Combinatorics, Probability and Computing (2006) 15, 731–751. © 2006 Cambridge University Press
doi:10.1017/S0963548306007553 Printed in the United Kingdom

Finding Large Independent Sets in Polynomial Expected Time

AMIN COJA-OGHLAN[†]

Humboldt-Universität zu Berlin, Institut für Informatik,
Unter den Linden 6, 10099 Berlin, Germany
(e-mail: `coja@informatik.hu-berlin.de`)

Received 11 February 2004; revised 28 September 2004

We consider instances of the maximum independent set problem that are constructed according to the following semirandom model. Let $G_{n,p}$ be a random graph, and let S be a set of k vertices, chosen uniformly at random. Then, let G_0 be the graph obtained by deleting all edges connecting two vertices in S. Finally, an adversary may add edges to G_0 that do not connect two vertices in S, thereby producing the instance $G = G^*_{n,p,k}$. We present an algorithm that on input $G = G^*_{n,p,k}$ finds an independent set of size $\geqslant k$ within *polynomial expected time*, provided that $k \geqslant C(n/p)^{1/2}$ for a certain constant $C > 0$. Moreover, we prove that in the case $k \leqslant (1-\varepsilon)\ln(n)/p$ this problem is hard.

1. Introduction and results

1.1. The maximum independent set problem and random graphs

An *independent set* in a graph $G = (V, E)$ is a set S of vertices of G such that no two vertices in S are adjacent. The *independence number* $\alpha(G)$ is the size of a largest independent set. The *maximum independent set problem* – given a graph G, find an independent set of maximum cardinality – is well known to be NP-hard. Indeed, Håstad [17] has shown that unless NP = ZPP, no polynomial time algorithm approximates $\alpha(G)$ within a factor of $n^{1-\varepsilon}$ ($\varepsilon > 0$ arbitrarily small but constant). Consequently, since we do not hope for efficient algorithms that compute good approximate solutions in the worst case, it is natural to ask for efficient algorithms that perform well on *average* instances, as proposed by Karp in 1976 [21].

The common way to describe 'average' instances is to consider a probability distribution on the instances, *i.e.*, on graphs. Since the seminal work of Erdős and Rényi, the binomial model $G_{n,p}$ has been the standard model of a random graph. Both the combinatorial structure and the algorithmic theory of $G_{n,p}$ have been studied intensively [5, 14, 18]. Given a parameter $0 < p = p(n) < 1$, the random graph $G_{n,p}$ is obtained by including each

[†] Research supported by the Deutsche Forschungsgemeinschaft (grant DFG FOR 413/1-1).

of the $\binom{n}{2}$ possible edges with probability p independently of all others. We say that $G_{n,p}$ enjoys some property P *with high probability* (w.h.p.) if the probability that $G_{n,p}$ satisfies P converges to 1 as $n \to \infty$.

The probable value of the independence number of $G_{n,p}$ has been determined by Bollobás and Erdős, Frieze and Matula [6, 13, 25] (see [18] for a unified treatment), who have proved that

$$\alpha(G) \sim \begin{cases} 2\log_{1/(1-p)}(n) & \text{for constant } 0 < p < 1, \\ \frac{2\ln(np)}{p} & \text{for } 1/n \ll p = o(1), \end{cases} \qquad \text{w.h.p.} \tag{1.1}$$

With respect to algorithms, it is known that, *e.g.*, in the case $p = 1/2$, a simple greedy heuristic (that just computes a maximal independent set w.r.t. inclusion) w.h.p. finds an independent set of size $\sim \log_2(n)$ and hence is $(2 + o(1))$-approximative w.h.p. The reason is that w.h.p. *all* maximal independent sets of $G = G_{n,1/2}$ are of size $\geqslant (\frac{1}{2} + o(1))\alpha(G)$ (see [5, p. 288]). Remarkably, no polynomial time algorithm is known to find an independent set of size $\geqslant (1 + \Omega(1))\log_2(n)$ w.h.p.

Although on input $G_{n,1/2}$ the greedy heuristic w.h.p. achieves a $(2 + o(1))$-approximation, there is a serious drawback. Namely, the greedy heuristic does not compute an upper bound on the independence number, and hence cannot *distinguish* between such input graphs $G = G_{n,p}$ with a 'low' $\alpha(G)$ as in (1.1), and 'exceptional' inputs with much larger $\alpha(G)$. In fact, it is easy to figure out graphs G for which the ratio between $\alpha(G)$ and the size of the independent set computed by the greedy heuristic is $\Omega(n)$.

To cope with the NP-hardness of approximating the independence number, Krivelevich and Vu [23] have proposed an approximation algorithm that runs in polynomial *expected* time applied to $G_{n,p}$, rather than in worst-case polynomial time. Let $R_{\mathscr{A}}(G)$ denote the running time of an algorithm $\mathscr{A}$ on input G. We say that $\mathscr{A}$ has a *polynomial expected running time* applied to $G_{n,p}$, if there is a constant $l > 0$ such that $\sum_G R_{\mathscr{A}}(G)\mathrm{P}(G = G_{n,p}) = O(n^l)$, where the sum ranges over all graphs on n vertices. The algorithm of Krivelevich and Vu achieves an approximation guarantee of $O((np)^{1/2}(\ln np)^{-1})$ on *all* input graphs, and runs in polynomial expected time applied to $G_{n,p}$, provided that $n^{-1/2} \leqslant p \leqslant 0.99$. Coja-Oghlan and Taraz [8, 10] have given a similar algorithm that works for the entire range of edge probabilities $1/n \leqslant p \leqslant 0.99$.

The algorithms suggested in [8, 10, 23] essentially combine the greedy heuristic with a technique for computing an upper bound on the independence number. For instance, the algorithm of Krivelevich and Vu [23] uses the greedy heuristic to find an independent set S of size $\Omega(\ln(np)/p)$. Then, the algorithm computes the largest eigenvalue λ of a certain auxiliary matrix, which is an upper bound on the independence number. If $\lambda \leqslant 4(n/p)^{1/2}$, then the output is just S, the cardinality of S being within the desired approximation ratio. However, if $\lambda > 4(n/p)^{1/2}$, the algorithm has a super-polynomial running time. Still, the expected running time remains polynomial, because on input $G_{n,p}$, the probability that $\lambda > 4(n/p)^{1/2}$ is extremely small.

1.2. Semirandom models

However, there are several reasons why $G_{n,p}$ may fail to provide an appropriate model of the 'average' instances we are confronted with. First, since all inclusion-maximal independent sets of $G_{n,p}$ are of size (1.1) up to a constant factor w.h.p., $G_{n,p}$ cannot model instances that actually contain one very large independent set that we are to find. Indeed, in the case $\alpha(G) > 4(n/p)^{1/2}$, the approximation algorithms for $G_{n,p}$ [8, 10, 23] have an exponential running time. Moreover, $G_{n,p}$ enjoys several well-studied structural properties (see [5, 18]) that the 'average' instances we have in mind may not have. For example, if $0 < p < 1$ is constant, then w.h.p. all vertex degrees of $G_{n,p}$ are roughly np, and the largest eigenvalue of the adjacency matrix is separated from the second eigenvalue (see [15]). Hence, it would be desirable to study a model that:

- covers the case of 'average' instances containing some large independent set, and
- describes instances that lack some typical properties of $G_{n,p}$.

In this paper, we study a *semirandom* model for the maximum independent set problem that meets the above requirements. The first semirandom models (for the k-colouring problem) have been studied by Blum and Spencer [4]. As instances of semirandom graph problems consist of a random share and a worst-case part constructed by an adversary, such models intermediate between the worst-case paradigm and random graphs.

The following semirandom model has been proposed by Feige and Krauthgamer [12]. Let $V = \{1, \dots, n\}$. First, a set $S \subset V$ of cardinality $\Omega(n^{1/2})$ is chosen uniformly at random. Then, every edge $\{v, w\}$, $v \in V$, $w \in V \setminus S$, is included in the graph G_0 with probability $p = 1/2$, independently of all other edges. Thus, G_0 is a random graph $G_{n,1/2}$ with a planted independent set of size $\Omega(n^{1/2})$. Finally, the adversary may add to G_0 further edges $\{v, w\}$, $v \in V$, $w \in V \setminus S$, thereby completing the instance G. Since S is an independent set of G, we have $\alpha(G) \geqslant \Omega(n^{1/2})$. Observe that the adversary can change the vertex degrees, the eigenvalues of the adjacency matrix, and other parameters of G_0. The algorithm studied by Feige and Krauthgamer always has a polynomial running time and finds the hidden independent set S with high probability over the choice of G_0.

In this paper, we study the following two semirandom models. Given an edge probability $0 < p = p(n) < 1$ and a number $k = k(n)$, the random share $G_{n,p,k}$ of the first semirandom model $G^*_{n,p,k}$ is constructed as follows.

M1. A set $S \subset V$ consisting of k vertices is chosen uniformly at random.
M2. The random graph $G_0 = G_{n,p,k}$ is obtained by including every edge $\{v, w\}$, $v \in V$, $w \in V \setminus S$, with probability p independently of all other edges.

Given G_0, the adversary completes the instance.

M3. The adversary may add to G_0 edges $\{v, w\}$ where $v \in V$ and $w \in V \setminus S$, thereby obtaining $G = G^*_{n,p,k}$.

Clearly, S remains an independent set in G, whence $\alpha(G) \geqslant k$. Thus, $G^*_{n,p,k}$ models instances of the maximum independent set problem that do contain a very large independent set. Note that $G^*_{n,\frac{1}{2},\Omega(\sqrt{n})}$ coincides with the model treated in [12].

Instances of our second semirandom model $G^*_{n,p}$ are constructed as follows.

M1′. Choose a random graph $G_0 = G_{n,p}$.
M2′. The adversary may add to G_0 arbitrary edges, thereby completing the instance $G = G^*_{n,p}$.

Thus, $G^*_{n,p}$ describes instances of the maximum independent set problem that do not contain a very large independent set. (To be precise, for values of $k \leqslant n$ larger than (1.1), the probability that $\alpha(G^*_{n,p}) \geqslant k$ is nonzero but $o(1)$.)

Let $G_0 = G_{n,p,k}$. Let $\mathscr{I}(G_0)$ signify the set of all graphs that can be obtained from G_0 according to M3. We say that *the expected running time of an algorithm $\mathscr{A}$ applied to $G^*_{n,p,k}$ is polynomial* if there is a constant $l > 0$ such that $\sum_{G_0} \mathrm{P}(G_0 = G_{n,p,k}) \cdot \max_{G \in \mathscr{I}(G_0)} R_{\mathscr{A}}(G) = O(n^l)$. Moreover, we say that the semirandom graph $G^*_{n,p,k}$ enjoys a certain property P *with high probability* if

$$\lim_{n\to\infty} \mathrm{P}\big(G_0 = G_{n,p,k} \text{ is such that } P \text{ holds for all } G \in \mathscr{I}(G_0)\big) = 1.$$

We have similar definitions for the model $G^*_{n,p}$. The main result of this paper is the following theorem.

Theorem 1.1. *Suppose that $\ln(n)^2/n \leqslant p \leqslant 0.99$ and that $k \geqslant C(n/p)^{1/2}$ for a sufficiently large constant C. There is an algorithm* `Find` *that satisfies the following conditions.*

(1) *For any input graph G,* `Find`(G,k) *outputs an independent set of size $\geqslant k$ if $\alpha(G) \geqslant k$. If $\alpha(G) < k$, then* `Find`(G,k) *outputs $\emptyset$.*

(2) `Find`(G,k) *runs in polynomial expected time, when applied to $G = G^*_{n,p,k}$ and to $G = G^*_{n,p}$*

Thus, Theorem 1.1 extends the result of Feige and Krauthgamer [12] in the two following respects.

- The algorithm in [12] *always* runs in polynomial time and succeeds in finding the planted independent set *with high probability* on input $G^*_{n,p,k}$. In contrast, `Find` satisfies a stronger requirement: applied to $G^*_{n,p,k}$, `Find` *always* succeeds in finding an independent set of size $\geqslant k$, and has a polynomial *expected* running time. Whereas, *e.g.*, for $p = \Theta(1)$ and $k = \Omega(n^{1/2})$ it is easy to modify the approach of Feige and Krauthgamer to satisfy the stronger requirement as well, for smaller values of p the stronger requirement is significant (see Section 1.3 for more details).
- The running time of `Find` is also polynomial when applied to $G^*_{n,p}$. Hence, `Find` can *distinguish* between $G^*_{n,p}$ and $G^*_{n,p,k}$ *efficiently.*

However, in contrast to the algorithm of Feige and Krauthgamer, `Find`$(G^*_{n,p,k}, k)$ does not provide a guarantee that its output is a largest independent set (see Section 5 for more details).

Furthermore, `Find` complements the work of Krivelevich and Vu [23] and Coja-Oghlan and Taraz [8, 10] on the independent set problem on $G_{n,p}$, as follows.

- `Find` can exhibit a large independent set efficiently (*i.e.*, in expected polynomial time), if there is any. In contrast, the running time of the algorithms studied in [8, 10, 23] is exponential if the input instance contains a large independent set.
- `Find` can handle *semirandom* instances, and is thus applicable to a wider class of input distributions than just $G_{n,p}$.

Why are algorithms with a polynomial expected running time interesting? Imagine the quest of the algorithm for either a large independent set or a proof that there is no such set as a search tree. Since the algorithm is supposed to work on all instances properly, this search tree can be of polynomial or exponential size, or anything in between. While a heuristic that just works well with high probability may either fail or produce an enormous search tree if the input instance lacks some 'typical' properties, an algorithm with polynomial expected running time must be such that minor 'atypical' defects in the input instance increase the running time only a little. Thus, on the one hand the interest lies in algorithmic techniques that lead to a search tree whose size is distributed 'smoothly' in such a way that it is small on average. On the other hand, we need to invent methods to analyse the size of the search tree. In general, such an analysis requires a more careful investigation than a proof that the algorithm works with high probability.

The following hardness result complements Theorem 1.1.

Theorem 1.2. *Let* $0 < \varepsilon < 1/100$ *be a constant. Suppose that* $\ln(n)^2 \leqslant np \leqslant n^{1/2}$*, and let* $k = (1-\varepsilon)\ln(n)/p$*. There is no polynomial time algorithm that, applied to* $G^*_{n,p,k}$*, finds an independent set of size* $\geqslant k$ *w.h.p., unless* NP $\subset$ RP.

Theorem 1.2 shows that for $p = \ln(n)^2/n$ the positive result (Theorem 1.1) is essentially best possible (up to the precise value of the constant C). Nevertheless, as p increases, the gap grows between the upper bound on the size of independent sets that can be found efficiently (see Theorem 1.2), and the size of the hidden independent set required by Theorem 1.1. Finally, observe that for edge probabilities $p > n^{-1/2}$ the statement of Theorem 1.2 is void, as in this range $(1-\varepsilon)\ln(n)/p$ is smaller than the independence number (1.1) of $G_{n,p}$. The proof of Theorem 1.2 follows [11, proof of Theorem 2] (see Section 1.3 for more detailed comments).

We present the algorithm `Find` and its analysis in Section 2. In Section 3 we prove Theorem 1.2. Section 4 contains the proof of a technical lemma that is part of the analysis of `Find`. Finally, our conclusion is in Section 5.

1.3. Techniques and further related work

The first to study the planted independent set model $G_{n,1/2,k}$ was Kučera [24], who observed that in the case $k = \Omega(n\ln n)^{1/2}$ one can recover the planted independent set simply by picking the k vertices of least degree. However, this approach fails if $k = o(n\ln(n))^{1/2}$. Furthermore, Jerrum [19] has shown that on $G_{n,p,k}$, where $p = 1/2$ and $k = n^\beta$ for a constant $\beta < 1/2$, (a restricted variant of) simulated annealing with high probability fails to find an independent set of size $> (1+\varepsilon)\log_2 n$ in polynomial time. Alon, Krivelevich and Sudakov [2] have presented an algorithm based on spectral techniques that on input $G_{n,1/2,k}$ finds the hidden independent set w.h.p., if $k = \Omega(n)^{1/2}$. No polynomial time algorithm is known to cope with the case $k = o(n^{1/2})$. However, it is easy to recover a planted independent set S of size $k = o(n^{1/2})$ in time $n^{O(\ln n)}$ (see [2]): enumerate all subsets T of size $10\ln(n)$ and consider the set of all non-neighbours of T; if $T \subset S$, then with high probability the non-neighbourhood of T is precisely S. Using spectral techniques, McSherry [26] has studied a general partitioning problem on random graphs. In particular, on input $G_{n,p,k}$ McSherry's heuristic recovers the planted independent set w.h.p., provided that $k = \Omega(n^{1/2})$.

The semirandom model $G^*_{n,p,k}$ with $p = 1/2$ and $k = \Omega(n^{1/2})$ was first considered by Feige and Krauthgamer [12]. Their algorithm relies on the fact that in the case $k \geqslant \Omega(n^{1/2})$ with high probability $\vartheta(G^*_{n,p,k}) = k$, where ϑ denotes the Lovász number (see [16, 22]). As the Lovász number can be seen as a semidefinite programming ('SDP') relaxation of the independence number, the inequality $\vartheta(G^*_{n,p,k}) \geqslant k$ holds trivially. Since it is not hard to extend the result of Feige and Krauthgamer [12] to smaller values of p (say, $p \geqslant n^{\varepsilon-1}$) and $k \geqslant \Omega(n/p)^{1/2}$, one obtains an algorithm that recovers the hidden independent set with high probability. Moreover, in the case $p = 1/2$ and $k = \Omega(n^{1/2})$, the method immediately yields an algorithm with polynomial expected running time, because the hidden independent set can most probably be recovered in time $n^{O(\ln(n))}$ as indicated above. However, at least in the case $p \leqslant n^{-1/2}$, the approach of Feige and Krauthgamer (even in combination with the concentration result [3] on the eigenvalues of random symmetric matrices) does not seem to lead to an algorithm with polynomial expected running time.

In the semirandom model $G^*_{n,p,k}$, the adversary can cause the spectral heuristic from [2] to fail to recover an independent set of size $\geqslant k$ (say, $p = 1/2$ and $k = \Omega(n^{1/2})$). Further, even in the case $p = 1/2$, $k = \Omega(n \ln n)^{1/2}$, the degree trick from [24] does not work on $G^*_{n,p,k}$. Thus, in a sense $G^*_{n,p,k}$ requires more robust algorithmic techniques than $G_{n,p,k}$.

Feige and Kilian [11] have presented an algorithm for finding an independent set of size $\Omega(n)$ hidden in a semirandom graph with high probability. Their model is even more adversarial than $G^*_{n,p,k}$, as random edges are only included between the hidden independent set S and $V \setminus S$ (but not between vertices inside $V \setminus S$). Complementing the result of Feige and Kilian, the author has given an algorithm for recovering an independent set of size $\Omega(n)$ in polynomial expected time [9]. The algorithms in [9, 11] are based on semidefinite programming techniques developed in [1, 20] (*e.g.*, computing the Lovász number ϑ and rounding fractional solutions via random hyperplanes). However, these SDP rounding techniques do not seem to apply if the planted independent set has size $o(n)$.

Although the algorithm `Find` for Theorem 1.1 is also based on computing Lovász's SDP relaxation ϑ of the independence number, we suggest a somewhat different approach from those of the previous papers [2, 9, 11, 12]. Indeed, we do not need to derive the probable value of $\vartheta(G^*_{n,p,k})$ explicitly, which has been the main technical difficulty in [12]. Instead, we just rely on results on the Lovász number of random graphs $G_{n,p}$, in particular on a concentration result on $\vartheta(G_{n,p})$ from [8]. Applying to an optimal fractional solution a more direct (deterministic) rounding technique than rounding via random hyperplanes as in [9, 11], the initial step of `Find`$(G^*_{n,p,k}, k)$ computes an independent set I. Using the concentration result on $\vartheta(G_{n,p})$, we prove that with probability $\geqslant 1 - \exp(-20k)$ this set I already contains 99% of the vertices in the hidden independent set S and only a few vertices in $V \setminus S$. This fact is crucial in order to obtain a polynomial expected running time. To remove the vertices that do not belong to S from I and to recover the remaining part $S \setminus I$, `Find` employs a procedure `Purify`, which relies on flow techniques. The flow techniques follow ideas from [11] and extend the approach used in [9] (see the remark at the end of Section 2.3 for more specific comments).

The proof of Theorem 1.2 is very similar to the proof of a hardness result of Feige and Kilian [11, Theorem 2] on finding an independent set of size $\Omega(n)$ in a semirandom graph. The main difference is that the semirandom graph $G^*_{n,p,k}$ treated in the present paper is a bit less adversarial than the model studied in [11], as in $G^*_{n,p,k}$ there are random edges present inside $V \setminus S$ (see Step M2 of the definition of $G^*_{n,p,k}$).

An extended abstract version of this paper, from which most of the proofs are omitted, has appeared in the proceedings of *STACS 2003*.

1.4. Notation

Throughout, we let $V = \{1, \ldots, n\}$. If X is a set, then $\#X$ signifies the cardinality of X. We omit floor and ceiling signs when these do not affect the arguments. Moreover, we frequently assume implicitly that n is sufficiently large.

Let G be a graph. We let $V(G)$ (resp. $E(G)$) denote the vertex (resp. edge) set of G. For $X \subset V(G)$, we let $N(X) = N_G(X)$ denote the *neighbourhood* $\{y \in V(G) | \{x, y\} \in E(G)$ for some $x \in X\}$ of X. Furthermore, $G[X]$ signifies the subgraph of G induced on X.

By $\langle \xi, \eta \rangle$ we denote the scalar product of two vectors ξ, η. Furthermore, $\vec{1}$ denotes the vector with all components equal to 1 (in any dimension).

If $G = G^*_{n,p,k}$, then $G_0 = G_{n,p,k}$ denotes the random share contained in G, and S signifies the hidden independent set (see the definition M1–M3 of $G^*_{n,p,k}$). Similarly, if $G = G^*_{n,p}$, then we let G_0 denote the random graph $G_{n,p}$ contained in G (see the definition M1′–M2′ of $G^*_{n,p}$).

2. The algorithm `Find`

2.1. Outline

Throughout Section 2, we assume that $\ln(n)^2 \leqslant np \leqslant 0.99n$. Suppose that the input graph is $G = G^*_{n,p,k}$, where $k \geqslant C(n/p)^{1/2}$ for a sufficiently large constant C. As a first step, `Find`(G, k) runs a subroutine `Filter`, which uses semidefinite programming in order to determine a set I of vertices of G that contains a large share of the hidden independent set S but only a few vertices of $V \setminus S$. If `Filter` succeeds in finding such a set I, then `Find` applies a further subroutine `Purify` to I in order to recover the entire set S (or another independent set of size $\geqslant k$). The subroutine `Purify` relies on a certain expansion property enjoyed with high probability by the random share G_0 contained in the input instance G.

However, since `Find` is supposed to find an independent set of size k on *any* input graph G with $\alpha(G) \geqslant k$, `Find` must take into account that the expansion property may be violated to a certain degree. To this end, `Find` uses the variable η. In the beginning, `Find` assumes that the expansion property is perfectly satisfied ($\eta = 0$). If the input graph resists the attempts of `Find` to exhibit an independent set of size $\geqslant k$, then `Find` increases η slowly. As the parameter η grows, the running time of the subroutine `Purify` increases, as does the probability that `Purify` succeeds. Finally, if `Purify` does not manage to exhibit an independent set of size $\geqslant k$ for any value of $\eta \leqslant k/2$, then `Find` calls the subroutine `Exact` that will always find an independent set of size $\geqslant k$ (if there is any).

The procedure `Filter` and its analysis will be given in Section 2.2. Showing that the output I of `Filter` with extremely high probability contains most vertices of S is the crucial point in the analysis of `Find`. Then, in Sections 2.3 and 2.4 we present the procedures `Purify` and `Exact`. Finally, we prove Theorem 1.1 in Section 2.5.

Algorithm 1. `Find`(G,k)

Input: A graph $G=(V,E)$ and an integer $1 \leqslant k \leqslant n$. *Output:* A subset of V.

```
1. Let $I = \texttt{Filter}(G,k)$. If $I = \emptyset$, then terminate with output $\emptyset$.
2. For $\eta = 0,\dots,k/2$ do:
3.     If $\mathscr{S} = \texttt{Purify}(G,I,\eta,k) \neq \emptyset$, then output an arbitrary element of $\mathscr{S}$.
4. Output $\tilde{S} = \texttt{Exact}(G,k)$.
```

Figure 1. The algorithm `Find`.

Algorithm 2. `Filter`(G,k)

Input: A graph $G=(V,E)$ and an integer $1 \leqslant k \leqslant n$. *Output:* A subset I of V.

```
1. Compute a vector $x = (x_v)_{v\in V} \in \mathrm{TH}(G)$ such that $\langle \vec{1}, x\rangle \geqslant \vartheta(G) - 1$. If $\langle \vec{1}, x\rangle < k-1$, then
   return $\emptyset$.
2. Return $I = \{v \in V \mid x_v \geqslant 2/3\}$.
```

Figure 2. The algorithm `Filter`.

2.2. The subroutine `Filter`

The procedure `Filter` is based on computing the *Lovász number* $\vartheta(G)$, which can be considered as an SDP relaxation of the independence number. Let us briefly recall the definition of ϑ (see [16, 22] for thorough treatments). Let $G=(V,E)$ be a graph, and let d be a positive integer. An *orthogonal labelling* of G is a tuple $(a_v)_{v\in V}$ of vectors in $\mathbb{R}^d$ such that, for any two vertices $v,w \in V$, $v \neq w$, with $\{v,w\} \notin E$ we have $\langle a_v, a_w\rangle = 0$ (this definition follows [22]). The *cost* of a d-dimensional vector $a = (a_1,\dots,a_d)^T$ is

$$c(a) = \begin{cases} a_1^2\|a\|^{-2} & \text{if } a \neq 0,\\ 0 & \text{otherwise.}\end{cases}$$

Let $\mathrm{TH}(G)$ be the set of all vectors $(x_v)_{v\in V} \in \mathbb{R}^n$ with nonnegative coordinates such that for all orthogonal labellings $(a_v)_{v\in V}$ of G we have $\sum_{v\in V} x_v c(a_v) \leqslant 1$. Then $\mathrm{TH}(G) \subset [0,1]^n$ is a compact convex set, which can be seen as a relaxation of the stable set polytope (see [16]). The Lovász number of G is $\vartheta(G) = \max\{\langle \vec{1}, x\rangle \mid x \in \mathrm{TH}(G)\}$. It is well known that $\alpha(G) \leqslant \vartheta(G) \leqslant \chi(\bar{G})$, *i.e.*, $\vartheta(G)$ is 'sandwiched' between the independence number of G and the chromatic number of the complement of G. Moreover, an immediate consequence of the definition is that ϑ is *monotone*:

$$\text{If } H_1 = (V,E_1) \text{ is a subgraph of } H_2 = (V,E_2)\text{, then } \vartheta(H_2) \leqslant \vartheta(H_1). \tag{2.1}$$

Using the ellipsoid method, one can compute a vector $x' \in \mathrm{TH}(G)$ such that $\vartheta(G) - \langle \vec{1}, x'\rangle \leqslant 1$ in polynomial time [16, p. 294].

The following proposition summarizes the analysis of `Filter`.

Proposition 2.1. *Suppose that $k \geqslant C(n/p)^{1/2}$ for a sufficiently large constant $C > 0$. Let I be the output of* `Filter`(G,k).

(1) *For all input graphs G, I is an independent set. Moreover, if $I = \emptyset$, then $\alpha(G) < k$.*
(2) *If $G = G^*_{n,p,k}$, then with probability $\geqslant 1 - \exp(-20k)$ we have*

$$\#I \cap S \geqslant \frac{99k}{100}. \tag{2.2}$$

(3) *If $G = G^*_{n,p}$, then with probability $\geqslant 1 - \exp(-20k)$ we have $I = \emptyset$.*

Thus, Proposition 2.1 claims that `Filter`$(G^*_{n,p,k})$ w.h.p. finds an independent set I that contains most vertices of the hidden independent set S. In addition, we shall see in Section 2.3 that such a set (most probably) contains only a few vertices of $V \setminus S$.

Lemma 2.2. *For any graph G the set I computed in Step 2 of* `Filter`(G,k) *is independent.*

Proof. Let $G = (V,E)$. Assume that there are two vertices $v_1, v_2 \in I$ such that $\{v_1, v_2\} \in E$. Then we obtain a (one-dimensional) orthogonal labelling $(a_v)_{v\in V}$ of G by letting $a_{v_1} = a_{v_2} = 1$, and $a_v = 0$ for $v \in V \setminus \{v_1, v_2\}$. Let $x \in \mathrm{TH}(G)$ be the vector computed in Step 1 of `Filter`. As $v_1, v_2 \in I$, we obtain $1 \geqslant \sum_{v\in V} x_v c(a_v) = x_{v_1} + x_{v_2} \geqslant \frac{4}{3}$, a contradiction. □

Lemma 2.3. *Suppose that $k \geqslant C(n/p)^{1/2}$ for some sufficiently large constant $C > 0$. Let $G = G^*_{n,p,k}$. Then, with probability $\geqslant 1 - \exp(-20k)$ the set I computed in Step 2 of* `Filter`(G,k) *satisfies $\#I \cap S > 99k/100$.*

To prove Lemma 2.3, we need the following fact concerning the Lovász number of $G_{n,p}$.

Lemma 2.4. *Assume that $t > c_1(n/p)^{1/2}$ for a certain constant $c_1 > 0$. Then*

$$\mathrm{P}\big(\vartheta(G_{n,p}) > t\big) \leqslant \exp(-20t).$$

Proof. This is an immediate consequence of [8, Theorems 1 and 4]. □

Lemma 2.4 shows that w.h.p. $\vartheta(G_{n,p})$ is much smaller than $\vartheta(G^*_{n,p,k}) \geqslant k \geqslant C(n/p)^{1/2}$ (provided that C is large enough).

Proof of Lemma 2.3. As a first step, we shall prove that in the case $\#I \cap S \leqslant 99k/100$ we have $\vartheta(G_0 - S) \geqslant \frac{k}{300} - 1$. Indeed, let $x = (x_v)_{v\in V} \in \mathrm{TH}(G)$ be the vector computed in Step 1. If $\#I \cap S \leqslant 99k/100$, then

$$\sum_{s\in S} x_s = \sum_{s\in I\cap S} x_s + \sum_{s\in S\setminus I} x_s \leqslant \#I \cap S + \frac{2\#S \setminus I}{3} \leqslant \frac{299k}{300}. \tag{2.3}$$

Since $(x_v)_{v\in V\setminus S} \in \mathrm{TH}(G - S)$, we have

$$\sum_{v\in V\setminus S} x_v \leqslant \vartheta(G - S). \tag{2.4}$$

Combining (2.3) and (2.4), we obtain

$$k \leqslant \alpha(G) \leqslant \vartheta(G) \leqslant 1 + \sum_{v \in V} x_v \leqslant 1 + \vartheta(G - S) + \frac{299k}{300} \leqslant 1 + \vartheta(G_0 - S) + \frac{299k}{300},$$

where the last inequality follows from the monotonicity (2.1) of ϑ. Thus, $\vartheta(G_0 - S) \geqslant \frac{k}{300} - 1$.

To conclude the proof of Lemma 2.3, observe that $G_0 - S$ is just a random graph $G_{n-k,p}$. Therefore, if $C \geqslant 400c_1$, then Lemma 2.4 yields our assertion. □

Proof of Proposition 2.1. The first two assertions in Proposition 2.1 follow from Lemmas 2.2 and 2.3 immediately. Now assume that the input graph is $G = G^*_{n,p}$. Then, by Lemma 2.4 and by the monotonicity of the Lovász number (2.1), $\mathrm{P}\big(\vartheta(G) \geqslant k-1\big) \leqslant \exp(-20k)$. Hence, with probability $\geqslant 1 - \exp(-20k)$ `Filter`$(G^*_{n,p}, k)$ outputs $\emptyset$. □

2.3. The subroutine `Purify`

Let $G = G^*_{n,p,k}$ be the input of `Find`. Suppose that `Filter`(G,k) has found an independent set I that contains 99% of the vertices in S (see Proposition 2.1). To recover the entire set S, `Purify` makes use of network flow techniques and the fact that the random bipartite graph consisting of the $V \setminus S$–S-edges of G_0 is a good expanding graph w.h.p.; that is, w.h.p. every set $T \subset V \setminus S$ of cardinality $\leqslant \frac{k}{2d}$ has at least $d\#T$ neighbours in S, for all d in a certain range.

Since `Purify` only relies on the aforementioned expansion property, we analyse the procedure in a slightly more general setting. Let $G = (V, E)$ be a graph, and let $R \subset V$ be a set of cardinality $k \geqslant 1$. To measure how far from being a good expanding graph G is (w.r.t. the sets R and $V \setminus R$), we define the *defect* $\mathrm{def}_G(R)$:

D1. If there is some $U \subset R$ such that $\#U \geqslant \frac{k}{2}$ and $\#V \setminus (R \cup N(U)) > \frac{k^2}{200n}$, then we let $\mathrm{def}_G(R) = \frac{k}{2}$.

D2. Otherwise, we let $\mathrm{def}_G(R)$ be the least number $0 \leqslant \eta \leqslant \frac{k}{2}$ such that for all $6 \leqslant d \leqslant \lceil 50n/k \rceil$ the following holds: every set $T \subset V \setminus R$ of size $\#T \leqslant \frac{k}{2d}$ has a d-fold matching to R with defect $\leqslant \eta$.

Here for $T \subset V$ and $6 \leqslant d \leqslant \lceil \frac{50n}{k} \rceil$ a *d-fold matching from T to R with defect* $\leqslant \eta$ is a set M of T–R-edges that satisfies the following conditions.

- At least $\#T - \eta$ vertices in T are incident with precisely d edges in M. The $\leqslant \eta$ remaining vertices in T, which are called *defect vertices*, are not incident with an edge in M.
- No vertex in R is incident with more than one edge in M.

Thus, a d-fold matching consists of disjoint stars on $d+1$ vertices with centres in T. The following lemma is an easy consequence of Hall's theorem.

Lemma 2.5. *Let $6 \leqslant d \leqslant \lceil \frac{50n}{k} \rceil$. If all sets $X \subset V \setminus R, \#X \leqslant \frac{k}{2d}$ satisfy $\#N(X) \cap R \geqslant d\#X - \eta$, then every set $T \subset V \setminus R$ of cardinality $\leqslant \frac{k}{2d}$ admits a d-fold matching to R with defect $\leqslant \eta$.*

The following proposition summarizes the analysis of `Purify`.

Proposition 2.6. *Let $G = (V, E)$ be a graph, and let $R \subset V$ be an independent set of cardinality k. Suppose that $\mathrm{def}_R(G) < \frac{k}{2}$. Let $I \subset V$ be an independent set such that $\#I \cap R \geqslant \frac{99k}{100}$. Let $0 \leqslant \eta \leqslant k/2$. Then the following holds.*

(a) *If $\eta \geqslant \mathrm{def}_R(G)$, then the output $\mathscr{S}$ of* $\texttt{Purify}(G, I, \eta, k)$ *contains R as an element.*

(b) *The running time of* $\texttt{Purify}(G, I, \eta, k)$ *is $\leqslant n^{O(1)}\binom{k}{\eta}^{14}$.*

To state the procedure `Purify`, we need some notation. A *network* N consists of a vertex set $V(N)$, an arc set $A = A(N) \subset V(N) \times V(N)$, a source $s \in V(N)$, a sink $t \in V(N) \setminus \{s\}$, and a capacity $c : A \to \mathbb{Z}_{\geqslant 0}$. A *flow* in N is a map $f : A \to \mathbb{Z}_{\geqslant 0}$ such that $f(a) \leqslant c(a)$ for all $a \in A$, satisfying Kirchhoff's law

$$\forall v \in V(N) \setminus \{s, t\} : \sum_{(v,w)\in A} f(v, w) = \sum_{(w,v)\in A} f(w, v). \tag{2.5}$$

Moreover, the *value* of f in N is $w(f) = \sum_{(s,v)\in A} f(s, v) - \sum_{(v,s)\in A} f(s, v)$. A *maximum flow* is a flow of maximum value, and can be computed in polynomial time [27, pp. 154–155]. If f_1, f_2 are flows in N, then we can define a flow $f_1 + f_2$ by letting $(f_1 + f_2)(a) = f_1(a) + f_2(a)$, provided that $f_1(a) + f_2(a) \leqslant c(a)$ for all $a \in A$.

$\texttt{Purify}(G, I, \eta, k)$ proceeds in two phases. In the first phase (Steps 1–2), `Purify` attempts to identify and remove the vertices in $I \setminus R$, thereby obtaining a set I'. We analyse the first phase in Lemma 2.7 below. In the second phase (Steps 3–8), the aim is to enlarge $I' \subset R$ several times, each time adding to the current set $I'' \subset R$ at least half of the remaining vertices in $R \setminus I''$. Thus, after at most $\log_2(n)$ steps, we have $I'' = R$, *i.e.*, R is recovered. Lemma 2.8 is devoted to the analysis of the second phase. Observe that the output $\mathscr{S}$ of `Purify` either consists of independent sets of size k or is empty.

Lemma 2.7. *Under the assumptions of Proposition 2.6* (a) *there exists a set $D \subset I$, $\#D \leqslant \eta$, such that the set I' computed in Step 2 of* $\texttt{Purify}(G, I, \eta, k)$ *satisfies $I' \subset R$ and $\#I' \geqslant \frac{97k}{100}$.*

Consider the network N constructed in Step 2 of `Purify`. Given a flow g in N and a set $U \subset I \setminus D$, we define the *restricted* flow g_U as follows. For every edge $\{v, w\} \in E$, where $v \in I \setminus D$ and $w \in V$, we let

$$g_U(s_v, t_w) = \begin{cases} 0 & \text{if } v \in I \setminus (D \cup U), \\ g(s_v, t_w) & \text{if } v \in U. \end{cases}$$

Furthermore,

$$g_U(s, s_v) = \sum_{u\in V:\{u,v\}\in E} g_U(s_v, t_u) \quad (v \in I \setminus D),$$

$$g_U(t_w, t) = \sum_{u\in I\setminus D:\{u,w\}\in E} g_U(s_u, t_w) \quad (w \in V).$$

Thus, g_U transports the same amount of flow from s through $(s_v)_{v\in U}$ via $(t_w)_{w\in V}$ to t as g does, but it does not carry any flow through $(s_v)_{v\in I\setminus(D\cup U)}$. Note that the flow that g_U sends through the arcs (s, s_v) and (t_w, t) is simply determined by Kirchhoff's law (2.5). For

```
Algorithm 3. Purify($G, I, \eta, k$)
Input: A graph $G = (V, E)$, integers $\eta, k$, $I \subset V$.   Output: A set $\mathscr{S}$ of subsets of $V$.

1.  Let $\mathscr{S} = \emptyset$. If $\#I > 2k$, then return $\emptyset$.
    Otherwise, for all $D \subset I$, $\#D \leqslant \eta$ do
2.      Construct the following network $N$:
        • The vertices of $N$ are $s$, $t$, $s_v$ for $v \in I \setminus D$, and $t_w$ for $w \in V$.
        • The arcs of $N$ are $(s, s_v)$ for $v \in I \setminus D$, $(t_w, t)$ for $w \in V$, and $(s_v, t_w)$ if $\{v, w\} \in E$.
        • The capacities are $c(s, s_v) = \lceil \frac{50n}{k} \rceil$, $c(t_w, t) = 1$, $c(s_v, t_w) = 1$ if $\{v, w\} \in E$.
        Compute a maximum flow $f$ in $N$, let $L = \{v \in I \setminus D \mid f(s, s_v) = c(s, s_v)\}$, and set
        $I' = I \setminus (L \cup D)$.
3.      If $\tilde{V} = V \setminus N(I')$ satisfies $\#\tilde{V} \leqslant 2k$ then
4.          For each set $Y \subset \tilde{V}$, $\#Y \leqslant 6\eta$, such that $I' \cup Y$ is an independent set of
            cardinality $k$ add $I' \cup Y$ to $\mathscr{S}$.
5.          For all $D' \subset \tilde{V}$, $\#D' \leqslant \eta$, do
                Let $I'' = I'$. For $\tau = 0, 1, \ldots, \lceil \log_2(n) \rceil$ do
                    Let $V' = V \setminus (N(I'') \cup D')$. Construct the following network $N'$.
                    • The vertices of $N'$ are $s'$, $t'$, $s'_v$ for $v \in V' \setminus I''$, and $t'_w$ for $w \in V'$.
                    • The arcs of $N'$ are $(s', s'_v)$ for $v \in V' \setminus I''$, $(t'_w, t')$ for $w \in V'$, and
                      $(s'_v, t'_w)$ if $\{v, w\} \in E$.
                    • The capacities are $c(s', s'_v) = 6$, $c(t'_w, t') = 1$, $c(s'_v, t'_w) = 1$ if $\{v, w\} \in$
                      $E$.
                    Compute a maximum flow $f'$ in $N'$. Let
                        $L' = \{v \in V' \setminus I'' \mid f'(s', s'_v) = c(s', s'_v)\}$
                    and $I'' = V' \setminus L'$. If $I''$ is an independent set of cardinality $k$ then add
                    $I''$ to $\mathscr{S}$.
6.  Return $\mathscr{S}$.
```

Figure 3. The algorithm Purify.

a flow g' in the network N' constructed in Step 5 and $U' \subset V' \setminus I''$ we define the restricted flow $g'_{U'}$ analogously.

Proof of Lemma 2.7. Let $d = \lceil \frac{50n}{k} \rceil$. Since $\mathrm{def}_R(G) < \frac{k}{2}$ and because I is an independent set, we have $\#I \setminus R \leqslant \frac{k}{2d}$ (by D1). Hence, by Lemma 2.5 the set $I \setminus R$ admits a d-fold matching M^* to R with defect $\leqslant \eta$. Let D be the set of defect vertices.

The matching M^* induces a flow h in the network N as follows. For each $e = \{v, w\} \in M^*$, where $v \in I \setminus (R \cup D)$, $w \in R$, we define a flow h_e in N by letting

$$h_e(s_v, v) = h_e(v, w) = h_e(w, t_w) = 1,$$
$$h_e(a) = 0 \text{ for all arcs } a \notin \{(s_v, v), (v, w), (w, t_w)\}.$$

Then, $h = \sum_{e \in M^*} h_e$ is a flow of value $w(h) = d\#I \setminus (R \cup D)$.

Let f be the maximum flow computed in Step 2 of `Purify`(G, I, η, k). Then the restricted flow $f_{I \setminus (R \cup D)}$ satisfies

$$w(f_{I \setminus (R \cup D)}) = d\#I \setminus (R \cup D). \tag{2.6}$$

For $f_{I \cap R} + h$ is a flow in N of value $w(f_{I \cap R} + h) = w(f_{I \cap R}) + w(h)$, because R is an independent set. Hence, (2.6) follows from the maximality of f. As a consequence, $L \supset I \setminus (R \cup D)$, so that $I' \subset R$. Since $n \geqslant d\#L \geqslant \frac{50n\#L}{k}$, we conclude that $\#L \leqslant \frac{k}{50}$. Therefore, $\#I' \geqslant \frac{97k}{100}$. □

Lemma 2.8. *Suppose that the assumptions of Proposition 2.6* (a) *hold, and that the set I' for which Step 3 is encountered satisfies $\#I' \geqslant \frac{97k}{100}$ and $I' \subset R$. Then the output $\mathscr{S}$ of* `Purify` *contains R as an element.*

Proof. Since $\mathrm{def}_R(G) < \frac{k}{2}$, we have $\#V \setminus (R \cup N(I')) \leqslant \frac{k^2}{200n}$ by D1, whence $\#\tilde{V} \leqslant k + \frac{k^2}{200n} \leqslant \frac{3k}{2}$. We claim that either $\#R \setminus I' \leqslant 6\eta$ or $\#\tilde{V} \setminus R \leqslant \frac{\#R \setminus I'}{3}$. For assume $\#\tilde{V} \setminus R > \frac{\#R \setminus I'}{3}$. Since $\#R \setminus I' < \frac{3k}{100}$, there exists $T \subset \tilde{V} \setminus R$ such that $\frac{\#R \setminus I'}{3} \leqslant \#T \leqslant \frac{k}{100}$. Consequently, there is a 6-fold matching M from T to R with defect $\leqslant \eta$. Since there is no T–I'-edge, we have

$$\#R \setminus I' \geqslant 6(\#T - \eta) \geqslant 6\left(\frac{\#R \setminus I'}{3} - \eta\right) = 2\#R \setminus I' - 6\eta.$$

Hence, $6\eta \geqslant \#R \setminus I'$.

If $6\eta \geqslant \#R \setminus I'$, Step 4 will add R to $\mathscr{S}$. Thus, let us assume that $\#\tilde{V} \setminus R \leqslant \frac{\#R \setminus I'}{3} \leqslant \frac{k}{100}$. Then, there is a 6-fold matching from $\tilde{V} \setminus R$ to $R \setminus I'$ with defect $\leqslant \eta$. Letting D' be the set of defect vertices, $I'' = I'$, and $V' = V \setminus (D' \cup N(I''))$, we have a 6-fold matching M^* from $V' \setminus R$ to $R \setminus I''$ with defect 0.

Let f' be the maximum flow computed in Step 5 of `Purify`, and let $L' = \{v \in V' \setminus I'' |\, f'(s', s'_v) = 6\}$. We claim that

$$\#L' \cap (R \setminus I'') \leqslant \frac{\#R \setminus I''}{36}. \tag{2.7}$$

For if $v \in L' \cap (R \setminus I'')$, then there are at least 6 vertices $w \in V' \setminus R$ such that $f'(s'_v, t'_w) = 1$. As $c(t'_w, t') = 1$ for all w, we get $\#L' \cap (R \setminus I'') \leqslant \frac{\#V' \setminus R}{6}$. Further, we have $\#V' \setminus R \leqslant \frac{1}{6}\#R \setminus I''$, because M^* induces a 6-fold matching with defect 0 from $V' \setminus R$ to $R \setminus I''$.

Finally, we claim $L' \supset V' \setminus R$. For consider the flow h in N' induced by the matching M^* as in the proof of Lemma 2.7. Let f'_R be the restriction of f' to R. Since R is an independent set, $f'_R + h$ is a flow in N'. Moreover, $w(f'_R + h) = w(f'_R) + w(h) \geqslant w(f')$. Hence, letting $f'_{V' \setminus R}$ signify the restriction of f' to $V' \setminus R$, we obtain $w(f'_{V' \setminus R}) \geqslant w(h) = 6\#V' \setminus R$. As a consequence, $L' \supset V' \setminus R$. Thus, (2.7) shows that after at most $\lceil \log_2 n \rceil$ iterations the set R will be added to $\mathscr{S}$. □

Algorithm 4. `Exact`(G,k)

Input: A graph $G=(V,E)$ and an integer $1 \leqslant k \leqslant n$. *Output:* A subset of V.

```
1.  If $k \leqslant n/10$ then
        For all $\tilde{S} \subset V$, $\#\tilde{S} = \frac{k}{\ln(n/k)}$, do
            For all $Y \subset V$, $\#Y \leqslant \frac{k}{\ln(n/k)}$, do
                If $V \setminus (N(\tilde{S}) \cup Y)$ is an independent set of cardinality $\geqslant k$, then return with
                output $V \setminus (N(\tilde{S}) \cup Y)$.
        If $\#V \setminus N(\tilde{S}) < k$ for all $\tilde{S} \subset V$, $\#\tilde{S} = \frac{k}{\ln(n/k)}$, then return with output $\emptyset$.
2.  For all $\tilde{S} \subset V$ of cardinality $k$ do
        If $\tilde{S}$ is an independent set, return with output $\tilde{S}$.
3.  Return $\emptyset$.
```

Figure 4. The algorithm `Exact`.

Proof of Proposition 2.6. The first assertion in Proposition 2.6 follows from Lemmas 2.7 and 2.8. Furthermore, the running time of `Purify` is at most

$$n^{O(1)}\binom{2k}{\eta}\left(\binom{2k}{\eta}+\sum_{l\leqslant 6\eta}\binom{2k}{l}\right) \leqslant n^{O(1)}\binom{2k}{\eta}^{7} \leqslant n^{O(1)}\binom{k}{\eta}^{14},$$

because $\eta \leqslant k/2$. □

Finally, we investigate the distribution of $\mathrm{def}_G(S)$ for $G = G^*_{n,p,k}$. The proof of the following lemma will be given in Section 4.

Lemma 2.9. *Suppose that $k \geqslant C(n/p)^{1/2}$ for a sufficiently large constant $C > 0$. Let $\eta \geqslant 0$. Then* $\mathrm{P}\big(\mathrm{def}_{G^*_{n,p,k}}(S) \geqslant \eta\big) \leqslant \binom{k}{\eta}^{-20}$.

Remark. The network flow techniques used in the procedure `Purify` extend the matching techniques of Feige and Kilian [11]. Such an extension is necessary in order to obtain a polynomial expected running time, as we need to 'correct' a small defect with only a little effort (this is precisely what the parameter η is intended for). Furthermore, `Purify` refines the flow techniques used by the author in [9]; as given in [9], the flow techniques do not apply to the case when the independent set has size $o(n)$. The procedure `Purify` developed in this paper has also been useful in the context of colouring semirandom graphs [7].

2.4. The subroutine `Exact`

If `Filter` and `Purify` were not able to either exhibit an independent set of size $\geqslant k$ or prove the absence of such a set, `Find` executes the subroutine `Exact`. *Throughout, we assume that $k \geqslant C(n/p)^{1/2}$ for a sufficiently large constant $C > 0$.*

Proposition 2.10. *Let $G = (V, E)$ be a graph. If $\alpha(G) \geqslant k$, then* $\texttt{Exact}(G, k)$ *outputs an independent set of cardinality $\geqslant k$. On the other hand, if $\alpha(G) < k$, then* $\texttt{Exact}(G, k)$ *outputs $\emptyset$. If $k < n/10$ and either $G = G^*_{n,p,k}$ or $G = G^*_{n,p}$, then the probability that* $\texttt{Exact}(G, k)$ *runs Step 2 is $\leqslant \binom{n}{k}^{-1}$.*

The next two lemmas bound the probability that $\texttt{Exact}(G^*_{n,p,k}, k)$ or $\texttt{Exact}(G^*_{n,p}, k)$ executes Step 2.

Lemma 2.11. *Suppose that $G = G^*_{n,p,k}$ and $k \leqslant n/10$. Then, with probability $\geqslant 1 - \binom{n}{k}^{-1}$ Step 1 of* $\texttt{Exact}(G)$ *finds an independent set of cardinality $\geqslant k$.*

Proof. Since $V = \{1, \ldots, n\}$ is an ordered set, we can consider the set S_0 consisting of the $k/\ln(n/k)$ smallest vertices in S. Let $Y_0 = \{v \in V \setminus S |\ N(v) \cap S_0 = \emptyset\}$. If $\#Y_0 \leqslant k/\ln(n/k)$, Step 1 of $\texttt{Exact}$ will eventually try $\tilde{S} = S_0$ and $Y = Y_0$, thereby recovering S. Moreover,

$$\mathrm{P}\left(\#Y_0 \geqslant \frac{k}{\ln(n/k)}\right) \leqslant \binom{n}{\frac{k}{\ln(n/k)}}(1-p)^{\frac{k^2}{\ln^2(n/k)}} \leqslant \left(\frac{\mathrm{e}\ln(n/k)n}{k}\right)^{\frac{k}{\ln(n/k)}} \exp\left(-\frac{k^2 p}{\ln^2(n/k)}\right)$$
$$\leqslant \exp\left(3k - \frac{k^2 p}{\ln^2(n/k)}\right) \leqslant \exp(-5k\ln(n/k)) \leqslant \binom{n}{k}^{-1},$$

because $kp \geqslant (np)^{1/2} \geqslant 10\ln(np)^3 \geqslant 10\ln(n/k)^3$. □

Lemma 2.12. *Let $k < n/10$. The probability that in $G = G^*_{n,p}$ there is a set $\tilde{S} \subset V$ of cardinality $\geqslant k/\ln(n/k)$ such that $\#V \setminus N(\tilde{S}) \geqslant k$ is at most $\binom{n}{k}^{-1}$.*

Proof. If $\tilde{S} \subset V$ has cardinality $\frac{k}{\ln(n/k)}$ and $\#V \setminus N(\tilde{S}) \geqslant k$, then $\#V \setminus (\tilde{S} \cup N(\tilde{S})) \geqslant \frac{k}{2}$, because $\ln(n/k) > 2$. Hence, the probability that $G_{n,p}$ admits a set $\tilde{S}$ as in the lemma is at most

$$\binom{n}{k/\ln(n/k)}\binom{n}{k/2}(1-p)^{\frac{k^2}{2\ln(n/k)}} \leqslant \exp\left(6k\ln(n/k) - \frac{k^2 p}{2\ln(n/k)}\right)$$
$$\leqslant \exp(-5k\ln(n/k)) \leqslant \binom{n}{k}^{-1},$$

because $kp \geqslant 100\ln(n/k)^3$. □

Proof of Proposition 2.10. Suppose that $k \leqslant n/10$ and $\alpha(G) \geqslant k$, and let S be an independent set of cardinality k. Let $\tilde{S} \subset S$ be a subset of cardinality $\frac{k}{\ln(n/k)}$. Then $V \setminus N(\tilde{S}) \supset S$, whence either Step 1 or Step 2 of $\texttt{Exact}$ finds an independent set of cardinality $\geqslant k$. On the other hand, if $\alpha(G) < k$, then $\texttt{Exact}(G, k)$ obviously outputs $\emptyset$. The assertion concerning the probability that Step 2 gets executed follows from Lemmas 2.11 and 2.12. □

2.5. Proof of Theorem 1.1

Let $G=(V,E)$ be a graph. It is an easy consequence of Proposition 2.10 and the fact that $\vartheta(G)\geqslant\alpha(G)$ that $\texttt{Find}(G,k)$ outputs an independent set of cardinality $\geqslant k$ if $\alpha(G)\geqslant k$, and outputs $\emptyset$ otherwise. Therefore, the remaining task is to bound the expected running time of $\texttt{Find}(G^*_{n,p,k},k)$ and of $\texttt{Find}(G^*_{n,p},k)$.

Let us first deal with $G=G^*_{n,p,k}$. Set

$$\eta^*=\begin{cases}k/2 & \text{if the output } I \text{ of } \texttt{Filter} \text{ does not satisfy (2.2),}\\ \mathrm{def}_G(S) & \text{otherwise.}\end{cases}$$

By Proposition 2.1 and Lemma 2.9,

$$\mathrm{P}(\eta^*\geqslant\eta)\leqslant\binom{k}{\eta}^{-16}. \tag{2.8}$$

For $\tilde\eta=0,1,\ldots,k/2$, let $E_{\tilde\eta}$ be the expected running time of $\texttt{Find}$ conditioned on $\eta^*=\tilde\eta$. If $\eta^*<k/2$, then the output I of $\texttt{Filter}(G,k)$ satisfies the assumptions of Proposition 2.6. Therefore, by Proposition 2.6 $\texttt{Purify}(G,I,\eta^*,k)$ outputs an independent set of cardinality $\geqslant k$. Hence, $\texttt{Find}(G,k)$ finds an independent set of cardinality $\geqslant k$ before the variable η exceeds η^*. Again by Proposition 2.6 the total running time of $\texttt{Find}(G,k)$ is $\leqslant n^{O(1)}\binom{k}{\eta^*}^{14}$. Thus, for $\tilde\eta<k/2$ we have

$$E_{\tilde\eta}\leqslant n^{O(1)}\binom{k}{\tilde\eta}^{14}. \tag{2.9}$$

If $\eta^*=k/2$, then either Steps 1–3 of $\texttt{Find}(G,k)$ succeed in finding an independent set of cardinality $\geqslant k$, or $\texttt{Find}$ calls $\texttt{Exact}$. By Proposition 2.6, the total running time before $\texttt{Find}$ calls $\texttt{Exact}$ is $\leqslant n^{O(1)}2^{14k}$. Furthermore, the running time of Step 1 of $\texttt{Exact}$ is $\leqslant n^{O(1)}\binom{n}{k/\ln(n/k)}^2\leqslant n^{O(1)}\exp(6k)$. If $k<n/10$, then by Proposition 2.10 the expected time spent on executing Step 2 of $\texttt{Exact}$ is polynomial. On the other hand, if $k\geqslant n/10$, then the running time of Step 2 of $\texttt{Exact}$ is $n^{O(1)}\binom{n}{k}\leqslant n^{O(1)}\exp(4k)$. Hence,

$$E_{k/2}\leqslant n^{O(1)}\big(2^{14k}+\exp(6k)+\exp(4k)\big)\leqslant n^{O(1)}2^{14k}. \tag{2.10}$$

Thus, (2.8), (2.9) and (2.10) entail that the expected running time of $\texttt{Find}(G^*_{n,p,k},k)$ is at most

$$\sum_{\tilde\eta=0}^{k/2}E_{\tilde\eta}\mathrm{P}\big(\eta^*=\tilde\eta\big)\leqslant n^{O(1)}\left(1+\sum_{0<\tilde\eta<k/2}\binom{k}{\tilde\eta}^{-1}+2^{-k}\right)=n^{O(1)}.$$

As for $G=G^*_{n,p}$, note that the running time of $\texttt{Find}(G,k)$ is polynomial if $\texttt{Filter}(G,k)$ outputs $\emptyset$. By Proposition 2.1,

$$\mathrm{P}\big(\texttt{Filter}(G,k)\text{ outputs }\emptyset\big)\geqslant 1-\exp(-20k). \tag{2.11}$$

Furthermore, if $\texttt{Filter}(G,k)$ does not output $\emptyset$, then by Proposition 2.6 $\texttt{Find}$ spends time $\leqslant n^{O(1)}\exp(14k)$ on executing Steps 2–3. Hence, (2.11) shows that the expected time consumed by Steps 2–3 of $\texttt{Find}$ is polynomial. Finally, $\texttt{Find}$ calls $\texttt{Exact}(G,k)$ (provided that no independent set of cardinality $\geqslant k$ has been found before). Step 1 of $\texttt{Exact}$

consumes time $\leqslant \exp(6k)$. Furthermore, if $k < n/10$, then by Proposition 2.10 the expected running time of Step 2 of `Exact` is $n^{O(1)}$, and if $k \geqslant n/10$, then the running time of Step 2 of `Exact` is $\leqslant n^{O(1)}\binom{n}{k} \leqslant \exp(4k)$. Thus, invoking (2.11), we conclude that the expected running time of `Find`$(G^*_{n,p}, k)$ is

$$\leqslant n^{O(1)}\big(1 + \exp(-16k)\big(1 + \exp(4k) + \exp(6k) + \exp(14k)\big)\big) = n^{O(1)},$$

thereby proving Theorem 1.1.

3. Proof of Theorem 1.2

Remember that deciding whether a given graph $G' = (V', E')$ satisfies $\alpha(G') \geqslant \#V'/2$ is NP-hard, and consider an instance $G' = (V', E')$ of this problem. Let $N = n^{\varepsilon}/2$, and suppose that $V' = \{1, \ldots, 2N\}$. Let $V_2 = \{2N+1, \ldots, k+N\}$ and $V_1 = \{k+N+1, \ldots, n\}$. Then $V', V_1, V_2 \subset V$ are disjoint, and $\#V_1 = n - k - N$, $\#V_2 = k - N$. We obtain a graph G'' with vertex set V by including all edges E' into G'' and connecting every vertex in $v \in V_1$ with all vertices in $V \setminus \{v\}$. Let σ be a permutation of V chosen uniformly at random, and let $H = (V, E(H))$ be the graph with edge set $E(H) = \{\{\sigma(v), \sigma(w)\} |\ \{v, w\} \in E(G'')\}$; that is, H is a randomly permuted copy of G''. Then $\alpha(H) \geqslant k$ if and only if $\alpha(G') \geqslant N$, so that deciding whether $\alpha(G) \geqslant k$ is NP-hard.

We claim that in the case $\alpha(G') \geqslant N$ the adversary can convert the random graph $G_{n,p,k}$ into the graph H w.h.p. Thus, let $G_0 = G_{n,p,k}$, and let S be the independent set planted in G_0. Let X be the set of all of vertices $v \in V \setminus S$ that have no neighbour in S in the graph G_0. Then $\#X$ is binomially distributed with mean $n(1-p)^k \sim n^{\varepsilon}$. By Chernoff bounds [18, p. 26], $n^{\varepsilon}/2 \leqslant \#X \leqslant 2n^{\varepsilon}$ w.h.p. Moreover, the expected number of edges spanned by the vertices in X is $\mathrm{E}(\#E(G_0[X])) \leqslant \binom{2n^{\varepsilon}}{2}p \leqslant 2n^{2\varepsilon}p = o(1)$, whence X is an independent set w.h.p. Since the adversary does not need to work in polynomial time, it can look for an independent set S' of cardinality N in G'. The adversary identifies the vertices in S' with N distinct randomly chosen vertices in S and the remaining vertices of G' with N distinct randomly chosen vertices $Y \subset X$; let $\tau : V' \to V$ be the resulting injective map. Then, in order to embed a copy of G' into G_0, for each edge $\{v, w\} \in E'$ the adversary inserts the edge $\{\tau(v), \tau(w)\}$. Finally, the adversary connects every vertex in $V \setminus (S \cup Y)$ with all other vertices. Let G be the resulting graph. Then, the distribution of G coincides with the distribution of H.

Consequently, if we had an algorithm $\mathscr{A}$ that w.h.p. finds an independent set of cardinality k in $G^*_{n,p,k}$, we would obtain the following randomized algorithm with one-sided error for deciding whether $\alpha(G') \geqslant N$. Construct the graph H. If $\mathscr{A}(H)$ finds an independent set of cardinality $\geqslant k$, then answer '$\alpha(G') \geqslant N$'. Otherwise, answer 'probably $\alpha(G') < N$'.

4. Proof of Lemma 2.9

In order to prove Lemma 2.9, we need some technical lemmas. *Throughout, we assume that $np \geqslant \ln(n)^2$ and that $k \geqslant C(n/p)^{1/2}$ for some large constant $C > 0$.*

Lemma 4.1. *Let $G = G^*_{n,p,k}$. With probability $\geqslant 1 - \exp(-21k)$ the graph G_0 enjoys the following property. If $T \subset V \setminus S$, $\#T \geqslant 100/p$, and $U \subset S$, $\#U \geqslant k/2$, then in G_0 there is a T–U-edge.*

Proof. We may assume that $\#U = k/2$ and $\#T = 100/p$. The probability that in G_0 there are sets T and U as above with no edge joining T and U is at most

$$\binom{n}{100/p}\binom{k}{k/2}(1-p)^{\frac{50k}{p}} \leqslant \exp\left(\frac{100\ln(np)}{p} - 49k\right) \leqslant \exp(-21k),$$

because $kp \geqslant (np)^{1/2} \geqslant 100\ln(np)$. □

Lemma 4.2. *The probability that in $G_0 = G_{n,p,k}$ there are $\eta \geqslant 1$ vertices in $V \setminus S$ that have $< kp/2$ neighbours in S is $\leqslant n^{-25\eta}$.*

Proof. Let $v \in V \setminus S$. Since $\#N(v) \cap S$ is binomially distributed with mean kp, Chernoff bounds (see [18, p. 26]) yield $\mathrm{P}(\#N(v) \cap S \leqslant kp/2) \leqslant \exp(-kp/8)$. As $np \geqslant \ln(n)^2$, the estimate

$$\binom{n}{\eta}\exp\left(-\frac{kp\eta}{8}\right) \leqslant n^{\eta}\exp\left(-\frac{kp\eta}{8}\right) \leqslant \exp\left(-\frac{C(np)^{1/2}\eta}{16}\right) \leqslant n^{-25\eta}$$

proves our assertion. □

Lemma 4.3. *Let $V_1 \subset V$ be a subset of cardinality n_1, and let $V_2 = V \setminus V_1$, $\#V_2 = n_2$. Let $\gamma > 0$ be an arbitrary constant, and let $2 \leqslant d \leqslant n_2/10$. Then there exists a number $\omega_0 = \omega_0(\gamma)$ such that the following holds. Let $\omega = \omega_0 \max\{d, \ln n\}$, and let H be a random bipartite graph obtained as follows. Every vertex in V_1 chooses a set of at least ω neighbours in V_2 uniformly at random; these choices occur independently for all vertices in V_1. Then, for all $\eta \in \{0, 1, \ldots, n_2/2\}$ we have*

$$\mathrm{P}\left(\exists T \subset V_1 : \#T \leqslant \frac{n_2}{2d} \wedge \#N_H(T) < d\#T - \eta\right) \leqslant \binom{n_2}{\eta}^{-\gamma}.$$

Proof. We are to bound the probability that there is a set $T \subset V_1$, $1 \leqslant \#T = t \leqslant \frac{n_2}{2d}$ that admits a set $U \subset V_2$, $\#U = dt - \eta$, such that $N(T) \subset U$. There are at most $\binom{n_1}{t}\binom{n_2}{dt}$ possible choices of T and U. Given T and U, the probability that for $v \in T$ we have $N(v) \subset U$ is at most

$$\binom{\#U}{\omega}\binom{n_2}{\omega}^{-1} \leqslant \prod_{j=0}^{\omega-1}\frac{dt-j}{n_2-j} \leqslant \left(\frac{dt}{n_2}\right)^{\omega}.$$

Consequently, $\mathrm{P}(N(T) \subset U) \leqslant \left(\frac{dt}{n_2}\right)^{t\omega}$. Thus, we are to show that

$$\sum_{t=\max\{1,\lceil\frac{\eta}{d}\rceil\}}^{\frac{n_2}{2d}} \binom{n_2}{\eta}^{\gamma}\binom{n_1}{t}\binom{n_2}{dt}\left(\frac{dt}{n_2}\right)^{t\omega} \leqslant 1.$$

Since $\eta \leqslant dt \leqslant \frac{n_2}{2}$, we have $\binom{n_2}{\eta}^\gamma \leqslant \binom{n_2}{dt}^\gamma$. Therefore,

$$\begin{aligned}\binom{n_2}{\eta}^\gamma \binom{n_1}{t}\binom{n_2}{dt}\left(\frac{dt}{n_2}\right)^{t\omega} &\leqslant \left(\left(\frac{en_2}{dt}\right)^{\gamma d}\frac{en_1}{t}\left(\frac{en_2}{dt}\right)^d\left(\frac{dt}{n_2}\right)^\omega\right)^t\\ &\leqslant \left(\frac{dt}{n_2}\right)^{\omega_0 dt/4}\left(\exp((\gamma+1)d+1)\frac{n_1}{t}2^{-\omega/4}\right)^t\\ &\leqslant \left(\frac{dt}{n_2}\right)^{\omega_0 dt/4},\end{aligned}$$

where the last inequality follows from the fact that $\omega \geqslant \omega_0 \ln(n)$. Hence,

$$\begin{aligned}\sum_{t=\max\{1,\lceil\eta/d\rceil\}}^{\lfloor(n_2/d)^{1/2}\rfloor}\binom{n_2}{\eta}^\gamma\binom{n_1}{t}\binom{n_2}{dt}\left(\frac{dt}{n_2}\right)^{t\omega} &\leqslant \sum_{t=\max\{1,\lceil\eta/d\rceil\}}^{\lfloor(n_2/d)^{1/2}\rfloor}\left(\frac{dt}{n_2}\right)^{\omega_0 dt/4}\\ &\leqslant \left(\frac{n_2}{d}\right)^{1/2}\left(\frac{d}{n_2}\right)^{\omega_0 d/8} \leqslant \frac{1}{2}.\end{aligned}$$

Moreover, we have

$$\sum_{t=\lceil(\frac{n_2}{d})^{1/2}\rceil}^{\lfloor\frac{n_2}{2d}\rfloor}\binom{n_2}{\eta}^\gamma\binom{n_1}{t}\binom{n_2}{dt}\left(\frac{dt}{n_2}\right)^{t\omega} \leqslant \sum_{t=\lceil(\frac{n_2}{d})^{1/2}\rceil}^{\lfloor\frac{n_2}{2d}\rfloor}2^{-\omega_0 dt/4} \leqslant 2^{-(n_2 d)^{1/2}} \leqslant \frac{1}{2},$$

thereby proving the lemma. □

Proof of Lemma 2.9. Given $G = G^*_{n,p,k}$, let $W = \{v \in V \setminus S | \ \#N(v) \cap S < kp/2\}$, $G' = G_0 - W$, and $\omega = kp/2$. Let $6 \leqslant d \leqslant \lceil\frac{50n}{k}\rceil$. Then in G' every vertex in $V \setminus (S \cup W)$ has chosen $\geqslant \omega$ neighbours in S uniformly at random and independently of all others. Choosing C sufficiently large, we can ensure that $\omega \geqslant \omega_0(25)d$, where $\omega_0(25)$ is as in Lemma 4.3. Let $\eta > 0$, $\eta_1 \in \{0, 1, \dots, \eta\}$, and $\eta_2 = \eta - \eta_1$. By Lemma 4.2,

$$\mathrm{P}(\#W = \eta_1) \leqslant n^{-25\eta_1}. \tag{4.1}$$

Let us call a set $T \subset V \setminus (S \cup W)$ *η_2-bad* if $\#T \leqslant \frac{k}{2d}$ and $\#N(T) \cap S < d\#T - \eta_2$. If we condition on $\#W = \eta_1$, then Lemma 4.3 yields that the probability that there is an η_2-bad T is $\leqslant \binom{k}{\eta_2}^{-25}$. Therefore, (4.1) entails that

$$\begin{aligned}&\mathrm{P}\left(\exists\, T \subset V \setminus S, \#T \leqslant \frac{k}{2d} : T \text{ has no } d\text{-fold matching with defect} \leqslant \eta\right)\\ &\quad\leqslant \sum_{\eta_1=0}^{\eta}\mathrm{P}(\text{there is an } \eta_2\text{-bad } T | \#W = \eta_1)\mathrm{P}(\#W = \eta_1)\\ &\quad\leqslant \sum_{\eta_1=0}^{\eta} n^{-25\eta_1}\binom{k}{\eta_2}^{-25} \leqslant (2np)^{-1}\binom{k}{\eta}^{-20}.\end{aligned}$$

Thus, the probability that condition D2 is violated for a certain value of η is $\leqslant \frac{1}{2}\binom{k}{\eta}^{-20}$. To bound the probability that D1 is violated, we invoke Lemma 4.1. □

5. Conclusion

In contrast to the algorithm of Feige and Krauthgamer [12], $\texttt{Find}(G^*_{n,p,k}, k)$ does not certify the optimality of its output. However, one could modify the algorithm easily in order to guarantee that the size of the independent set found by the algorithm is within a factor of $1+\varepsilon$ from the independence number, for an arbitrarily small constant $\varepsilon > 0$. Indeed, a similar argument as in the proof of Lemma 2.3 shows that $\mathrm{P}(\vartheta(G^*_{n,p,k}) > (1+\varepsilon)k) \leqslant \exp(-20k)$, provided that $k \geqslant C(n/p)^{1/2}$ for a sufficiently large constant $C = C(\varepsilon)$. Hence, with probability $\geqslant 1 - \exp(-20k)$ we obtain a guarantee that $\alpha(G^*_{n,p,k}) \leqslant (1+\varepsilon)k$.

Although Theorem 1.2 gives a lower bound on the size of independent sets that can be recovered within polynomial expected time, there remains a gap between this lower bound and the upper bound provided by Find. Therefore, it is an open problem to either construct an algorithm that beats the upper bound provided by Find, or to prove a better lower bound. An algorithm that can distinguish efficiently between graphs $G_{n,p,k}$, with planted independent sets of size $k = o(n/p)^{1/2}$, and random graphs $G_{n,p}$ might also lead to a better performance guarantee for colouring $G_{n,p}$ than provided by [8, 10, 23].

References

[1] Alon, N. and Kahale, N. (1998) Approximating the independence number via the ϑ-function. *Math. Programming* **80** 253–264.

[2] Alon, N., Krivelevich, M. and Sudakov, B. (1998) Finding a large hidden clique in a random graph. *Random Struct. Alg.* **13** 457–466.

[3] Alon, N., Krivelevich, M. and Vu, V. H. (2002) On the concentration of the eigenvalues of random symmetric matrices. *Israel J. Math.* **131** 259–267.

[4] Blum, A. and Spencer, J. (1995) Coloring random and semirandom k-colorable graphs. *J. Algorithms* **19** 203–234.

[5] Bollobás, B. (2001) *Random Graphs*, 2nd edn, Cambridge University Press.

[6] Bollobás, B. and Erdős, P. (1976) Cliques in random graphs. *Math. Proc. Camb. Phil. Soc.* **80** 419–427.

[7] Coja-Oghlan, A. (2004) Coloring semirandom graphs optimally. In *Proc. 31st ICALP*, pp. 383–395.

[8] Coja-Oghlan, A. (2005) The Lovász number of random graphs. *Combin. Probab. Comput.* **14** 439–465.

[9] Coja-Oghlan, A. Solving NP-hard semirandom graph problems in polynomial expected time. To appear in *J. Algorithms.* A preliminary version has appeared in *Proc. 6th RANDOM*, pp. 139–148.

[10] Coja-Oghlan, A. and Taraz, A. (2004) Exact and approximative algorithms for coloring $G(n,p)$. *Random Struct. Alg.* **24** 259–278.

[11] Feige, U. and Kilian, J. (2001) Heuristics for semirandom graph problems. *J. Comput. System Sci.* **63** 639–671.

[12] Feige, U. and Krauthgamer, J. (2000) Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Alg.* **16** 195–208.

[13] Frieze, A. (1990) On the independence number of random graphs. *Discrete Math.* **81** 171–175.

[14] Frieze, A. and McDiarmid, C. (1997) Algorithmic theory of random graphs. *Random Struct. Alg.* **10** 5–42.

[15] Füredi, Z. and Komloś, J. (1981) The eigenvalues of random symmetric matrices. *Combinatorica* **1** 233–241.

[16] Grötschel, M., Lovász, L. and Schrijver, A. (1988) *Geometric Algorithms and Combinatorial Optimization*, Springer.
[17] Håstad, J. (1999) Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica* **182** 105–142.
[18] Janson, S., Łuczak, T. and Ruciński, A. (2000) *Random Graphs*, Wiley.
[19] Jerrum, M. (1992) Large cliques elude the metropolis process. *Random Struct. Alg.* **3** 347–359.
[20] Karger, D., Motwani, R. and Sudan, M. (1998) Approximate graph coloring by semidefinite programming. *J. Assoc. Comput. Mach.* **45** 246–265.
[21] Karp, R. (1976) Probabilistic analysis of some combinatorial search problems. In *Algorithms and Complexity: New Directions and Recent Results* (J. F. Traub, ed.), Academic Press, pp. 1–19.
[22] Knuth, D. (1994) The sandwich theorem. *Electron. J. Combin.* **1**.
[23] Krivelevich, M. and Vu, V. H. (2002) Approximating the independence number and the chromatic number in expected polynomial time. *J. Combin. Optimization* **6** 143–155.
[24] Kučera, L. (1995) Expected complexity of graph partitioning problems. *Discrete Appl. Math.* **57** 193–212.
[25] Matula, D. (1976) The largest clique size in a random graph. Technical report, Southern Methodist University, Dallas, Texas.
[26] McSherry, F. (2001) Spectral partitioning of random graphs. In *Proc. 42nd FOCS*, pp. 529–537.
[27] Schrijver, A. (2003) *Combinatorial Optimization*, Vol. A, Springer.