Electronic Colloquium on Computational Complexity, Report No. 50 (2005)

Finding a Maximum Independent Set in a Sparse Random Graph

Uriel Feige and Eran Ofek

Department of Computer Science and Applied Mathematics
Weizmann Institute, Rehovot 76100, Israel
{uriel.feige,eran.ofek}@weizmann.ac.il

April 15, 2005

Abstract

We consider the problem of finding a maximum independent set in a random graph. The random graph G is modelled as follows. Every edge is included independently with probability $\frac{d}{n}$, where d is some sufficiently large constant. Thereafter, for some constant α, a subset I of αn vertices is chosen at random, and all edges within this subset are removed. In this model, the planted independent set I is a good approximation for the maximum independent set I_{max}, but both $I \setminus I_{max}$ and $I_{max} \setminus I$ are likely to be nonempty. We present a polynomial time algorithms that with high probability (over the random choice of random graph G, and without being given the planted independent set I) finds the maximum independent set in G when $\alpha \geq \sqrt{\frac{c_0 \log d}{d}}$, where c_0 is some sufficiently large constant independent of d.

ISSN 1433-8092

1 Introduction

Let $G = (V, E)$ be a graph. An independent set I is a subset of vertices which contains no edges. The problem of finding a maximum size independent set in a graph is a fundamental problem in Computer Science and it was among the first problems shown to be NP-hard [13]. Moreover, Hastad shows [11] that for any $\epsilon > 0$ there is no $n^{1-\epsilon}$ approximation algorithm for the maximum independent set problem unless NP=ZPP. The best approximation ratio currently known for maximum independent set [5] is $O(n(\log\log n)^2/(\log n)^3)$.

In light of the above mentioned negative results, we may try to design a heuristic which performs well on typical instances. Karp [14] proposed trying to find a maximum independent set in a random graph. However, even this problem appears to be beyond the capabilities of current algorithms. For example let $G_{n,1/2}$ denote the random graph on n vertices obtained by choosing randomly and independently each possible edge with probability $1/2$. A random $G_{n,1/2}$ graph has almost surely maximum independent set of size $2(1 + o(1)) \log_2 n$. A simple greedy algorithm almost surely finds an independent set of size $\log_2 n$ [10]. However, there is no known polynomial time algorithm which almost surely finds an independent set of size $(1 + \epsilon) \log_2 n$ (for any $\epsilon > 0$).

To further simplify the problem, Jerrum [12] and Kucera [15] proposed a *planted model* $G_{n,1/2,k}$ in which a random graph $G_{n,1/2}$ is chosen and then a clique of size k is randomly placed in the graph. (A clique in a graph G is an independent set in the edge complement of G, and hence all algorithmic results that apply to one of the problems apply to the other.) Alon Krivelevich and Sudakov [2] gave an algorithm based on spectral techniques that almost surely finds the the planted clique for $k = \Omega(\sqrt{n})$. More generally, one may extend the range of parameters of the above model by planting an independent set in $G_{n,p}$, where p need not be equal to $1/2$, and may also depend on n. The $G_{n,p,\alpha}$ model is as follows: n vertices are partitioned at random into two sets of vertices, I of size αn and C of size $(1 - \alpha)n$. No edges are placed within the set I, thus making it an independent set. Every other possible edge (with at least one endpoint not in I) is added independently at random with probability p. The goal of the algorithm, given the input G (but without being given the partition into I and C) is to find a maximum independent set. Intuitively, as α becomes smaller the size of the planted independent is closer to the probable size of the maximum independent set in $G_{n,p}$ and the problem becomes harder.

We consider values of p as small as d/n where d is a large enough constant. A difficulty which arises in this sparse regime (e.g. when d is constant) is that the planted independent set I is not likely to be a maximum independent set. Moreover, with high probability I is not contained in a maximum independent set of G. For example, there are expected to be $e^{-d}n$ vertices in C of degree one. It is very likely that two (or more) such vertices $v, w \in C$ will have the same neighbor, and that it will be some vertex $u \in I$. This implies that every maximum independent set will contain v, w and not u, and thus I contains vertices that are not contained in the maximum independent set.

A similar argument shows that there are expected to be $e^{-\Omega(d)}n$ isolated edges. This implies that there will be an exponential number of maximum independent sets.

1.1 Our result

We give a polynomial time algorithm that searches for a maximum independent set of G. Given a random instance of $G_{n,\frac{d}{n},\alpha}$, the algorithm almost surely succeeds, when $d > d_0$ and $\alpha \geq \sqrt{c_0 \log d/d}$ (d_0, c_0 are some universal constants). The parameter d can be also an arbitrary increasing function of n.

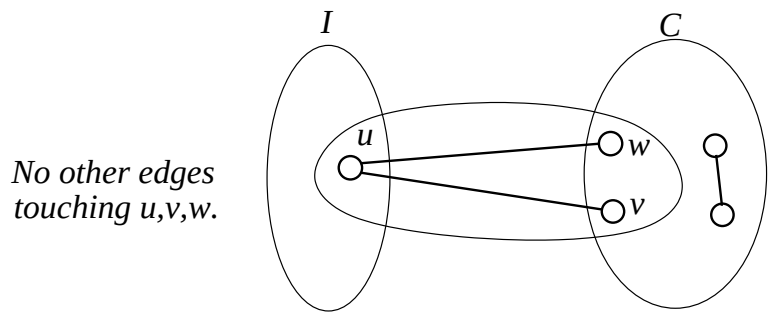

Figure 1: I is not contained in a maximum independent set.

1.2 Related work

For $p = 1/2$, Alon Krivelevich and Sudakov [2] gave an efficient spectral algorithm which almost surely finds the planted independent set when $\alpha = \Omega(1/\sqrt{n})$. For the above mentioned parameters, the planted independent set is almost surely the unique maximum independent set.

A few papers deal with *semi-random* models which extend the planted model by enabling a mixture of random and adversarial decisions. Feige and Kilian [6] considered the following model: a random $G_{n,p,\alpha}$ graph is chosen, then an adversary may add arbitrarily many edges between I and C, and make arbitrary changes (adding or removing edges) inside C. For any constant $\alpha > 0$ they give a heuristic that almost surely outputs a list of independent sets containing the planted independent set, whenever $p > (1+\epsilon)\ln n/\alpha n$ (for any $\epsilon > 0$). The planted independent set may not be the only independent set of size αn since the adversary has full control on the edges inside C. Possibly, this makes the task of finding the planted independent set harder.

In [7] Feige and Krauthgamer considered a less adversarial semi-random model in which an adversary is allowed to add edges to a random $G_{n,\frac{1}{2},\frac{1}{\sqrt{n}}}$ graph. Their algorithm almost surely extracts the planted independent set and certifies its optimality.

Heuristics for optimization problems different than max independent set will be discussed in the following section.

1.2.1 Technique and outline of the algorithm

Our algorithm builds on ideas from the algorithm of Alon and Kahale [1], which was used for recovering a planted 3-coloring in a random graph. The algorithm we propose has the following 3 phases:

1. Get an approximation of I, C denoted by I', C', OUT, where I' is an independent set. The error term $|C \triangle C'| + |I \triangle I'|$ should be at most $e^{-c \log^d n}$ where c is a big enough universal constant (this phase is analogous to the first two phases of [1]).

2. Remove vertices of I', C' which have non typical degrees to OUT.

   We stop when I', C' become *promising*: every vertex of C' has at least 4 edges to I' and no vertex of I' has edges to OUT. At this point we have a promising partial solution I', C' and the error term (with respect to I, C) is still small. Using the fact that random graphs (almost surely) have no small dense sets, it can be shown that I' is *extendable*: $I' \subseteq I_{max}$ for some optimal solution I_{max}.

3. Extend the independent set I' optimally using the vertices of OUT. This is done by finding a maximum independent set among the vertices of OUT and adding it to I'.

   Almost surely the structure of OUT will be easy enough so that maximum independent sets of its subgraphs can be efficiently found (OUT is a random graph of size $n/poly(d)$ with each edge chosen with probability d/n). Notice however, that the set OUT depends on the graph itself thus we can not argue that it is a random $G_{\frac{n}{poly(d)},\frac{d}{n}}$ graph.

The technique of [1] was implemented successfully on various problems in the planted model: planted hypergraph coloring, planted 3-SAT, planted 4-NAE, min-bisection (by Chen and Frieze [3] , Flaxman [8] , Goerdt and Lanka [9], Coja-Oghlan [4] respectively).

Perhaps the work closest in nature to the work in the current paper is that of Amin Coja-Oghlan [4] on finding a bisection in a sparse random graph. Both in our work and in that of [4], one is dealing with an optimization problem, and the density of the input graph is such that the planted solution is not an optimal solution. The algorithm for bisection in [4] is based on spectral techniques, and has the advantage that it provides a certificate showing that the solution that it finds is indeed optimal. Our algorithm for maximum independent set does not use spectral techniques and does not provide a certificate for optimality.

An important difference between planted models for independent set and those for other problems such as 3-coloring and min-bisection is that in our case the planted classes I, C are not symmetric. The lack of symmetry between I and C makes some of the ideas used for the more symmetric problems insufficient. In the approach of [1], a vertex is removed from its current color class and placed in OUT if its degree into some other current color class is very different than what one would typically expect to see between the two color classes. This procedure is shown to "clean" every color class C from all vertices that should have been from a different color class, but were wrongly assigned to class C in phase 1 of the algorithm. (The argument proving this goes as follows. Every vertex remaining in the wrong color class by the end of phase 2 must have many neighbors that are wrongly assigned themselves. Thus the set of wrongly assigned vertices induces a small subgraph with large edge density. But G itself does not have any such subgraphs, and hence by the end of phase 2 it must be the case that all wrongly assigned vertices were moved into OUT.) It turns out that this approach works well when classes are of similar nature (such as color classes, or two sides of a bisection), but does not seem to suffice in our case where I' is supposed to be an independent set whereas C' is not. Specifically, the set I' might still contain wrongly assigned vertices, and might not be a subset of a maximum independent set in the graph. Under these circumstances, phase 3 will not result in a maximum independent set. Our solution to this problem involves the following aspects, not present in previous work. In phase 2 we remove from I' every vertex that has even one edge connecting it to OUT. This adds more vertices to OUT and may possibly create large connected components in OUT. Indeed, we do not show that OUT has no large connected components, which is a key ingredient in previous approaches. Instead, we analyze the 2-core of OUT and show that the 2-core has no large components. Then, in phase 3, we use dynamic programming to find a maximum independent set in OUT, and use the special structure of OUT to show that the algorithm runs in polynomial time.

Remark: A significantly more complicated version of our algorithm works for a wider range of parameters, namely, for $\alpha \geq \sqrt{\frac{c_0}{d}}$ rather than $\alpha \geq \sqrt{\frac{c_0 \log d}{d}}$. In the current version of the paper we prefer to present the simpler version of the algorithm, so as not to obscure the new aspects of our work, such as the use of 2-cores. Future versions of this paper may include the improved algorithm.

1.3 Notation

Let $G = (V, E)$ and let $U \subset V$. The subgraph of G induced by the vertices of U is denoted by $G[U]$. We will use $\deg(v)_U$ to denote the degree of a vertex v into a set $U \subset V$. We use $\Gamma(U)$ to denote the vertex neighborhood of $U \subset V$ (excluding U). The parameter d (specifying the expected degree in the random graph G) is assumed to be sufficiently large, and some of the inequalities that we shall derive implicitly use this assumption, without stating it explicitly.

2 The Algorithm

Algorithm $FindIS$
Input: G.

1. (a) Set: $I_1 = \{v : \deg(v) < d - \alpha d/2\}$,
   $C_1 = \{v : \deg(v) \geq d - \alpha d/2\}$,
   $OUT_1 = \emptyset$.
   (b) For every edge (u, v) such that both u, v are in I_1, move u, v to OUT_1.
2. Set $I_2 = I_1,\ C_2 = C_1,\ OUT_2 = OUT_1$.
   A vertex $v \in C_2$ is *removable* if $\deg(v)_{I_2} < 4$.
   Iteratively: find a removable vertex v and move it and its neighbors from I_2 to OUT_2.
3. Output the union of I_2 and a maximum independent set of $G[OUT_2]$. We will explain later how this is done efficiently.

Figure 2 depicts the situation after step 2 of the algorithm. At that point, I_2 is an independent set, there are no edges between I_2 and OUT_2, and every vertex $v \in C_2$ has at least four neighbors in I_2.

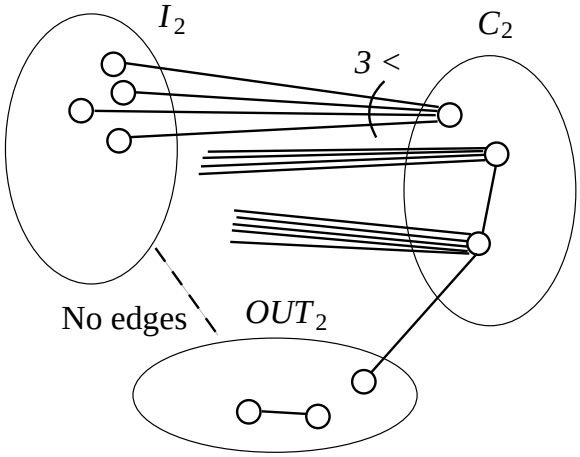

Figure 2: $FindIS$ outcome

3 Correctness

Let I_{max} be a maximum independent set of G. We establish two claims. Claim 3.1 guarantees the correctness of the algorithm and Claim 3.3 guarantees its efficient running time. Here we present these two claims, and their proofs are deferred to later sections.

Claim 3.1. *With high probability there exists I_{max} such that $I_2 \subseteq I_{max}, C_2 \cap I_{max} = \emptyset$.*

Definition 3.2. The 2-*core* of a graph G is the maximal subgraph in which the minimal degree is 2.

It is easy to see that the 2-core is unique and can be found by iteratively removing vertices whose degree is smaller than 2.

Claim 3.3. *With high probability the largest connected component in the 2-core of $G[OUT_2]$ has cardinality of at most $3 \log n$.*

Let G be any graph. All the vertices of G, which do not belong to the 2-core, form trees. Each such tree is either disconnected from the 2-core or it is connected by exactly one edge to the 2-core. To find a maximum independent set of $G[OUT_2]$ we need to find a maximum independent set in each connected component of $G[OUT_2]$ separately. For each connected component D_i of $G[OUT_2]$ we find the maximum independent set as follows: let C_i be the intersection of D_i with the 2-core of G. We enumerate all possible independent sets in C_i (there are at most $2^{|C_i|}$ possibilities), each one of them can be optimally extended to an independent set of D_i by solving (separately) a maximum independent set problem on each of the trees connected to C_i. For some trees we may have to exclude the tree vertex which is connected to D_i if it is connected to a vertex of the independent set that we try to extend. On each tree the problem can be solved by dynamic programming.

Corollary 3.4. *Finding $I(OUT_2)$ can be done in polynomial time.*

3.1 Dense Sets and Degree Deviations

In proving the correctness of the algorithm, we will use structural properties of the random graph G. In particular, such a random graph most likely has no small dense sets (small sets of vertices that induce many edges). This fact will be used on several occasions to derive a proof by contradiction. Namely, certain undesirable outcomes of the algorithm cannot occur, as otherwise they will lead to a discovery of a small dense set in G. The lemmas relating to these properties are rather standard and for a lack of space their statement and proof are given in appendix A, B respectively.

3.2 The dense case: $d \geq n^{32/c_0}$

When d is large enough (say $> n^{32/c_0}$) it can be shown that the planted independent set is the unique maximum independent set (we omit the details). In this case step 1a suffices for finding I, C: the probability that a fixed vertex $v \in C$ has degree below $d(1-\alpha/2)$ is at most $e^{-\alpha^2 d/8} = e^{-c_0 \log d/8} < e^{-4 \log n}$. Using the union bound we get that with probability $1 - 1/n^3$ in step 1a it holds that $C \subset C_1$. A similar argument shows that almost surely in step 1a it holds that $I \subset I_1$. When d is smaller (say constant) the above argument using the union bound fails (the probabilities are not small enough). Also when d is smaller it is no longer true that the planted independent set is the unique maximum one (as explained in Section 1). In the following sections we will assume that $d \leq n^{32/c_0}$.

3.3 Proof of Claim 3.1

We would have liked to prove that with high probability $I_2 \subseteq I \subseteq I_2 \cup OUT_2$. However, this is not correct when d is a constant.

Lemma 3.5 (Extention Lemma)**.** *Let I be any independent set of G and let $C \triangleq V \setminus I$. Let I', C', OUT' be an arbitrary partition of V for which I' is an independent set. If the following hold:*

1. $|(I' \cap C) \cup (I \cap C')| < n/d^5$.
2. *Every vertex of C' has 4 neighbors in I'. None of the vertices of I' have edges to OUT'.*
3. *The graph G has no small dense subsets as described in Lemma A.1 part 1.*

then there exists an independent set I_{new} (and $C_{new} \triangleq V \setminus I_{new}$) such that $I' \subseteq I_{new}, C' \subseteq C_{new}$ and $|I_{new}| \geq |I|$.

Proof. If we could show that on average a vertex of $U = (I' \cap C) \cup (I \cap C')$ contributes at least $4/3$ internal edges to U, then U would form a small dense set that contradicts Lemma A.1. This would imply that $U = (I' \cap C) \cup (I \cap C')$ is the empty set, and we could take $I_{new} = I$ in the proof of Lemma 3.5. The proof below extends this approach to cases where we cannot take $I_{new} = I$.

Every vertex $v \in C'$ has at least 4 edges into vertices of I'. Since I is an independent set it follows that every vertex of $I \cap C'$ has at least 4 edges into $I' \cap C$. To complete the argument we would like to show that every vertex of $I' \cap C$ has at least 2 edges into $I \cap C'$. However, some vertices $v \in I' \cap C$ might have less than two neighbors in $I \cap C'$. In this case, we will modify I to get an independent set I_{new} (and $C_{new} \triangleq V \setminus I_{new}$) at least as large as I, for which every vertex of $I' \cap C_{new}$ has 2 neighbors in $I_{new} \cap C'$. This is done iteratively; after each iteration we set $I = I_{new}, C = C_{new}$. Consider a vertex $v \in (I' \cap C)$ which has strictly less than 2 edges into $I \cap C'$:

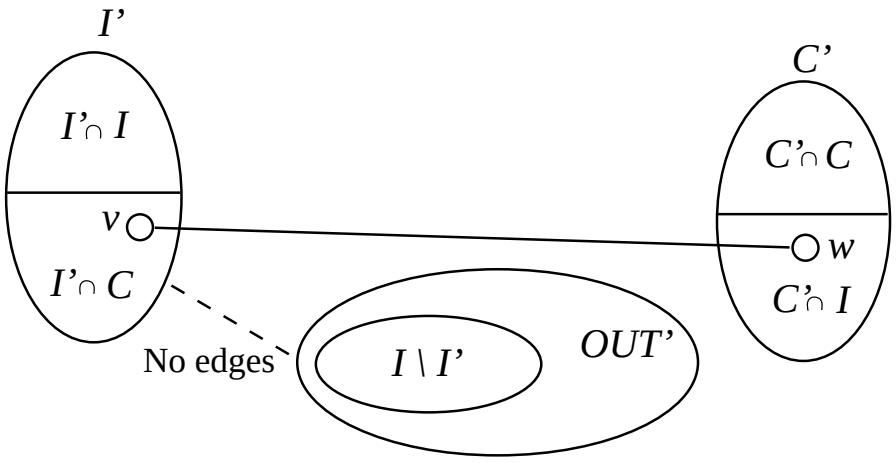

Figure 3: A vertex $v \in (I' \cap C)$ which has strictly less than 2 edges into $I \cap C'$

- If v has no neighbors in $I \cap C'$, then defi ne $I_{new} = I \cup \{v\}$. I_{new} is an independent set because v has no neighbors in $I \cap I'$ nor in $I \setminus I' \subseteq OUT'$ (there are no edges inside I' nor between I' and OUT).

- If v has only one edge into $w \in (I \cap C')$ then define $I_{new} = (I \setminus \{w\}) \cup \{v\}$. I_{new} is an independent set: v has no neighbors in $I \cap I'$ nor in $I \setminus (I' \cup C') \subseteq OUT$. The only neighbor of v in $I \cap C'$ is w.

The three properties are maintained also with respect to I_{new}, C_{new} (replacing I, C): properties $2, 3$ are independent on the sets I, C and property 3 is maintained since after each iteration it holds that $|(I' \cap C_{new}) \cup (I_{new} \cap C')| < |(I' \cap C) \cup (I \cap C')|$.

When the process ends, we have the following situation: each vertex of $I \cap C'$ has 4 edges into $I' \cap C$. Each vertex of $I' \cap C$ has at least 2 edges into $I \cap C'$ and thus using Corollary A.2 we get $|I \cap C'| \geq \frac{1}{2}|I' \cap C|$. The number of edges in $(I \cap C') \cup (I' \cap C)$ is at least $2|(I \cap C') \cup (I' \cap C)|$ and also $|(I \cap C') \cup (I' \cap C)| < n/d^5$, which implies that this set is empty (see Lemma A.1 part 1). □

To use Lemma 3.5 with $I = I_{max}, I' = I_2$, we need to show that condition 1 is satisfied, i.e. $|I_{max} \triangle I_2| < n/d^5$.

Lemma 3.6. *Almost surely* $|I_{max} \triangle I_2| < n/d^5$.

Proof.

$$|I_{max} \triangle I_2| \leq |I_{max} \triangle I| + |I \triangle I_2|$$

By Lemma A.3 the number of wrongly assigned vertices (in step 1a) is $e^{-\alpha^2 d/64} n$. By Claim 3.7 $|OUT| < 14e^{-\epsilon^2 d/41} n$. It follows that $|I \triangle I_2| << n/d^5$. It remains to bound $|I_{max} \triangle I|$:

$$|I_{max} \triangle I| = |I_{max} \setminus I| + |I \setminus I_{max}| \leq 2|I_{max} \setminus I| = 2|I_{max} \cap C|$$

$I_{max} = (I_{max} \cap I) \cup (I_{max} \cap C)$. One can always replace $I_{max} \cap C$ with $\Gamma(I_{max} \cap C) \cap I$ to get an independent set $(I_{max} \cap I) \cup (\Gamma(I_{max} \cap C) \cap I)$. The maximum independent set of C of cardinality at most $\frac{n \log d}{d}$, this upper bounds $|I_{max} \cap C|$. From Corollary A.5 if $|I_{max} \cap C| > n/(2d^5)$ then $|\Gamma(C \cap I_{max}) \cap I| > |I_{max} \cap C|$ which contradicts the maximality of I_{max}.

□

So far we have proved that I_2 is an independent set which is contained in some maximum independent set I_{max}. It remains to show that the 2-core of OUT_2 has no large connected components.

3.4 Proof of Claim 3.3

We first establish that OUT_2 is small.

Claim 3.7. *With probability of at least* $1 - e^{-c_1 \alpha^2 d \log n}$ *the cardinality of* OUT_2 *is at most* $e^{-\alpha^2 d/70} n$ *(where* c_1 *is a universal constant independent of* c_0*).*

Proof. We will use the assumption $d < n^{-32/c_0}$. If u, v were moved to OUT_1 in step 1b then at least one of them belongs to $C \cap I_1$. Thus:

$$|OUT_1| \leq |C \cap I_1| + |\Gamma(C \cap I_1)| < e^{-\alpha^2 d/64} n + 3de^{-\alpha^2 d/64} n + 2e^{-d} n < e^{-\alpha^2 d/64 + \log d + O(1)} n$$

with probability $1 - e^{-n^{0.2+o(1)}}$ (we use Lemma A.3 parts 3, 4).

Let $C' = \{v \in C : \deg(v)_I < \alpha d/2\}$. The cardinality of C' is at most $e^{-\alpha^2 d/64}$ with probability at least $1 - e^{-n^{0.5}}$ (using Lemma A.3 part 2). We will now show that $|OUT_2| \leq 2(|OUT_1| + |C'|)$. Let $U = OUT_1 \cup C'$. Start adding to U vertices of $OUT_2 \setminus U$ by the order in which the algorithm moved them

to OUT_2. In each step we add at most 4 vertices to U. Assume that at some point $|U|$ becomes larger than $2(|OUT_1| + |C'|)$. The number of edges inside U is at least:

$$\frac{1}{4}\frac{1}{2}|U|\alpha d/2 = \frac{\alpha d}{16}|U|. \tag{1}$$

At this point $|U| \leq e^{-\alpha^2 d/64 + \log d + O(1)}n + 4 << n/d^5$ and U contains $\frac{\alpha d}{16}|U|$ edges. By Lemma A.1 the probability that G contains such a dense set U is at most $e^{-\log n(\alpha d/16-1)} \leq e^{-c_1\alpha^2 d\log n}$.

□

Having established that OUT_2 is small, we would now like to establish that its structure is simple enough to allow one to find the maximum independent set of $G[OUT_2]$ in polynomial time. Establishing such a structure would have been easy if the vertices of OUT_2 were chosen independently at random, because a small random subgraph of a random graph G is likely to decompose into connected components no larger than $O(\log n)$. However, OUT_2 is extracted from G using some deterministic algorithm, and hence might have more complicated structure. For this reason, we shall now consider the 2-core of $G[OUT_2]$, and bound the size of its connected components.

Let A denote the 2-core of $G[OUT_2]$. In order to show that A has no large component, it is enough to show that A has no large tree. We were unable to show such a result for a general tree. Instead, we prove that A has no large *balanced* trees, that is trees in which at least $1/3$ fraction of the vertices belong to C. Fortunately, this turns out to be enough. Any set of vertices $U \subset V$ is called *balanced* if it contains at least $\frac{|U|}{3}$ vertices from C. We use the following reasoning: any maximal connected component of A is balanced - see Proposition 3.8 below. Furthermore, any balanced connected component of size at least $2\log n$ (in vertices) must contain a balanced tree of size is in $[\log n, 2\log n - 1]$ – see Lemma 3.9. We then complete the argument by showing that OUT_2 does not contain a balanced tree with size in $[\log n, 2\log n]$.

Proposition 3.8. *Every maximal connected component of the 2-core of OUT_2 is balanced.*

Proof. Let A_i be such a maximal connected component. Every vertex of A_i has degree of at least 2 in A_i because A_i is a maximal connected component of a 2-core. $|A_i| \leq |OUT_2| < \frac{n}{d^5}$. If $\frac{|A_i \cap I|}{|A_i|}$ is more than $\frac{2}{3}$, then the number of internal edges in A_i is $> 2 \cdot \frac{2}{3}|A_i| > \frac{4}{3}|A_i|$ which contradicts Lemma A.1. □

The proof of the following Lemma is deferred to Section A.

Lemma 3.9. *Let G be a connected graph whose vertices are partitioned into two sets: C and I. Let $\frac{1}{k}$ be a lower bound on the fraction of C vertices, where k is an integer. For any $1 \leq t \leq |V(G)|/2$ there exists a tree whose size is in $[t, 2t-1]$ and at least $\frac{1}{k}$ fraction of its vertices are C.*

We shall now prove that OUT_2 contains no balanced tree of size in $[\log n, 2\log n]$. Fix t to be some

value in $[\log n, 2\log n]$. The probability that OUT_2 contains a balanced tree of size t is at most:

$$\sum_{\substack{\text{T is balanced,}\\ |T|=t}} \Pr[T \subseteq E] \cdot \Pr[V(T) \subseteq OUT_2 \mid T \subseteq E] \leq \quad (2)$$

$$t \cdot \max_{\substack{t_1+t_2=t,\\ t_2 \geq t/3}} \binom{\alpha n}{t_1}\binom{(1-\alpha)n}{t_2} t^{t-2}\left(\frac{d}{n}\right)^{t-1} \cdot \max_{\substack{\text{T is balanced,}\\ |T|=t}} \{\Pr[V(T) \subseteq OUT_2 \mid T \subseteq E]\} \leq \quad (3)$$

$$t \max_{\substack{t_1+t_2=t,\\ t_2 \geq t/3}} \left(\frac{e\alpha n}{t_1}\right)^{t_1}\left(\frac{e(1-\alpha)n}{t_2}\right)^{t_2} t^{t-2}\left(\frac{d}{n}\right)^{t-1} \cdot \max_{\substack{\text{T is balanced,}\\ |T|=t}} \{\Pr[V(T) \subseteq OUT_2 \mid T \subseteq E] \quad (4)$$

$$nt\left((3ed(1-\alpha)^{1/3})\right)^t \max_{\substack{\text{T is balanced,}\\ |T|=t}} \{\Pr[V(T) \subseteq OUT_2 \mid T \subseteq E]\} \quad (5)$$

$$(1-\alpha)^{t/3}\, e^{\log n + t(\log d + 3)} \max_{\substack{\text{T is balanced,}\\ |T|=t}} \{\Pr[V(T) \subseteq OUT_2 \mid T \subseteq E]\} \quad (6)$$

To upper bound the above expression by $n^{-\log d}$ (so we can use union bound over all choices of t), it is enough to prove that for some universal constant c_1 and any fixed balanced tree T of size t it holds that:

$$\Pr[V(T) \subseteq OUT_2 \mid T \subseteq E] \leq e^{-c_1(\alpha^2 dt)}/(1-\alpha)^{t/3}$$

the last term is bounded by $\leq e^{-c_1 c_0 t \log d}/(1-\alpha)^{t/3}$ because $\alpha^2 d \geq c_0 \log d$. By choosing $c_0 > 1/c_1$ we derive the required bound. We will use the following equality

$$\Pr_E[V(T) \subseteq OUT_2 \mid T \subseteq E] = \Pr_E[V(T) \subseteq OUT_2(E \cup T)] \quad (7)$$

which is true because the distribution of E given that $T \subseteq E$ is exactly the distribution of $E \cup T$. We have to show that for any balanced tree T of size t, $\Pr[V(T) \subseteq OUT_2(E \cup T)] = e^{-c_1\alpha^2 dt}$. The difficulty in bounding the above probability is that $OUT_2(E \cup T)$ is not a uniformly chosen random set.

We will use a technique introduced at [1]. We give a review of this technique and its implementation in our setting. Using this technique in our setting involves some complications which do not exist in [1]. The new complications in our case are due to the fact that I, C are not symmetric as opposed to the coloring classes in [1]. The basic idea is as follows: given a fixed tree T of size $\log n$, we define a new algorithm $\tilde{FindIS}$ that outputs a set $O\tilde{U}T_2$. The algorithm $\tilde{FindIS}$ depends on T and has the following properties:

1. For every fixed configuration of edges E it holds that $OUT_2(E \cup T) \subseteq O\tilde{U}T_2(E)$.
2. There is a set $L(T) \subseteq V(T) \cap C$ of size $V(T)/6$ which is *oblivious* to $\tilde{FindIS}$ (the meaning of the term oblivious will be clear in the following proof).
3. The size of $O\tilde{U}T_2$ is bounded by $e^{-\alpha^2 d/70}n$, with probability $> 1 - e^{-c_1\alpha^2 d\log n}$.

These three properties (and $t \in [\log n, 2\log n]$) imply that:

$$\Pr[L(T) \subseteq OUT_2(E \cup T)] \leq \Pr[L(T) \subseteq O\tilde{U}T_2] < e^{-c_1\alpha^2 d|L(T)|} = e^{-c_1\alpha^2 dt}.$$

The first inequality follows from property 1. The second inequality follows from properties $2, 3$. The calculation is as follows:

$$\Pr[L(T) \subseteq O\tilde{U}T_2(E)] \leq$$

$$\Pr[L(T) \subseteq O\tilde{U}T_2(E) \mid \#(O\tilde{U}T_2 \cap C) < e^{-\alpha^2 d/70}] + \Pr[\#O\tilde{U}T_2 \geq e^{-\alpha^2 d/70} n]$$

The second term is at most $e^{-c_1 \alpha^2 d \log n}$ by property 3. It remains to upper bound the first term. Let $H(T) = (C \cap V(T)) \setminus L(T)$. Given that the intersection of $O\tilde{U}T_2$ with $C \setminus H(T)$ is of size m, its distribution is uniform over all subsets of $C \setminus H(T)$ of size m – this is the meaning of $L(T)$ being oblivious to $A\tilde{L}G$. It then follows that $\Pr[L(T) \subseteq O\tilde{U}T_2]$ is bounded by the probability that a binary random variable $X \sim Bin(m, p = \frac{|L(T)|}{|C \setminus H(T)| - m})$ has $|L(T)|$ successes. Since $m \leq e^{-\alpha^2 d/70} n, |L(T)| \geq t/6$ this probability is bounded by:

$$\binom{m}{t/6} p^{t/6} \leq \left(\frac{mep}{t/6}\right)^{t/6} = \left(\frac{me}{t/6} \cdot \frac{t/6}{|C \setminus H(T)| - m}\right)^{t/6} \leq \left(\frac{e^{-\alpha^2 d/70+1} n}{(1-\alpha)n/2}\right)^{t/6} \leq e^{-c_1 \alpha^2 dt}/(1-\alpha)^{t/3}$$

In the second inequality we used $(1-\alpha)n - \log n >> m$ which is true for $1 - \alpha >> e^{-c_0 \log d/70}$ (as otherwise almost surely a random $G_{n,\frac{d}{n},\alpha}$ graph has no connected components of size more than $\log n$).

We will now describe the procedure $\tilde{FindIS}$ and show that it has the three above mentioned properties. Let T be a balanced tree. We partition the vertices of T into three sets:

$I(T) = V(T) \cap I$,
$H(T) = V(T) \cap \{v \in C : \deg^T(v) > 11\}$,
$L(T) = V(T) \setminus (H(T) \cup I(T))$.

Algorithm $\tilde{FindIS}$
Input: $G, I, C, I(T), H(T)$.

1. (a) Set: $O\tilde{U}T_1 = I(T) \cup H(T)$,
   $\tilde{I}_1 = \{v : \deg(v) < d - \alpha d/2 - 11\} \setminus (C \cup O\tilde{U}T_1)$,
   $\tilde{C}_1 = \{v : \deg(v) \geq d - \alpha d/2\} \setminus O\tilde{U}T_1$,
   $O\tilde{U}T_1 = O\tilde{U}T_1 \cup (V \setminus (\tilde{I}_1 \cup \tilde{C}_1))$.
   (b) For every edge (u, v) such that both u, v are in $\tilde{I}_1$, move u, v to $O\tilde{U}T_1$.
   (c) Find those vertices of $\tilde{I}_1$ that have edges to $O\tilde{U}T_1$ and then move them to $O\tilde{U}T_1$.
2. Set $\tilde{I}_2 = \tilde{I}_1,\ \tilde{C}_2 = \tilde{C}_1,\ O\tilde{U}T_2 = O\tilde{U}T_1$. A vertex $v \in \tilde{C}_2$ is *removable* if $\deg(v)_{\tilde{I}_2} < 4$. Iteratively: find a removable vertex v and move it and its neighbors from $\tilde{I}_2$ to $O\tilde{U}T_2$.

A property that we will use in the proof of Lemma 3.10 is that after step 1a there are no T edges which touch $\tilde{I}_1$. There are no such edges because $\tilde{I}_1 \cap V(T) \subseteq (\tilde{I}_1 \cap C) \cup (V(T) \cap I)$ and it holds that $\tilde{I}_1 \cap C = \emptyset$ (by the definition of $\tilde{I}$), $V(T) \cap I = I(T) \subseteq O\tilde{U}T_1$ (see step $1(a)$). This property was achieved by moving to $O\tilde{U}T_1$ all the vertices of $I \cap V(T)$ immediately when $\tilde{FindIS}$ starts. If T was not balanced (e.g. T contains one vertex of C connecting all the rest which are vertices of I) it is possible that all the vertices of T but one are moved to OUT_1 at step 1a. In this case the set $|L(T)|$ is significantly smaller than $|V(T)|$ and thus not useful for our purpose. To get $|L(T)| \geq |V(T)|/6$ we need T to be balanced.

Lemma 3.10. $OUT_2(E \cup T) \subseteq O\tilde{U}T_2(E)$.

Proof. Consider the execution of $FindIS(E \cup T)$, we show a parallel execution of $\tilde{FindIS}(E)$, for which the invariant $\tilde{I}_j \subseteq I_j, \tilde{C}_j \subseteq C_j$ is kept. The final outcome of $FindIS$ does not depend on the order by which removable vertices are moved to OUT_2, since once a vertex becomes removable, it (and its neighbors form I_2) will be removed.

- Step 1(a): We consider only vertices of $L(T)$ as $H(T) \cup I(T) \subseteq O\tilde{U}T_1$ and the other vertices have the same degree in $E, E \cup T$. Vertices of $\tilde{C}_1$ do not pose a problem as their degree in $E \cup T$ may only increase. A vertex of $\tilde{I}_1 \cap L(T)$ has degree $\leq \alpha d/2 - 11$, thus its degree in $E \cup T$ is at most $\alpha d/2$ and it belongs to I_1.

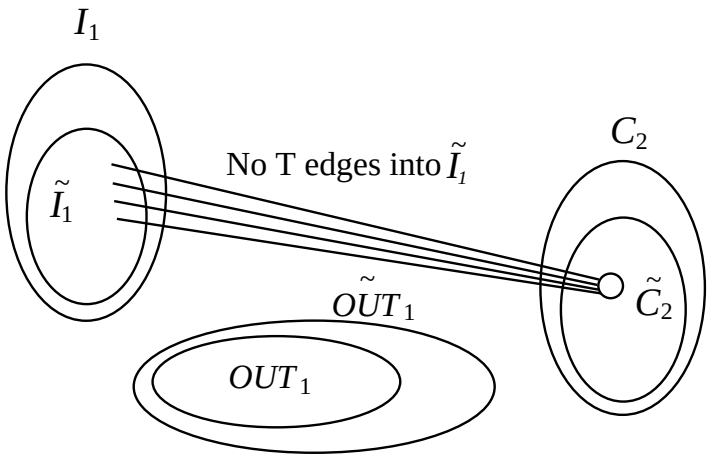

Figure 4: After step $1(a)$ there are no edges of T touching $\tilde{I}_1$.

- Steps 1(b,c): Consider a vertex v removed from I_1 because of the edge (v, w) inside I_1. If this edge is in E, then v is also removed from $\tilde{I}_1$ since w is either in $\tilde{I}_1$ or in $O\tilde{U}T_1$. If the edge (v, w) belongs to T then v is already in $O\tilde{U}T_1$. The reason is that no edges of T touch vertices of $\tilde{I}_1$.
- Step (2): The situation is as follows: for every vertex $v \in C_2 \cap \tilde{C}_2$ it holds: $\deg^{E \cup T}(v)_{I_1} \geq \deg^{E}(v)_{\tilde{I}_1}$, thus if v is removable in C_2 it is also removable in $\tilde{C}_2$. Moreover, the set of neighbors of v in I_2 contains the set of neighbors of v in $\tilde{I}_2$, this is because there are no T edges touching $\tilde{I}_2$.

□

Lemma 3.11. *For a balanced tree T, the set $L(T)$ is oblivious to $\tilde{FindIS}$ and its size is at least $|V(T)|/6$.*

Proof. The tree T contains at least $|V(T)|/3$ vertices from C. At least $1/2$ of them are of degree at most 11 in T, as otherwise the sum of degrees in T will be at least $|T|(\frac{11}{6} + \frac{1}{6}) > 2|T| - 2$. The set $L(T)$ looks as any other subset of $C \setminus H(T)$ to $\tilde{ALG}$. The input for $\tilde{ALG}$ is $G, I, C, I(T), H(T)$ (does not contain the edges of T) so $\tilde{ALG}$ has is no way to distinguish between $L(T)$ and any other set in $C \setminus H(T)$ with the same size. □

Lemma 3.12. *The size of $O\tilde{U}T_2$ is at most $e^{-\alpha^2 d/70}n$ with probability of at least $1 - e^{-c_1 \alpha^2 d \log n}$.*

Proof. The proof is similar to the proof of Claim 3.7. The only difference is that in step 1 we remove more vertices than in step 1 of ALG. Still, the number of vertices removed to $O\tilde{U}T_1$ in step 1 is of order $e^{-\alpha^2 d/64}n$ and this is enough to bound also the number of vertices removed in step 2. □

Acknowledgements

This work was supported in part by a grant from the G.I.F., the German-Israeli Foundation for Scientific Research and Development. Part of this work was done while the authors were visiting Microsoft Research in Redmond, Washington.

References

[1] N. Alon and N. Kahale. A spectral technique for coloring random 3-colorable graphs. *SIAM Journal on Computing*, 26(6):1733–1748, 1997.

[2] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. *Random Structures and Algorithms*, 13(3-4):457–466, 1988.

[3] H. Chen and A. Frieze. Coloring bipartite hypergraphs. In *Proceedings of the 5th International Conference on Integer Programming and Combinatorial Optimization*, pages 345–358, 1996.

[4] A. Coja-Oghlan. A spectral heuristic for bisecting random graphs. In *To appear in Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.

[5] U. Feige. Approximating maximum clique by removing subgraphs. *Siam J. on Discrete Math.*, 18(2):219–225, 2004.

[6] U. Feige and J. Kilian. Heuristics for semirandom graph problems. *Journal of Computing and System Sciences*, 63(4):639–671, 2001.

[7] U. Feige and R. Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Structures and Algorithms*, 16(2):195–208, 2000.

[8] A. Flaxman. A spectral technique for random satisfiable 3cnf formulas. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 357–363, 2003.

[9] A. Goerdt and A. Lanka. On the hardness and easiness of random 4-sat formulas. In *Proceedings of the 15th International Symposium on Algorithms and Computation (ISAAC)*, pages 470–483, 2004.

[10] G. Grimmet and C. McDiarmid. On colouring random graphs. *Math. Proc. Cam. Phil. Soc.*, 77:313–324, 1975.

[11] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[12] M. Jerrum. Large clique elude the metropolis process. *Random Structures and Algorithms*, 3(4):347–359, 1992.

[13] R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103. 1972.

[14] R. M. Karp. *The Probabilistic Analysis of Some Combinatorial Search Algorithms*, pages 1–19. Academic Press, NY, 1976.

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

A Technical lemmas

Lemma A.1. *Let G be a random graph taken from $G_{n,p}$ ($p = \frac{d}{n}$). The following holds:*

1. *With probability at least $1 - d^2/n$ for every set $U \subset V$ of cardinality smaller than n/d^5 the number of edges inside U is bounded by $\frac{4}{3}|U|$.*
2. *Let $c \geq 3$. With probability of at least $n^{-0.9(c-1)}$ for every set of vertices U of size smaller than n/d^2 the number of edges inside U is less than $c|U|$.*

Given two small enough disjoint sets of vertices A, B, if every vertex of B has at least 2 edges going to A then $|A|$ can not be too small relative to $|B|$. This is true as otherwise $|A \cup B|$ would contain too many internal edges with contradiction to Lemma A.1.

Corollary A.2. *Let G be a graph which has the property from Lemma A.1 part 1. Let A, B be any two disjoint sets of vertices each of size smaller than n/d^5. If every vertex of B has at least 2 edges going into A, then $|A| \geq |B|/2$.*

The following lemmas bound the number of vertices whose degree largely deviates from the expectation.

Lemma A.3. *Let $d < n^{32/c_0}$. The following properties hold with probability of at least $1 - e^{-\sqrt{n}}$*

1. *The number of vertices from I which are not I_1 is at most $e^{-\alpha^2 d/64} n$.*
2. *The number of vertices from C whose degree into I is $< \alpha d/2$ is at most $e^{-\alpha^2 d/64} n$.*
3. *The number of vertices from C which are not in C_1 is most $e^{-\alpha^2 d/64} n$.*

The following property holds with probability of $1 - e^{-n^{0.2+o(1)}}$:

4. *The number of edges that contain a vertex with degree at least $3d$ is at most $3e^{-d} dn$.*

For sets of size $\frac{10n \log d}{d}$ the statement in Lemma A.1 is not correct. For this set size we use the following weaker lemma.

Lemma A.4. *With probability at least $1 - 1/n$ there is no set U of size at most $\frac{10n \log d}{d}$ that contains $50 \log d|U|$ edges.*

Corollary A.5. *With probability of at least $1 - 1/n$ there is no $C' \subseteq C$ such that $n/2d^5 \leq |C'| \leq \frac{n \log d}{d}$ and $|\Gamma(C') \cap I| \leq |C'|$.*

B Proof of technical lemmas

Proof of lemma A.1 part 1. Denote $c = 4/3, k = n/d^5$. The statement of the lemma trivially holds for sets with at most 2 vertices. The probability that the statement in the lemma is false is bounded by:

$$\sum_{i=3}^{k} \binom{n}{i} \underbrace{\left(\sum_{j=\lceil ic \rceil}^{\binom{i}{2}} \binom{\binom{i}{2}}{j} \left(\frac{d}{n}\right)^j \right)}_{\geq \text{ prob. for at least ic successes}} \leq \sum_{i=3}^{k} \binom{n}{i} 2 \binom{\binom{i}{2}}{\lceil ic \rceil} \left(\frac{d}{n}\right)^{\lceil ic \rceil} \leq 2 \sum_{i=3}^{k} \left(\frac{n}{ei}\right)^i \left(\frac{i^2}{2eic}\right)^{\lceil ic \rceil} \left(\frac{d}{n}\right)^{\lceil ic \rceil} \tag{8}$$

the first inequality holds because the inside summation is geometric with a factor of $\frac{\binom{i}{2}}{\binom{i}{j+1}}\frac{d}{n} \leq \frac{i^2}{2(j+1)}\frac{d}{n} \leq \frac{i^2}{ic}\frac{d}{n} \leq \frac{id}{cn} \leq \frac{kd}{cn} < 1/2$ (for $k \leq n/2d$). The last term in (8) is bounded by:

$$\leq 2\sum_{i=3}^{k}\frac{n^i}{i^i}\left(\frac{dn}{2eic}\right)^{\lceil ic\rceil}\left(\frac{i}{n}\right)^{2\lceil ic\rceil} \leq \frac{2}{(4e)^2}\sum_{i=3}^{k}\left(\frac{d}{c}\right)^{\lceil ic\rceil}\left(\frac{i}{n}\right)^{\lceil ic\rceil - i} \leq \frac{1}{32}\sum_{i=3}^{k}\left(\frac{d}{c}\right)^{ic+1}\left(\frac{i}{n}\right)^{ic-i}$$

$$\leq \frac{d}{32c}\sum_{i=3}^{k}\left[\left(\frac{d}{c}\right)^{c}\left(\frac{i}{n}\right)^{c-1}\right]^{i} \leq \frac{d}{6c}\left(\left(\frac{d}{c}\right)^{c}\left(\frac{3}{n}\right)^{c-1}\right)^{3} \leq \frac{1}{2}\left(\frac{d^{3c+1}}{n^{3c-3}}\right) \leq \frac{d^5}{n}$$

(the inequality before the last one holds because the sum is geometrically decreasing with a factor of $(\frac{d}{c})^c(\frac{i+1}{n})^{c-1}$ which is smaller than $4/5$ for $i \leq n/d^5, c \geq 4/3, d \geq 2$). □

Proof of lemma A.1 part 2. The proof is essentially the same as the proof of part 1. The only difference is in the last inequality where we use:

$$\frac{d}{6c}\left(\left(\frac{d}{c}\right)^{c}\left(\frac{3}{n}\right)^{c-1}\right)^{3} \leq e^{-(c-1)\log n+(c+1)\log d} \leq e^{-0.9(c-1)\log n}$$

□

Proof of Corollary A.2. By contradiction, assume that $|A| = \delta|B|$ for some $0 < \delta < 1/2$. The number of internal edges of $A \cup B$ is at least $\frac{2|B|}{(1+\delta)|B|} = \frac{2}{1+\delta} > 4/3$. The last inequality contradicts Lemma A.1. □

Proof of Lemma A.3 Set $\delta = e^{-\alpha^2 d/64}$. We will use the fact that $\alpha > 10\delta$ which is true for large enough d since $\alpha > \sqrt{\frac{c_0 \log d}{d}}$ and $\delta < e^{-d/64}$. It also holds that $e^{-\delta n} = e^{-e^{-\alpha^2 d/64}n} < e^{-n^{0.5}}$ because $e^{-\alpha^2 d/64} > n^{-0.5}$ (we use here the assumption that $d < n^{32/c_0}$, which is equivalent to $c_0 \log d/64 < \log n/2$).

Proof of part 1. Bounding the fraction of vertices from I with large degrees is relatively easy, since the degrees of vertices in I into C are independent. For a fixed set of size δn the expected sum of degrees in $\mu = \delta n(1-\alpha)d$. The probability for a δn size set with degrees $> \alpha d/2$ is at most:

$$\binom{\alpha n}{\delta n}e^{-\frac{1}{2}(\frac{\alpha}{2(1-\alpha)})^2(1-\alpha)d\delta n} \leq e^{-\delta n(-\log(e/\delta)+\alpha^2 d/8)} \leq e^{-\delta n\epsilon^2 d/16} \leq e^{-\delta n}.$$

□

Proof of part 2. The degrees into I are independent random variables. For a fixed set of size δn the expected sum of degrees in $\mu = \delta n\alpha d$. A bad set has only $\delta n\alpha d/2$ edges to I. The probability for a bad set of size δn is bounded by:

$$\binom{n}{\delta n}e^{-\frac{1}{2}\mu(\frac{1}{2})^2} \leq e^{-\delta n(\log(e/\delta)-\frac{1}{8}\alpha d)} \leq e^{-\delta n\alpha d/4} \leq e^{-\delta n}.$$

□

Proof of part 3. Let B be a subset of C of size δn, in which every vertex has a degree$< \alpha d/2$. For every vertex of B the expected degree into $V \setminus B$ is $(1-\delta)d$ and its expected degree into B is δd. Let X_1 be the sum of degrees in B of edges inside B. Let X_2 be the sum degrees in B of edges which go out of B. It holds that $\mu_1 = E[X_1] = \delta n \delta d$, $\mu_2 = E[X_2] = \delta n(1-\delta)d$. It holds that $X_1 + X_2 \leq \delta n \alpha d/2$. Since $\mu_1 + \mu_2 = \delta n d$, at least one of the following holds:

1. X_1 deviates from μ_1 by $\delta n \alpha d/4$. Since $\alpha > 10\delta$ the probability for this event is bounded by $e^{-\frac{1}{2}\mu_1 \frac{\alpha}{4\delta}} = e^{-\delta n \alpha d/8}$ (X_1 is a binomial random variable multiplied by a factor of 2).

2. X_2 deviates from μ_2 by $\delta n \alpha d/4$. The probability for this event is bounded by $e^{-\frac{1}{2}\mu_2(\frac{\alpha}{4(1-\delta)})^2} \leq e^{-\delta n \alpha^2 d/32}$ ($1-\delta$ is very close to 1).

The probability that either of these two events happens is at most:

$$\binom{n}{\delta n} e^{-\delta n \alpha^2 d/32} \leq \left(\frac{ne}{\delta n}\right)^{\delta n} e^{-\delta n \alpha^2 d/8}$$

$$\leq e^{\delta n(\log(e/\delta) - \alpha^2 d/32)} \leq e^{-\delta n(\alpha^2 d/64 - 1)} \leq e^{-\delta n}.$$

Remark: We use here $\alpha^2 d/64 > 2$ (which is true for large enough d).

□

Proof of part 4. The proof is standard, details are omitted. □

Proof of lemma A.4. Modify the proof of Lemma A.1 by setting the parameters: $c = 50 \log d$ and $k = \frac{10n \log d}{d}$ (the set size bound). In the proof of Lemma A.1 we used the following inequalities:

$$\left(\frac{d}{2c}\right)^c \left(\frac{2}{n}\right)^{c-1} < \frac{4}{5}; \quad \frac{kd}{cn} < \frac{1}{2}; \quad c < d$$

for showing that certain sums are geometric. These inequalities hold also for the current values of k, c. □

Proof of Corollary A.5. Let C' be such a bad set. It holds that $|\Gamma(C') \cap I| \leq |C'|$. By Lemma A.3 part 2 at least $|C'| - e^{-\alpha^2 d/64} n > \frac{9}{10}|C'|$ vertices of C' have $\frac{\alpha d}{2}$ at least edges to I (for large enough c_0 it holds that $e^{-\alpha^2 d/64} n = e^{-c_0 \log d/64} n < \frac{n}{20d^5} \leq \frac{9}{10}|C'|$). It follows that $C' \cup (\Gamma(C') \cap I)$ has at least $\frac{\alpha d}{5}|C' \cup (\Gamma(C') \cap I)| > \sqrt{d}|C' \cup (\Gamma(C') \cap I)|$ internal edges with contradiction to Lemma A.4. □

Proof of Lemma 3.9. We use the following well know fact: any tree T contains a *center* vertex v such that each subtree hanged on v contains strictly less than half of the vertices of T.

Let T be an arbitrary spanning tree of G, with center v. We proceed by induction on the size of T. Consider the subtrees $T_1, ..., T_k$ hanged on v. If there exists a subtree T_j with at least t vertices then also $T \setminus T_j$ has at least t vertices. In at least one of $T_j, T \setminus T_j$ the fraction of C vertices is at least $\frac{1}{k}$ and the lemma follows by induction on it. Consider now the case in which all the trees have less than t vertices. If in some subtree T_j the fraction of C vertices is at most $\frac{1}{k}$, then we remove it and apply induction to $T \setminus T_j$. The remaining case is that in all the subtrees the fraction of C vertices is strictly more than $\frac{1}{k}$. In this case we start adding subtrees to the root v until for the first time the number of vertices is at least t. At this point we have a tree with at most $2t-1$ vertices and the fraction of C vertices is at least $\frac{1}{k}$. To see that the fraction of C vertices is at least $\frac{1}{k}$, we only need to prove that the tree formed by v and the first subtree has $\frac{1}{k}$ fraction of C vertices. Let r be the number of C vertices in the first subtree and let b be the number of vertices in it. Since k is integer we have: $\frac{r}{b} > \frac{1}{k} \Longrightarrow \frac{r}{b+1} \geq \frac{1}{k}$. □

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'