How to Hide a Clique?

Uriel Feige
The Weizmann Institute, Rehovot, Israel
uriel.feige@weizmann.ac.il

Vadim Grinberg
Toyota Technological Institute at Chicago, IL, USA
vgm@ttic.edu

Abstract

In the well known planted clique problem, a clique (or alternatively, an independent set) of size k is planted at random in an Erdos-Renyi random $G(n,p)$ graph, and the goal is to design an algorithm that finds the maximum clique (or independent set) in the resulting graph. We introduce a variation on this problem, where instead of planting the clique at random, the clique is planted by an adversary who attempts to make it difficult to find the maximum clique in the resulting graph. We show that for the standard setting of the parameters of the problem, namely, a clique of size $k = \sqrt{n}$ planted in a random $G(n, \frac{1}{2})$ graph, the known polynomial time algorithms can be extended (in a non-trivial way) to work also in the adversarial setting. In contrast, we show that for other natural settings of the parameters, such as planting an independent set of size $k = \frac{n}{2}$ in a $G(n,p)$ graph with $p = n^{-\frac{1}{2}}$, there is no polynomial time algorithm that finds an independent set of size k, unless NP has randomized polynomial time algorithms.

2012 ACM Subject Classification Theory of computation

Keywords and phrases planted clique, semi-random model, Lovasz theta function, random graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.44

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/2004.12258`.

Funding *Uriel Feige*: Supported in part by the Israel Science Foundation (grant No. 1388/16).

1 Introduction

The planted clique problem, also referred to as hidden clique, is a problem of central importance in the design of algorithms. We introduce a variation of this problem where instead of planting the clique at random, an adversary plants the clique. Our main results are that in certain regimes of the parameters of the problem, the known polynomial time algorithms can be extended to work also in the adversarial settings, whereas for other regimes, the adversarial planting version becomes NP-hard. We find the results interesting for three reasons. One is that they concern an extensively studied problem (planted clique), but from a new direction, and we find that the results lead to a better understanding of what aspects of the planted clique problem are made use of by the known algorithms. Another is that extending the known algorithms (based on semidefinite programming) to the adversarial planted setting involves some new techniques regarding how semidefinite programming can be used and analysed. Finally, the NP-hardness results are interesting as they are proven in a semi-random model in which most of the input instance is random, and the adversary controls only a relatively small aspect of the input instance. One may hope that this brings us closer to proving NP-hardness results for purely random models, a task whose achievement would be a breakthrough in complexity theory.

1.1 The random planted clique model

Our starting point is the Erdos-Renyi $G(n,p)$ random graph model, which generates graphs on n vertices, and every two vertices are connected by an edge independently with probability p. We start our discussion with the special case in which $p = \frac{1}{2}$, and other values of p will be considered later. Given a graph G, let $\omega(G)$ denote the size of the maximum clique in G, and let $\alpha(G)$ denote the size of the maximum independent set. Given a distribution D over graphs, we use the notation $G \sim D$ for denoting a graph sampled at random according to D. The (edge) complement of a graph $G \sim G(n, \frac{1}{2})$ is by itself a graph sampled from $G(n, \frac{1}{2})$, and the complement of a clique is an independent set, and hence the discussion concerning cliques in $G(n, \frac{1}{2})$ extends without change to independent sets (and vice versa).

It is well known (proved by computing the expectation and variance of the number of cliques of the appropriate size) that for $G \sim G(n, \frac{1}{2})$, w.h.p. $\omega(G) \simeq 2\log n$ (the logarithm is in base 2). However, there is no known polynomial time algorithm that can find cliques of size $2\log n$ in such graphs. A polynomial time greedy algorithm can find a clique of size $(1+o(1))\log n$. The existence of $\rho > 1$ for which polynomial time algorithms can find cliques of size $\rho \log n$ is a longstanding open problem.

In the classical planted clique problem, one starts with a graph $G' \sim G(n, \frac{1}{2})$ and a parameter k. In G' one chooses at random a set K of k vertices, and makes this set into a clique by inserting all missing edges between pairs of vertices with K. We refer to K as the planted clique, and say that the resulting graph G is distributed according to $G(n, \frac{1}{2}, k)$. Given $G \sim G(n, \frac{1}{2}, k)$, the algorithmic goal can be one of the following three: find K, find a clique of maximum size, or find any clique of size at least k. It is not difficult to show that when k is sufficiently large (say, $k > 3\log n$), then with high probability K is the unique maximum size clique in $G \sim G(n, \frac{1}{2}, k)$, and hence all three goals coincide. Hence in the planted clique problem, the goal is simply to design polynomial time algorithms that (with high probability over the choice of $G \sim G(n, \frac{1}{2}, k)$) find the planted clique K. The question is how large should k be (as a function of n) so as to make this task feasible.

For some sufficiently large constant $c > 0$ (throughout, we use c to denote a sufficiently large constant), if $k > c\sqrt{n \log n}$, with high probability the the vertices of K are simply the k vertices of highest degree in G (see [13]), and hence K can easily be recovered. Alon, Krivelevich and Sudakov [1] managed to shave the $\sqrt{\log n}$ factor, designing a spectral algorithm that recovers K when $k > c\sqrt{n}$. They also showed that c can be made an arbitrarily small constant, by increased the running time by a factor of $n^{O(\log(\frac{1}{c}))}$ (this is done by "guessing" a set K' of $O(\log(\frac{1}{c}))$ vertices of K, and finding the maximum clique in the subgraph induced on their common neighbors). Subsequently, additional algorithms were developed that find the planted clique when $k > c\sqrt{n}$. They include algorithms based on the Lovasz theta function, which is a form of semi-definite programming [7], algorithms based on a "reverse-greedy" principle [10, 3], and message passing algorithms [4]. There have been many attempts to find polynomial time algorithms that succeed when $k = o(\sqrt{n})$, but so far all of them failed (see for example [11, 8, 16]). It is a major open problem whether there is any such polynomial time algorithm.

Planted clique when $p \neq \frac{1}{2}$ was not studied as extensively, but it is quite well understood how results from the $G(n, \frac{1}{2}, k)$ model transfer to the $G(n, p, k)$ model. For p much smaller that $\frac{1}{2}$, say $p = n^{\delta-1}$ for some $0 < \delta < 1$ (hence average degree n^{δ}), the problem changes completely. Even without planting, with high probability over the choice of $G \sim G(n,p)$ (with $p = n^{\delta-1}$) we have that $\omega(G) = O(\frac{1}{1-\delta})$, and the maximum clique can be found in polynomial time. This also extends to finding maximum cliques in the planted setting, regardless of the value of k. (We are not aware of such results being previously published,

but they are not difficult. See Section 2.2.) For $p > \frac{1}{2}$, it is more convenient to instead look at the equivalent problem in which $p < \frac{1}{2}$, but with the goal of finding a planted independent set instead of a planted clique. We refer to this model as $\bar{G}(n,p,k)$. For $G \sim G(n,p)$ (with $p = n^{\delta-1}$) we have that with high probability $\alpha(G) = \Theta(n^{1-\delta}\log n)$. For $G \sim \bar{G}(n,p,k)$ the known algorithms extend to finding planted independent sets of size $k = cn^{1-\frac{\delta}{2}}$ in polynomial time. We remark that the approach of [1] of making c arbitrarily small does not work for such sparse graphs.

1.2 The adversarial planted clique model

In this paper we introduce a variation on the planted clique model (and planted independent set model) that we refer to as the adversarial planted clique model. As in the random planted clique model, we start with a graph $G' \sim G(n,p)$ and a parameter k. However, now a computationally unbounded adversary may inspect G', select within it a subset K of k vertices of its choice, and make this set into a clique by inserting all missing edges between pairs of vertices with K. We refer to this model as $AG(n,p,k)$ (and the corresponding model for planted independent set as $A\bar{G}(n,p,k)$). As shorthand notation shall use $G \sim AG(n,p,k)$ to denote a graph generated by this process. Let us clarify that $AG(n,p,k)$ is not a distribution over graphs, but rather a family of distributions, where each adversarial strategy (where a strategy of an adversary is a mapping from G' to a choice of K) gives rise to a different distribution.

In the adversarial planted model, it is no longer true that the planted clique is the one of maximum size in the resulting graph G. Moreover, finding K itself may be information theoretically impossible, as K might be statistically indistinguishable from some other clique of size k (that differs from K by a small number of vertices). The three goals, that of finding K, finding a clique of maximum size, or finding any clique of size at least k, are no longer equivalent. Consequently, for our algorithmic results we shall aim at the more demanding goal of finding a clique of maximum size, whereas for our hardness results, we shall want them to hold even for the less demanding goal of finding an arbitrary clique of size k.

1.3 Our results

Our results cover a wide range of values of $0 < p < 1$, where p may be a function of n. For simplicity of the presentation and to convey the main insights of our results, we present here the results for three representative regimes: $p = \frac{1}{2}$, $p = n^{\delta-1}$ for $0 < \delta < 1$, and $p = 1 - n^{\delta-1}$. For the latter regime, it will be more convenient to replace it by the equivalent problem of finding adversarially planted independent sets when $p = n^{\delta-1}$.

Informally, our results show the following phenomenon. We consider only the case that $p \leq \frac{1}{2}$, but consider both the planted clique and the planted independent set problems, and hence the results can be translated to $p > \frac{1}{2}$ as well. For clique, we show (Theorem 1 and Theorem 2) how to extend the algorithmic results known for the random planted clique setting to the adversarial planted clique setting. However, for independent set, we show that this is no longer possible. Specifically, when p is sufficiently small, we prove (Theorem 3) that finding an independent set of size k (any independent set, not necessarily the planted one) in the adversarial planted independent set setting is NP-hard. Moreover, the NP-hardness result holds even for large values of k for which finding a random planted independent set is trivial.

▶ **Theorem 1.** *For every fixed $\varepsilon > 0$ and for every $k \geq \varepsilon\sqrt{n}$, there is an (explicitly described) algorithm running in time $n^{O(\log(\frac{1}{\varepsilon}))}$ which almost surely finds the maximum clique in a graph $G \sim AG(n, \frac{1}{2}, k)$. The statement holds for every adversarial planting strategy (choice of k vertices as a function of $G' \sim G(n, \frac{1}{2})$), and the probability of success is taken over the choice of $G' \sim G(n, \frac{1}{2})$.*

▶ **Theorem 2.** *Let $p = n^{\delta-1}$ for $0 < \delta < 1$. Then for every k, there is an (explicitly described) algorithm running in time $n^{O(\frac{1}{1-\delta})}$ which almost surely finds the maximum clique in a graph $G \sim AG(n, p, k)$. The statement holds for every adversarial planting strategy, and the probability of success is taken over the choice of $G' \sim G(n, p)$.*

▶ **Theorem 3.** *For $p = n^{\delta-1}$ with $0 < \delta < 1$, $0 < \gamma < 1$, and $cn^{1-\delta} \log n \leq k \leq \frac{2}{3}n$ (where c is a sufficiently large constant, and the constant $\frac{2}{3}$ was chosen for concreteness – any other constant smaller than 1 will work as well) the following holds. There is no polynomial time algorithm that has probability at least γ of finding an independent set of size k in $G \sim A\bar{G}(n, p, k)$, unless NP has randomized polynomial time algorithms (NP=RP). (The algorithm is required to succeed against every adversarial planting strategy, and the probability of success is taken over the choice of $G' \sim G(n, p)$.)*

1.4 Related work

Some related work was already mentioned in Section 1.1.

Our algorithm for Theorem 1 is based on an adaptation of the algorithm of [7] that applied to the random planted clique setting. In turn, that algorithm is based on the theta function of Lovasz [14].

A work that is closely related to ours and served as an inspiration both to the model that we study, and to the techniques that are used in the proof of the NP-hardness result (Theorem 3) is the work of David and Feige [2] on adversarially planted 3-colorings. That work uncovers a phenomenon similar to the one displayed in the current work. Specifically, for the problem of 3-coloring (rather than clique or independent set) it shows that for certain values of p, algorithms that work in the random planted setting can be extended to the adversarial planted setting, and for other values of p, finding a 3-coloring in the adversarial planted setting becomes NP-hard. However, there are large gaps left open in the picture that emerges from the work of [2]. For large ranges of the values of p, specifically, $n^{-1/2} < p < n^{-1/3}$ and $p < n^{-2/3}$, there are neither algorithmic results nor hardness results in the work of [2]. Unfortunately, the most interesting values of p for the 3-coloring problem, which are $p \leq \frac{c \log n}{n}$, lie within these gaps, and hence the results of [2] do not apply to them. Our work addresses a different problem (planted clique instead of planted 3-coloring), and for our problem, our analysis leaves almost no such gaps. We are able to determine for which values of p the problem is polynomial time solvable, and for which values it is NP-hard. See Section 3 for more details.

Our model is an example of a *semi-random* model, in which part of the input is determined at random and part is determined by an adversary. There are many other semi-random models, both for the clique problem and for other problems. Describing all these models is beyond the scope of this paper, and the interested reader is referred to [5] and references therein for additional information.

2 Overview of the proofs

In this section we provide an overview of the proofs for our three main theorems. Further details, as well as extensions to the results, appear in the full version of our paper [6].

The term *almost surely* denotes a probability that tends to 1 as n grows. The term *extremely high probability* denotes a probability of the form $1 - e^{-n^r}$ for some $r > 0$. By $\exp(x)$ for some expression x we mean e^x.

2.1 Finding cliques using the theta function

In this section we provide an overview of the proof of Theorem 1. Our algorithm is an adaptation of the algorithm of [7] that finds the maximum clique in the random planted model. We shall first review that algorithm, then describe why it does not apply in our setting in which an adversary plants the clique, and finally explain how we modify that algorithm and its analysis so as to apply it in the adversarial planted setting.

The key ingredient in the algorithm of [7] is the theta function of Lovasz, denoted by ϑ. Given a graph G, $\vartheta(G)$ can be computed in polynomial time (up to arbitrary precision, using semidefinite programming (SDP)), and satisfies $\vartheta(G) \geq \alpha(G)$. As we are interested here in cliques and not in independent sets, we shall consider $\bar{G}$, the edge complement of G, and then $\vartheta(\bar{G}) \geq \omega(G)$. The theta function has several equivalent definitions, and the one that we shall use here (referred to as ϑ_4 in [14]) is the following.

Given a graph $G = G(V, E)$, a collection of unit vectors $s_i \in \mathbb{R}^n$ (one vector for every vertex $i \in V$) is an *orthonormal representation* of G, if s_i and s_j are orthogonal ($s_i \cdot s_j = 0$) whenever $(i, j) \in E$. The theta function is the maximum value of the following expression, where maximization is over all orthonormal representations $\{s_i\}$ of G and over all unit vectors h (h is referred to as the *handle*):

$$\vartheta(G) = \max_{h,\{s_i\}} \sum_{i \in V} (h \cdot s_i)^2 \tag{1}$$

The optimal orthonormal representation and the associated handle that maximize the above formulation for ϑ can be found (up to arbitrary precision) in polynomial time by formulating the problem as an SDP (details omitted). Observe that for any independent set S the following is a feasible solution for the SDP: choose $s_i = h$ for all $i \in S$, and choose all remaining vectors s_j for $j \notin S$ to be orthogonal to h and to each other. Consequently, $\vartheta(G) \geq \alpha(G)$, as claimed.

The main content of the algorithm of [7] is summarized in the following theorem. We phrased it in a way that addresses cliques rather than independent sets, implicitly using $\alpha(\bar{G}) = \omega(G)$. We also remind the reader that in the random planted model, the planted clique K is almost surely the unique maximum clique.

▶ **Theorem 4** (Results of [7]). *Consider $G \sim G(n, \frac{1}{2}, k)$, a graph selected in the random planted clique model, with $k \geq c\sqrt{n}$ for some sufficiently large constant c. Then with extremely high probability (over choice of G) it holds that $\vartheta(\bar{G}) = \omega(G)$.*

Moreover, for every vertex i that belongs to the planted clique K, the corresponding vector s_i has inner product larger than $1 - \frac{1}{n}$ with the handle h, and for every other vertex, the corresponding inner product is at most $\frac{1}{n}$.

Given Theorem 4, the following algorithm finds the planted clique when $G \sim G(n, \frac{1}{2}, k)$, and $k \geq c\sqrt{n}$ for some sufficiently large constant c. Solve the optimization problem (1) (on $\bar{G}$) to sufficiently high precision, and output all vertices whose corresponding inner product with h is at least $\frac{1}{2}$.

The algorithm above does not apply to $G \sim AG(n, \frac{1}{2}, k)$, a graph selected in the adversarial planted clique model, for the simple reason that Theorem 4 is incorrect in that model. The following example illustrates what might go wrong,

▶ **Example 5.** Consider a graph $G' \sim G(n, \frac{1}{2})$. In G' first select a random vertex set T of size slightly smaller than $\frac{1}{2}\log n$. Observe that the number of vertices in G' that are in the common neighborhood of all vertices of T is roughly $2^{-|T|}n > \sqrt{n}$. Plant a clique K of size k in the common neighborhood of T. In this construction, K is no longer the largest clique in G. This is because T (being a random graph) is expected to have a clique K' of size $2\log|T| \simeq 2\log\log n$, and $K' \cup K$ forms a clique of size roughly $k + 2\log\log n$ in G. Moreover, as T itself is a random graph with edge probability $\frac{1}{2}$, the value of the theta function on T is roughly $\sqrt{|T|}$ (see [12]), and consequently one would expect the value of $\vartheta(\bar{G})$ to be roughly $k + \sqrt{\log n}$.

Summarizing, it is not difficult to come up with strategies for planting cliques of size k that result in the maximum clique having size strictly larger than k, and the value of $\vartheta(\bar{G})$ being even larger. Consequently, the solution of the optimization problem (1) by itself is not expected to correspond to the maximum clique in G.

We now explain how we overcome the above difficulty. A relatively simple, yet important, observation is the following.

▶ **Proposition 6.** *Let $G \sim AG(n, p, k)$ with $p = 1/2$ and $k > \sqrt{n}$, and let K' be the maximum clique in G (which may differ from the planted clique K). Then with extremely high probability over the choice of $G' \sim G(n, \frac{1}{2})$, for every possible choice of k vertices by the adversary, K' contains at least $k - O(\log n)$ vertices from K, and at most $O(\log n)$ additional vertices.*

Proof. Standard probabilistic arguments show that with extremely high probability, the largest clique in G' (prior to planting a clique of size k) is of size at most $\frac{k}{2}$. When this holds, K' contains at least $\frac{k}{2}$ vertices from K. Each of the remaining vertices of K' needs to be connected to all vertices in $K' \cap K$. Consequently, with extremely high probability, K' contains at most $2\log n$ vertices not from K. This is because a $G' \sim G(n, \frac{1}{2})$ graph, with extremely high probability, does not contain two sets of vertices A and B, with $|A| = 2\log n$, $|B| = \Omega(\sqrt{n})$, such that all pairs of vertices in $A \times B$ induce edges in G.

As $|K'| \geq k$, we conclude that all but $O(\log n)$ vertices of K must be members of K'. ◀

A key theorem that we prove is:

▶ **Theorem 7.** *Let $G \sim AG(n, p, k)$ with $p = 1/2$ and $k = k(n) \geq 10\sqrt{n}$. Then $k \leq \vartheta(\bar{G}) \leq k + O(\log n)$ with extremely high probability over the choice of $G' \sim G(n, \frac{1}{2})$, for every possible choice of k vertices by the adversary.*

We now explain how Theorem 7 is proved. The bound $\vartheta(\bar{G}) \geq k$ was already explained above. Hence it remains to show that $\vartheta(\bar{G}) \leq k + O(\log n)$. In general, to bound $\vartheta(G)$ from above for a graph $G(V, E)$, one considers the following dual formulation of ϑ, as a minimization problem.

$$\vartheta(G) = \min_{M}[\lambda_1(M)] \tag{2}$$

Here M ranges over all n by n symmetric matrices in which $M_{ij} = 1$ whenever $(i, j) \notin E$, and $\lambda_1(M)$ denotes the largest eigenvalue of M. (Observe that if G has an independent set S of size k, then M contains a k by k block of 1 entries. A Rayleigh quotient argument then implies that $\lambda_1(M) \geq k$, thus verifying the inequality $\vartheta(G) \geq \alpha(G)$.) To prove Theorem 7 we exhibit a matrix M as above (for the graph $\bar{G}$) for which we prove that $\lambda_1(M) \leq k + O(\log n)$.

We first review how a matrix M was chosen by [7] in the proof of Theorem 4. First, recall that we consider $\bar{G}$, and let E be the set of edges of $\bar{G}$ (non-edges of G). We need to associate values with the entries M_{ij} for $(i,j) \in E$ (as other entries are 1). The matrix block corresponding to the planted clique K (planted independent set in $\bar{G}$) is all 1 (by necessity). For every $(i,j) \in E$ where both vertices are not in K one sets $M_{ij} = -1$. For every other pair $(i,j) \in E$ (say, $i \in K$ and $j \notin K$) one sets $M_{i,j} = -\frac{k-d_{i,K}}{d_{i,K}}$, where $d_{i,K}$ is the number of neighbors that vertex i has in the set K. In order to show that $\lambda_1(M) = k$, one first observes that the vector x_K (with value 1 at entries that correspond to vertices of K, and value 0 elsewhere) is an eigenvector of M with eigenvalue k. Then one proves that $\lambda_2(M)$, the second largest eigenvalue of M, has value smaller than k. This is done by decomposing M into a sum of several matrices, bounding the second largest eigenvalue for one of these matrices, and the largest eigenvalue for the other matrices. By Weyl's inequality, the sum of these eigenvalues is an upper bound on $\lambda_2(M)$. This upper bound is not tight, but it does show that $\lambda_2(M) < k$. It follows that the eigenvalue k associated with x_K is indeed $\lambda_1(M)$. Further details are omitted.

We now explain how to choose a matrix M so as to prove the bound $\vartheta(\bar{G}) \leq k + O(\log n)$ in Theorem 7. Recall (see Example 5) that we might be in a situation in which $\vartheta(\bar{G}) > \alpha(\bar{G}) > k$ (with all inequalities being strict). In this case, let K' denote the largest independent set in $\bar{G}$, and note that K' is larger than K. In M, the matrix block corresponding to K' is all 1. One may attempt to complete the construction of M as described above for the random planting case, but replacing K by K' everywhere in that construction. If one does so, the vector $x_{K'}$ (with value 1 at entries that correspond to vertices of K', and value 0 elsewhere) is an eigenvector of M with eigenvalue $\alpha(\bar{G}) > k$. However, M would necessarily have another eigenvector with a larger eigenvalue, because $\vartheta(\bar{G}) > \alpha(\bar{G})$. Hence we are still left with the problem of bounding $\lambda_1(M)$, rather than bounding $\lambda_2(M)$. Having failed to identify an eigenvector for $\lambda_1(M)$, we may still obtain an upper bound on $\lambda_1(M)$ by using approaches based on Weyl's inequality (or other approaches). However, these upper bounds are not tight, and it seems difficult to limit the error that they introduce to be as small as $O(\log n)$, which is needed for proving the inequality $\lambda_1(M) \leq k + O(\log n)$.

For the above reason, we choose M differently. For some constant $\frac{1}{2} < \rho < 1$, we extend the clique K to a possibly larger clique Q, by adding to it every vertex that has ρk neighbors in K. (In Example 5, the corresponding clique Q will include all vertices of $K \cup T$. In contrast, if K is planted at random and not adversarially, then we will simply have $Q = K$.) Importantly, we prove that if $G' \sim G(n, \frac{1}{2})$, then with high probability $|Q| < k + O(\log n)$ (for every possible choice of planting a clique of size k by the adversary). For the resulting graph G_Q, we choose the corresponding matrix M in the same way as it was chosen for the random planting case. Now we do manage to show that the eigenvector x_Q (with eigenvalue $|Q|$) associated with this M indeed has the largest eigenvalue. This part is highly technical, and significantly more difficult than the corresponding proof for the random planting case. The reason for the added level of difficulty is that, unlike the random planting case in which we are dealing with only one random graph, here the adversary can plant the clique in any one of $\binom{n}{k}$ locations, and our analysis needs to hold simultaneously for all $\binom{n}{k}$ graphs that may result from such plantings. Further details can be found in [6].

Having established that $\vartheta(\bar{G}_Q) = |Q| \leq k + O(\log n)$, we use monotonicity of the theta function to conclude that $\vartheta(\bar{G}) \leq k + O(\log n)$. This concludes our overview for the proof of Theorem 7.

Given Theorem 7, let us now explain our algorithm for finding a maximum clique in $G \sim AG(n, \frac{1}{2}, k)$.

Given a graph $G \sim AG(n, \frac{1}{2}, k)$, the first step in our algorithm is to solve the optimization problem (1) on the complement graph $\bar{G}$. By Theorem 7, we will have $\vartheta(\bar{G}) \leq k + c \log n$ for some constant $c > 0$. Let $\{s_i\}$ denote the orthonormal representation found by our solution, and let h be the corresponding handle.

The second step of our algorithm it to extract from G a set of vertices that we shall refer to as H, that contains all those vertices i for which $(h \cdot s_i)^2 \geq \frac{3}{4}$.

▶ **Lemma 8.** *For H as defined above, with extremely high probability, at least $k - O(\log n)$ vertices of K are in H, and most $O(\log n)$ vertices not from K are in H.*

Proof. Let T denote the set of those vertices in K for which $(h \cdot s_i)^2 < \frac{3}{4}$. Remove T from G, thus obtaining the graph G_T. This graph can be thought of as a subgraph with $n - |T|$ vertices of the random graph $G' \sim G(n, \frac{1}{2})$, in which an adversary planted a clique of size $k - |T|$. We also have that $\vartheta(\bar{G}_T) \geq \vartheta(\bar{G}) - \sum_{i \in T} (h \cdot s_i)^2 \geq k - \frac{3}{4}|T|$. If $|T|$ is large (larger than $c' \log n$ for some sufficiently large constant $c' > 0$), the gap of $\frac{|T|}{4}$ between the size of the planted clique and the value of the theta function contradicts Theorem 7 for the graph G_T. (Technical remark: this last argument uses the fact that Theorem 7 holds with extremely high probability, as we take a union bound over all choices of T.)

Having established that T is small, let R be the set of vertices not in K for which $(h \cdot s_i)^2 \geq \frac{3}{4}$. We claim that every such vertex $i \in R$ is a neighbor of every vertex $j \in K \setminus T$. This is because in the orthogonal representation (for $\bar{G}$), if i and j are not neighbors we have that $s_i \cdot s_j = 0$, and then the fact that s_i,s_j and h are unit vectors implies that $(h \cdot s_i)^2 < 1 - (h \cdot s_j)^2 \leq \frac{1}{4}$. Having this claim and using the fact that $|K \setminus T| > \sqrt{n}$, it follows that $|R| \leq 2 \log n$. This is because a $G' \sim G(n, \frac{1}{2})$ graph, with extremely high probability, does not contain two sets of vertices A and B, with $|A| = 2 \log n$, $|B| = \sqrt{n}$, such that all pairs of vertices in $A \times B$ induce edges in G. ◀

The third step of our algorithm constructs a set F that contains all those vertices that have at least $\frac{3k}{4}$ neighbors in H.

▶ **Lemma 9.** *With extremely high probability, the set F described above contains the maximum clique in G, and at most $O(\log n)$ additional vertices.*

Proof. We may assume that H satisfies the properties of Lemma 8. Proposition 6 then implies that with extremely high probability, every vertex of the maximum clique in G has at least $\frac{3k}{4}$ neighbors in H, and hence is contained in F. A probabilistic argument (similar to the end of the proof of Lemma 8) establishes that F has at most $O(\log n)$ vertices not from K. As K has at most $O(\log n)$ vertices not from the maximum clique (by Proposition 6), the total number of vertices in F that are not members of the maximum clique is at most $O(\log n)$. ◀

Finally, in the last step of our algorithm we find a maximum clique in F, and this is a maximum clique in G. This last step can be performed in polynomial time by a standard algorithm (used for example to show that vertex cover is fixed parameter tractable). For every non-edge in the subgraph induced on F, at least one of its end-vertices needs to be removed. Try both possibilities in parallel, and recurse on each subgraph that remains. The recursion terminates when the graph is a clique. The shortest branch in the recursion gives the maximum clique. As only $O(\log n)$ vertices need to be removed in order to obtain a clique, the depth of the recursion is at most $O(\log n)$, and consequently the running time (which is exponential in the depth) is polynomial in n.

This completes our overview of our algorithm for finding a clique in $G \sim AG(n, \frac{1}{2}, k)$ when $k > c\sqrt{n}$ for a sufficiently large constant $c > 0$. To complete the proof of Theorem 1 we need to also address the case that $k > \varepsilon\sqrt{n}$ for arbitrarily small constant ε. This we do (as in [1]) by guessing $t \simeq 2\log\frac{c}{\epsilon}$ vertices from K (there are n^t possibilities to try, and we try all of them), and considering the subgraph of G induced on their common neighbors. This subgraph corresponds to a subgraph of $G' \simeq G(n, \frac{1}{2})$ with roughy $n' \simeq 2^{-t}n$ vertices, and a planted clique of size $\varepsilon\sqrt{n} - t \simeq c\sqrt{n'}$. Now on this new graph G" we can invoke the algorithm based on the theta function. (Technical remark. The proof that $\vartheta(\bar{G}") \leq k + O(\log n)$ uses the fact that Theorem 7 holds with extremely high probability.)

The many details that were omitted from the above overview of the proof of Theorem 1 can be found in [6].

2.2 Finding cliques by enumeration

In this section we prove Theorem 2.

Let $p = n^{\delta-1}$ for $0 < \delta < 1$, and consider first $G' \sim G(n,p)$ (hence G' has average degree roughly n^δ). For every size $t \geq 1$, let N_t denote the number of cliques of size t in G'. The expectation (over choice of $G' \sim G(n,p)$) satisfies:

$$\mathbb{E}[N_t] = \binom{n}{t} p^{\binom{t}{2}} \leq \frac{1}{t!} n^{\frac{\delta-1}{2}t^2 + \frac{3-\delta}{2}t}$$

The exponent is maximized when $t = \frac{3-\delta}{2(1-\delta)}$. For the maximizing (not necessarily integer) t, the exponent equals $\frac{(3-\delta)^2}{8(1-\delta)}$. We denote this last expression by e_δ, and note that $e_\delta = O(\frac{1}{1-\delta})$. The expected number of cliques of all sizes is then:

$$\sum_{t\geq 1} \mathbb{E}[N_t] \leq n + \sum_{t\geq 2} \frac{1}{t!} n^{\frac{\delta-1}{2}t^2 + \frac{3-\delta}{2}t} \leq n^{e_\delta}$$

(The last inequality holds for sufficiently large n.) By Markov's inequality, with probability at least $1 - \frac{1}{n}$, the actual number of cliques in G' is at most $n^{e_\delta+1}$. (Stronger concentration results can be used here, but are not needed for the proof of Theorem 2.)

Now, for arbitrary $1 \leq k \leq n$, let the adversary plant a clique K of size k in G', thus creating the graph $G \sim G(n,p,k)$. As every subgraph of K is a clique, the total number of cliques in G is at least 2^k, which might be exponential in n (if k is large). However, the number of maximal cliques in G (a clique is maximal if it is not contained in any larger clique) is much smaller. Given a maximal clique C in G, consider C', the subgraph of C not containing any vertex from K. C' is a clique in G' (which is nonempty, except for one special case of $C = K$). C' uniquely determines C, as the remaining vertices in C are precisely the set of common neighbors of C' in K (this is because the clique C is maximal). Consequently, the number of maximal cliques in G is not larger than the number of cliques in G'.

As all maximal cliques in a graph can be enumerated in time linear in their number times some polynomial in n (see e.g. [15] and references therein), one can list all maximal cliques in G in time $n^{e_\beta+O(1)}$ (this holds with probability at least $1 - \frac{1}{n}$, over the choice of G', regardless of where the adversary plants clique K), and output the largest one.

This completes the proof of Theorem 2.

2.3 Proving NP-hardness results

In this section we provide an overview of the proof of Theorem 3. Our proof is an adaptation to our setting of a proof technique developed in [2].

Recall that we are considering a graph $G \sim A\bar{G}(n,p,k)$ (adversarial planted independent set) with $p = n^{\delta-1}$ and $0 < \delta < 1$. Let us first explain why the algorithm described in Section 2.1 fails when $k = cn^{1-\frac{\delta}{2}}$ (whereas if the independent set is planted at random, algorithms based on the theta function are known to succeed). The problem is that the bound in Theorem 7 is not true anymore, and instead one has the much weaker bound of $\vartheta(G) \leq k + n^{1-\delta}\log n$. Following the steps of the algorithm of Section 2.1, in the final step, we would need to remove a minimum vertex cover from F. However, now the upper bound on the size of this vertex cover is $O(n^{1-\delta}\log n)$ rather than $O(\log n)$. Consequently, we do not know of a polynomial time algorithm that will do so. It may seem that we also do not know that no such algorithm exists. After all, F is not an arbitrary worst case instance for vertex cover, but rather an instance derived from a random graph. However, our NP-hardness result shows that indeed this obstacle is insurmountable, unless NP has randomized polynomial time algorithms. We remark that using an approximation algorithm for vertex cover in the last step of the algorithm of Section 2.1 does allow one to find in G an independent set of size $k - O(n^{1-\delta}\log n) = (1-o(1))k$, and the NP-hardness result applies only because we insist on finding an independent set of size at least k.

Let us proceed now with an overview of our NP hardness proof. We do so for the case that $k = \frac{n}{3}$ (for which we can easily find the maximum independent set if the planted independent set is random). Assume for the sake of contradiction that ALG is a polynomial time algorithm that with high probability over choice of $G' \sim G(n,p)$, for every planted independent set of size $k = \frac{n}{3}$, it finds in the resulting graph G an independent set of size k.

We now introduce a class $\mathcal{H}$ of graphs that, in anticipation of the proofs that will follow, is required to have the following three properties. (Two of the properties are stated below in a qualitative manner, but they have precise quantitative requirements in the proofs that follow.)

1. Solving maximum independent set on graphs from this class is NP-hard.
2. Graphs in this class are very sparse.
3. The number of vertices in each graph is small.

Given the above requirements, we choose $0 < \varepsilon < \min[\frac{\delta}{2}, 1-\delta]$, and let $\mathcal{H}$ be the class of *balanced* graphs on n^{ϵ} vertices, and of average degree $2+\delta$. (A graph H is *balanced* if no subgraph of H has average degree larger than the average degree of H.) Given a graph $H \in \mathcal{H}$ and a parameter k', it is NP-hard to determine whether H has an independent of size at least k' or not. We will reach a contradiction to the existence of ALG by showing how ALG could be used in order to find in H an independent set of size k', if one exists. For this, we use the following randomized algorithm ALGRAND.

1. Generate a random graph $G' \sim G(n,p)$.
2. Plant in G' a random copy of H (that is, pick $|H|$ random vertices in G' and replace the subgraph induced on them by H). We refer to the resulting distribution as $G_H(n,p)$, and to the graph sampled from this distribution as G_H. Observe that the number of vertices in G_H that have a neighbor in H is with high probability not larger than $|H|n^{\delta} \leq \frac{n}{2}$.
3. Within the non-neighbors of H, plant at random an independent set of size $k-k'$. We refer to the resulting distribution as $G_H(n,p,k)$, and to the graph sampled from this distribution as $\tilde{G}_H$. Observe that with extremely high probability, $\alpha(\tilde{G}_H \setminus H) = k-k'$. Hence we may assume that this indeed holds. If furthermore $\alpha(H) \geq k'$, then $\alpha(\tilde{G}_H) \geq k$.
4. Run ALG on $\tilde{G}_H$. We say that ALGRAND succeeds if ALG outputs an independent set IS of size k. Observe that then at least k' vertices of H are in IS, and hence ALGRAND finds an independent set of size k' in H.

If H does not have an independent set of size k', ALGRAND surely fails to output such an independent set. But if H does have an independent set of size k', why should ALGRAND succeed? This is because ALG (which is used in ALGRAND) is fooled to think that the graph $\tilde{G}_H$ generated by ALGRAND was generated from $A\bar{G}(n,p,k)$, and on such graphs ALG does find independent sets of size k. And why is ALG fooled? This is because the distribution of graphs generated by ALGRAND is statistically close to a distribution that can be created by the adversary in the $A\bar{G}(n,p,k)$ model. Specifically, consider the following distribution that we refer to as $A_HG(n,p,k)$.

1. Generate $G' \sim G(n,p)$.
2. The computationally unbounded adversary finds within G' all subsets of vertices of size $|H|$ such that the subgraph induced on them is H. (If there is no such subset, fail.) Choose one such copy of H uniformly at random.
3. As H is assumed to have an independent set of size k', plant an independent set K of size k as follows. k' of the vertices of K are vertices of an independent set in the selected copy of H. The remaining $k-k'$ vertices of K are chosen at random among the vertices of G' that have no neighbor at all in the copy of H. (Observe that we expect there to be at least roughly $n - |H|n^{\delta} \geq \frac{n}{2}$ such vertices, and with extremely high probability the actual number will be at least $\frac{n}{3} > k - k'$.)

▶ **Theorem 10.** *The two distributions, $\tilde{G}_H \sim G_H(n,p,k)$ generated by ALGRAND and $G \sim A_HG(n,p,k)$ generated by the adversary, are statistically similar to each other.*

The proof of Theorem 10 appears in [6]. Here we explain the main ideas in the proof. A minimum requirement for the theorem to hold is that $G' \sim G(n,p)$ typically contains at least one copy of H (otherwise $A_HG(n,p,k)$ fails to produce any output). But this by itself does not suffice. Intuitively, the condition we need is that G' typically contains many copies of H. Then the fact that $G_H(n,p)$ of ALGRAND adds another copy of H to G' does not appear to make much of a difference to G', because G' anyway has many copies of H. Hopefully, this will imply that $G' \sim G(n,p)$ and $G_H \sim G_H(n,p)$ come from two distributions that are statistically close. This intuition is basically correct, though another ingredient (a concentration result) is also needed. Specifically, we need the following lemma (stated informally).

▶ **Lemma 11.** *For $G' \in G(n,p)$ (with p and H as above), the expected number of copies of H in G' is very high ($2^{n^{\eta}}$ for some $\eta > 0$ that depends on δ and ϵ). Moreover, with high probability, the actual number of copies of H in G' is very close to its expectation.*

The proof of Lemma 11 is based on known techniques (first and second moment methods). It uses in an essential way the fact that the graph H is sparse (average degree barely above 2) and does not have many vertices (these properties hold by definition of the class $\mathcal{H}$). Armed with Lemma 11, we then prove the following Lemma.

▶ **Lemma 12.** *The two distributions $G(n,p)$ and $G_H(n,p)$ are statistically similar to each other.*

Lemma 12 is proved by considering graphs $G' \sim G(n,p)$ that do contain a copy of H (Lemma 11 establishes that this is a typical case), and comparing for each such graph the probability of it being generated by $G_H(n,p)$ with the probability of it being generated by $G(n,p)$. Conveniently, the ratio between these probabilities is the same as the ratio between the actual number of copies of H in the given graph G', and the expected number of copies of H in a random $G' \sim G(n,p)$. By Lemma 11, for most graphs, this ratio is close to 1.

Theorem 10 follows quite easily from Lemma 12. Consequently ALG's performance on the distributions $G_H(n,p,k)$ and $A_HG(n,p,k)$ is similar. By our assumption, ALG finds (with high probability) an independent set of size k in $G \sim A_HG(n,p,k)$, which now implies that it also does so for $\tilde{G}_H \sim G_H(n,p,k)$. But as argued above, finding an independent set of size k in $\tilde{G}_H \sim G_H(n,p,k)$ implies that ALGRAND finds an independent set of size k' in $H \in \mathcal{H}$, thus solving an NP-hard problem. Hence the assumption that there is a polynomial time algorithm ALG that can find independent sets of size k in $G \sim A\bar{G}(n,p,k)$ implies that NP has randomized polynomial time algorithms.

3 Additional results

In the main part of the paper we only described what we view as our main results. The appendix contains all missing proofs, and some additional results and extensions, not described above. For example, one may ask for which value of $p \leq \frac{1}{2}$ the transition occurs from being able to find the maximum independent set in $G \sim A\bar{G}(n,p,k)$ in polynomial time, to the problem becoming NP hard. Our results show a gradual transition. For constant p the problem remains polynomial time solvable, and then, as p continues to decrease, the running time of our algorithms becomes super polynomial, and grows gradually towards exponential complexity. Establishing this type of behavior does not require new proof ideas, but rather only the substitution of different parameters in the existing proofs. Consequently, some theorems that were stated here only in special cases (e.g., Theorem 7 that was stated only for $p = \frac{1}{2}$) are restated in the appendix in a more general way (e.g., replacing $\frac{1}{2}$ by p), and a more general proof is provided.

Though this is not shown in the appendix, our hardness results (for finding adversarially planted independent sets) also imply a gradual transition, providing NP-hardness results when $p = n^{\delta-1}$, and as p grows (e.g., into the range $p = \frac{1}{(\log n)^c}$) the NP-hardness results are replaced by hardness results under stronger assumptions, such as (a randomized version of) the exponential time hypothesis. This is because for $p = \frac{1}{(\log n)^c}$ we need to limit the size of the graphs $H \in \mathcal{H}$ to be only polylogarithmic in n, as for larger sizes the proofs in Section 2.3 fail.

An interesting range of parameters that remains open is that of $p = \frac{d}{n}$ for some large constant d. The case of a random planted independent set of size $\sqrt{\frac{c}{d}}n$ (for some sufficiently large constant $c > 0$ independent of d) was addressed in [9]. In such sparse graphs, the planted independent set is unlikely to be the maximum independent set. The main result in [9] is a polynomial time algorithm that with high probability finds the maximum independent set in that range of parameters. It would be interesting to see whether the positive results extend to the case of adversarial planted independent set. We remark that neither Theorem 1 nor Theorem 3 apply in this range of parameters.

References

1 Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. *Random Struct. Algorithms*, 13(3-4):457–466, 1998.

2 Roee David and Uriel Feige. On the effect of randomness on planted 3-coloring models. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 77–90, 2016.

3 Yael Dekel, Ori Gurel-Gurevich, and Yuval Peres. Finding hidden cliques in linear time with high probability. *Combinatorics, Probability & Computing*, 23(1):29–49, 2014.

4 Yash Deshpande and Andrea Montanari. Finding hidden cliques of size $\sqrt{N/e}$ in nearly linear time. *Foundations of Computational Mathematics*, 15(4):1069–1128, 2015.

5 Uriel Feige. Introduction to semi-random models. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*. to appear, 2020. URL: `http://www.wisdom.weizmann.ac.il/~feige/mypapers/bwca.pdf`.

6 Uriel Feige and Vadim Grinberg. How to hide a clique?, 2020. `arXiv:2004.12258`.

7 Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms*, 16(2):195–208, 2000.

8 Uriel Feige and Robert Krauthgamer. The probable value of the Lovász–Schrijver relaxations for maximum independent set. *SIAM J. Comput.*, 32(2):345–370, 2003.

9 Uriel Feige and Eran Ofek. Finding a maximum independent set in a sparse random graph. *SIAM J. Discrete Math.*, 22(2):693–718, 2008.

10 Uriel Feige and Dorit Ron. Finding hidden cliques in linear time. In *21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10)*, pages 189–204, 2010.

11 Mark Jerrum. Large cliques elude the metropolis process. *Random Struct. Algorithms*, 3(4):347–360, 1992.

12 Ferenc Juhász. The asymptotic behaviour of Lovász' ϑ function for random graphs. *Combinatorica*, 2:153–155, 1982.

13 Ludek Kucera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57:193–212, 1995.

14 László Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, 25(1):1–7, 1979.

15 Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *SWAT*, pages 260–272, July 2004. `doi:10.1007/978-3-540-27810-8_23`.

16 Raghu Meka, Aaron Potechin, and Avi Wigderson. Sum-of-squares lower bounds for planted clique. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 87–96, 2015. `doi:10.1145/2746539.2746600`.