,j_1}, \chi_{i_2,j_2}$ are independent, so that the first condition in Claim 11 is satisfied. Next, for any pair $i, j \in [s']$, $i < j$ we have that

$$\mu = \Pr[\chi_{i,j} = 1] = \sum_{v' \in R_{x^*}} \frac{\ell_2(v, v')}{\ell_2(v, R)} \cdot \frac{\ell_2(v, v') - 1}{\ell_2(v, R)} \geq \frac{1}{2|R| \log n} \; . \tag{4}$$

Therefore, we have that $s' \geq 16 \cdot \sqrt{2|R| \log n} = 16/\sqrt{\mu}$, and so the third condition in Claim 11 is satisfied (for $c_2 = 16$, where s' serves as the parameter s in the claim).

It remains to verify that the second condition holds. For any four indices $i_1, j_1, i_2, j_2 \in [s']$, $i_1 < j_1$, $i_2 < j_2$ such that exactly two of the four indices are the same,

$$\Pr[\chi_{i_1,j_1} = \chi_{i_2,j_2} = 1] = \sum_{v' \in R_{x^*}} \frac{\ell_2(v, v')}{\ell_2(v, R)} \cdot \left(\frac{\ell_2(v, v') - 1}{\ell_2(v, R)}\right)^2 \leq \mu \cdot \frac{2^{x^*} - 1}{\ell_2(v, R)} \; . \tag{5}$$

Since by Equation (4), $\mu = \sum_{v' \in R_{x^*}} \frac{\ell_2(v,v')}{\ell_2(v,R)} \cdot \frac{\ell_2(v,v')-1}{\ell_2(v,R)} \geq \frac{2^{2(x^*-1)}}{2(\ell_2(v,R))^2}$ (as $\ell_2(v, v') \geq 2^{x^*-1}$ for every $v' \in R_{x^*}$ and $\ell_2(v, v') - 1 \geq \ell_2(v, v')/2$), we get that $\Pr[\chi_{i_1,j_1} = \chi_{i_2,j_2} = 1] < \sqrt{2} \cdot \mu^{3/2}$, and so the second condition in Claim 11 holds as well (for $c_1 = \sqrt{2}$). Thus, the current claim follows. ◁

We are now ready to prove Theorem 6.

Proof of Theorem 6. Since the algorithm only rejects a graph G if it detects a C_4, it will always accept graphs that are C_4-free. Hence, we focus on the case that G is ϵ-far from being C_4-free.

Recall that $\theta_0 = 4\alpha/\epsilon$ and let $E_{>\theta_0}$ be the subset of edges in G where both endpoints have degree greater than θ_0. Since the arboricity of G is at most α, there are at most $2m/\theta_0$ vertices with degree greater than θ_0, so that $|E_{>\theta_0}| \leq (2m/\theta_0) \cdot \alpha = \epsilon m/2$ edges.

Since G is ϵ-far from C_4-free, if we remove all edges in $E_{>\theta_0}$, then we get a graph that is at least $(\epsilon/2)$-far from C_4-free. It follows that there exists a set of edge-disjoint C_4s, denoted $\mathcal{C}$, such that no C_4 in $\mathcal{C}$ contains an edge in $E_{>\theta_0}$, and $|\mathcal{C}| \geq \epsilon m/8$.

We next partition $\mathcal{C}$ into two disjoint subsets: $\mathcal{C}_1$ contains those C_4s that have at most one vertex with degree at least θ_1 in them, and $\mathcal{C}_2$ contains those that have at least two such vertices (where in the case $\theta_0 \leq \theta_1$ there will be exactly two). Since $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$, either $|\mathcal{C}_1| \geq \epsilon m/16$ or $|\mathcal{C}_2| \geq \epsilon m/16$ (possibly both).

The case $|\mathcal{C}_1| \geq \epsilon m/16$. Consider first the case that $|\mathcal{C}_1| \geq \epsilon m/16$. In order to analyze this case, we apply a process of "coloring" vertices and edges. Initially, all vertices and edges that participate in C_4s that belong to $\mathcal{C}_1$ are colored *green*, and all other vertices and edges are colored *red*. We next apply the following iterative process. As long as there is a green vertex v whose number of incident green edges is less than $\epsilon d(v)/64$, color v and its green incident edges by red. Observe that the total number of edges colored red by this process is at most $\epsilon m/32$. Furthermore, at the end of this process, every green vertex v has at least

$\epsilon d(v)/64$ incident green edges (and if a vertex is red, then all its incident edges are red). Let $\mathcal{C}'_1$ be the subset of $\mathcal{C}_1$ that consists of those C_4s in $\mathcal{C}_1$ whose edges all remain green after the process (and hence they are green), so that $|\mathcal{C}'_1| \geq \epsilon m/32$.

By the definition of $\mathcal{C}_1$, and hence also $\mathcal{C}'_1$, in each C_4 in $\mathcal{C}'_1$ there is at most one vertex with degree greater than θ_1, and no edges such that both endpoints have degree greater than θ_0. Assume without loss of generality that for each four-cycle $\rho \in \mathcal{C}'_1$, where $\rho = (v_0(\rho), v_1(\rho), v_2(\rho), v_3(\rho))$, $v_2(\rho)$ is the highest degree vertex (where $d(v_2(\rho))$ could be any value between 1 to n). Let $V_0(\mathcal{C}'_1) = \bigcup_{\rho\in\mathcal{C}'_1}\{v_0(\rho)\}$ denote this set of vertices (i.e., the ones that are across from the highest degree vertex in a (green) four-cycle in $\mathcal{C}'_1$).

Observation. *For every $\rho \in \mathcal{C}'_1$,*

1. *$d(v_0(\rho)) \leq \theta_1$, and*
2. *$v_1(\rho)$ and $v_3(\rho)$ are of degree at most $\theta_{min} = \min\{\theta_0, \theta_1\}$.*

To verify this observation, note that by the definition of $\mathcal{C}'_1$, for every $\rho \in \mathcal{C}'_1$, there is at most one vertex with degree greater than θ_1, and since $v_2(\rho)$ is the highest degree vertex in ρ, it follows that all three other vertices in ρ are of degree at most θ_1.

We now show that $d(v_1(\rho)) \leq \theta_0$, and the proof for $v_3(\rho)$ is identical. If $d(v_2(\rho)) > \theta_0$, then it must be the case that $d(v_1(\rho)) < \theta_0$, as otherwise both have degree greater than θ_0 and so they cannot be connected, which is a contradiction to them both being incident on the four-cycle ρ. If $d(v_2(\rho)) \leq \theta_0$, then since $v_2(\rho)$ is the highest degree vertex in ρ, $d(v_1(\rho)) \leq d(v_2(\rho)) \leq \theta_0$.

Therefore, for every $v \in V_0(\mathcal{C}'_1)$, it has at least $\epsilon d(v)/64$ neighbors u such that (v, u) is green and $d(u) \leq \theta_{min}$. Hence, overall in the graph, the set of vertices $V_0(\mathcal{C}'_1)$ has at least $\epsilon m/32$ green edges that are incident to it and their second endpoint is of degree at most $\theta_{min} \leq \theta_0$. It follows that conditioned on an edge being returned by procedure Select-an-Edge, by Claim 12, it returns an edge incident to a vertex $v \in V_0(\mathcal{C}'_1)$ with probability at least $(\epsilon m/32)/2m' > \epsilon/128$ (since $m' > \frac{1}{2}m$). So the probability that in some iteration of Test-C_4-freeness a vertex $v_0 \in V_0(\mathcal{C}'_1)$ is selected, is at least $1 - (1 - \frac{\epsilon}{128})^t > 9/10$ (recall that $t = \Theta(1/\epsilon)$ so that it suffices to set $t = 500/\epsilon$).

Conditioning on this event, we apply Claim 13. Specifically:

- $\theta_0 = 4\alpha/\epsilon$ (as defined in Step 1 in Algorithm Test-C_4-freeness);
- $C(v_0, \theta_{min})$ is the set of C_4s in $\mathcal{C}'_1$ that are incident to v_0;
- $\epsilon' = |C(v_0, \theta_{min})|/d(v) \geq \epsilon/128$ (since v_0 has at least $\epsilon d(v)/64$ incident green edges, and they can be partitioned into pairs such that each pair belongs to exactly one C_4 in $C(v_0, \theta_{min})$);
- $d(v_0) \leq \theta_1$ (by the above observation);

In order to apply the claim, we must ensure that $s > 16\sqrt{d(v_0)/\epsilon'}$. By the above, it is sufficient to set s_1 in Step 2c, to be $s_1 = 512\sqrt{d(v_0)/\epsilon}$.

Hence, by Claim 13, if Step 2c is applied to v_0, then a C_4 is observed with probability at least 9/10.

The analysis for the case that $|\mathcal{C}_2| \geq \epsilon m/16$ is similar, and due to space constraints, it is deferred to the full version.

We next turn to analyze the query complexity. By the settings of θ_0, θ_1, t, s_1 and s_2 in the algorithm, the query complexity of the algorithm is upper bounded as follows.

$$O\left(\frac{1}{\epsilon}\cdot\left(\frac{\alpha}{\epsilon}+\max\{s_1,s_2\}\right)\right) = O\left(\frac{1}{\epsilon}\left(\frac{\alpha}{\epsilon}+\max\left\{\sqrt{\frac{\theta_1}{\epsilon}}\cdot\theta_{min}, \frac{1}{\epsilon^2}\cdot\sqrt{\frac{n\alpha}{\theta_1}\cdot\log n}\right\}\right)\right) \quad (6)$$

For the case that $\alpha \le (c_1/4)\sqrt{n}$, we have that $\theta_{min} = \theta_0 = \Theta(\alpha/\epsilon)$ and that $\theta_1 = \Theta(\sqrt{n}/\epsilon)$, and so we get a complexity of

$$O\left(\epsilon^{-3} \cdot n^{1/4}\alpha^{1/2} \cdot \max\{\alpha^{1/2}, \log^{1/2} n\}\right) = O(\epsilon^{-3} \cdot n^{1/4}\alpha) \ , \tag{7}$$

where the last inequality is for $\alpha > \log n$, and otherwise the complexity is $O(\epsilon^{-3} \cdot n^{1/4}\alpha^{1/2} \log^{1/2} n)$.

For the case that $\alpha > (c_1/4)\sqrt{n}$, we have that $\theta_{min} = \theta_1 = \Theta(\sqrt{n}/\epsilon)$. Therefore, the complexity is

$$O(\epsilon^{-3} \cdot (\alpha + n^{3/4})) \ . \tag{8}$$

Thus, the proof is complete. ◀

References

1 Noga Alon. Testing subgraphs in large graphs. *Random Struct. Algorithms*, 21(3-4):359–370, 2002. doi:10.1002/rsa.10056.

2 Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000. doi:10.1007/s004930070001.

3 Noga Alon, Tali Kaufman, Michael Krivelevich, and Dana Ron. Testing triangle-freeness in general graphs. *SIAM Journal on Discrete Mathematics*, 22(2):786–819, 2008.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Color coding. *Journal of the ACM*, 42(4):844–856, 1995.

5 Uri Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6):450–461, 2007.

6 Jonathan W. Berry, Bruce Hendrickson, Randall A. LaViolette, and Cynthia A. Phillips. Tolerating the Community Detection Resolution Limit with Edge Weighting. *Physical Review E*, 83(5):056119, May 2011.

7 Ronald S. Burt. Structural holes and good ideas. *American journal of sociology*, 110(2):349–399, 2004.

8 James S. Coleman. Social capital in the creation of human capital. *American Journal of Sociology*, 94:S95–S120, 1988. URL: http://www.jstor.org/stable/2780243.

9 Artur Czumaj, Oded Goldreich, Dana Ron, C. Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures and Algorithms*, 45(2):139–184, 2014.

10 Artur Czumaj and Christian Sohler. A characterization of graph properties testable for general planar graphs with one-sided error (it's all about forbidden subgraphs). In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1525–1548, 2019. doi:10.1109/FOCS.2019.00089.

11 Jean-Pierre Eckmann and Elisha Moses. Curvature of co-links uncovers hidden thematic layers in the World Wide Web. *Proceedings of the National Academy of Sciences*, 99(9):5825–5829, 2002. doi:10.1073/pnas.032093399.

12 Talya Eden, Dana Ron, and Will Rosenbaum. The arboricity captures the complexity of sampling edges. In *46th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 52:1–52:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.52.

13 Talya Eden, Dana Ron, and Will Rosenbaum. Almost optimal bounds for sublinear-time sampling of k-cliques in bounded arboricity graphs. In *49th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 56:1–56:19, 2022. doi:10.4230/LIPIcs.ICALP.2022.56.

14 Talya Eden, Dana Ron, and C. Seshadhri. Sublinear time estimation of degree distribution moments: The arboricity connection. *SIAM J. Discret. Math.*, 33(4):2267–2285, 2019. doi:10.1137/17M1159014.

15 Talya Eden, Dana Ron, and C. Seshadhri. Faster sublinear approximation of the number of k-cliques in low-arboricity graphs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1467–1478, 2020. doi:10.1137/1.9781611975994.89.

16 Paul Erdős. Extremal problems in graph theory. In *In Proc. Symp. Theory of Graphs and its Applications*, pages 29–36, 1963.

17 Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. Three notes on distributed property testing. In *31st International Symposium on Distributed Computing, DISC*, pages 15:1–15:30, 2017. doi:10.4230/LIPIcs.DISC.2017.15.

18 Brooke Foucault Welles, Anne Van Devender, and Noshir Contractor. Is a friend a friend?: Investigating the structure of friendship networks in virtual worlds. In *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 4027–4032, 2010.

19 Oded Goldreich. Open problems in property testing of graphs. In *Electron. Colloquium Comput. Complex.*, volume 28, page 88, 2021.

20 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

21 Paul W. Holland and Samuel Leinhardt. A method for detecting structure in sociometric data. *American Journal of Sociology*, 76:492–513, 1970.

22 Kazuo Iwama and Yuichi Yoshida. Parameterized testability. *ACM Trans. Comput. Theory*, 9(4):16:1–16:16, 2018. doi:10.1145/3155294.

23 Matthew O. Jackson, Tomas Rodriguez-Barraquer, and Xu Tan. Social capital and social quilts: Network patterns of favor exchange. *American Economic Review*, 102(5):1857–1897, 2012.

24 Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004. doi:10.1137/S0097539703436424.

25 Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors III: poly($d\varepsilon^{-1}$)–time partition oracles for minor-free graph classes. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 257–268, 2021. doi:10.1109/FOCS52979.2021.00034.

26 Reut Levi. Testing triangle freeness in the general model in graphs with arboricity $o(\sqrt{n})$. In *48th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 198, pages 93:1–93:13, 2021.

27 Reut Levi and Nadav Shoshan. Testing Hamiltonicity (and other problems) in minor-free graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 61:1–61:23, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.61.

28 Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

29 Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013. doi:10.1137/120890946.

30 Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Struct. Algorithms*, 20(2):165–183, 2002. doi:10.1002/rsa.10013.

31 Alejandro Portes. Social capital: Its origins and applications in modern sociology. In Eric L. Lesser, editor, *Knowledge and Social Capital*, pages 43–67. Butterworth-Heinemann, 2000. doi:10.1016/B978-0-7506-7222-1.50006-4.

32 C. Seshadhri, Tamara G. Kolda, and Ali Pinar. Community structure and scale-free collections of Erdös-Rényi graphs. *Physical Review E*, 85(5):056109, May 2012. doi:10.1103/PhysRevE.85.056109.