ALEA, Lat. Am. J. Probab. Math. Stat. **17**, 775–790 (2020)
DOI: 10.30757/ALEA.v17-30

Finding a planted clique by adaptive probing

Miklós Z. Rácz and Benjamin Schiffer

Princeton University
204 Sherrerd Hall,
Princeton, NJ 08544, USA.
E-mail address: mracz@princeton.edu, bgs3@princeton.edu
URL: http://mracz.princeton.edu/

Abstract. We consider a variant of the planted clique problem where we are allowed unbounded computational time but can only investigate a small part of the graph by adaptive edge queries. We determine (up to logarithmic factors) the number of queries necessary both for detecting the presence of a planted clique and for finding the planted clique.

Specifically, let $G \sim G(n, 1/2, k)$ be a random graph on n vertices with a planted clique of size k. We show that no algorithm that makes at most $q = o(n^2/k^2 + n)$ adaptive queries to the adjacency matrix of G is likely to find the planted clique. On the other hand, when $k \geq (2 + \varepsilon)\log_2 n$ there exists a simple algorithm (with unbounded computational power) that finds the planted clique with high probability by making $q = O((n^2/k^2)\log^2 n + n\log n)$ adaptive queries. For detection, the additive n term is not necessary: the number of queries needed to detect the presence of a planted clique is n^2/k^2 (up to logarithmic factors).

1. Introduction

In the planted clique problem the goal is to find a clique that is planted within an Erdős-Rényi random graph. This problem has received widespread attention in the past few decades because there exists a (wide) range of clique sizes for which it is information-theoretically possible to find the planted clique but there are no known polynomial-time algorithms to do so (Jerrum, 1992; Kučera, 1995; Alon et al., 1998; Feige and Ron, 2010; Dekel et al., 2014; Deshpande and Montanari, 2015). In this regime it is conjectured to be computationally hard to find the planted clique and this conjecture forms the basis of numerous average-case complexity results in recent years (Berthet and Rigollet, 2013b,a; Gao et al., 2017; Brennan et al., 2018; Brennan and Bresler, 2019).

Received by the editors November 16th, 2019; accepted August 1st, 2020.
2010 *Mathematics Subject Classification.* 05C80, 60C05.
Key words and phrases. Planted clique, random graph, adaptive query model.
Research supported in part by NSF grant DMS 1811724.

In this paper we consider a variant of the planted clique problem where we are allowed unbounded computational time but can only investigate a small part of the graph by adaptive edge queries. We consider the problems of detection and estimation under this model, and determine (up to logarithmic factors) the number of queries necessary both for detecting the presence of a planted clique and for finding the planted clique.

In the problems we consider there is an underlying n vertex graph G with vertex set $[n] := \{1, 2, \ldots, n\}$. The algorithms that we consider are allowed unbounded computational power but we restrict the number of edges they are allowed to inspect. Specifically, we consider algorithms that evolve dynamically over a certain number of steps. In the first step, the algorithm chooses a pair (i_1, j_1), $1 \leq i_1 < j_1 \leq n$, and asks whether this pair is an edge or not. Depending on the outcome, the algorithm selects a second pair (i_2, j_2), $1 \leq i_2 < j_2 \leq n$, and asks whether this pair is an edge or not. It then selects (i_3, j_3), and so on. The algorithm may ask q such edge queries and use unbounded computational time to produce an output.

The detection problem can be phrased as a simple hypothesis testing problem. Under the null hypothesis H_0, the graph G is an Erdős-Rényi random graph with edge density 1/2. Under the alternative hypothesis H_1, the graph G is drawn from the planted clique model with clique size k. That is, we first choose a (uniformly) random subset of the vertices $K \subseteq [n]$ of size $|K| = k$, we connect all pairs of vertices in K—that is, the vertices in K form a clique—and every other pair of vertices is connected independently with probability 1/2. In short:

$$H_0 : G \sim G(n, 1/2), \qquad\qquad H_1 : G \sim G(n, 1/2, k). \tag{1.1}$$

We denote the two probability distributions over n vertex graphs by $\mathbb{P}_0$ and $\mathbb{P}_1$, respectively. An algorithm A for detection under the adaptive edge query model makes up to q adaptive edge queries to G and then outputs a hypothesis in $\{0, 1\}$. We measure the performance of an algorithm A by its risk, which is defined as the sum of its type I and type II errors:

$$R(A) := \mathbb{P}_0 \left(A(G) = 1 \right) + \mathbb{P}_1 \left(A(G) = 0 \right).$$

If an algorithm A achieves vanishing risk—$R(A) = o(1)$ as $n \to \infty$—then we say that A can detect the presence of a planted clique; otherwise, we say that it cannot do so.

The following theorem determines (up to logarithmic factors) the number of queries necessary to detect the presence of a planted clique. All logarithms in this paper are in base 2.

Theorem 1.1 (Detecting a planted clique)**.** *Consider the hypothesis testing problem in* (1.1).

(a) Let $q = o(n^2/k^2)$ as $n \to \infty$. If an algorithm A makes at most q adaptive edge queries then its risk must satisfy $R(A) \geq 1 - o(1)$ as $n \to \infty$.

(b) Suppose that $k \geq (2 + \varepsilon) \log n$ for some constant $\varepsilon > 0$ and let $\varepsilon_0 > 0$ be arbitrary. There exists an algorithm (with unlimited computational power) that can detect the presence of a planted clique by querying

$$q = (2 + \varepsilon_0) \frac{n^2}{k^2} \log^2 n$$

pairs of vertices. Moreover, the queries can be nonadaptive.

If we can detect the presence of a planted clique, the natural next goal is to find it. The following theorem determines (up to logarithmic factors) the number of queries necessary to find the planted clique. In particular, it shows that an extra $n \log n$ queries suffice compared to detection and that this is tight (up to logarithmic factors).

Theorem 1.2 (Finding the planted clique)**.** *Let $G \sim G(n, 1/2, k)$, where $1 \leq k < n$.*

(a) Let $q = o(n^2/k^2 + n)$ as $n \to \infty$. No algorithm that makes at most q adaptive edge queries can find the planted clique. That is, any estimator $\widehat{K}$ of the planted clique K that is based on at most q adaptive edge queries satisfies $\mathbb{P}_1\left(\widehat{K} = K\right) = o(1)$ as $n \to \infty$.

(b) Suppose that $k \geq (2+\varepsilon)\log n$ for some constant $\varepsilon > 0$ and let $\varepsilon_0 > 0$ be arbitrary. There exists an algorithm (with unlimited computational power) that adaptively queries

$$q = (2+\varepsilon_0)\frac{n^2}{k^2}\log^2 n + (1+\varepsilon_0) n \log n$$

pairs of vertices and finds the planted clique with probability $1 - o(1)$ as $n \to \infty$.

Theorems 1.1 and 1.2 give a complete phase diagram of when detection and estimation are possible as a function of the clique size k and the number of queries q (up to some boundary cases). A natural parametrization is to take both k and q to be polynomial in n: $k = n^\gamma$ and $q = n^\delta$ for some $\gamma \in (0,1)$ and $\delta \in (0,2)$. Corollary 1.3 summarizes the results with this parametrization—see also Figure 1 for an illustration. Note in particular the region of the phase space where detection is possible but estimation is not. Note also that the conjectured computational threshold is at $\gamma = 1/2$.

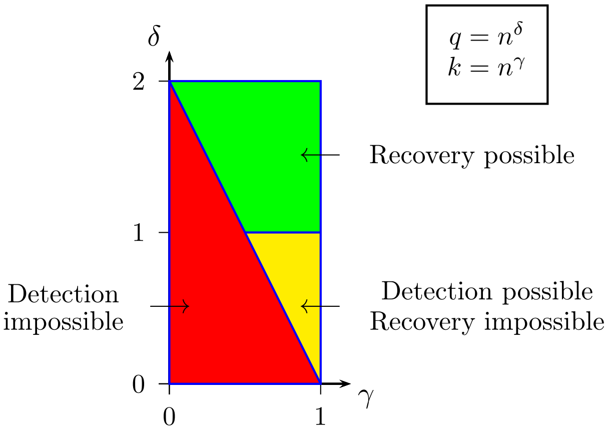

Figure 1.1. Phase diagram for detecting the presence of a planted clique and for finding the planted clique, as a function of the clique size $k = n^\gamma$ and the number of adaptive edge queries $q = n^\delta$. The horizontal axis contains $\gamma \in (0,1)$, the vertical axis contains $\delta \in (0,2)$.

Corollary 1.3 (Phase diagram)**.** *Suppose that $k = n^\gamma$ and $q = n^\delta$ for some $\gamma \in (0,1)$ and $\delta \in (0,2)$.*

(a) If $\delta < 2 - 2\gamma$, then detecting the presence of a planted clique is impossible.
(b) If $\delta > 2 - 2\gamma$ and $\delta < 1$, then it is possible to detect the presence of a planted clique, but it is impossible to find the planted clique.
(c) If $\delta > 2 - 2\gamma$ and $\delta > 1$, then it is possible to find the planted clique.

These results raise several open questions. First, can the logarithmic factors be closed, to obtain results that are tight up to constant factors? Second, how do these results change if we plant a different subgraph instead of a clique? For instance, one can plant a dense random graph $G(k, q)$ with $q > 1/2$. Finally, while we neglected all computational considerations in this paper, are there any connections to average-case computational hardness? We leave exploring these questions to future work.

The rest of this paper is outlined as follows. After discussing motivation and related work in the remainder of the introduction, we turn to algorithms for detection and estimation in Section 2, proving Theorem 1.1(b) and Theorem 1.2(b). Finally, we prove the impossibility results of Theorem 1.1(a) and Theorem 1.2(a) in Section 3.

1.1. *Motivation.* There are several potential applications where understanding the query complexity necessary to finding cliques may be of interest. For instance, in scientific applications one may wish to find closely related entities (corresponding to a clique or dense subgraph), and querying an edge may correspond to performing a physical experiment which is costly and/or time-consuming.

Another potential application is to the analysis of social media connections. Here the nodes of a graph represent individuals and the edges represent connections between individuals such as Facebook friends, Twitter following, or LinkedIn connections. Access to these connections may be expensive to obtain or limited (due to privacy limitations or any other source of incomplete information), and hence query complexity may be relevant when trying to reconstruct a specific close-knit group within the network.

The planted clique problem and related subgraph inference problems have been applied to a variety of applications, including biological networks (Milo et al., 2002), cryptography (Juels and Peinado, 2000), and finance (Arora et al., 2011). Obtaining full information about the underlying networks in these applications may not be possible due to queries being expensive and/or limited, and hence the planted clique problem with limited adaptive probing could be relevant to these same applications.

1.2. *Related work.* This paper is a natural follow-up to the recent work of Feige et al. (2020), where the authors consider the problem of finding cliques in an Erdős-Rényi random graph under the same adaptive edge query model. While the largest clique in an Erdős-Rényi random graph with edge density 1/2 has size approximately $2 \log n$, the current best algorithm that makes at most $q = O(n^{\delta})$ adaptive edge queries finds a clique of size approximately $(1 + \delta/2) \log n$. Feige et al. (2020) show an impossibility result if the adaptivity of the algorithm is limited: any algorithm that makes $q = O(n^{\delta})$ edge queries ($\delta < 2$) in ℓ rounds finds cliques of size at most $(2 - \varepsilon) \log n$ where $\varepsilon = \varepsilon(\delta, \ell) > 0$. Very recently, Alweiss et al. (2019) improved upon this result, showing that there exists such ε that depends only on δ and not on ℓ. However, closing the gap between the upper and lower bounds remains an open problem.

Several recent works consider finding structure in a random graph under such an adaptive edge query model. Ferber, Krivelevich, Sudakov, and Vieira studied finding a Hamilton cycle (Ferber et al., 2016) and finding long paths (Ferber et al., 2017), while Conlon et al. (2019) studied finding a copy of a fixed target graph (such as a constant size clique). All of these works focus on sparse random graphs.

As mentioned in the introduction, the planted clique problem has been studied from many angles in the past few decades (Jerrum, 1992; Kučera, 1995; Alon et al., 1998; Feige and Ron, 2010; Dekel et al., 2014; Deshpande and Montanari, 2015; Berthet and Rigollet, 2013b,a; Gao et al., 2017; Brennan et al., 2018; Brennan and Bresler, 2019). To the best of our knowledge, it has not been considered under an edge query model before. It would be interesting to see if there are any connections to computational aspects of the planted clique problem. The recent work of Mardia et al. (2020) develops sublinear[1] time algorithms for finding the planted clique in the regime $k = \omega\left(\sqrt{n \log\log n}\right)$ and makes such connections. As the authors point out, our results imply an $\Omega(n)$ running time lower bound for finding the planted clique, which shows that their algorithms are optimal (up to logarithmic factors) whenever $k = \Omega\left(n^{2/3}\right)$.

Finally, we mention that query complexity arises naturally in many other areas, such as clustering (Vinayak and Hassibi, 2016; Mazumdar and Saha, 2017a,b), where answers to queries are often noisy due to them being crowdsourced, and community detection (Hartmann et al., 2016; Anagnostopoulos et al., 2016), where the evolution of the underlying graph necessitates repeated queries. Statistical queries have also been widely studied (Kearns, 1998), including in the setting of the planted clique model (Feldman et al., 2017). More generally, our work fits into the framework of online learning, a large and rapidly growing area which is beyond the scope of this article to survey.

2. Algorithms

We start with a simple sampling-based algorithm to detect the presence of a planted clique. This is contained in Section 2.1 and proves Theorem 1.1(b). We then extend this algorithm in Section 2.2 to find all vertices of the planted clique, thus proving Theorem 1.2(b).

First, recall that the largest clique in an Erdős-Rényi random graph has size approximately $2\log n$. In fact, very precise results are known. Define $\omega_n = 2\log n - 2\log\log n + 2\log e - 1$. Matula (1972) showed that for any $\varepsilon > 0$, the clique number (the size of the largest clique) $\omega(G)$ of a random graph G drawn from $G(n, 1/2)$ satisfies $\lfloor \omega_n - \varepsilon \rfloor \leq \omega(G) \leq \lfloor \omega_n + \varepsilon \rfloor$ with probability tending to 1 as $n \to \infty$; see also Bollobás and Erdős (1976). For our purposes much weaker estimates suffice. Indeed, a first moment argument shows that $\mathbb{P}_0\left(\omega(G) \leq 2\log n + 3\right) \to 1$ as $n \to \infty$ (see Lugosi, 2017).

2.1. *Detecting the presence of a planted clique.* The basic idea in detecting the presence of a planted clique is to sample all pairs of vertices among a set $S \subseteq [n]$ of size $m := (2 + \varepsilon')\,(n/k)\log n$. After these queries we learn the induced subgraph on S and we can use the size of the largest clique in S as a statistic to distinguish between the hypotheses H_0 and H_1, as follows. We know (see above) that under

[1] Here the input size is $\Theta(n^2)$, hence sublinear refers to $o(n^2)$.

$\mathbb{P}_0$ this statistic is at most $2\log n + 3$ with probability $1 - o(1)$. On the other hand, under $\mathbb{P}_1$ the set S contains, in expectation, $(2+\varepsilon')\log n$ vertices from the planted clique, so with probability $1-o(1)$ this statistic is at least $(2+\varepsilon'/2)\log n$.

The following proof makes this reasoning precise. Note that this algorithm to detect the presence of a planted clique is nonadaptive, making all queries at the same time.

Proof of Theorem 1.1(b)*:* Let $\varepsilon' > 0$ be such that $2\varepsilon' + \varepsilon'^2/2 \le \varepsilon_0$ and $\varepsilon' \le \varepsilon$. First, we choose an arbitrary subset S of the n vertices of size $m := |S| = (2+\varepsilon')\,(n/k)\log n$; for instance, choose $S := \{1, 2, \ldots, m\}$. We then query all pairs of vertices among S. This results in

$$\binom{m}{2} \le \frac{m^2}{2} = (2 + 2\varepsilon' + \varepsilon'^2/2)\frac{n^2}{k^2}\log^2 n \le (2+\varepsilon_0)\frac{n^2}{k^2}\log^2(n)$$

queries. After the queries we know the induced subgraph on S. In particular—due to the fact that we have no restrictions on computational power—we can compute the size of the largest clique in this induced subgraph. The algorithm then chooses a hypothesis based on this statistic: if S contains a clique of size at least $(2+\varepsilon'/2)\log n$, then it accepts the alternative hypothesis H_1; otherwise, it accepts the null hypothesis H_0.

We now argue that this algorithm achieves vanishing risk. First, as we discussed at the beginning of Section 2, if $G \sim G(n, 1/2)$, then the largest clique in G has size approximately $2\log n$. In particular, $\mathbb{P}_0\left(\omega(G) \ge (2+\varepsilon'/2)\log n\right) \to 0$ as $n \to \infty$; that is, the type I error vanishes in the limit.

Next, assuming H_1, let X denote the number of planted clique vertices in S. Observe that X has a hypergeometric distribution with parameters n, k, and m. Thus we have that $\mathbb{E}[X] = m \times (k/n) = (2+\varepsilon')\log(n)$ and $\mathrm{Var}(X) \le m\frac{k}{n}(1-\frac{k}{n}) \le (2+\varepsilon')\log(n)$, so Chebyshev's inequality implies that $\mathbb{P}_1\left(X \le (2+\varepsilon'/2)\log(n)\right) \le c/\log(n)$ for some constant $c < \infty$ depending on ε'. To conclude, note that if $X \ge (2+\varepsilon'/2)\log n$ then S contains a clique of size at least $(2+\varepsilon'/2)\log n$. □

2.2. *Finding the planted clique.* In order to find the planted clique, we start with the same step as in Section 2.1 above: we sample all pairs of vertices among a set $S \subseteq [n]$ of size $m := (2+\varepsilon')\,(n/k)\log n$. As we show below, with probability $1-o(1)$ under $\mathbb{P}_1$, the set of vertices in the largest clique in S is exactly the set of vertices in S that are in the planted clique. Thus the remaining goal is to identify the vertices of the planted clique that are not in S.

To do this, a natural idea is to query all pairs of vertices where one vertex is part of the largest clique in S and the other vertex is not in S. Any vertex that is in the planted clique and not in S will necessarily be connected to all planted clique vertices in S, while vertices not in S and not in the planted clique will not be connected to all of the planted clique vertices in S (with probability $1-o(1)$ under $\mathbb{P}_1$). Thus in the second step the algorithm selects all vertices not in S that were connected to all vertices in S where the pair was queried.

Finally, the algorithm outputs the union of the two sets of vertices identified in the two steps. The following proof makes all this precise and proves Theorem 1.2(b). Note that in the second step we take only a subset of the largest clique in S—this is done in order to lessen the number of queries made. Note furthermore that this algorithm has limited adaptivity, as it can be implemented in two "rounds".

Proof of Theorem 1.2(b): The algorithm for finding the planted clique consists of two steps, the first step being the same as the one used for detection.

- **Step 1:** Choose a subset S of the n vertices of size
$$m := |S| = (2+\varepsilon')\,(n/k)\log n,$$
where ε' is chosen as in Section 2.1. We then query all pairs of vertices among S.
- **Step 2:** Let D be the set of vertices in the largest clique in S. Let $D' \subseteq D$ be a fixed subset of size $(1+\varepsilon_0)\log n$ (e.g., take D' to be the $(1+\varepsilon_0)\log n$ nodes in D with lowest label). (If ε_0 is large such that $|D| \leq (1+\varepsilon_0)\log n$, then let $D' := D$.) We then query all pairs of vertices where one of the vertices is in D' and the other is in $[n] \setminus S$.

  Let T denote the vertices in $[n] \setminus S$ that are connected to all vertices in D'. The algorithm then outputs $D \cup T$ as its estimate for the planted clique.

We have seen in Section 2.1 that we make at most $(2+\varepsilon_0)(n^2/k^2)\log^2 n$ queries in the first step, while in the second step we make at most $(1+\varepsilon_0)n\log n$ queries. We now argue that this algorithm succeeds in finding the planted clique with probability $1-o(1)$.

As we argued in Section 2.1, we have that $\mathbb{P}_1\left(|D| \geq (2+\varepsilon'/2)\log n\right) = 1 - o(1)$ as $n \to \infty$. Furthermore, with probability $1-o(1)$, the set D contains only planted clique vertices. Indeed, as we discussed at the beginning of Section 2, with probability $1-o(1)$ the largest clique in an Erdős-Rényi random graph with edge density $1/2$ has size at most $2\log n + 3$, so no vertex outside of the planted clique is in a clique of size greater than $2\log n + 3$. Thus in the first step of the algorithm we have found at least $(2+\varepsilon'/2)\log n$ vertices of the planted clique. Moreover, we have found all vertices of the planted clique that are in S.

Any vertex in $[n] \setminus S$ that is in the planted clique will be connected to every planted clique vertex and hence every vertex in D. Thus all vertices of the planted clique are contained in $D \cup T$. To see that there are no false positives in this set, note that the probability that a vertex not in the planted clique is connected to a fixed set of $(1+\varepsilon_0)\log n$ planted clique vertices is $2^{-(1+\varepsilon_0)\log n} = n^{-(1+\varepsilon_0)}$. Taking a union bound over vertices in $[n] \setminus S$, we see that the probability that there exists a vertex not in the planted clique that is in T is at most $n^{-\varepsilon_0}$. □

Note that this algorithm succeeds in finding the planted clique even though it does not check that all pairs of vertices within the planted clique are connected. In fact, it checks the edge between $O\left(k\log n + \log^2 n\right)$ pairs of vertices within the planted clique, instead of the $\Theta\left(k^2\right)$ pairs that exist.

3. Lower bounds

To prove our lower bounds we introduce a simpler variant problem that removes all graph structure from the problem. In this hypothesis testing problem we consider the set $[n] = \{1, 2, \ldots, n\}$, where each element of the set is either marked or unmarked. Under the null hypothesis $\widetilde{H}_0$, all elements are unmarked. Under the alternative hypothesis $\widetilde{H}_1$, a uniformly randomly chosen subset $K \subseteq [n]$ of size

$|K| = k$ is chosen and its elements are marked, and the elements of $[n] \setminus K$ are unmarked. We denote the two probability distributions over $\{\text{unmarked, marked}\}^{[n]}$ by $\widetilde{\mathbb{P}}_0$ and $\widetilde{\mathbb{P}}_1$, respectively.

We consider algorithms that can adaptively query (i, j) pairs, where $1 \leq i < j \leq n$. We refer to such queries as *pair queries* to distinguish them from the *edge queries* of the original problem. When pair (i, j) is queried, the query evaluates to true if both i and j are marked and it evaluates to false otherwise. The algorithm may ask q such adaptive pair queries and use unbounded computational time to produce an output in $\{0, 1\}$ (corresponding to $\widetilde{H}_0$ or $\widetilde{H}_1$). We again measure the performance of an algorithm $\widetilde{A}$ by its risk, defined as

$$\widetilde{R}(\widetilde{A}) := \widetilde{\mathbb{P}}_0 \left(\widetilde{A} = 1 \right) + \widetilde{\mathbb{P}}_1 \left(\widetilde{A} = 0 \right).$$

We consider randomized algorithms as well, in which case the type I and type II error probabilities in the display above are taken over the internal randomness of the algorithm as well.

The following lemma connects this variant problem with the original hypothesis testing problem.

Lemma 3.1 (Reduction)**.** *Suppose that there exists an algorithm A that makes at most q adaptive edge queries and achieves risk $R(A) \leq r$ for the hypothesis testing problem in* (1.1)*. Then there exists an algorithm $\widetilde{A}$ that makes at most q adaptive pair queries in the variant problem described above—distinguishing between $\widetilde{H}_0$ and $\widetilde{H}_1$—and achieves risk $\widetilde{R}(\widetilde{A}) \leq r$.*

Proof: There is a direct correspondence between the two hypothesis testing problems, which allows the answers to pair queries to simulate answers to edge queries. Specifically, marked elements of $[n]$ correspond to planted clique vertices. Thus a pair query that evaluates to true corresponds to querying two planted clique vertices, while a pair query that evaluates to false corresponds to querying two vertices between which the edge is random. Thus given the answer to a pair query, the answer to an edge query can be simulated as follows: if the answer to the pair query is true, the answer to the corresponding edge query is that the edge exists, while if the answer to the pair query is false, then flip a fair coin to answer the corresponding edge query.

Thus for any algorithm A that makes at most q adaptive edge queries, there exists a corresponding algorithm $\widetilde{A}$ that makes at most q adaptive pair queries in the variant problem and simulates A. We then let the output of $\widetilde{A}$ be the same as the output of the simulated algorithm A. Since the simulation of A involves extra randomness, $\widetilde{A}$ is thus a randomized algorithm. By conditioning on the extra randomness, it follows that the risk of $\widetilde{A}$ is the same as the risk of A. □

This lemma implies that to prove Theorem 1.1(a) it suffices to prove the analogous result for the variant problem. Consequently, we turn our focus to the variant problem. Observe that under $\widetilde{H}_0$ all answers to all pair queries will be false. The next lemma considers the alternative hypothesis $\widetilde{H}_1$.

Lemma 3.2. *Let $q \leq \frac{n(n-1)}{k(k-1)} - 1$. Let $\widetilde{A}$ be any algorithm that makes at most q adaptive pair queries. Let $\mathcal{E}_q$ denote the event that all of the pair queries of $\widetilde{A}$*

evaluate to false. We then have that

$$\widetilde{\mathbb{P}}_1\left(\mathcal{E}_q\right) \geq 1 - q\frac{k(k-1)}{n(n-1)}. \tag{3.1}$$

In particular, if $q = o(n^2/k^2)$ as $n \to \infty$, then $\widetilde{\mathbb{P}}_1\left(\mathcal{E}_q\right) = 1 - o(1)$ as $n \to \infty$.

Proof: To highlight the key elements of the proof, we first prove the claim for deterministic algorithms, where each query is a deterministic function of the previous queries and the answers to them; at the end of the proof we address how the proof changes for randomized algorithms, which may use additional randomness. Thus for now assume that the algorithm $\widetilde{A}$ is deterministic. To describe the structure of deterministic algorithms we introduce some notation. We let $e_1, e_2, \ldots$ denote the pair queries made by the algorithm. Furthermore, let $X_1, X_2, \ldots$ denote the answers to the pair queries, as follows: $X_\ell = 0$ if the ℓth pair query e_ℓ evaluates to false, and $X_\ell = 1$ if the ℓth pair query e_ℓ evaluates to true. Any deterministic algorithm $\widetilde{A}$ can thus be described as follows:

- First, $\widetilde{A}$ makes the pair query $e_1 = (i_1, j_1)$. The algorithm receives the answer X_1 (which depends on the realization of K).
- The next pair query of $\widetilde{A}$ depends on the answer X_1:
  - if $X_1 = 0$, then $\widetilde{A}$ makes the pair query $e_2 = \left(i_2^{(0)}, j_2^{(0)}\right)$;
  - if $X_1 = 1$, then $\widetilde{A}$ makes the pair query $e_2 = \left(i_2^{(1)}, j_2^{(1)}\right)$.

  The algorithm receives the answer X_2 (which again depends on the realization of K).
- The third pair query of $\widetilde{A}$ depends on the answers X_1 and X_2:
  - if $X_1 = 0$ and $X_2 = 0$, then $\widetilde{A}$ makes the pair query $e_3 = \left(i_3^{(0,0)}, j_3^{(0,0)}\right)$;
  - if $X_1 = 0$ and $X_2 = 1$, then $\widetilde{A}$ makes the pair query $e_3 = \left(i_3^{(0,1)}, j_3^{(0,1)}\right)$;
  - if $X_1 = 1$ and $X_2 = 0$, then $\widetilde{A}$ makes the pair query $e_3 = \left(i_3^{(1,0)}, j_3^{(1,0)}\right)$;
  - if $X_1 = 1$ and $X_2 = 1$, then $\widetilde{A}$ makes the pair query $e_3 = \left(i_3^{(1,1)}, j_3^{(1,1)}\right)$.

  The algorithm receives the answer X_3.
- And so on. The pair query $e_{\ell+1}$ of $\widetilde{A}$ depends on the answers $X_1, X_2, \ldots, X_\ell$ as follows: for every $x \in \{0,1\}^\ell$, if $(X_1, X_2, \ldots, X_\ell) = x$, then $e_{\ell+1} = \left(i_{\ell+1}^x, j_{\ell+1}^x\right)$.

Thus the set of pairs

$$\{(i_1, j_1)\} \cup \left\{(i_\ell^x, j_\ell^x) : \ell \geq 2, x \in \{0,1\}^{\ell-1}\right\} \tag{3.2}$$

completely determines how the algorithm $\widetilde{A}$ behaves for any realization of K; and vice versa: any set of pairs as in (3.2) determines a deterministic pair query algorithm $\widetilde{A}$. (Note that in the description of a deterministic algorithm $\widetilde{A}$ we have only described how the algorithm makes the pair queries and not how the algorithm produces an output after making q adaptive pair queries—for the purposes of the claim this is all that we care about.)

In the following we thus fix the deterministic algorithm $\widetilde{A}$ by fixing the set of pairs in (3.2). Also, for notational convenience, we write (i'_ℓ, j'_ℓ) for (i_ℓ^x, j_ℓ^x) when $\ell \geq 2$ and $x = (0, 0, \ldots, 0) \in \{0,1\}^{\ell-1}$; furthermore, let $i'_1 \equiv i_1$ and $j'_1 \equiv j_1$. Recall that $\mathcal{E}_q$ denotes the event that the first q adaptive pair queries of $\widetilde{A}$ all evaluate to false. We prove (3.1) by determining the conditional law of K given the event $\mathcal{E}_q$. Note that we fixed the set of pairs in (3.2), and thus we know that, given $\mathcal{E}_q$, the first q pair queries were $e_1 = (i'_1, j'_1), \ldots, e_q = \left(i'_q, j'_q\right)$. Let $\mathcal{S}_q$ denote the set of k-tuples such that there was a pair query among these first q pair queries that queried a pair from this k-tuple; that is,

$$\mathcal{S}_q := \left\{ S \subseteq [n] : |S| = k, \exists\, \ell \in [q] : i'_\ell, j'_\ell \in S \right\}.$$

Since all pair queries evaluated to false, no k-tuple in $\mathcal{S}_q$ can be the marked subset given $\mathcal{E}_q$ (since otherwise a pair query would have evaluated to true). That is,

$$\widetilde{\mathbb{P}}_1 \left(K = S \,|\, \mathcal{E}_q \right) = 0$$

for all $S \in \mathcal{S}_q$. Now let us consider a k-tuple that is not in $\mathcal{S}_q$. By Bayes's rule we have that

$$\widetilde{\mathbb{P}}_1 \left(K = S \,|\, \mathcal{E}_q \right) = \frac{\widetilde{\mathbb{P}}_1 \left(\mathcal{E}_q \,|\, K = S \right) \widetilde{\mathbb{P}}_1 \left(K = S \right)}{\widetilde{\mathbb{P}}_1 \left(\mathcal{E}_q \right)}.$$

Since the prior on K is uniform, we have that $\widetilde{\mathbb{P}}_1 \left(K = S \right) = 1/\binom{n}{k}$. Now since $S \notin \mathcal{S}_q$, if $K = S$, then the answers to the pair queries $e_1 = (i'_1, j'_1), \ldots, e_q = \left(i'_q, j'_q\right)$ are necessarily all false (due to the definition of $\mathcal{S}_q$). Therefore for every $S \notin \mathcal{S}_q$ we have that $\widetilde{\mathbb{P}}_1 \left(\mathcal{E}_q \,|\, K = S \right) = 1$. Thus we have shown that for every $S \notin \mathcal{S}_q$ we have that

$$\widetilde{\mathbb{P}}_1 \left(K = S \,|\, \mathcal{E}_q \right) = \frac{1}{\binom{n}{k} \widetilde{\mathbb{P}}_1 \left(\mathcal{E}_q \right)}.$$

Altogether, we have shown that the conditional law of K given $\mathcal{E}_q$ is given by

$$\widetilde{\mathbb{P}}_1 \left(K = S \,|\, \mathcal{E}_q \right) = \frac{1}{\binom{n}{k} \widetilde{\mathbb{P}}_1 \left(\mathcal{E}_q \right)} \mathbf{1}_{\{S \notin \mathcal{S}_q\}}.$$

Notice that this conditional probability is equal for all k-tuples that are not in $\mathcal{S}_q$. Therefore we also have that

$$\widetilde{\mathbb{P}}_1 \left(K = S \,|\, \mathcal{E}_q \right) = \frac{1}{\binom{n}{k} - |\mathcal{S}_q|} \mathbf{1}_{\{S \notin \mathcal{S}_q\}}.$$

Putting together the previous two displays we have that

$$\widetilde{\mathbb{P}}_1 \left(\mathcal{E}_q \right) = 1 - \frac{|\mathcal{S}_q|}{\binom{n}{k}}. \tag{3.3}$$

Note that every pair (i, j) (where $1 \leq i < j \leq n$) is part of exactly $\binom{n-2}{k-2}$ different subsets of k elements. This implies the following upper bound on the size of $\mathcal{S}_q$:

$$|\mathcal{S}_q| \leq q \binom{n-2}{k-2} = q \frac{k(k-1)}{n(n-1)} \binom{n}{k}.$$

Plugging this bound back into (3.3) we obtain (3.1), as desired.

Finally, we discuss randomized algorithms, which may use additional randomness. We may condition on the extra randomness and then use the argument above for deterministic algorithms. This shows that no matter what the realization of the additional randomness is, the conditional probability of all q pair queries evaluating to false is at least $1 - q\frac{k(k-1)}{n(n-1)}$. Taking an expectation over the additional randomness then shows the desired claim. □

We are now ready to prove the analogue of Theorem 1.1(a) for the variant problem.

Corollary 3.3 (Detecting a marked set of elements)**.** *Consider the hypothesis testing problem $\widetilde{H}_0$ versus $\widetilde{H}_1$. Let $q = o(n^2/k^2)$ as $n \to \infty$. If an algorithm $\widetilde{A}$ makes at most q adaptive pair queries then its risk must satisfy $\widetilde{R}(\widetilde{A}) \geq 1 - o(1)$ as $n \to \infty$.*

Proof: No matter what algorithm $\widetilde{A}$ does, all of its pair queries will evaluate to false under $\widetilde{H}_0$ (by definition), and all of its pair queries will evaluate to false under $\widetilde{H}_1$ with probability $1 - o(1)$ (by Lemma 3.2). Suppose that $\widetilde{A}$ outputs 0 with probability α and 1 with probability $1 - \alpha$ when all of its queries evaluate to false, where $\alpha \in [0, 1]$. The first sentence of the proof then implies that its risk is at least $\widetilde{R}(\widetilde{A}) \geq (1 - \alpha) + \alpha(1 - o(1)) = 1 - o(1)$. □

*Proof of Theorem 1.1(*a*):* This follows directly from Lemma 3.1 and Corollary 3.3. □

We now turn to proving Theorem 1.2(a). Here too we leverage the connection to the corresponding estimation problem for the simplified variant problem, where we aim to estimate the set of marked elements.

Lemma 3.4. *Let $K \subseteq [n]$ be a uniformly randomly chosen set of size $|K| = k$, where $1 \leq k < n$. Let the elements of K be marked and let the elements of $[n] \setminus K$ be unmarked. Let $q = o(n^2/k^2 + n)$ as $n \to \infty$. If $\widehat{K}$ is any estimator of the marked set K that is based on at most q adaptive pair queries, then $\widehat{K}$ satisfies $\widetilde{\mathbb{P}}_1\left(\widehat{K} = K\right) = o(1)$ as $n \to \infty$.*

Proof: There are two cases to consider. First, consider the case when $q = o(n^2/k^2)$. The proof of Lemma 3.2 shows that, with probability $1 - o(1)$, after q adaptive pair queries there remain a $(1 - o(1))$ fraction of subsets of size k that are equally likely to be the marked set. No estimator can do better than pick randomly among these, and this will succeed with probability $(1 + o(1)) / \binom{n}{k} = o(1)$.

Next, consider the case when $q = o(n)$. In this case we show that it is not possible to estimate the marked set even for algorithms with significantly more information. Specifically, we consider algorithms that can adaptively query (i, j) pairs, where $1 \leq i < j \leq n$, and when pair (i, j) is queried, the algorithm learns, for both i and j, whether they are marked or unmarked. We refer to such queries as *strong pair queries* to distinguish them from *pair queries*. From the answer to a strong pair query it is possible to determine the answer to the appropriate pair query. Therefore any algorithm that makes q adaptive pair queries can be simulated by an algorithm that makes q adaptive strong pair queries. Thus in order to prove the claim it suffices to show that if $\widehat{K}$ is any estimator of the marked set K that is based on at most q adaptive strong pair queries, then $\widehat{K}$ satistifies $\widetilde{\mathbb{P}}_1\left(\widehat{K} = K\right) = o(1)$ as $n \to \infty$. This is what we will show; thus in the following we consider algorithms that make at most q adaptive strong pair queries, and we assume that $q = o(n)$.

We now argue that for $q < n/2$ we may, without loss of generality, consider the algorithm that makes the strong pair queries $(1, 2), (3, 4), \ldots, (2q - 1, 2q)$. We argue this by induction. Since the marked set K is chosen uniformly at random, the first strong pair query may be $(1, 2)$ without loss of generality. There are now three cases to consider, depending on the answer to this first strong pair query.

- **Both elements are unmarked.** Suppose that the answer to the strong pair query $(1, 2)$ is that both 1 and 2 are unmarked. Thus $1, 2 \notin K$ and therefore neither 1 nor 2 will be in any estimator $\widehat{K}$ (since otherwise the estimator will be incorrect). The algorithm thus knows that $K \subseteq [n] \setminus \{1, 2\}$. Moreover, by Bayes's rule, the conditional distribution of K, given that both 1 and 2 are unmarked, is uniform among k-tuples in $[n] \setminus \{1, 2\}$.
- **One element is unmarked, the other is marked.** Suppose that the answer to the strong pair query $(1, 2)$ is that 1 is marked and 2 is unmarked. Thus $1 \in K$ and therefore 1 will be in any estimator $\widehat{K}$ (since otherwise the estimator will be incorrect). We also learn that $2 \notin K$, so 2 will not be in any estimator $\widehat{K}$ (since otherwise the estimator will be incorrect). Moreover, by Bayes's rule, the conditional distribution of K, given that 1 is marked and 2 is unmarked, is uniform among k-tuples in $[n]$ that contain the element 1 and do not contain the element 2. Thus the conditional distribution of $K \setminus \{1, 2\}$, given that 1 is marked and 2 is unmarked, is uniform among $(k - 1)$-tuples in $[n] \setminus \{1, 2\}$.

  The case where 1 is unmarked and 2 is marked is analogous.
- **Both elements are marked.** Suppose that the answer to the strong pair query $(1, 2)$ is that both 1 and 2 are marked. Thus $1, 2 \in K$ and therefore both 1 and 2 will be in any estimator $\widehat{K}$ (otherwise the estimator will be incorrect). Moreover, by Bayes's rule, the conditional distribution of K, given that both 1 and 2 are marked, is uniform among k-tuples in $[n]$ that contain both 1 and 2. Thus the conditional distribution of $K \setminus \{1, 2\}$, given that both 1 and 2 are marked, is uniform among $(k-2)$-tuples in $[n] \setminus \{1, 2\}$.

In summary, no matter what the answer to the strong pair query $(1, 2)$ is, the algorithm deduces the following two points.

- For elements 1 and 2, the algorithm knows whether or not to include them in any estimator $\widehat{K}$ that has any possibility of being correct.
- The conditional distribution of $K \setminus \{1, 2\}$, given the answer to the strong pair query $(1, 2)$, is uniform among m-tuples in $[n] \setminus \{1, 2\}$; here $m = k$ if both 1 and 2 are unmarked, $m = k - 1$ if one is unmarked and the other is marked, and $m = k - 2$ if both 1 and 2 are marked.

Due to the *uniformity* of the conditional distribution in the last bullet point, the next strong pair query may, without loss of generality, be $(3, 4)$. More generally, after having made the strong pair queries $(1, 2), (3, 4), \ldots, (2\ell-1, 2\ell)$, the algorithm deduces the following two points.

- For each element in $\{1, 2, \ldots, 2\ell\}$, the algorithm knows whether or not to include them in any estimator $\widehat{K}$ that has any possibility of being correct.
- The conditional distribution of $K \setminus \{1, 2, \ldots 2\ell\}$, given the answers to the strong pair queries $(1, 2), (3, 4), \ldots, (2\ell - 1, 2\ell)$, is uniform among m-tuples in $[n] \setminus \{1, 2, \ldots, 2\ell\}$, where m is equal to k minus the number of marked elements in $\{1, 2, \ldots, 2\ell\}$.

Again, due to the uniformity of the conditional distibution in the bullet point above, the next strong pair query may, without loss of generality, be $(2\ell + 1, 2\ell + 2)$. This finishes the proof of the induction.

Finally, we analyze the algorithm that makes the strong pair queries

$$(1, 2), (3, 4), \ldots, (2q - 1, 2q).$$

After the answers to these strong pair queries, the algorithm knows for each element in $[2q] = \{1, 2, \ldots, 2q\}$ whether they are marked or unmarked. Let S denote the subset of marked elements in $[2q]$, and let $X := |S|$. Any estimator $\widehat{K}$ that has any possibility of being correct must include S as a subset (since otherwise the estimator will be incorrect); similarly, any estimator $\widehat{K}$ that has any possibility of being correct must not include any elements in $[2q] \setminus S$. If $X = k$, then this determines that the estimator should be $\widehat{K} = S$, and indeed in this case the estimator is correct: $\widehat{K} = K$. If $X < k$, then the estimator has to choose a subset $\widehat{T} \subseteq [n] \setminus [2q]$ of size $k - X$ and outputs $\widehat{K} = S \cup \widehat{T}$. The estimator is then correct (that is, $\widehat{K} = K$ holds) if and only if $\widehat{T} = K \setminus [2q]$. As we have argued above, the conditional distribution of $K \setminus [2q]$, given the answers to the strong pair queries $(1, 2), (3, 4), \ldots, (2q-1, 2q)$, is uniform among $(k - X)$-tuples in $[n] \setminus [2q]$. Due to the uniformity of this conditional distribution, for *any* estimator $\widehat{T}$ the conditional probability of $\widehat{T} = K \setminus [2q]$ is equal to $1/\binom{n-2q}{k-X}$. Putting everything together we have thus obtained that

$$\widetilde{\mathbb{P}}_1\left(\widehat{K} = K\right) = \widetilde{\mathbb{E}}_1\left[\widetilde{\mathbb{P}}_1\left(\widehat{K} = K \,\middle|\, X\right)\right] = \widetilde{\mathbb{E}}_1\left[\widetilde{\mathbb{P}}_1\left(\widehat{T} = K \setminus [2q] \,\middle|\, X\right)\right]$$
$$= \widetilde{\mathbb{E}}_1\left[\frac{1}{\binom{n-2q}{k-X}}\right]. \tag{3.4}$$

Since the distribution of K is uniform among k-tuples in $[n]$, the distribution of X is hypergeometric with parameters n, k, and $2q$. We now distinguish three cases based on how the parameters n, k, and $2q$ relate to each other, and in each case we bound the expected value in (3.4).

- *Case 1:* $k \leq 2q$. Since $q = o(n)$, we have that $q < n/4$ for all n large enough. Thus for all n large enough we have that $k - X \leq k \leq 2q < n - 2q$. This implies that, for all n large enough, if $X < k$, then $\binom{n-2q}{k-X} \geq n - 2q \geq n/2$. Also, if $X = k$, then $\binom{n-2q}{k-X} = 1$. We thus have, for all n large enough, that
$$\widetilde{\mathbb{E}}_1\left[\frac{1}{\binom{n-2q}{k-X}}\right] \leq \frac{2}{n} + \widetilde{\mathbb{P}}_1\left(X = k\right). \tag{3.5}$$
Since X is a hypergeometric random variable with parameters n, k, and $2q$, we have that
$$\widetilde{\mathbb{P}}_1\left(X = k\right) = \frac{\binom{k}{k}\binom{n-k}{2q-k}}{\binom{n}{2q}} = \frac{(2q)(2q-1)\cdot\ldots\cdot(2q-k+1)}{n(n-1)\cdot\ldots\cdot(n-k+1)} \leq \frac{2q}{n}. \tag{3.6}$$
Combining (3.5) and (3.6) we have that $\widetilde{\mathbb{E}}_1\left[1/\binom{n-2q}{k-X}\right] \leq (2q+2)/n = o(1)$ as $n \to \infty$.
- *Case 2:* $2q < k < n - 2q$. Since $X \leq 2q$, in this case we always have that $k - X > 2q - X \geq 0$, so $k - X \geq 1$. Also, $k - X \leq k < n - 2q$. Put together, we have that $1 \leq k - X < n - 2q$, which implies that $\binom{n-2q}{k-X} \geq n - 2q$. Therefore in this case we have that
$$\widetilde{\mathbb{E}}_1\left[\frac{1}{\binom{n-2q}{k-X}}\right] \leq \frac{1}{n - 2q}.$$
- *Case 3:* $n - 2q \leq k$. Since $q = o(n)$, we have that $q < n/4$ for all n large enough. Note also that $X \leq 2q$. Thus for all n large enough we have that

$k - X \geq (n-2q) - 2q > 0$, so $k - X \geq 1$. Also note that $k - X \leq n - 2q$ by definition. If $1 \leq k - X < n - 2q$, then $\binom{n-2q}{k-X} \geq n - 2q \geq n/2$, where the second inequality holds for all n large enough. We thus have, for all n large enough, that

$$\widetilde{\mathbb{E}}_1 \left[\frac{1}{\binom{n-2q}{k-X}} \right] \leq \frac{2}{n} + \widetilde{\mathbb{P}}_1 \left(X = k - (n-2q) \right). \tag{3.7}$$

Since X is a hypergeometric random variable with parameters n, k, and $2q$, we have that

$$\begin{aligned} \widetilde{\mathbb{P}}_1 \left(X = k - (n-2q) \right) &= \frac{\binom{k}{k-(n-2q)} \binom{n-k}{n-k}}{\binom{n}{2q}} \\ &= \frac{2q(2q-1) \cdot \ldots \cdot (k - (n-2q) + 1)}{n(n-1) \cdot \ldots \cdot (k+1)} \leq \frac{2q}{n}. \end{aligned} \tag{3.8}$$

Combining (3.7) and (3.8) we have that $\widetilde{\mathbb{E}}_1 \left[1 / \binom{n-2q}{k-X} \right] \leq (2q+2)/n = o(1)$ as $n \to \infty$.

In summary, in all three cases above we have that $\widetilde{\mathbb{E}}_1 \left[1 / \binom{n-2q}{k-X} \right] = o(1)$ as $n \to \infty$. Combining this with (3.4) proves the claim. □

Proof of Theorem 1.2(a): There exists a direct correspondence between the problem of estimating the planted clique and the problem of estimating the marked set in the variant problem. This correspondence for the estimation problem is analogous to the correspondence for the detection problem described in the proof of Lemma 3.1. The proof then follows directly from Lemma 3.4 and this correspondence. □

Acknowledgements

M.Z.R. is grateful to David Gamarnik for helpful discussions. We also thank Jay Mardia, Joe Neeman, an anonymous reviewer, and an anonymous associate editor for helpful comments, feedback, and questions which helped improve the manuscript.

References

Alon, N., Krivelevich, M., and Sudakov, B. Finding a large hidden clique in a random graph. *Random Structures Algorithms*, **13** (3-4), 457–466 (1998). MR1662795.

Alweiss, R., Hamida, C. B., He, X., and Moreira, A. On the subgraph query problem. *ArXiv Mathematics e-prints* (2019). arXiv: 1911.04413.

Anagnostopoulos, A., Łacki, J., Lattanzi, S., Leonardi, S., and Mahdian, M. Community detection on evolving graphs. In *Advances in Neural Inf. Proc. Systems*, pp. 3522–3530 (2016).

Arora, S., Barak, B., Brunnermeier, M., and Ge, R. Computational Complexity and Information Asymmetry in Financial Products. *Communications of the ACM*, **54** (5), 101–107 (2011).

Berthet, Q. and Rigollet, P. Complexity Theoretic Lower Bounds for Sparse Principal Component Detection. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT)*, volume 30 of *Proceedings of Machine Learning Research*, pp. 1046–1066 (2013a).

Berthet, Q. and Rigollet, P. Optimal detection of sparse principal components in high dimension. *Ann. Statist.*, **41** (4), 1780–1815 (2013b). MR3127849.

Bollobás, B. and Erdős, P. Cliques in random graphs. *Math. Proc. Cambridge Philos. Soc.*, **80** (3), 419–427 (1976). MR498256.

Brennan, M. and Bresler, G. Optimal Average-Case Reductions to Sparse PCA: From Weak Assumptions to Strong Hardness. *ArXiv Mathematics e-prints* (2019). arXiv: 1902.07380.

Brennan, M., Bresler, G., and Huleihel, W. Reducibility and computational lower bounds for problems with planted sparse structure. In *Proceedings of the 31st Annual Conference on Learning Theory (COLT)*, volume 75 of *Proceedings of Machine Learning Research*, pp. 48–166 (2018).

Conlon, D., Fox, J., Grinshpun, A., and He, X. Online Ramsey Numbers and the Subgraph Query Problem. In *Building Bridges II*, pp. 159–194. Springer (2019). DOI: 10.1007/978-3-662-59204-5_4.

Dekel, Y., Gurel-Gurevich, O., and Peres, Y. Finding hidden cliques in linear time with high probability. *Combin. Probab. Comput.*, **23** (1), 29–49 (2014). MR3197965.

Deshpande, Y. and Montanari, A. Finding hidden cliques of size $\sqrt{N/e}$ in nearly linear time. *Found. Comput. Math.*, **15** (4), 1069–1128 (2015). MR3371378.

Feige, U., Gamarnik, D., Neeman, J., Rácz, M. Z., and Tetali, P. Finding cliques using few probes. *Random Structures Algorithms*, **56** (1), 142–153 (2020). MR4052849.

Feige, U. and Ron, D. Finding hidden cliques in linear time. In *21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10)*, Discrete Math. Theor. Comput. Sci. Proc., AM, pp. 189–203. Assoc. Discrete Math. Theor. Comput. Sci., Nancy (2010). MR2735341.

Feldman, V., Grigorescu, E., Reyzin, L., Vempala, S. S., and Xiao, Y. Statistical algorithms and a lower bound for detecting planted cliques. *J. ACM*, **64** (2), Art. 8, 37 (2017). MR3664576.

Ferber, A., Krivelevich, M., Sudakov, B., and Vieira, P. Finding Hamilton cycles in random graphs with few queries. *Random Structures Algorithms*, **49** (4), 635–668 (2016). MR3570983.

Ferber, A., Krivelevich, M., Sudakov, B., and Vieira, P. Finding paths in sparse random graphs requires many queries. *Random Structures Algorithms*, **50** (1), 71–85 (2017). MR3583027.

Gao, C., Ma, Z., and Zhou, H. H. Sparse CCA: adaptive estimation and computational barriers. *Ann. Statist.*, **45** (5), 2074–2101 (2017). MR3718162.

Hartmann, T., Kappes, A., and Wagner, D. Clustering evolving networks. In *Algorithm engineering*, volume 9220 of *Lecture Notes in Comput. Sci.*, pp. 280–329. Springer, Cham (2016). MR3609427.

Jerrum, M. Large cliques elude the Metropolis process. *Random Structures Algorithms*, **3** (4), 347–359 (1992). MR1179827.

Juels, A. and Peinado, M. Hiding cliques for cryptographic security. *Des. Codes Cryptogr.*, **20** (3), 269–280 (2000). MR1779310.

Kearns, M. Efficient noise-tolerant learning from statistical queries. *J. ACM*, **45** (6), 983–1006 (1998). MR1678849.

Kučera, L. Expected complexity of graph partitioning problems. *Discrete Appl. Math.*, **57** (2-3), 193–212 (1995). MR1327775.

Lugosi, G. Lectures on Combinatorial Statistics (2017). Available at http://www.econ.upf.edu/∼lugosi/SaintFlour.pdf.

Mardia, J., Asi, H., and Chandrasekher, K. A. Finding Planted Cliques in Sublinear Time. *ArXiv Mathematics e-prints* (2020). arXiv: 2004.12002.

Matula, D. W. The Employee Party Problem. In *Notices of the American Mathematical Society*, volume 19, pp. A–382 (1972).

Mazumdar, A. and Saha, B. Clustering with noisy queries. In *Advances in Neural Information Processing Systems*, pp. 5788–5799 (2017a).

Mazumdar, A. and Saha, B. Query complexity of clustering with side information. In *Advances in Neural Information Processing Systems*, pp. 4682–4693 (2017b).

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, **298** (5594), 824–827 (2002). DOI: 10.1126/science.298.5594.824.

Vinayak, R. K. and Hassibi, B. Crowdsourced clustering: Querying edges vs triangles. In *Advances in Neural Information Processing Systems*, pp. 1316–1324 (2016).