Finding and certifying a large hidden clique in a semi-random graph *

Uriel Feige and Robert Krauthgamer
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel
{feige,robi}@wisdom.weizmann.ac.il

November 11, 1999

Abstract

Alon, Krivelevich and Sudakov (Random Structures and Algorithms, 1998) designed an algorithm based on spectral techniques that almost surely finds a clique of size $\Omega(\sqrt{n})$ hidden in an otherwise random graph. We show that a different algorithm, based on the Lovász theta function, almost surely both finds the hidden clique and certifies its optimality. Our algorithm has an additional advantage of being more robust: it also works in a semi-random hidden clique model, in which an adversary can remove edges from the random portion of the graph.

1 Introduction

A *clique* in a graph G is a subset of the vertices every two of which are connected by an edge. The maximum clique problem, that is, finding a clique of maximum size in a graph, is fundamental in the area of combinatorial optimization, and is closely related to the independent set problem (clique on the edge complement graph $\overline{G}$), the vertex cover problem (the vertex complement of the independent set) and chromatic number problem (minimum cover by independent sets). The problem of finding the size of the maximum clique in G, called the *clique number*, $\omega(G)$, is one of the first problems shown to be NP-hard [17].

*This is a preprint of an article accepted for publication in Random Structures and Algorithms© (copyright owned as specified in the journal).

A common way to cope with NP-hardness of a problem is to devise algorithms which give approximate solutions. An algorithm is said to have an *approximation ratio* $\rho > 1$ for the clique problem if for every input graph the ratio between $\omega(G)$ and the size of the clique returned by the algorithm is at most $\rho = \rho(n)$ (throughout, n denotes the number of vertices in G). It is known through work culminating in [12] that for any fixed $\epsilon > 0$ it is impossible to approximate the clique number $\omega(G)$ within a ratio of $n^{1-\epsilon}$, unless NP has randomized polynomial time algorithms (NP=ZPP). The best approximation algorithm known for $\omega(G)$, by [5], has approximation ratio of $O(n/\log^2 n)$.

The intractability of the clique problem in the worst case suggests studying the performance of algorithms on average instances. A possible rigorous description of average instances is by probabilistic models (see [9] for a survey on using random graphs as models for average case analysis of several graph algorithms).

The problem of finding a maximum clique on a random graph appears to be difficult. Let $G_{n,1/2}$ denote the random graph of n labeled vertices obtained by choosing, randomly and independently, each pair of vertices to be an edge with probability 1/2. It is known that the clique number of $G_{n,1/2}$ is roughly $2\log_2 n$, *almost surely*, i.e. with probability that approaches 1 as n tends to infinity. Several simple and natural algorithms (e.g. the greedy one) find a clique of size roughly $\log_2 n$, almost surely. Karp [18] suggested the problem of finding a clique of size significantly larger than $\log_2 n$, but no efficient algorithm is known to achieve that. Finding cliques of size $\frac{3}{2}\log_2 n$ in random graphs was even suggested as a hard computational problem on which to base cryptographic applications, see [15].

Jerrum [14] and Kučera [20] suggested independently a “hidden clique” variant $G_{n,1/2,k}$ in which a random graph $G_{n,1/2}$ is chosen and then a clique of size k is randomly placed in the graph. Jerrum showed that the Metropolis process will not find the clique when $k = o(\sqrt{n})$. Kučera observed that when $k > c\sqrt{n \log n}$ for an appropriate constant c, the vertices of the planted clique would almost surely be the ones with the largest degrees in G, and hence it is easy to recognize them efficiently. Alon, Krivelevich and Sudakov [2] showed an algorithm based on spectral properties of the graph, for finding the planted clique, almost surely, when $k \geq \Omega(\sqrt{n})$.

A major motivation for studying various probabilistic input models in general is to evaluate algorithms performance in real-life applications. It would be encouraging if we could rigorously show that really difficult instances are very rare, and we are more likely to encounter in practice a “solvable” instance. However, it is difficult to establish a connection between probabilistic input models and instances that occur in practice. For example, random graph models are usually highly regular (all vertices are of roughly the same degree). While most graphs indeed have this property, real-life instances not necessarily do. It is therefore desirable to have an algorithm which is effective on a wider range of instances.

One approach to enrich the class of solvable instances is to consider a *semi-random* *model*, in which the input is generated by a mixture of random and adversarial choices. Blum and Spencer [4] introduced two variants of the semi-random model. In one variant, an adversary makes its choices for the graph edges, but each of these choices is flipped with some small probability ("noise"). In the other variant, a random graph is chosen first, and then an adversary can modify this graph subject to some restrictions. This last variant of the semi-random model was formalized by Feige and Kilian [8] as a *sandwich model.* First, two instances $G_{\min}$ and $G_{\max}$, both containing the same planted clique of size k, are generated using random decisions. Then, an adversary is allowed to choose any graph G^* which is sandwiched in between, i.e. $G_{\min} \subseteq G^* \subseteq G_{\max}$, where inclusion is with respect to edges.

The algorithm of [8] finds a clique of linear size ($k = \Omega(n)$), in the following sandwich setting. $G_{\min}$ is the empty graph except a clique of size k. $G_{\max}$ is the complete graph except for roughly $n \log n$ random missing edges chosen from those edges connecting the (same) clique and the rest of the graph. An adversary then chooses $G_{\min} \subseteq G^* \subseteq G_{\max}$, and thus has complete control over the edges which are not adjacent to the clique vertices, and large control over the edges connecting the clique to the rest of the graph. The algorithm of [8] uses semidefinite programming and matchings techniques and finds, almost surely, a clique of k vertices in the graph.

Since average case algorithms do not have an a priori guarantee on their performance, it is important to certify that the algorithm is indeed successful on the particular instance at hand. Boppana [6] shows an algorithm with such a certification property for the minimum bisection problem (see also [8, Appendix B]). The algorithm outputs a bisection together with a lower bound on the size of the optimal bisection. The analysis shows that the output bisection and lower bound are equal, almost surely, in which case the algorithm proves the optimality of its output bisection.

We present an algorithm based on the Lovász theta function for finding a clique of size $k \geq \Omega(\sqrt{n})$ planted in a random graph $G_{n,1/2}$. Our algorithm improves over the algorithm of [2] in two respects:

1. Extends to the sandwich model. An adversary may remove edges from the [2] graph except for the edges forming the clique of size k. In the sandwich model terminology , let $G_{\max} = G_{n,1/2,k}$ be the [2] graph, and let $G_{\min}$ be the empty graph except the same clique of size k. Then our algorithm finds the clique of size k in an arbitrary graph G^* sandwiched in between $G_{\min}$ and $G_{\max}$, almost surely over the distribution of $G_{\min}$ and $G_{\max}$. Observe that in this sandwich model, the vertices of the clique will not necessarily have higher degree, even if $k >> \sqrt{n \log n}$, so also the algorithm of Kučera would not work in this model.

2. Certifies optimality of its solution. Using a semidefinite programming relaxation of the clique problem, the algorithm provides an upper bound on the size of the maximum clique in the graph. The upper bound matches, almost surely, the

solution of the algorithm, proving that the clique output by the algorithm is the optimal one.

1.1 The sandwich model

The semi-random graph G^* on n labeled vertices is constructed by a combination of random and adversarial decisions, as follows.

1. Random graph: for any pair of vertices i, j the edge (i, j) is placed in the graph with probability $1/2$. This gives the random graph $G_{n,1/2}$.

2. Planted clique: a subset Q of k vertices is chosen at random. Let $G_{\min}$ be the empty graph except a clique on the vertices of Q. Let $G_{\max}$ be the random graph $G_{n,1/2}$ with a planted clique on the vertices of Q (i.e. an edge is forced between every pair of vertices in Q). $G_{\max}$ is the $G_{n,1/2,k}$ graph of [2].

3. Adversarial component: having complete knowledge of $G_{\max}$, an adversary may remove from the graph arbitrary edges except those which form the clique on Q (i.e. it is not allowed to remove edges both of whose endpoints are in Q). This gives the input graph G^* which is sandwiched in between $G_{\min}$ and $G_{\max}$.

Both graphs $G_{\min}, G_{\max}$ contain the same clique Q, and thus any sandwiched graph G^* must also have a clique on Q. An adversary can remove any of the edges of $G_{\max}$ outside the clique Q, so it has control over roughly half of all possible edges in the graph.

Observe that the algorithm is essentially required to output $G_{\min}$. The adversary is given $G_{\max}$ and may only remove edges, as long as G^* still contains $G_{\min}$. Thus, it may appear as if the adversarial moves only make the problem easier. Nevertheless, many algorithms that would recover a large clique in $G_{\max}$ would fail on G^*. A major motivation for the sandwich graph model is to identify those algorithms which are robust enough to withstand such an adversarial "help".

1.2 Our results

The performance of our algorithm on the sandwich model is summarized in the next theorem. Throughout, we say that an event occurs *almost surely* if its probability, over the distribution of $G_{\max}$, tends to 1 when $n \to \infty$. The adversarial component is, of course, always assumed to have the worst possible effect.

Theorem 1 *For any $k = \Omega(\sqrt{n})$, there is a polynomial time algorithm that, given a semi-random graph G^*, outputs, almost surely, a clique of size k together with a tight upper bound of k on the size of the largest clique in G^*.*

1.3 Techniques

Lovász [21] introduced a relaxation of the independent set problem, known as the Lovász *theta* function of a graph, usually denoted $\vartheta(G)$. The theta function can be approximated within arbitrary precision in polynomial time, using semidefinite programming (cf. [11]). From the equivalence between the independent set problem and the clique problem through the edge complement graph $\overline{G}$, it follows that $\vartheta(\overline{G})$ is an efficiently computable relaxation of the clique number $\omega(G)$.

In terms of approximation ratio, the theta function (and similar semidefinite programs) appear to have little to offer. The ratio between $\vartheta(\overline{G})$ and the clique number $\omega(G)$ can be as large as $n^{1-o(1)}$, as shown in [7]. Indeed, Hastad [12] shows that no polynomial time computable function approximates $\omega(G)$ within a ratio of $n^{1-\epsilon}$, unless NP has random polynomial time algorithms.

Also on the average there is a gap between the clique number $\omega(G)$ and the Lovász theta function $\vartheta(\overline{G})$. While the clique number of the random graph $G_{n,1/2}$ is roughly $2\log n$ (cf. [3]), the theta function gives, almost surely, $\Theta(\sqrt{n})$ [16].

It is not difficult to see that planting a clique of size k in a graph cannot increase the theta function by more than k, using monotonicity properties of the theta function, cf. [19, Sections 18-19]. Hence, the theta function of a random graph $G_{n,1/2}$, with a planted clique of size $k = o(\sqrt{n})$ is also $\Theta(\sqrt{n})$, and the planted clique probably does not have any noticeable effect on the theta function.

It is interesting to study what is the effect of planting a clique of size $k \geq \Omega(\sqrt{n})$ on the theta function. Note that this is exactly the graph $G_{\max}$ of our sandwich model. We answer this question in the following Lemma, which is also the key to the proof of Theorem 1.

Lemma 2 *Let $G = G_{n,1/2,k} = G_{\max}$, where $k > c'\sqrt{n}$ for a large enough constant c'. Then $\vartheta(\overline{G}) = k$, with extremely high probability.*

Throughout the paper, the term *with extremely high probability* will be used to denote a probability $1 - e^{-n^r}$ for some constant $r > 0$.

Organization. Section 2 shows a proof of Theorem 1, based on Lemma 2. When $k > c\sqrt{n}$ and $c \geq c'$ (where c' is as in Lemma 2), a direct application of Lemma 2 gives an algorithm which recognizes, almost surely, the vertices of the planted clique in $G_{\max}$. We use an idea from [2] to extend this algorithm to cases where $c < c'$. It will follow quite easily that our algorithm has two additional performance guarantees, which are the robustness against the sandwich model adversary and a certificate (almost surely) for the optimality of its solution.

The proof of Lemma 2 is deferred to Section 3. There we use a formulation of the theta function as an eigenvalue minimization problem, which can be interpreted as duality of semidefinite programming, cf. [1].

2 The Algorithm

In this section we prove Theorem 1 and describe its algorithm, extensively relying on the result of Lemma 2. We shall start with the strictly random model, in which the input graph is $G_{\max}$ and there is no adversary. First (Section 2.1) we address the basic case of $k > c'\sqrt{n}$ for a large enough constant c', where we can easily use Lemma 2. Then (Section 2.2) we improve the result to any $k = \Omega(\sqrt{n})$. We use the approach of [2] of guessing a fixed number of vertices from the planted clique, in order to reduce the problem to the basic case. Finally (Section 2.3), we show how our analysis easily extends to the semi-random model with the exact same algorithm.

The monotonicity properties of the theta function, cf. [19, Sections 18-19], are used throughout this section. Specifically, addition (or removal) of an edge (or a vertex) from a graph has a monotone effect on the theta function, similarly to the independence number $\alpha(G)$. For example, adding (resp. removing) an edge may only decrease (resp. increase) the theta function.

Observe that in all cases Q is almost surely the unique maximum clique. Indeed, any clique containing a vertex from $V \setminus Q$, almost surely contains at most $(1+o(1))k/2$ vertices from Q, and at most $2\log n$ vertices from $V \setminus Q$, which is altogether much smaller that $|Q|$.

2.1 The basic case

In the basic case we assume that the input graph is $G_{\max} = G_{n,1/2,k}$ (so there is no adversary), and assume also that $k > c'\sqrt{n}$ for a large enough constant c'.

Finding the clique vertices can then be performed by testing separately each vertex v to see whether it belongs to the planted clique or not. To test if v belongs to the planted clique, remove v from the graph $G_{\max}$ to get the graph $G_{\max} \setminus v$, and check how this removal affects the theta function. We can analyze $G_{\max} \setminus v$ by using the principle of deferred decisions. If the vertex v belongs to the planted clique, then $G_{\max} \setminus v$ has a clique of size $k-1$ in an otherwise random graph. If v does not belong to the planted clique, then $G_{\max} \setminus v$ has a clique of size k in an otherwise random graph.

Applying Lemma 2 on $G_{\max}$ and on $G_{\max} \setminus v$ for all vertices v, and using the union bound we get the following observation.

Observation. With extremely high probability, $\vartheta(\overline{G_{\max}}) = k$ and

$$\vartheta(\overline{G_{\max} \setminus v}) = \begin{cases} k-1 & \text{if } v \text{ belongs to the planted clique;} \\ k & \text{otherwise;} \end{cases}$$

This suggests the following simple algorithm for the basic case:

Algorithm **BasicFind.**
Input: Graph $G = G_{\max}$ where $k > c'\sqrt{n}$ for a large enough constant c'.

1. $P \leftarrow \{v : \vartheta(\overline{G \setminus v}) < \vartheta(\overline{G}) - 1/2\}$
2. Output P and $\vartheta(\overline{G})$.

The discussion above shows that with extremely high probability, the output P of algorithm **BasicFind** is the planted clique Q of size k, and $\vartheta(\overline{G}) = |P|$, proving its optimality.

Improved Algorithm

The same observation leads to a more efficient algorithm, which uses only one computation of the theta function. The theta function has several equivalent formulations, cf. [21, 11, 19], but we will use a particular geometric maximization form, described as ϑ_4 in [11, Chapter 9.3], as follows. An *orthonormal representation* of G is a sequence of unit length vectors $\{u_i \in \mathbb{R}^n : i \in V\}$, such that $u_i \cdot u_j = 0$ whenever i, j are non-adjacent vertices. Then

$$\vartheta_4(G) = \max_{d,\{u_i\}} \sum_{i \in V} (d \cdot u_i)^2$$

where $d \in \mathbb{R}^n$ ranges over all vectors of unit length, and $\{u_i \in \mathbb{R}^n : i \in V\}$ is an orthonormal representation of $\overline{G}$.

Using semidefinite programming, it is possible to solve $\vartheta_4(G)$ within arbitrary small additive error, and arrive at a corresponding vector d and an orthonormal representation $\{u_i\}$. (More precisely, formulation $\vartheta_3(G)$ describes a semidefinite program, whose solution can be efficiently transformed to an equivalent orthonormal representation for $\vartheta_4(G)$. See the above references for details).

Suppose now that we solve $\vartheta_4(G)$ for our input graph $G = \overline{G_{\max}}$, and we get a solution $d, \{u_i\}$. We claim that the k vertices whose contribution $(d \cdot u_i)^2$ is the largest, are the vertices of the clique, with extremely high probability. This, of course, will enable us to recognize the vertices of the planted clique with only one such computation of the theta function.

To prove the claim, assume that the result of the above observation regarding $\vartheta(\overline{G_{\max}})$ and $\vartheta(\overline{G_{\max} \setminus v})$ indeed holds (which happens with extremely high probability). Then $\vartheta(\overline{G_{\max}}) = k$, and thus the solution $d, \{u_i\}$ we get from the semidefinite programming has a value of at least $k - \epsilon$ (for arbitrary small fixed $\epsilon > 0$). In this solution, the contribution of every vertex of the planted clique, $q \in Q$, is at least $1 - \epsilon$, or otherwise, the same orthonormal representation would give that $\vartheta_4(\overline{G_{\max} \setminus q}) \geq \sum_{i \neq q} (d \cdot u_i)^2 > k - 1$, contradicting the above observation. However, the contribution of any vertex $j \in V \setminus Q$ is bounded, because j is (with extremely high probability) a neighbor of some $q \in Q$, so $u_j \cdot u_q = 0$ and therefore $(d \cdot u_j)^2 + (d \cdot u_q)^2 \leq 1$, and $(d \cdot u_j)^2 \leq \epsilon$. This shows that the contribution of every vertex $q \in Q$ is larger than that of vertex $j \in V \setminus Q$, as claimed.

As an estimate for $k = |Q|$ we can take the value we get from the semidefinite programming for $\vartheta_4(\overline{G_{\max}})$, rounded to the nearest integer. This gives the following algorithm, which outputs $P = Q$ and $\vartheta = k$ with extremely high probability.

Algorithm **ImprovedBasicFind.**
Input: Graph $G = G_{\max}$ where $k > c'\sqrt{n}$ for a large enough constant c'.

1. Compute $\vartheta \leftarrow \vartheta_4(\overline{G})$ within small additive error $\epsilon = 1/3$, together with a corresponding orthonormal representation $\{u_i\}$ and its handle d.
2. $P \leftarrow \{i : (d \cdot u_i)^2 > 1/2\}$.
3. Output P and ϑ.

2.2 Smaller values of k

Following [2], the main idea in improving the algorithm for $G_{\max}$ to any $k = \Omega(\sqrt{n})$ is to guess a constant number of vertices from the planted clique. We can then work on the subgraph induced on those vertices which are common neighbors to all of our guess set. The induced subgraph is much smaller the $G_{\max}$, and is random except for the planted clique of size k. Thus we improve the ratio between the size of the clique and the size of the graph by a constant factor, and can use the algorithm of the basic case (either **BasicFind** or **ImprovedBasicFind** of Section 2.1).

Guessing a constant number of vertices from the graph can be replaced by an exhaustive search on a polynomial number of possibilities. Let $\hat{N}(S)$ denote the set of vertices neighboring to all of S. The algorithm for $k \geq c\sqrt{n}$ with arbitrary fixed $c > 0$ is as follows.

Algorithm **FindClique.**
Input: Graph $G = G_{\max}$ where $k \geq c\sqrt{n}$.

1. $s \leftarrow 2\lceil \log_2(c'/c) \rceil + 3$
2. For all subsets S of s vertices, do
   - 2.1. $V_1 \leftarrow \hat{N}(S)$.
   - 2.2. $(P_S, \vartheta_S) \leftarrow$ **BasicFind**(the subgraph induced on V_1)
   - 2.3. End-for.
3. Output the set $S \cup P_S$ which is a clique in G and has maximum size over all choices of S.
4. Output $\vartheta = s + \max_S \vartheta_S$.

We claim that for any fixed c, algorithm **FindClique** almost surely outputs the planted clique Q and a tight upper bound $\vartheta = k$. First observe that for any fixed subset S of size s, the cardinality of $\hat{N}(S)$ in the random graph $G_{n,1/2}$ is a binomially distributed random variable with parameters $n - s$ and $1/2^s$. Thus, almost surely, $|\hat{N}(S)| = (1 + o(1))n/2^s$ for all subsets of vertices of size s in $G_{n,1/2}$. Planting a clique of size k can increase $|\hat{N}(S)|$ by at most k. Therefore, for all S, $|\hat{N}(S)| = (1+o(1))n/2^s$ almost surely also in $G_{\max}$.

Since algorithm **FindClique** checks all possible subsets S of size s, in some step it will reach some S which is a subset of the planted clique Q. At this iteration we almost surely find the planted clique $Q = S \cup P_S$. Indeed, by the principle of deferred decisions, the subgraph induced on $\hat{N}(S)$ is a random graph $G_{|\hat{N}(S)|,1/2}$ with a planted clique of size $k - s$. By our choice of s, the planted clique size satisfies $k - s \geq c\sqrt{n}/2 \geq c'\sqrt{2n/2^s} \geq c'\sqrt{|\hat{N}(S)|}$. Thus, algorithm **BasicFind** will almost surely find the planted clique $Q \setminus S$ in $\hat{N}(S)$, and as a result we will find the planted clique of size k in $G_{\max}$.

With extremely high probability, for all subsets S of size s checked by the algorithm **FindClique**, the theta function of the subgraph induced on $\hat{N}(S)$ is at most $k-s$, i.e. $\vartheta_S \leq k - s$ for all S. Indeed, for each subset S, either $S \subset Q$ or S contains a vertex not from Q. In the first case the subgraph induced on $\hat{N}(S)$ is a random graph with a planted clique of size at most $k-s$. In the second case, with extremely high probability $\hat{N}(S)$ contains only $(1 + o(1))k/2 < k - s$ vertices of the planted clique Q. In either case, using the monotonicity properties, we may assume $\hat{N}(S)$ contains exactly $k - s$ vertices of the planted clique Q, and by Lemma 2, with extremely high probability the theta function of the induced subgraph is at most $k - s$. Using the union bound on polynomially many choices of S, the event that all these theta functions will be at most $k - s$ takes place with extremely high probability.

2.3 A sandwiched graph G^*

To show that algorithm **FindClique** is robust enough to withstand a monotone adversary of the sandwich model, we argue that an adversary cannot prevent the algorithm from succeeding. Thus, the proof for the strictly random model $G_{\max}$ extends to the semi-random model G^*.

For simplicity, consider first the basic case where $k > c'\sqrt{n}$ for a large enough constant c'. Let G^* be an arbitrary sandwiched graph, i.e. $G_{\min} \subset G^* \subset G_{\max}$. Then by the monotonicity properties of the theta function,

$$k \leq \vartheta(\overline{G_{\min}}) \leq \vartheta(\overline{G^*}) \leq \vartheta(\overline{G_{\max}}) \leq k$$

where the last inequality holds, almost surely, by our previous analysis. A similar phenomenon happens when we remove one vertex, i.e. also in $\vartheta(\overline{G_{\max} \setminus v})$, as in the observation of Section 2.1. This shows that an adversary cannot prevent the algorithm from finding the planted clique nor from certifying the optimality of its solution.

In the general case where $k \geq \Omega(\sqrt{n})$, we consider the possible effect of an adversary in any of the applications of the theta function in algorithm **FindClique**. These applications are equivalent to the removals of one vertex in the observation of Section 2.1 (which are used in the proof of algorithm **ImprovedFindClique**).

The algorithm **FindClique** only applies the theta function on induced subgraphs of the input graph G^*. Let H be any such induced subgraph of the corresponding $G_{\max}$. It follows from the principle of deferred decisions, that the induced subgraph H is a random graph (with edge probability 1/2) with a randomly planted clique of size $k' \leq k$. Consider the effect of an adversary on this application of the theta function, i.e. on $\vartheta(\overline{H})$. Recall that an adversary is only allowed to remove edges. Thus, its effect is limited to either removing some vertices from H (by reducing $\hat{N}(S)$), or removing some edges from H (by removing the same edges from $G_{\max}$).

On the one hand, these operations may only decrease the theta function, $\vartheta(\overline{H})$, by the aforementioned monotonicity properties. On the other hand, whenever the theta function is tight on H due to its planted clique, (i.e. $\vartheta(\overline{H}) = k'$), an adversary cannot affect the theta function because it cannot remove any of the edges forming the planted clique, and H must still contain a clique of size k'. Note that these are exactly the properties used in the analysis of algorithm **FindClique**.

Hence, an adversary cannot prevent the algorithm from finding the planted clique of size k, and cannot increase the upper bound ϑ, which thus must remain k. Overall, whenever algorithm **FindClique** succeeds on $G_{\max}$, it also succeeds on an arbitrary sandwiched G^*, regardless of the adversary operations, as claimed.

3 Main Technical Result

In this section we prove Lemma 2. Throughout this section, let us denote by G the graph $\overline{G_{\max}}$, which can be equivalently described as a random graph $G_{n,1/2}$ with a planted independent set on k randomly chosen vertices. Since G has a (planted) independent set of size k and the theta function is a relaxation of the independent set problem, $\vartheta(G) \geq \alpha(G) \geq k$.

The main part is to show the other direction, i.e. that with extremely high probability $\vartheta(G) \leq k$. The theta function has several equivalent formulations (cf. [21, 11, 19]. We will use the formulation as an eigenvalue minimization problem:

$$\vartheta_2(G) = \min_M \{\lambda_1(M)\}$$

where M is an $n \times n$ real symmetric matrix with $M_{ij} = 1$ whenever vertices i, j are non-adjacent in G, and $\lambda_i(M)$ denotes the i-th largest eigenvalue of the matrix M. We derive an upper bound on $\vartheta(G)$ by "guessing" a particular matrix M, and clearly

$$\vartheta_2(G) \leq \lambda_1(M) \tag{1}$$

So it suffices to show that with extremely high probability, our choice of M is such that $\lambda_1(M) \leq k$.

Assume wlog that the vertices of the planted independent set are the first k coordinates. Then our chosen M looks like

$$M = \left[\begin{array}{c|c} J_k & B^t \\ \hline B & C \end{array}\right] = \left[\begin{array}{c|cccc} J_k & & 1/-1{+}x_j & & \\ \hline & 1 & & & 1/-1 \\ & & 1 & & \\ 1/-1{+}x_i & & & \ddots & \\ & 1/-1 & & & 1 \end{array}\right]$$

where J_k is the all ones matrix of order k, B^t is the transpose of B, and B, C are as follows. Roughly half of the entries of B and C have to be $+1$ because they correspond to pairs of non-adjacent vertices in the graph G. We set all other entries of C to be -1. The remaining entries of B are chosen so that the sum of each row of B is 0. At each row, we choose all entries to be equal. Formally, we define the entries of C to be $c_{ij} = 1$ if $(i,j) \notin E$ and $c_{ij} = -1$ otherwise (for all $k < i, j \leq n$). B is defined by $b_{ij} = 1$ if $(i,j) \notin E$ and $b_{ij} = -1 + x_i$ otherwise (for all $i > k$ and $j \leq k$), where $x_i = (2deg(x_i, Q) - k)/deg(x_i, Q)$, and $deg(x_i, Q)$ denotes the number of vertices in Q which are adjacent to x_i in the graph.

It is easy to see that k is an eigenvalue of M. By our choice of B the vector with 1 in its first k entries and 0 otherwise is an eigenvector whose eigenvalue is k. In order to prove that this vector corresponds to the largest eigenvalue, i.e. $\lambda_1(M) = k$, it suffices to show that $\lambda_2(M) < k$.

Let us describe M as the sum of three matrices $M = U + V + W$. the three matrices are random but correlated, as follows.

U is a random 1/-1 symmetric matrix, with 1 on the diagonal.

V is a random 0/2 symmetric matrix in the upper left $k \times k$ block (with 0 in the diagonal), and 0 otherwise. The matrix V is correlated with U by $v_{ij} = 1 - uij$ for all $1 \leq i, j \leq k$.

W is the correction matrix for having the row sums of B equal to 0.

$$U = \begin{bmatrix} 1 & & & 1/-1 \\ & 1 & & \\ & & \ddots & \\ 1/-1 & & & 1 \end{bmatrix} \quad V = \left[\begin{array}{c|c} 0/2 & 0 \\ \hline 0 & 0 \end{array}\right] \quad W = \left[\begin{array}{c|c} 0 & 0/x_j \\ \hline 0/x_i & 0 \end{array}\right]$$

From [10] we know that with probability $1 - e^{-n^r}$ for some constant $r > 0$:

$$\forall i \geq 1, |\lambda_i(U)| \leq c_1\sqrt{n} \qquad \forall i \geq 2, |\lambda_i(V)| \leq c_2\sqrt{k}$$

By the Weyl theorem (cf. [13, page 181]),

$$\lambda_2(M) \leq \lambda_1(U) + \lambda_2(V) + \lambda_1(W) \leq c_1\sqrt{n} + c_2\sqrt{k} + \lambda_1(W) \tag{2}$$

To bound $\lambda_1(W)$, it suffices to bound $Tr(W^2)$ because

$$Tr(W^2) = \sum_i \lambda_i(W^2) = \sum_i (\lambda_i(W))^2 \geq (\lambda_1(W))^2 \tag{3}$$

Since W is symmetric $Tr(W^2) = \sum_i (W_i W_i^t) = \sum_{i,j} W_{ij}^2 = 2\sum_{i<j} W_{ij}^2$. Look at the i-th row of B and the corresponding row of W. Let us denote by S_i (for $k < i \leq n$) the number of non-zero (i.e. x_i) entries in this row of W, i.e. $S_i = deg(i, Q)$. Then $Tr(W^2) = 2\sum_{i=k+1}^{n} S_i x_i^2$. Recall x_i was chosen so that the corresponding row sum in B would be zero, so

$$x_i = (2S_i - k)/S_i$$

Since $S_i = deg(i, Q)$ is binomially distributed $S_i \sim B(k, 1/2)$, then $k = 2ES_i$ and we get

$$Tr(W^2) = 2\sum_{i=k+1}^{n} S_i((2S_i - k)/S_i)^2 = 8\sum_{i=k+1}^{n} (S_i - ES_i)^2/S_i \tag{4}$$

The following Lemma allows us to bound these quantities.

Lemma 3 *With extremely high probability (at least $1 - e^{-n^r}$ for some constant $r > 0$)*

1. $\sum_{i=k+1}^{n}(S_i - ES_i)^2 \leq k^3/96$.

2. $S_i \geq k/3$.

Proof.

1. We shall bound $Y = \sum_{i=k+1}^{n}(S_i - ES_i)^2$ by Azuma's inequality, cf. [3]. Let us first compute its expectation:

$$E(S_i - ES_i)^2 = Var(S_i) = k/4$$

$$EY = (n-k)k/4 \leq nk/4$$

Let $Y_0, Y_1, Y_2, \ldots, Y_{k(n-k)} = Y$ be a Doob martingale of Y defined by exposing one by one each of the $k(n-k)$ Bernoulli trials (recall $S_i \sim B(k, 1/2)$). Let us now bound the Martingale difference. To see how an exposure of a single Bernoulli trial can affect the final result Y, assume that all other Bernoulli trials have been fixed. The sum $Y = \sum_i (S_i - ES_i)^2$ is then fixed except for the contribution of a single S_i (which includes the yet unexposed Bernoulli trial). When that trial is exposed, the contribution of the corresponding S_i will be either $(v - ES_i)^2$ or $(v + 1 - ES_i)^2$, for some value $0 \leq v \leq k/2 - 1$. The difference between the two values is maximized when $v = 0$ or $v = k/2 - 1$. The Martingale difference is thus bounded by $\Delta = (\frac{k}{2})^2 - (\frac{k}{2} - 1)^2 = k - 1$. By Azuma's inequality, for any $\lambda > 0$

$$Pr\left[|Y_{k(n-k)} - EY_{k(n-k)}| \geq \lambda\Delta\sqrt{k(n-k)}\right] \leq 2e^{-\lambda^2/2}$$

Take $\lambda = n^{1/4}$ to get

$$Pr\left[|Y - EY| \geq n^{1/4}k\sqrt{k(n-k)}\right] \leq e^{-\sqrt{n}/4}$$

Since $EY \leq nk/4$ and $k > c'\sqrt{n}$ for a sufficiently large constant $c' > 0$ we conclude that with extremely high probability, $Y \leq EY + n^{3/4}k^{3/2} \leq k^3/4c'^2 + k^3/c'^{3/2} \leq k^3/96$, as claimed.

2. Follows immediately from Chernoff bound:

$$Pr[S_i \leq k/3] = Pr[S_i \leq 2/3 \cdot ES_i] \leq e^{-k/36} \leq e^{-c'\sqrt{n}/36}$$

□

Using (3),(4) and Lemma 3 we get that with extremely high probability

$$(\lambda_1(W))^2 \leq Tr(W^2) < 8 \cdot k^3/96 \cdot (3/k) = k^2/4$$

and using (1),(2) we arrive at

$$\lambda_2(M) \leq \lambda_1(U) + \lambda_2(V) + \lambda_1(W) < c_1\sqrt{n} + c_2\sqrt{k} + k/2 < k$$

as claimed.

3.1 Concluding remarks

The approach presented in this paper can be generalized to find a hidden clique in a random graph $G_{n,p}$ where the edge probability p is different than $1/2$.

For every fixed p, and thus fixed $q = 1 - p$, the main result holds, except, maybe, for a change in the (polynomial) running time of the algorithm. That is, a hidden clique of size $c\sqrt{n}$ can be found, for every fixed $c > 0$. Indeed, it is known from [16] that with extremely high probability $\vartheta(\overline{G_{n,p}}) = \vartheta(G_{n,q}) = \Theta(\sqrt{np/q})$. To find in a $G_{n,p}$ graph a hidden clique of size $k \geq c'\sqrt{np/q}$, for a sufficiently large constant $c' > 0$ (and thus extend Lemma 2), one can take the matrix M of our proof in the spirit of [16]. To finish the argument, observe that the idea from [2] of guessing a constant number of vertices allows handling a hidden clique of size $c\sqrt{n}$ for arbitrary fixed $c > 0$, and that the sandwich properties of the theta function give robustness against a monotone adversary.

When $p = o(1)$ finding hidden cliques becomes easier. For example, if $p = 1/n^\delta$ for a fixed $\delta > 0$, it is possible to find a hidden clique of size $O(1/\delta) << n^{(1-\delta)/2} \simeq \sqrt{np/q}$ by exhaustively trying all subsets of this size in the graph. Indeed, with high probability the maximum clique in the random graph is of size $O(1/\delta)$, and can thus be found in polynomial time by exhaustive search. If a clique of size c'/δ is planted in the graph, for a sufficiently large constant $c' > 0$, then with high probability it will be the unique maximum clique in the graph, and can similarly be found in polynomial time.

When $q = o(1)$ finding a hidden clique becomes more difficult. For example, the idea of [2] of guessing a constant number of vertices in the hidden clique has only a negligible effect in reducing the size of the graph. Nevertheless, we believe that our analysis of the algorithm based on the theta function can be extended to work for a large range of values of $q = o(1)$, finding cliques of size $c'\sqrt{n/q}$ for sufficiently large c'. We remark that the case of extremely small q, namely $q = \frac{c \log n}{n}$ for a sufficiently large $c > 0$, was handled in [8] (in a model that is more adversarial than the one studied here), where it was shown how to find hidden cliques of linear size $k = \Omega(n)$. Note that this value of q is larger by a factor of $\log n$ then the one which a general bound of $c'\sqrt{n/q}$ would have required for a linear sized clique. It appears to us that the loss of the $\log n$ factor is unavoidable for the semirandom graph model when q is so small (for reasons that are explained in [8]), but can perhaps be avoided in a random graph model that involves no adversary.

Acknowledgements

This research was supported in part by a Minerva grant (Project 8354). The first author is the incumbent of the Joseph and Celia Reskin Career Development Chair. The work of the second author was supported in part by a Dora Ostre memorial scholarship.

References

[1] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

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

[3] Noga Alon and Joel H. Spencer. *The probabilistic method*. John Wiley & Sons, Inc., New York, 1992.

[4] Avrim Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, September 1995.

[5] Ravi Boppana and Magnús M. Halldórsson. Approximating Maximum Independent Sets by Excluding Subgraphs. *BIT*, 32:180–196, 1992.

[6] Ravi B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *28th Annual Symposium on Foundations of Computer Science*, pages 280–285, Los Angeles, California, 12–14 October 1987. IEEE.

[7] Uriel Feige. Randomized graph products, chromatic numbers, and the Lovász ϑ-function. *Combinatorica*, 17(1):79–90, 1997.

[8] Uriel Feige and Joe Kilian. Heuristics for finding large independent sets, with applications to coloring semi-random graphs. In *39th Annual Symposium on Foundations of Computer Science*, Palo Alto, California, November 1998. IEEE.

[9] Alan Frieze and Colin McDiarmid. Algorithmic theory of random graphs. *Random Structures Algorithms*, 10(1-2):5–42, 1997.

[10] Z. Füredi and J. Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3):233–241, 1981.

[11] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization.* Springer-Verlag, Berlin, second edition, 1993.

[12] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *37th Annual Symposium on Foundations of Computer Science*, pages 627–636, Burlington, Vermont, 14–16 October 1996. IEEE.

[13] Roger A. Horn and Charles R. Johnson. *Matrix analysis.* Cambridge University Press, Cambridge-New York, 1985.

[14] Mark Jerrum. Large cliques elude the Metropolis process. *Random Structures Algorithms*, 3(4):347–359, 1992.

[15] Ari Juels and Marcus Peinado. Hiding cliques for cryptographic security. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998)*, pages 678–684. ACM, New York, 1998.

[16] Ferenc Juhász. The asymptotic behaviour of Lovász' θ function for random graphs. *Combinatorica*, 2(2):153–155, 1982.

[17] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations (Proceedings of a Symposium on the Complexity of Computer Computations, March, 1972, Yorktown Heights, NY)*, pages 85–103. Plenum Press, New York, 1972.

[18] Richard M. Karp. The probabilistic analysis of some combinatorial search algorithms. In *Algorithms and complexity (Proc. Sympos., Carnegie-Mellon Univ., Pittsburgh, Pa., 1976)*, pages 1–19. Academic Press, New York, 1976.

[19] Donald E. Knuth. The sandwich theorem. *Electron. J. Combin.*, 1:Article 1, approx. 48 pp. (electronic), 1994.

[20] Luděk Kučera. Expected complexity of graph partitioning problems. *Discrete Appl. Math.*, 57(2-3):193–212, 1995.

[21] László Lovász. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory*, 25(1):1–7, 1979.