

Finding a Maximum Independent Set in a Sparse Random Graph *

Uriel Feige Eran Ofek

September 21, 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 algorithm that with high probability (over the random choice of random graph G , and without being given the planted independent set I) finds a maximum independent set in G when $\alpha \geq \sqrt{c_0 \log d/d}$, where c_0 is some sufficiently large constant independent of d .

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 NP-hard. Moreover, for any $\epsilon > 0$ there is no $n^{1-\epsilon}$ (polynomial time) approximation algorithm for it unless NP=ZPP [12]. The best approximation ratio currently known for maximum independent set [6] is $O(n(\log \log n)^2/(\log n)^3)$.

In light of the above mentioned negative results, one 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 $\log_2 n$ [11]. However, there is no known polynomial

*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.

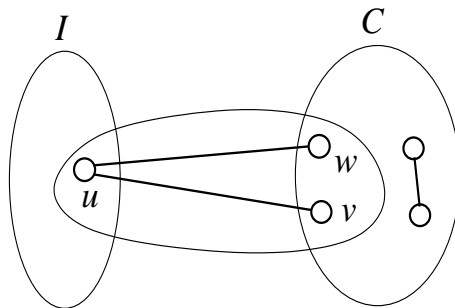


Figure 1: The vertex $u \in I$ is not contained in any maximum independent set because no other edges touch v, w .

time algorithm that almost surely finds an independent set of size $(1 + \epsilon) \log_2 n$ (for any $\epsilon > 0$).

To further simplify the problem, Jerrum [13] and Kucera [15] proposed a *planted model* 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 , 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 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 $1/2$, 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 (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 any 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 any 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 .

Remark: A significantly more complicated version of our algorithm works for a wider range of parameters, namely, for $\alpha \geq \sqrt{c_0/d}$ rather than $\alpha \geq \sqrt{c_0 \log d/d}$. The improved algorithm will appear in the full version of this paper ¹.

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 [7] 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 [8] 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 $|C \Delta C'| + |I \Delta I'|$ should be at most $e^{-c \log d/n}$ where c is a large enough universal constant (this phase is analogous to the first two phases of [1]).
2. Move to OUT vertices of I', C' which have non typical degrees.

¹Can be found in <http://wisdom.weizmann.ac.il/~erano>

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 (almost surely) random graphs 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 in graph induced on OUT and adding it to I' .

With high probability the structure of OUT will be easy enough so that a maximum independent set can be efficiently found. OUT is a random graph of size $n/poly(d)$. 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: hypergraph coloring, 3-SAT, 4-NAE, min-bisection (by Chen and Frieze [3], Flaxman [9], Goerdt and Lanka [10], 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.

1.3 Notation and Terminology

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 denote by $\deg^E(v)_U$ the number of edges from E that connect a vertex v to $U \subset V$; when E is clear from the context we will use $\deg(v)_U$. 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. The term *with high probability* (w.h.p.) is used to denote a sequence of probabilities that tends to 1 as n tends to infinity.

2 The Algorithm

FindIS(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 . Move v and $\Gamma(v) \cap I_2$ 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.

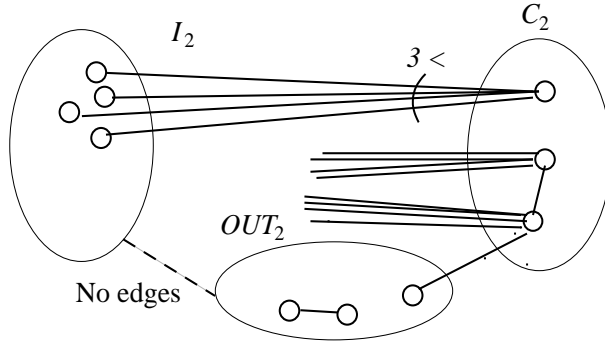


Figure 2: After step 2 of the algorithm, I_2 is an independent set, there are no edges between I_2 and OUT_2 , and every vertex $v \in C_2$ has at least four neighbors in I_2 .

3 Correctness

When $d > n^{32/c_0}$ a simple argument (using the union and the Chernoff bounds), which we omit, shows that after step 1a of the algorithm it holds that $I_1 = I, C_1 = C$. In fact, when $d > n^{32/c_0}$ the planted independent set is the unique maximum independent set and this case was already covered in [5]. From now we will assume that $d < n^{32/c_0}$.

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

Theorem 3.1. *Wh.p. 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 at least 2.*

Clearly the 2-core is unique and can be found by iteratively removing vertices whose degree is smaller than 2.

Theorem 3.3. *Wh.p. the largest connected component in the 2-core of $G[OUT_2]$ has cardinality of at most $2 \log n$.*

Let G be any graph. Those vertices of G that 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 C_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. *A maximum independent set of $G[OUT_2]$ can be found efficiently.*

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 lead to a discovery of a small dense set. The lemmas relating to these properties are rather standard and their proofs are omitted due to lack of space.

Lemma 3.5. *Let G be a random graph taken from $G_{n,p,\alpha}$ ($p = \frac{d}{n}$, $d < n^{32/c_0}$). The following holds:*

1. *W.h.p. for every set $U \subset V$ of cardinality smaller than $2n/d^5$ 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 $c|U|$.*
3. *W.h.p. there is no $C' \subseteq C$ such that $\frac{n}{2d^5} \leq |C'| \leq \frac{2n \log d}{d}$ and $|\Gamma(C') \cap I| \leq |C'|$.*

Corollary 3.6. *Let G be a graph which has the property from Lemma 3.5 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 $|A| \geq |B|/2$.*

Lemma 3.7. *Let $d < n^{32/c_0}$. The following hold with probability $> 1 - e^{-n^{0.1}}$:*

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 $e^{-\alpha^2 d/64}n$.*
4. *The number of edges that contain a vertex with degree at least $3d$ is at most $3e^{-d}dn$.*

3.2 Proof of Theorem 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 incorrect when d is a constant.

Lemma 3.8. *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' . There are no edges between I' and OUT' .*

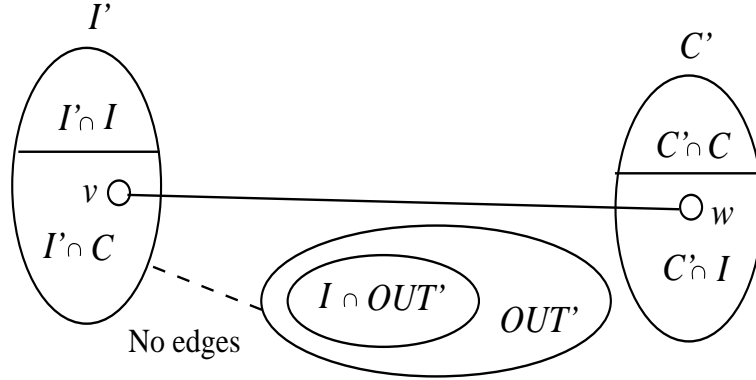


Figure 3: A vertex $v \in (I' \cap C)$ which has exactly one edge into $I \cap C'$.

3. The graph G has no small dense subsets as described in Lemma 3.5 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 3.5. 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 3.8. 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 $I \cap C'$:

- If v has no neighbors in $I \cap C'$, then define $I_{new} = I \cup \{v\}$. I_{new} is an independent set because v (being in I') has no neighbors in I' nor in 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 because v (being in I') has no neighbors in I' nor in OUT' . The only neighbor of v in $I \cap C'$ is w .

The three properties (from the statement of this lemma) 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 1 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, let U denote $(I' \cap C) \cup (I \cap C')$. Each vertex of $I' \cap C$ has at least 2 edges into $I \cap C'$, thus $|I \cap C'| \geq \frac{1}{2}|I' \cap C|$ (see Corollary 3.6). Each vertex of

$I \cap C'$ has 4 edges into $I' \cap C$ so the number of edges in U is at least $4|I \cap C'| \geq 4|U|/3$ and also $|U| < n/d^5$, which implies that U is empty (by Lemma 3.5 part 1). \square

To use Lemma 3.8 with $I = I_{max}, I' = I_2$, we need to show that condition 1 is satisfied, i.e. $|I_{max} \Delta I_2| < n/d^5$.

Lemma 3.9. *With high probability $|I_{max} \Delta I_2| < n/d^5$.*

Proof. $|I_{max} \Delta I_2| \leq |I_{max} \Delta I| + |I \Delta I_2|$. By Lemma 3.7 the number of wrongly assigned vertices (in step 1a) is $e^{-\alpha^2 d/64}n$. By Lemma 3.10 $|OUT_2| < e^{-\alpha^2 d/70}n$. It follows that $|I \Delta I_2| \ll n/d^5$. It remains to bound $|I_{max} \Delta I|$:

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

$I_{max} = (I_{max} \cap I) \cup (I_{max} \cap C)$. 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 in $G[C]$ is w.h.p. of cardinality at most $\frac{2n \log d}{d}$ (the proof is standard using first moment), this upper bounds $|I_{max} \cap C|$. From Lemma 3.5 part 3 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} . \square

It follows that I_2 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.

3.3 Proof of Theorem 3.3

Lemma 3.10. *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:

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

with probability $> 1 - e^{-n^{0.1}}$ (we use Lemma 3.7 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 $1 - e^{-n^{0.1}}$ (using Lemma 3.7 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:

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

At this point $|U| \leq e^{-\alpha^2 d/64 + \log d + O(1)}n + 4 \ll n/d^5$ and U contains $\frac{\alpha d}{16} |U|$ edges. By Lemma 3.5 part 2 the probability that G contains such a dense set U is at most $e^{-0.9 \log n (\alpha d/16-1)} \leq e^{-c_1 \alpha^2 d \log n}$ (we use here $\alpha d \gg \alpha^2 d$). \square

Having established that OUT_2 is small, we would now like to establish that its structure is simple enough to allow one to find a 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 connected 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 Lemma 3.11 below. Furthermore, any balanced connected component of size at least $2 \log n$ (in vertices) must contain a balanced tree of size in $[\log n, 2 \log n - 1]$ - see Lemma 3.12. We then complete the argument by showing that OUT_2 does not contain a balanced tree with size in $[\log n, 2 \log n]$.

Lemma 3.11. *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 $\frac{|A_i \cap C|}{|A_i|}$ 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 3.5. \square

Lemma 3.12. *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 .*

Proof. 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, \dots, 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: $\frac{r}{b} > \frac{1}{k} \implies \frac{r}{b+1} \geq \frac{1}{k}$. \square

We shall now prove that OUT_2 contains no balanced tree of size in $[\log n, 2 \log n]$. To simplify the following computations, we will assume that $\alpha < 1/2$ (the proof can be modified to work for any α , we omit the details). Fix t to be some value in $[\log n, 2 \log n]$. We will use the fact that the number of spanning trees of a t -clique is t^{t-2} (Cayley's formula). The probability that OUT_2 contains a balanced tree of size t is at most:

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

$$\binom{n}{t} t^{t-1} \left(\frac{d}{n}\right)^{t-1} \max_{\substack{T \text{ balanced,} \\ |T|=t}} \{\Pr[V(T) \subseteq OUT_2 \mid T \subseteq E]\} \leq \quad (3)$$

$$n (ed)^t \max_{\substack{T \text{ is balanced,} \\ |T|=t}} \{\Pr[V(T) \subseteq OUT_2 \mid T \subseteq E]\} \leq \quad (4)$$

$$e^{t(\log d + 1) + \log n} \max_{\substack{T \text{ is balanced,} \\ |T|=t}} \{\Pr[V(T) \subseteq OUT_2 \mid T \subseteq E]\} \quad (5)$$

To upper bound the above expression by $o(1/\log n)$ (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:

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

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

$$\Pr_E[V(T) \subseteq OUT_2 \mid T \subseteq E] = \Pr_E[V(T) \subseteq OUT_2(E \cup T)], \quad (6)$$

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 , $\Pr[V(T) \subseteq OUT_2(E \cup T)] \leq e^{-c_1 \alpha^2 dt}$. We use the technique introduced at [1] and modify it to our setting. To simplify the notation we will use F to denote the algorithm FindIS defined at Section 2. Given a fixed balanced tree T of size $\leq 2 \log n$, we define an intermediate algorithm F' that knows the partition I, C and also $I(T)$ (which is $I \cap V(T)$). F' has no knowledge of which vertices are in $C(T)$ (in fact they can be chosen after the algorithm is run). F' is identical to F except for the following difference: after step 1(a), it uses a step 1'(a) that puts all vertices of $I(T)$ and all vertices of $C \cap I_1$ in OUT_1 , it then continues to step 1(b) after which, it also throws from I_1 (to OUT_1) all the vertices connected to OUT_1 (step 1(c)).

We use $OUT'_2(E)$ to denote OUT_2 in the outcome of $F'(E)$. As we shall see, F' dominates F in the following sense: $OUT'_2(E)$ contains $OUT_2(E \cup T)$. Nevertheless, since F' has no knowledge of $C(T)$, the set $C(T)$ is likely to be in $OUT'_2(E)$ as any other subset of C with the same size (where the random variable is E). An argument similar to the one in Lemma 3.10 (which we omit) shows that $OUT'_2(E)$ is likely to be

small: $|OUT_2'(E)| < e^{-\alpha^2 d/70} n$ with probability $> 1 - e^{-c_1 \alpha^2 d \log n}$. We get:

$$\begin{aligned} \Pr_E[C(T) \subseteq OUT_2(E \cup T)] &\leq \Pr_E[C(T) \subseteq OUT_2'(E)] \\ &\leq \Pr_E[C(T) \subseteq OUT_2'(E) \mid \#OUT_2'(E) \cap C < e^{-\alpha^2 d/70}] + e^{-c_1 \alpha^2 d \log n}. \end{aligned}$$

Given that $|OUT_2'(E) \cap C| = m$, the set $OUT_2'(E) \cap C$ is just a random subsets of C of size m . It then follows that $\Pr[C(T) \subseteq OUT_2'(E) \mid \#OUT_2'(E) \cap C < e^{-\alpha^2 d/70}]$ is bounded by the probability that a binary random variable $X \sim \text{Bin}(m, p = \frac{|C(T)|}{|C|-m})$ has $|C(T)|$ successes. Since $m \leq e^{-\alpha^2 d/70} n$ and $|C(T)| \geq t/3$ this probability is bounded by:

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

In the second inequality we use $(1 - \alpha)n - m > n/3$ which is true for $\alpha < 1/2$. (If we let $(1 - \alpha)$ to be too small we get a small factor in the denominator which becomes significant. Nevertheless, a more careful estimation in equation (3) yields a factor that cancels it out.) It follows that $\Pr_E[V(T) \subseteq OUT_2(E \cup T)] < e^{-c_1 \alpha^2 dt}$ as needed.

It remains to show that $OUT_2(E \cup T) \subseteq OUT_2'(E)$. This is done in Lemma 3.13.

Lemma 3.13. $OUT_2(E \cup T) \subseteq OUT_2'(E \cup T) \subseteq OUT_2'(E)$.

Proof. There are three executions that we consider:

- (i) $F(E \cup T)$ which produces $I_1^{(i)}, C_1^{(i)}$ in step 1 and $I_2^{(i)}, C_2^{(i)}$ in step 2,
- (ii) $F'(E \cup T)$ which produces $I_1^{(ii)}, C_1^{(ii)}$ in step 1 and $I_2^{(ii)}, C_2^{(ii)}$ in step 2,
- (iii) $F'(E)$ which produces $I_1^{(iii)}, C_1^{(iii)}$ in step 1 and $I_2^{(iii)}, C_2^{(iii)}$ in step 2.

First we analyse step 1 and show that: $I_1^{(i)} \supseteq I_1^{(ii)} \supseteq I_1^{(iii)}$ and $C_1^{(i)} \supseteq C_1^{(ii)} \supseteq C_1^{(iii)}$. The inclusions $I_1^{(i)} \supseteq I_1^{(ii)}, C_1^{(i)} \supseteq C_1^{(ii)}$ are easy as after step 1(a) they are in fact equalities and in steps 1'(a), 1(b), 1(c) F' removes more vertices than what F removes in 1(b) to OUT_1 . We now prove the inclusions $I_1^{(ii)} \supseteq I_1^{(iii)}, C_1^{(ii)} \supseteq C_1^{(iii)}$. The only difference (due to T edges) in step 1(a) is that some vertices of $I_1^{(iii)} \cap V(T)$ will be put in $C_1^{(ii)}$ (instead of being in $I_1^{(ii)}$). This does not pose a problem since anyway all the vertices of $I_1^{(iii)} \cap V(T)$ are moved to OUT_1 by F' at step 1'(a) (because $I_1^{(iii)} \cap V(T)$ is contained in $I(T) \cup (C \cap I_1^{(iii)})$). Any two vertices which are removed in step 1(b) from $I_1^{(ii)}$ will be removed from $I_2^{(iii)}$ either in step 1(b) or in step 1(c).

Given the inclusions of step 1 we are ready to prove the inclusions of step 2: $I_2^{(i)} \supseteq I_2^{(ii)} \supseteq I_2^{(iii)}$ and $C_2^{(i)} \supseteq C_2^{(ii)} \supseteq C_2^{(iii)}$. Notice that these inclusions imply the Lemma. We begin with $I_2^{(i)} \supseteq I_2^{(ii)}, C_2^{(i)} \supseteq C_2^{(ii)}$. Consider the execution of step 2 of $F(E \cup T)$. We show a parallel execution of step 2 of $F'(E \cup T)$, for which the invariant $I_2^{(i)} \supseteq I_2^{(ii)}, C_2^{(i)} \supseteq C_2^{(ii)}$ is kept. It is enough to show one possible execution of step 2 of $F'(E \cup T)$, because the order by which vertices are removed does not affect the final outcome (once a vertex becomes removable it will be removed).

Initially, the required inclusions are true due to the inclusions after step 1. Whenever a vertex $v \in C_2^{(ii)}$ is removed, it becomes removable from $C_2^{(ii)}$ and we remove it (if it had already been removed then also its neighbors from $I_2^{(ii)}$ had been removed because F' ensures there are no edges between I_2 and OUT_2). Proving the inclusions $I_2^{(ii)} \supseteq I_2^{(iii)}$, $C_2^{(ii)} \supseteq C_2^{(iii)}$ is done in a similar way, only that now we have the edges of T which influence the execution of $F'(E \cup T)$, but do not exist in the execution of $F'(E)$. Again, whenever a vertex $v \in C_2^{(ii)}$ is removed, it becomes removable from $C_2^{(iii)}$ and we remove it (if it is was not already moved to OUT_2). Let u be a neighbor of v from $I_2^{(ii)}$ that is moved together with v to OUT_2 . If $(u, v) \in E$ then u can not stay in $I_2^{(iii)}$ as there are no edges between I_2 and OUT_2 in F' . If $(u, v) \in T$ then u is either in $I_2^{(iii)} \cap C(T)$ or in $I_2^{(iii)} \cap I(T)$. In both cases u belonged to either $I_1^{(iii)} \cap C$ or $I(t)$ and was already removed in step 1'(a). \square

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. IPCO 1996, 345–358.
- [4] A. Coja-Oghlan. A spectral heuristic for bisecting random graphs. SODA 2005, 850–859.
- [5] A. Coja-Oghlan. Finding Large Independent Sets in Polynomial Expected Time. STACS 2003, 511–522.
- [6] U. Feige. Approximating maximum clique by removing subgraphs. *Siam J. on Discrete Math.*, 18(2):219–225, 2004.
- [7] U. Feige and J. Kilian. Heuristics for semirandom graph problems. *Journal of Computing and System Sciences*, 63(4):639–671, 2001.
- [8] 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.
- [9] A. Flaxman. A spectral technique for random satisfiable 3cnf formulas. SODA 2003, 357–363.
- [10] A. Goerdt and A. Lanka. On the hardness and easiness of random 4-sat formulas. ISAAC 2004, 470–483.
- [11] G. Grimmet and C. McDiarmid. On colouring random graphs. *Math. Proc. Cam. Phil. Soc.*, 77:313–324, 1975.

- [12] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.
- [13] M. Jerrum. Large clique elude the metropolis process. *Random Structures and Algorithms*, 3(4):347–359, 1992.
- [14] R. M. Karp. The probabilistic analysis of some combinatorial search algorithms. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 1–19. Academic Press, New York, 1976.
- [15] L. Kučera. Expected complexity of graph partitioning problems. *Discrete Appl. Math.*, 57(2-3):193–212, 1995.