Heuristics for finding large independent sets, with applications to coloring semi-random graphs (Extended abstract)

Uriel Feige[*] Joe Kilian[†]

Abstract

We study a semi-random graph model for finding independent sets. For $\alpha > 0$, an n-vertex graph with an independent set S of size αn is constructed by blending random and adversarial decisions. Randomly and independently with probability p, each pair of vertices, such that one is in S and the other is not, is connected by an edge. An adversary can then add edges arbitrarily (provided that S remains an independent set). The smaller p is, the larger the control the adversary has over the semi-random graph. We design heuristics that with high probability recover S when $p > (1+\epsilon)\ln n/|S|$, for any constant $\epsilon > 0$. We show that when $p < (1-\epsilon)\ln n/|S|$, an independent set of size $|S|$ cannot be recovered, unless $NP \subseteq BPP$.

We use our results to obtain greatly improved coloring algorithms for the model of k-colorable semi-random graphs introduced by Blum and Spencer.

1. Introduction

An independent set in a graph is a set of vertices no two of which are connected by an edge. Finding a maximum size independent set (MIS) is a fundamental problem in combinatorial optimization, and is related to many other problems such as vertex cover (the complement of an independent set), clique (an independent set in the complement of the graph), and coloring (covering the vertices of a graph by the minimum number of independent sets). As is well known, MIS is NP-hard.

Much work was devoted to developing heuristics (we use the term *heuristic* to denote algorithms that are not always guaranteed to return the optimal solution) for finding large independent sets (see [8], for example). However, it is not easy to evaluate the performance of heuristics. The empirical approach is to run the heuristic on a collection of input graphs ("benchmarks"), and record the sizes of independent sets returned by the heuristic. If one heuristic consistently outperforms another, then we have empirical evidence of it being a better heuristic. Though running heuristics on benchmarks is sometimes informative, we seek a more rigorous measure for evaluating heuristics.

One such measure is the approximation ratio. An algorithm is said to approximate MIS within a ratio $\rho > 1$ if for every input graph, the size of the maximum independent set is a factor of at most ρ larger than the size of the independent set returned by the algorithm. For NP-hard optimization problems in general, one can evaluate heuristics based on their approximation ratios. However, it is known (through work culminating in [13]) that for any constant $\epsilon > 0$, MIS cannot be approximated within ratio of $n^{1-\epsilon}$ (where n is the number of vertices in the input graph) unless NP has randomized polynomial time algorithms. Hence, if we were to evaluate heuristics for MIS based on their approximation ratios, all heuristics would perform badly. The best approximation ratio known to be achievable for MIS is $O(n/(\log n)^2)$ [7].

As very little can be done with the MIS problem on worst case instances (unless P=NP), one would like to compare the performance of heuristics on average instances, or those that typically occur in practice. But how does one model such instances? One possible model is that of a random graph (see the survey in [10]). The question then arises of how well random graphs model inputs that interest us in "real life" applications. But even regardless of this question, the random graph model does not seem to provide a good way of distinguishing between good and bad heuristics in the case of MIS. For example, if each edge is chosen independently at random with probability 1/2, then the size of the maximum independent set is almost surely roughly $2\log n$. An elementary greedy heuristic almost surely finds an independent set of size $\log n$. No heuristic, not even the

[*]Department of Applied Math and Computer Science, the Weizmann Institute, Rehovot 76100, Israel. feige@wisdom.weizmann.ac.il. Incumbent of the Joseph and Celia Reskin Career Development Chair. Part of this work done while the author was visiting the NEC Research Institute.

[†]NEC Research Institute, 4 Independence Way, Princeton, New Jersey, 08540. joe@research.nj.nec.com

most sophisticated one, is known to find independent sets significantly larger than $\log n$. Hence, most heuristics have roughly the same performance in this random graph model, making it an inadequate framework for comparing between them. Another random model that was suggested is similar to the random graph model, but with an independent set of size K planted in the graph. The larger K is, the easier it is to find the independent set. If $K > c\sqrt{n \log n}$ for sufficiently large c, the vertices of the independent set are easily recognized to be the K vertices of lowest degree in G. Hence a trivial heuristic will solve MIS in this case. But even the most sophisticated heuristics guarantee only a marginal improvement in the size of K. The lowest value of K that can be provably handled by known heuristics (spectral techniques, in this case) is $\Omega(\sqrt{n})$ [4]. Hence also the planted independent set model does not provide sharp distinctions between good and bad heuristics.

In this paper we study a semi-random model for MIS, which has some similarities with semi-random models studied for the coloring problem in [5]. In our model there is an independent set S of size αn planted in a graph, for some $\alpha > 0$. Edges connecting S to the rest of the graph are placed at random, with probability $p = O(\frac{\log n}{n})$, creating a random graph G_{min}. An adversary can then add (but not remove) arbitrary edges to G_{min} (as long as S remains an independent set), thus obtaining the input graph G. The goal of the heuristic is to recover an independent set of size αn. The semirandom model has a random component G_{min}, and when p is large enough, hardness results for worst case instances no longer apply. It also has an adversarial (worst case) component, which can be used to make G more similar to inputs that may occur in practice – with the restriction that G_{min} must remain a subgraph. Technically, the adversarial component can be used to alter degrees of vertices (and foil heuristics based on vertex degrees), to create independent sets that are "local maxima" (and foil heuristics based on local search), and to modify the spectrum of the graph. We are not aware of any previously published algorithm that can handle our semirandom model.

We design an algorithm that recovers an independent set of size αn in the semirandom graph model. Our algorithm requires a value of p slightly above $\ln n/\alpha n$. We also show that this value of p is best possible up to low order terms, unless NP has randomized polynomial time algorithms.

Our algorithm for finding independent sets can also be used for coloring. In the random k-colorable graph model, a graph is partitioned into k color classes, and edges are placed at random with probability p between color classes. In this model Alon and Kahale [2] show that when $p = c/n$ and $c > 0$ is large enough, a k-coloring can be recovered. Blum and Spencer [5] introduced the framework of semi-random k-colorable graphs, where in addition to the random edges, an adversary can add arbitrary edges between color classes. (Several variations on this model are presented in [5], and the results discussed here apply to all of them. This model was further studied in [17].) Algorithms designed for the random graph model often do not work in the semi-random model. For the semirandom model, when color classes are of the same size (the *balanced* case), Blum and Spencer [5] design algorithms that handle values of p above n^{δ}/n, for some $\delta > 0$ that depends on k. No better results were known, and the algorithm of [2] does not seem to apply to this model. Our new algorithm for finding independent sets can be used also for k-coloring (in the balanced case), by extracting the color classes one by one. It offers major improvement over the results of [5] on semirandom graphs, as it handles values of p as low as $(1+\epsilon)k \ln n/n$. This answers an open question of [5] and [10]. Moreover, the value of p is almost best possible, as coloring semirandom k-colorable graphs with values of p below $(1-\epsilon)\ln n/n$ is NP-hard. (Note that in the random model values of $p = O(1/n)$ can be handled [2].)

1.1. The semi-random graph model

The graph G has n vertices. Its edge set E is generated partly at random and partly by an adversary.

Let $0 < \alpha < 1$ be a constant, and $p = (c \ln n)/n$, where c is a large enough constant that depends on α.

1. An independent set S of size αn is chosen at random. Let $\bar{S}$ denote the vertices of G not belonging to S.

2. Random component: for any pair of vertices u, v such that $u \in S$ and $v \in \bar{S}$, the edge (u, v) is placed in E with probability p, independently of all other events. This gives the random graph G_{min}.

3. Adversarial component: having complete knowledge of G_{min}, an adversary may add to E any number of edges, provided that S remains an independent set. This gives the graph G.

Let G_{max} be the graph obtained by planting the independent set S of size αn in an otherwise complete graph on n vertices. Then G is an arbitrary graph "sandwiched" between G_{min} and G_{max}. This fixes S as an independent set, allows the adversary complete control for placing edges both of whose endpoints are in $\bar{S}$, and partial control over those edges with one endpoint in S and the other in $\bar{S}$ (the adversary controls a fraction of $(1-p)$ of these edges). The lower p is, the more control the adversary has over the graph.

Our algorithm is required to recover an independent set of size at least αn. We wish the algorithm to succeed with high probability, where probability is computed over the choice of G_{min}, and over the random coin tosses of the algorithm, regardless of the adversarial component of the graph.

We note that G may contain several independent sets of size αn. In this case the algorithm is required to output just one of them, not necessarily the original S.

Observe that the algorithm is in essence required to output G_{max}. The random graph G_{min} differs from G_{max} in that it misses some of the edges contained in G_{max}. The only changes that the adversary makes to G_{min} is to put back some of these missing edges. Hence, it may appear that the adversary only makes the problem easier. Nevertheless, many algorithms that would recover a large independent set in the random graph G_{min} would fail on G. A major motivation for the semi-random graph model is to identify those algorithms that work on random graphs, and are also robust enough to withstand adversarial "help".

Notation: throughout, α, p and c will be used only as above. That is, $|S| = \alpha n$, and the edge probability of the random component is $p = c \ln n/n$. For a set T of vertices, $N(T)$ denotes the set of their neighbors in graph G. We denote set subtraction by $A \setminus X$ (that is, those elements in A that are not in X).

1.2. Our results

The properties of our main algorithm are summarized in the next theorem.

Theorem 1 *For $c = (1+\epsilon)/\alpha$, where $\epsilon > 0$, there is a random polynomial time algorithm that with overwhelming probability recovers an independent set of size αn in the semi-random graph model.*

"Overwhelming probability" means with probability $1 - o(1)$ for n sufficiently large. Using the adversarial component of the graph, it can be shown that the value of c in Theorem 1 is best possible, up to low order terms.

Theorem 2 *In the semi-random model, if $c = (1-\epsilon)/\alpha$ for some $\epsilon > 0$, then unless $NP \subset BPP$, every random polynomial time algorithm will with overwhelming probability fail to find an independent set of size αn in G (against an optimal adversary).*

The proof of Theorem 2 is sketched in the appendix.

Theorem 1 implies new results about coloring k-colorable graphs in the semi-random model of Blum and Spencer [5]. In this model, a graph is partitioned into k color classes. Each edge e_i between different color classes is included with probability p_i, where p_i is controlled by an adversary, subject to $p_i \geq p$, for some $p < 1$. Clearly, the smaller p, the stronger the adversary. Blum and Spencer give algorithms to color semi-random 3-colorable graphs when $p > n^{2/3}/n$, and to color k-colorable semi-random graphs in which the sizes of the color classes are balanced, whenever $p > n^{\delta}/n$, for $\delta > (k^2-k-2)/(k^2+k-2)$. We significantly lower the values of p for which a coloring can be found. Our bounds on p are best possible up to constant multiplicative factors (via an argument similar to the proof of Theorem 2).

Theorem 3 *For any constant k, there is a polynomial time algorithm that with overwhelming probability recovers the largest independent set in k-colorable graphs in the semi-random graph model of [5], whenever $p > \frac{(1+\epsilon)k \ln n}{n}$.*

It is not hard to show that in semirandom k-colorable graphs with p as in Theorem 3, the largest independent set corresponds to a color class. Hence after the largest independent set is recovered, it can be removed from the graph, which then remains $(k-1)$-colorable. When $k = 3$, the remaining graph is bipartite and can be two-colored in polynomial time. For $k > 3$, the other color classes can be recovered by repeatedly applying Theorem 3, but only if the color classes are large enough. If they are too small, then the number of vertices n' remaining in the graph may be too low, causing $p < \frac{(k-1)\ln n'}{n'}$, and Theorem 3 may not apply. In fact, in [5] it is shown that coloring semi-random 4-colorable graphs is NP-hard even for much larger values of p, when the color classes are highly unbalanced.

We remark that our algorithm for finding independent sets works also in models that have less randomness (and hence, are more adversarial) than our semirandom graph model. One such model is the d-neighbors model, where in G_{min}, each vertex of $\bar{S}$ has d random neighbors in S. The graph G is then an arbitrary graph sandwiched between G_{min} and G_{max}. It can be shown that a simple modification of our algorithm recovers in this model independent sets of size αn, when d is a large enough constant that depends only on α. Observe that in this model G_{min} has only $O(n)$ edges, whereas in our original semirandom model G_{min} has $\Omega(n \log n)$ edges.

1.3. Techniques and related work

Lovasz introduced the *theta* function as an upper bound on the size of the maximum independent set [15]. The theta function can be approximated within arbitrary precision in polynomial time, using semidefinite programming. Goemans and Williamson [12] showed how semidefinite programming can be used in order to approximate problems such as max-cut. Inspired by their work, Karger, Motwani and Sudan [14] used semidefinite programming to obtain improved coloring algorithms. Alon and Kahale [3] used the work of [14] to show that the theta function can be used to find medium size (n^{δ} vertex) independent sets in graphs that have linear size independent sets (improving the values of δ previously obtained in [7]).

In terms of approximation ratio, the theta function (and similar semidefinite programs) appear to have little to offer. In [9] it is shown that for every $\epsilon > 0$ there are graphs with multiplicative gaps of $n^{1-\epsilon}$ between the size of the maximum independent set and the value of the theta function. Indeed, Håstad's result [13] implies that, unless NP is easy, no easily computable function will give better than a $n^{1-\epsilon}$ approximation in the worst case.

However, our current work singles out semidefinite programming as an approach that can cope with the semirandom graph model, unlike other heuristics for MIS. In more detail, our algorithm has two phases. In the first phase (Sections 2.1 and 2.2) G is partitioned into a small number of parts, such that some of these parts are composed mostly of vertices of S. This first phase uses semidefinite programming. Its analysis is based only on G_{min}, and goes through regardless of what the adversary does. This illustrates the robustness of (some) algorithms based on semidefinite programming.

In the second phase (Sections 2.3, 2.4 and 2.5) we "clean up" the output of the first phase, and extract S (or a different independent set of the same size). Many of the difficulties introduced by the adversary manifest themselves in the second phase. In particular, there is the problem of getting out of local maxima. To illustrate this problem, assume that the algorithm already found a maximal independent set I composed mostly of vertices of S (though not containing all of S). One may then hope that local heuristics such as k-opt (exchanging a constant k number of vertices of I with $V \setminus I$ so as to hopefully get a new independent set that is not maximal and hence can be expanded) would allow one to eventually extract S. However, in our semi-random model, the adversary is strong enough so as to make no k-exchange possible, even when I is almost as large as αn (details omitted). Our method of improving over local maxima is based on global computations (finding maximum matchings) rather than local ones, and may be of independent interest.

As pointed out above, our work shows that algorithms based on semidefinite programming perform well on random instances of MIS, and are robust enough to withstand adversaries that add edges to the graph. We remark that a similar phenomenon occurs for graph bisection. Boppana [6] considers a model of random graphs in which the edge probability for edges crossing the intended bisection is slightly smaller than that of edges outside the bisection. In this model, he shows how to find the planted bisection. We propose an algorithm similar to that of Boppana, formulated as a semidefinite program. Based on Boppana's techniques, we show that this algorithm finds the bisection in a semirandom model in which an adversary adds arbitrary edges within each side of the bisection, and removes arbitrary edges connecting the two sides of the bisection. This gives another example of the robustness of algorithms based on semidefinite programming. More details are sketched in the appendix.

1.4. Useful properties of semi-random graphs

Finding independent sets is NP-hard. Hence our algorithm will have to use some special property of G, inherited from G_{min}. The property we use is *expansion*, as formulated in the following lemmata. Recall that overwhelming probability means with probability $1 - o(1)$ for n sufficiently large.

Lemma 4 *Let $c > 0$, and let $t = n(\log n)^{-\delta}$, for some δ, $0 < \delta < 1$. Then when n is large enough, with overwhelming probability over the choice of G_{min}, for every $T \subseteq \bar{S}$ and $S' \subseteq S$, each of cardinality t, there is some edge in G_{min} joining T and S'.*

Proof: (Sketch) There are at most $\binom{n}{t}^2$ ways of choosing T and S'. For each such choice, the probability of the bad event that there is no edge joining T and S' is $(1-p)^{t^2}$. For large enough n, the lemma follows from the union bound on the probabilities of the bad events. $\Box$

Similarly, we can show:

Lemma 5 *Let $c = (1+\epsilon)/\alpha$, let $d > 0$ be an arbitrary constant, and assume that n is large enough. Then with overwhelming probability over the choice of G_{min}, for every $T \subset \bar{S}$ of cardinality at most $31\alpha n/32d$, $|N(T) \bigcap S| \geq d|T|$.*

Throughout we shall assume that G_{min} has the above expansion properties. As G contains all edges of G_{min}, these properties are preserved in G.

In Section 2.2, we shall also use the following property of G_{min}, namely:

Lemma 6 *Let $c = (1+\epsilon)/\alpha$, with $0 < \epsilon < 1$. Then with overwhelming probability over the choice of G_{min}, the number of edges in G_{min} is at most $2n \ln n/\alpha$.*

Of course, G may contain many more edges than G_{min}. We shall use Lemma 6 in our analysis, but shall NOT assume that G is sparse.

2. Our algorithm and its analysis

Our algorithm has five phases, described and analyzed in the following subsections. Many of the constants involved are arbitrary and are specified only for concreteness. Throughout our analysis, we ignore divisibility issues, eschewing careful roundoff analyses. Such considerations do not materially affect our argument.

2.1. A coarse partition using semidefinite programming

The following lemma is implicit in [3, 14].

Lemma 7 *Let $G(V, E)$ be a graph on n vertices that contains an independent set of size K. Then one can find in polynomial time a set Q of $K/2$ vertices (not necessarily belonging to the independent set), and a set of $K/2$ unit vectors in R^n associated with these vertices, such that for any two vertices $v_i, v_j \in Q$, if $(v_i, v_j) \in E$, then the inner product of the associated vectors satisfies*

$$\langle z_i, z_j \rangle < -K/(2n - K)$$

(i.e., the angle between v_i and v_j is large).

For completeness, the proof of Lemma 7 is presented in the appendix.

Using Lemma 7, we extract a small number of large sets $V_1, V_2, \dots$ as follows. We define $G_0 = G$ and inductively, we let G_i be the subgraph of G induced on $V \setminus \bigcup_{j \le i} V_j$. We generate V_{i+1} by applying the algorithm of Lemma 7 to G_i, setting the value of K to be $\alpha n/4$. Hence the set V_{i+1} contains $K/2 = \alpha n/8$ vertices, and we also have vectors associated with these vertices (z_j denotes the vector associated with vertex v_j) such that whenever $(v_j, v_k) \in E$,

$$\langle z_j, z_k \rangle < -(\alpha n/4)/(2n - \alpha n/4) < -\alpha/8.$$

We stop producing new V_i when the algorithm fails to find such a set. In this case, the set of remaining vertices contains less than $\alpha n/4$ vertices of S (or the algorithm would succeed).

The subgraph of G induced on V_i will be denoted by Q_i. Let $S_i = V_i \bigcap S$.

Definition 1 *A set V_i is* good *if $|S_i| \ge \alpha^2 n/32$.*

Proposition 8 *The process described above gives a partition for which*

$$\sum_{\{i|V_i \text{ is good}\}} |S_i| \ge \alpha n/2.$$

Proof: Since $V_i = \alpha n/8$, by construction, there are at most $8/\alpha$ sets. Less than $\alpha n/4$ vertices of S are not contained in any V_i (i.e., those discarded at the end). The number of vertices of S contained in sets V_i that are not good is at most $\frac{\alpha^2 n}{32} \cdot \frac{8}{\alpha} \le \frac{\alpha n}{4}$. As $|S| = \alpha n$, the proof follows. □

2.2. Refining the partition using random hyperplanes

We further partition each V_i. The desired outcome of this phase is summarized in Lemma 10.

Let $\bar{S}_i = V_i \backslash S_i$, and let E_i be the set of edges connecting S_i and $\bar{S}_i$ in G_{min}. (Note, E_i does not contain the edges added by the adversary.)

By our construction, for each vertex $v_j \in V_i$ we can associate a unit vector $z_j \in R^n$ such that for any two vertices $v_j, v_k \in V_i$, if $(v_j, v_k) \in E$ (and as a special case, if $(v_j, v_k) \in E_i$), then the inner product of the associated vectors satisfies $\langle z_j, z_k \rangle < -\alpha/8$. This inequality implies that the angle between the vectors z_j and z_k is at least some constant $\Theta > \pi/2$. (One can take $\Theta = \cos^{-1}(-\alpha/8)$. The exact value of Θ is irrelevant to our analysis.)

We partition Q_i using a technique developed by [12, 14]. We pass a random hyperplane through the origin, and separate the vertices V_i into two sets, depending on the side of the hyperplane on which they lie. As shown in [12], if $(v_j, v_k) \in E$, a random hyperplane will separate z_j and z_k with probability at least $\rho = \Theta/\pi > 1/2$.[1]

Choosing h random hyperplanes partitions the vertices V_i into $m \le 2^h$ sets of vertices, $V_{i1}, \dots, V_{im}$. This randomized partition procedure, which we denote $\mathsf{partition}(Q_i, h)$, tends to shatter Q_i in the following sense:

Definition 2 *Let $E(V_{i1}, \dots, V_{im})$ denote all pairs $(v_j, v_k) \in E_i$ such that $v_j, v_k \in V_{iq}$ for some q.*

Lemma 9 *For a good V_i and for E_i as defined above, the expectation of $|E(V_{i1}, \dots, V_{im})|$ is at most $|E_i|(1 - \rho)^h$, where the expectation is taken over the coin tosses of* $\mathsf{partition}(Q_i, h)$.

Proof: Each pair $(v_j, v_k) \in E_i$ has probability $(1 - \rho)^h$ of not being separated by at least one of the hyperplanes. The proof follows from the linearity of the expectation. □

Modifying an approach of [14], our algorithm computes $(V_{i1}, \dots, V_{im}) = \mathsf{partition}(Q_i, h)$ for each Q_i, and then removes a maximal matching from each set V_{ij}. For each V_i, this gives independent sets $I_{i1}, \dots, I_{im}$ (some of which might be empty), and some left over vertices (removed by the matching) M_i. These independent sets and the sets of leftover vertices form the refined partition of G.

Definition 3 *An independent set I_{ij} is* useful *if $|I_{ij} \bigcap S_i| \ge 3|I_{ij}|/4$.*

Recall that $\rho = \Theta/\pi > 1/2$ and let h be the least integer satisfying $(1 - \rho)^h \le \alpha^2/16 \ln n$. The following lemma is of central importance to our analysis. Its proof is given in the appendix.

Lemma 10 *Let h be as above and let $\{I_{ij}\}$ be the independent sets obtained by running $\mathsf{partition}(Q_i, h)$ on each of the subgraphs Q_i, and removing a maximal independent set*

[1]Better partitioning techniques are also suggested in [14], but are not needed for our results.

from each V_{ij}. Then with probability at least $1/2$ (over the random choices of $\mathsf{partition}(Q_i, h)$*), at least $\alpha n/8$ vertices of S are in useful independent sets I_{ij}.*

2.3. Creating a linear size independent set I

In this section we find in G an independent set of size $\Omega(n)$.

The number of independent sets constructed in Section 2.2 is at most $8m/\alpha \leq \log n$ (for large enough n). Lemma 10 describes an event that happens with probability at least 1/2, and we assume that this event holds. (The randomized algorithm can be repeated several times with independent coin tosses so as to make the probability of failure arbitrarily small.) Hence $\alpha n/8$ vertices of S are in useful independent sets.

We now guess which are the useful independent sets. The number of possible combinations here is less than n, and we can just try out all possibilities. (More efficiently, one may guess just one useful independent set, and deduce the rest via matching techniques. Details are omitted from this preliminary version.) Combine the useful independent sets to obtain a set J, and remove a maximal matching from J to obtain an independent set I. At most one fourth of the vertices of J do not belong to S, and for each edge removed from J at least one of its endpoints is not in S. It follows that $|I \bigcap S| \geq |J| - 2|J|/4 \geq |J|/2 \geq |S|/16$.

2.4. Purifying I.

Let I be a an independent set in G with $|I \bigcap S| \geq |S|/16$. Observe that by Lemma 4, almost all vertices of I belong to S, and only a small number of vertices from I might belong to $\bar{S}$. In this section we extract from I a subset I', all of which is contained in S.

Denote the vertices of G by $v_1, \ldots, v_n$. Let l be the smallest integer greater than $32/\alpha$.

Based on $G(V, E)$ and I, we describe a bipartite graph $G'(V', E')$, where $V' = R \bigcup L$. The right hand side R of G' contains the $n - |I|$ vertices $V \setminus I$. The left hand side L contains l copies of I. Namely, each vertex $v_i \in I$ is represented l times in L, as $v_{i1}, v_{i2}, \ldots v_{il}$. The edges E' are obtained from G in a natural way: $(v_{ij}, v_k) \in E'$ if $v_{ij} \in L$, $v_k \in R$, and $(v_i, v_k) \in E$.

We now describe the algorithm $\mathsf{purify}(I)$.

```
purify(I)
 1. Construct $G'$ as described above.
 2. Find a maximum matching $M$ in $G'$.
 3. Return $I'$, where $I' \subseteq I$ contains those vertices $v_i \in I$
    for which there is some $1 \leq j \leq l$ for which the vertex
    $v_{ij}$ was left unmatched by $M$.
```

Clearly, the algorithm runs in polynomial time. Lemma 11 shows that in semi-random graphs, it indeed purifies large independent sets.

Lemma 11 *Let G be a semi-random graph satisfying Lemma 5, and assume that n is large enough so that the parameter d in Lemma 5 can be chosen to be larger than l. Let I be an independent set in G such that $|I \bigcap S| > |S|/16$. Then* $\mathsf{purify}(I)$ *returns an independent set $I' \subseteq S$, with $I' > |S|/32$.*

Proof: To see that $|I'| > |S|/32$, observe that the size of M is at most $|R| < n$. For every vertex $v_i \in I \setminus I'$, there are l vertices $v_{i1}, \ldots, v_{il} \in L$ matched with vertices in R. Hence $|I \setminus I'| < n/l$, and $|I'| > |I| - n/l = \alpha n/16 - \alpha n/32 = \alpha n/32$.

It remains to show that $I' \subseteq S$. We first analyze the graph G. Let $A = I \bigcap S$ and $B = I \bigcap \bar{S}$. By assumption, $|A| > |S|/16$. As $N(B) \bigcap S \subseteq \bar{I}$, it follows that $|N(B) \bigcap S| < 15|S|/16$. By Lemma 5, it follows that $|B| < (15|S|/16)/d$; it then follows that for every $B' \subseteq B$, $|N(B') \bigcap (S \bigcap \bar{I})| > l|B'|$ (note that $N(B') \bigcap (S \bigcap \bar{I}) = N(B') \bigcap S$). Observe that $N(B) \bigcap (S \bigcap \bar{I})$ is disjoint from $N(A) \bigcap \bar{I}$.

We now analyze G'. Let B' denote those vertices of L that originate from B. (That is, $v_{ij} \in B'$ if $v_i \in B$.) We claim that there is a matching M' from B' to the vertices of $R \bigcap S$. Since no other vertex from L can be matched to vertices of $R \bigcap S$, the existence of M' implies that any maximal matching must match every vertex of B'. Consider an arbitrary $B'' \subseteq B'$. B'' must contain at least $|B''|/l$ distinct representatives from B. Then $|N(B'')| > (|B''|/l) \cdot l = |B''|$. Hence by Hall's theorem, all vertices of B' will be matched. It follows that $B \subseteq I \setminus I'$, implying $I' \subseteq A \subseteq S$. □

2.5. Expanding I'.

To expand I', ultimately recovering S, we perform the following procedure.

```
expand(I')
 1. Set $V' = V - N(I')$. Let $G'$ be $G$ induced on $V'$.
 2. Compute a maximum matching on $G'$.
 3. Return $I''$, the set of unmatched vertices of $G'$.
```

Lemma 12 *Let $I' \subseteq S$ and $|S|/32 < |I'| < |S|$. If Lemma 5 holds, then $I'' \subseteq S$ and $|I''| > |I'|$.*

Proof: Since $I' \subseteq S$, V' will contain S; define $Q = V' \setminus S$. Since S is an independent set, the maximum size of a matching on G' is $|Q|$, and this size can be achieved only if every vertex in Q is contained in the matching. By construction, $S \cap N(Q)$ (the neighborhood of Q, restricted to S) is contained in $S - I'$. Noting that $|S - I'| < 31|S|/32$, it follows from the expansion properties of G that $|Q| < |S|/32$, or $S \cap N(Q)$ would have at least $31|S|/32$ vertices. It then follows from the expansion properties that $|S \cap N(Q)| > |Q|$, unless Q is empty (in which case $\mathsf{expand}(I') = S$), and that $|S \cap N(Q')| > |Q'|$ for every nonempty $Q' \subseteq Q$. Hence there exists a complete matching from Q to S. This implies that any maximum matching found will include every vertex from Q and that at least one vertex from $S - I'$ will be left over, since $|S - I'| \geq |S \cap N(Q)| > |Q|$. $\mathsf{expand}(I')$ will be contained in S and properly contain I. $\square$

Lemma 12 implies that repeated applications of $\mathsf{expand}(I')$ (each time with I' being the previous I'') will recover S. (In fact, the maximum matching found in the first application of $\mathsf{expand}(I')$ can be reused, making repeated applications of $\mathsf{expand}(I')$ unnecessary. Details are omitted.)

Acknowledgements

Allan Frieze gave useful comments on an earlier version of this manuscript.

References

[1] F. Alizadeh. “Interior point methods in semidefinite programming with applications to combinatorial optimization”, *SIAM J. Optimization*, 5(1), 13–51, 1995.

[2] N. Alon and N. Kahale. “A spectral technique for coloring random 3-colorable graphs”. *SIAM J. Comput.*, 26(6), 1733–1748, 1997.

[3] N. Alon and N. Kahale. “Approximating the independence number via the ϑ-function”. *Math. Programming*, to appear.

[4] N. Alon, M. Krivelevich and B. Sudakov. “Finding a large hidden clique in a random graph”. *In Proc. Ninth SODA, 594–598, 1998*.

[5] A. Blum and J. Spencer. “Coloring random and semi-random k-colorable graphs”. *Journal of Algorithms*, 19(2):204–234, September 1995.

[6] R. Boppana. “Eigenvalues and graph bisection: An average-case analysis”. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 280–285, Los Angeles, CA, October 1987. IEEE Computer Society Press.

[7] R. Boppana and M. Halldorsson. “Approximating maximum independent sets by excluding subgraphs”. *BIT, 32 (1992), 180–196*.

[8] David S. Johnson and Michael A. Trick (editors). “Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993”, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 26, American Mathematical Society, 1996*.

[9] U. Feige. Randomized graph products, chromatic numbers, and the Lovasz ϑ-function. *Combinatorica 17 (1) (1997) 79–90*.

[10] A. Frieze and C. McDiarmid. “Algorithmic theory of random graphs”. *Random Structures and Algorithms 10 (1997), 5–42*.

[11] Z. Furedi and J. Komlos. “The eigenvalues of random symmetric matrices”. *Combinatorica*, 1(3), 233–241, 1981.

[12] Michel X. Goemans and David P. Williamson. “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming”, *Journal of the ACM*, 42 (6) (1995), 1115–1145.

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

[14] D. Karger, R. Motwani, and M. Sudan. “Approximate graph coloring by semidefinite programming.” *Journal of the ACM*, 45(2) , 246–265, 1998.

[15] Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory IT-25*, pp. 1-7, 1979.

[16] S. Poljak, F. Rendl. “Nonpolyhedral relaxations of graph-bisection problems”. *SIAM J. Optimization*, 5(3), 467–487, 1995.

[17] C. Subramanian. “Minimum coloring random and semi-random graphs in polynomial expected time”. *Proc. 36th Annual Symposium on Foundations of Computer Science, 1995, 463–472*.

A. Proofs

Proof of Theorem 2: (Sketch) Recall that $p = \frac{c \ln n}{n}$ and consider the random component of G. With overwhelming probability, $\bar{S}$ will have a set T of n^{δ} isolated vertices not connected to S, where δ is some constant that depends on

ϵ. Using T, an adversary can embed a hard instance of independent set in the semi-random graph.

Let G' be a graph on $3n^\delta/2$ vertices, V', in which we seek to find an independent set of size $n^\delta/2$ (this problem is NP-hard). Let V_1 consist of $\alpha n - n^\delta/2$ vertices, V_2 consist of $n - \alpha n - n^\delta$ vertices. Construct the graph G with vertices $V' \cup V_1 \cup V_2$. The edges of G consist of those of G' (those connect only vertices in V'), and all edges between every vertex in V_2 and any other vertex (hence vertices in V_2 have degree $n-1$, and vertices in V_1 have degree $|V_2|$). Then, permute the vertex labels of G at random. It follows from the construction that G has an independent set of size αn iff G' has an independent set of size $n^\delta/2$. Observe that given G', the graph G is constructed in random polynomial time.

We next argue that when constructing a semirandom graph with an independent set of size αn, the adversary can create a graph isomorphic to G whenever G_{min} has n^δ isolated vertices (which happens with overwhelming probability) and G' has an independent set of size $n^\delta/2$. Hence, if there existed an algorithm that didn't fail with overwhelming probability, it must succeed on G with some constant probability, revealing that G' has such an independent set. Theorem 2 would then follow. Note that the adversary need not run in polynomial time.

Our adversary computes an independent set I of G', where $|I| = n^\delta/2$. Given G_{min}, with a set T of n^δ isolated vertices in $\bar{S}$, the adversary maps the $n^\delta/2$ vertices of I arbitrarily to distinct vertices of S and maps the n^δ vertices of $V' \setminus I$ (distinctly) to T. Given this mapping, it then adds the edges corresponding to those in G'. By the construction, it is allowed to add these edges, and furthermore, every nonedge in G' corresponds to a nonedge in G_{min}. Finally, the adversary connects every vertex of $\bar{S} \setminus T$ to every other vertex. It can be verified that this graph is isomorphic to G. V_1 corresponds to those vertices in S that do not correspond to I and V_2 corresponds to $\bar{S} \setminus T$. $\square$

Proof of Lemma 7: Consider the following semidefinite program (which can be solved up to arbitrary precision in polynomial time, using the Ellipsoid algorithm). Given the graph G, we find an order n matrix $X = \{x_{ij}\}$ satisfying the following constraints:

1. $\forall i, j,\ 0 \le x_{ij} \le 1$,
2. $\forall (i,j) \in E,\ x_{ij} = 0$,
3. $\sum_i x_{ii} = K$,
4. $\forall i,\ \sum_j x_{ij} = K x_{ii}$,
5. The matrix X is positive semidefinite.

As G has an independent set S of size K, the above semidefinite program is feasible. Setting $x_{ij} = 1$ whenever $v_i, v_j \in S$, and $x_{ij} = 0$ otherwise, gives a matrix X satisfying the above constraints. To see that X is semidefinite, let q_i be 1 if $v_i \in S$, and 0 otherwise, and observe that $X = YY^T$, where $Y = [q_1\ q_2 \cdots q_n]^T$ and Y^T denotes the transpose of Y.

A positive semidefinite matrix X can be decomposed in polynomial time into $X = YY^T$, where Y is a matrix with n rows, and Y^T is its transpose. We denote the row vectors of Y by $y_1, \ldots, y_n$. The entry x_{ij} is the inner product $\langle y_i, y_j \rangle$. Let $y_0 = \sum_i y_i/K$. From constraints 3 and 4 it follows that $\langle y_0, y_0 \rangle = K^2/K^2 = 1$. Hence y_0 is a unit vector. Moreover, constraint 4 implies that for every i, $\langle y_i, y_i \rangle = \langle y_0, y_i \rangle$. (Geometrically, this means that the points y_i, $1 \le i \le n$, all lie on an n-dimensional sphere of radius $1/2$, and the points 0 and y_0 are antipodal.)

Assume w.l.o.g. that the vectors y_i are sorted by their lengths, and consider now only the vectors $y_1, \ldots, y_{K/2}$. For every such vector, $K/2n < \langle y_i, y_i \rangle \le 1$ (from constraints 1 and 3). Associate now with the vertices $v_1, \ldots, v_{K/2}$ unit vectors $z_1, \ldots, z_{K/2}$, where vector z_i is in direction $y_i - \langle y_0, y_i \rangle y_0$ (i.e., we project out direction y_0).

Proposition 13 *For* $1 \le i < j \le K/2$, *if* $(v_i, v_j) \in E$, *then* $\langle z_i, z_j \rangle < -K/(2n-K)$.

Proof: Let $w_i = y_i - \langle y_0, y_i \rangle y_0$. If $(v_i, v_j) \in E$ then $\langle y_i, y_j \rangle = 0$. This gives

$$\begin{aligned}\langle w_i, w_j \rangle &= 0 - 2\langle y_0, y_i \rangle \langle y_0, y_j \rangle + \langle y_0, y_0 \rangle \langle y_0, y_i \rangle \langle y_0, y_j \rangle \\ &= -\langle y_0, y_i \rangle \langle y_0, y_j \rangle.\end{aligned}$$

This implies that $\langle z_i, z_j \rangle < 0$, but does not bound its magnitude, since the w_i are not unit vectors. By using the interpretation of the y_i as lying on a sphere of radius $1/2$ we obtain that their lengths satisfy $(|w_i|)^2 + (1/2 - \langle y_0, y_i \rangle)^2 = 1/4$, implying that $z_i = w_i/\sqrt{\langle y_0, y_i \rangle - (\langle y_0, y_i \rangle)^2}$. It follows that

$$\langle z_i, z_j \rangle = -\frac{\langle y_0, y_i \rangle}{\sqrt{\langle y_0, y_i \rangle - (\langle y_0, y_i \rangle)^2}} \cdot \frac{\langle y_0, y_j \rangle}{\sqrt{\langle y_0, y_j \rangle - (\langle y_0, y_j \rangle)^2}}.$$

By elementary calculus it can be shown that the function $f(x) = x/\sqrt{x - x^2}$ is positive and monotone increasing over $(0,1)$; hence $\langle z_i, z_j \rangle$ is minimized when $\langle y_0, y_i \rangle$ and $\langle y_0, y_j \rangle$ are minimized. As these minimums are in both cases more than $K/2n$, it follows that $\langle z_i, z_j \rangle < -(K/2n)^2/(K/2n - (K/2n)^2) = -K/(2n-K)$. $\square$

This completes the proof of Lemma 7. $\square$

Proof of Lemma 10: Consider an arbitrary set V_{ij} that is the outcome of $\mathsf{partition}(Q_i, h)$. We define the *surplus* of V_{ij} as $\mathsf{sur}(V_{ij}) = |S \bigcap V_{ij}| - |\bar{S} \bigcap V_{ij}|$. From V_{ij}, an arbitrary maximal matching is removed so as to obtain an independent set I_{ij} (that may possibly be empty).

Clearly, $\mathsf{sur}(V_{ij}) \leq |I_{ij} \bigcap S|$. Recall that I_{ij} is useful if $|I_{ij} \bigcap S| \geq 3|I_{ij}|/4$. From Lemma 4 it can easily be deduced that if $|I_{ij} \bigcap S| > n/(\log n)^{\delta}$ for some $0 < \delta < 1$, then $|I_{ij} \bigcap \bar{S}| = o(|I_{ij}|)$. Hence I_{ij} is useful. From the above it follows that if $\mathsf{sur}(V_{ij}) > n/(\log n)^{\delta}$ then necessarily I_{ij} will be useful. We now show that many sets V_{ij} have a large surplus, which then implies Lemma 10.

By Proposition 8, at least $\alpha n/2$ vertices from S belong to good V_i. A good V_i is partitioned by h random hyperplanes into m parts $\{V_{ij}\}$. From Lemma 9, the expectation of $|E(V_{i1}, \ldots, V_{im})|$ is at most $|E_i|(1-\rho)^h$. Hence the expectation of $\sum_i |E(V_{i1}, \ldots, V_{im})|$ is at most $\sum_i |E_i|(1-\rho)^h$. From Lemma 6 it follows that $\sum_i |E_i| \leq 2n \ln n/\alpha$. Hence with probability at least $1/2$ we have that

$$\sum_i |E(V_{i1}, \ldots, V_{im})| \leq 4n \ln n (1-\rho)^h/\alpha.$$

By our choice of h, $(1-\rho)^h \leq \alpha^2/16 \ln n$, implying

$$\sum_i |E(V_{i1}, \ldots, V_{im})| \leq \alpha n/4.$$

Recall that our algorithm removes a maximal matching from each V_{ij}. For the sake of analysis, we as a thought experiment remove a maximal matching containing only edges from E_i, giving a set U_{ij}. (The algorithm itself does not have the luxury of knowing which edges of Q_i belong to E_i). Then over all good V_i, the total number of edges removed is at most $\alpha n/4$, meaning that at least $\alpha n/2 - \alpha n/4 = \alpha n/4$ vertices from S remain in the sets $\{U_{ij}\}$. Call a set U_{ij} *large* if $|U_{ij} \bigcap S| \geq \alpha^2 n/128m$. As there are at most $8m/\alpha$ sets U_{ij}, those that are not large can contain in total at most $\alpha n/16$ vertices from S. Hence at least $\alpha n/4 - \alpha n/16 = 3\alpha n/16$ vertices from S are contained in large U_{ij}.

By our choice of h and the fact that $\rho > 1/2$, large U_{ij} contain $\Omega(n/(\log n)^{\delta})$ vertices of S for some $\delta < 1$. As they contain no edges from G_{min}, Lemma 4 implies that for large U_{ij}, $|U_{ij} \bigcap \bar{S}| \leq o(|U_{ij}|)$. To obtain U_{ij} from V_{ij}, exactly $(|V_{ij}| - |U_{ij}|)/2$ vertices of $S \bigcap V_{ij}$ were removed. Hence the surplus of V_{ij} must be $(1 - o(1))|U_{ij}|$. So if we consider only those V_{ij} that had large U_{ij} in the above experiment, they give rise to useful I_{ij} containing a total of at least

$$(1 - o(1)) \sum_{\text{large } U_{ij}} |U_{ij}| \geq (1 - o(1))3\alpha n/16 > \alpha n/8$$

vertices from S. $\square$

B. Graph bisection

The bisection size $b(G)$ of a graph G is the minimum number of edges in a balanced cut (each side contains $n/2$ vertices). This problem is NP-hard. Boppana [6] develops a heuristic for this problem, and analyses its performance on random graphs with planted bisections. Specifically, he considers random graphs whose vertex set is partitioned into two equal size sets S and $\bar{S}$, and two vertices are connected by an edge with probability p if they belong to the same set, and with probability q if they belong to different sets. If q is sufficiently smaller than p, then w.h.p. $S, \bar{S}$ is a unique minimum bisection. Boppana shows that when

$$p - q \geq 10\sqrt{(p+q)\frac{\log n}{n}}$$

then with high probability (over the choice of input graph) his heuristic recovers the minimum bisection.

We consider a semirandom model for graph bisection. In this model, a graph G_{rand} is chosen at random as above. Then an adversary removes edges of his choice from the cut $S, \bar{S}$, and adds edges of his choice outside the cut (i.e., connecting pairs of vertices in S, or pairs of vertices in $\bar{S}$). This gives the semirandom graph G. Clearly, if $S, \bar{S}$ is the minimum bisection in G_{rand}, then it is a minimum bisection also in G.

We show that w.h.p. over the choice of G_{rand}, an algorithm similar to that of Boppana outputs $b(G)$. To also output the bisection itself, determine which pairs of vertices belong to the same side of the bisection by adding/removing an edge between them to obtain a graph G', and checking whether $b(G') = b(G)$. (We use here the fact that G' is sufficiently random for Boppana's algorithm to work.)

Let $h(G)$ be an arbitrary function on graphs which has the following properties:

1. *Lower bound.* For every graph G, $h(G) \leq b(G)$.

2. *Bounded monotonicity.* Let G^+ be the graph G with one additional edge. Then $h(G) \leq h(G^+) \leq h(G) + 1$.

3. *Probably good.* With high probability over the choice of G_{rand},
$$h(G_{\text{rand}}) = b(G_{\text{rand}}).$$

For any function h satisfying the above properties, we have that with high probability over the choice of G_{rand}, $h(G) = b(G)$ also in the semirandom model. This follows from the fact that whenever $h(G_{\text{rand}}) = b(G_{\text{rand}})$, then necessarily $h(G) = h(G_{\text{rand}}) - r$, where r is the number of edges removed from the cut by the adversary. (When $h(G_{\text{rand}}) = b(G_{\text{rand}})$, each edge removed from the cut decreases $b(G)$ by one and $h(G)$ by at most one (bounded monotonicity), and hence $h(G)$ by exactly one (lower bound). Each edge added outside the cut does not increase $b(G)$, and hence can neither increase $h(G)$ (lower bound), nor decrease $h(G)$ (monotonicity).)

Boppana [6] describes a polynomial time computable function h and shows that it satisfies properties 1 and 3 above. It is not easy (for us) to verify that Boppana's function satisfies property 2. Instead, we propose the following semidefinite relaxation of bisection, for which proving properties 1 and 2 is straightforward, and property 3 is proved using the techniques of [6]. For a graph $G(V,E)$ find an order n matrix $X = \{x_{ij}\}$ satisfying:

1. $\forall i, x_{ii} = 1$,
2. $\sum_{ij} x_{ij} = 0$,
3. The matrix X is positive semidefinite,

and let $h(G)$ be given by

$$h(G) = \min_X \sum_{\substack{(i,j) \in E \\ i<j}} \frac{1 - x_{ij}}{2},$$

where X ranges over all matrices satisfying the above constraints. Using semidefinite programming, $h(G)$ can be computed in polynomial time within arbitrary precision limits. To see that $h(G) \leq b(G)$, consider the indicator vector $s \in \{+1, -1\}^n$ having entry $+1$ for vertices in S and -1 for vertices in $\bar{S}$, and let $X = ss^T$. To see that h is bounded monotone use the fact that $|x_{ij}| \leq 1$ and $0 \leq \frac{1-x_{ij}}{2} \leq 1$. To see that h is probably good, consider the dual of the above semidefinite minimization problem (see [1] for rules how to obtain the dual).

Maximize $m/2 + (\sum_{i=1}^n y_i)/4$ s.t.
$-A - y_0 J - Y$ is p.s.d.
Here m is the number of edges in the graph, A is its adjacency matrix, J is the all 1 matrix, y_0 is an auxiliary variable which affects feasibility but not the objective function, and Y is a diagonal matrix with $y_1, \ldots, y_n$ along its diagonal.

For every $1 \leq i \leq n$, let y_i be the difference between the number of neighbors that vertex i has on the other side of the bisection and the number of neighbors that vertex i has on the same side of the bisection. The value of the maximization semidefinite program is then exactly $b(G)$ (though we have not yet shown that the solution is feasible). Furthermore, regardless of the value of y_0, the vector s described above is an eigenvector of the matrix $M = -A - y_0 J - Y$ with eigenvalue 0. It remains to show that w.h.p. (over the choice of G_{rand}), it is possible to choose y_0 such that s is the eigenvector of smallest eigenvalue. For this, choose $y_0 = -1$, giving $-A + J = A^c + I$, where A^c is the adjacency matrix of the complement of the graph G_{rand}, and I is the identity matrix. The complement graph can be viewed as a sum of two random graphs – one with edge probability $1-p$, and the other a bipartite graph with edge probability $p-q$. From [11], with high probability, $-O(\sqrt{pn})$ lower bounds hold for the smallest eigenvalue of the first graph and for the second smallest eigenvalue of the second graph. These bounds can be combined with $(p-q)n/2 - O(\sqrt{pn \log n})$ lower bounds on the entries of the matrix $(-Y)$, that hold with high probability, showing that M is almost surely positive semidefinite when $p-q$ is large enough. Details omitted.

Remark: Possibly, for every graph G, our function h and Boppana's function give the same value. See also [16] for other semidefinite formulation for graph bisection.