Beyond the Worst-Case Analysis of Algorithms

Edited by
Tim Roughgarden

Contents

1
Introduction to Semi-Random Models

Uriel Feige

Abstract

This chapter introduces *semi-random models*, in which input instances are generated by a process that combines random components with adversarial components. These models may bridge the gap between worst case assumptions on input instances, that often are too pessimistic, and purely random "average case" assumptions, which might be too optimistic. We discuss several semi-random frameworks. We present algorithmic paradigms that have been proved effective in handling semi-random instances, and explain some principles used in their analysis. We also discuss computational hardness results for the semi-random setting.

1.1 Introduction

In semi-random models, input instances are generated by a process that combines random components with adversarial components. There are different ways by which these components can be combined, and indeed, many different semi-random models have been proposed. In this section we present several such models. In Section 1.2 we explain considerations that motivate the introduction of semi-random models. In Section 1.3 we survey some representative past work on semi-random models. In Section 1.4 we list some open questions.

1.1.1 Examples of semi-random models

In distributional models (see Chapter 8 in this book) the input instance is generated by a random process. In semi-random models, generation of input instances involves both a random component and an adversarial (worst case) component. We present here examples of semi-random models, and contrast them with related distributional models that have no adversarial component. In all models considered in this chapter, the algorithm that is faced with a computational problem (3SAT, 3-coloring, min

bisection, maximum independent set, in our examples) gets to see only the resulting input instance, but not the way by which it was generated.

In our first example, the adversary first chooses a tentative input instance, and then the final input instance is generated by applying a small random perturbation to the tentative input instance. Semi-random models of this nature are studied in the area of *smoothed analysis* (see Part IV in this book).

3SAT.

Worst case. The input is an arbitrary 3CNF formula ϕ. A 3CNF formula is a collection of clauses, where each clause contains three literals, where a literal is a Boolean variable or a negation of a Boolean variable. A satisfying assignment is a truth assignment to the Boolean variables such that in every clause at least one literal is set to *true*. The goal is to determine whether ϕ is satisfiable, namely, whether there is a satisfying assignment.

Distributional. Given positive integer parameters n (for the number of variables) and m (the the number of clauses), one generates independently at random m clauses, where each clause contains three variables chosen uniformly at random from the $\binom{n}{3}$ triples of variables, and the polarity of the variables in each clause (determining whether the literal associated with the variable is negated) is chosen uniformly at random. The goal is to determine whether the resulting 3CNF formula ϕ is satisfiable.

Semi-random. Given integer parameters n and m and a parameter p (where $0 < p < 1$), an adversary generates a 3CNF formula ϕ' containing m clauses of its choice. This completes the adversarial part of the construction. Thereafter, for each literal in ϕ', its polarity is flipped independently with probability p. The goal is to determine whether the resulting 3CNF formula ϕ is satisfiable.

In our next example, a tentative input instance is first generated in a distributional manner, and thereafter an adversary is allowed to modify the tentative input instance in some restricted way. Often, the forms of modification that are allowed are meant to capture modifications under which the resulting final instance should not be more difficult to solve than the original tentative instance. We refer to such an adversary as a *monotone adversary*.

Minimum bisection.

Worst case. The input is an arbitrary graph $G(V, E)$ with an even number n of vertices. The goal is to output a set $S \subset V$ of cardinality $\frac{n}{2}$ for which the number $|E(S, V \setminus S)|$ of cut edges is minimized.

Distributional. Given an even integer n and parameters $\frac{1}{n} \leq p < q \leq 1 - \frac{1}{n}$ one generates a graph $G(V, E)$ (with $|V| = n$) as follows. A subset $S \subset V$ of size $\frac{n}{2}$ is chosen at random. For every pair of vertices (u, v) with $u \in S$ and

$v \in V \setminus S$, the edge (u, v) is included in E independently with probability p. For other pairs (u, v) of distinct vertices (either both vertices in S or both not in S) the edge (u, v) is included in E independently with probability q. The resulting graph $G(V, E)$ is the input graph, and the goal is to output the minimum bisection. The set S is referred to as the *planted bisection*. If $q - p$ is sufficiently large then with high probability the unique minimum bisection is S.

Semi-random. Given n, p, q, first one generates a random input graph exactly as explained above. This completes the random component of the construction. Thereafter, an adversary may observe the random graph and remove from the cut $(S, V \setminus S)$ arbitrary edges of its choice, and add elsewhere (within S or within $V \setminus S$) arbitrary edges of its choice. The goal is to output the minimum bisection in the resulting input graph. If S was the minimum bisection in the random graph, it remains so in the semi-random graph.

In the following example, one of the random steps in a distributional model is replaced by an adversarial step. We refer here to such models as *separable*, as we separate between the components generated at random and those generated by the adversary. The semi-random model presented for 3SAT, in the special case in which $p = \frac{1}{2}$, is one such separable model, as the choice of variables is completely adversarial, whereas the choice of polarities is completely random. We now present a separable model for 3-coloring.

3-coloring.

Worst case. The input is an arbitrary graph $G(V, E)$. The goal is to legally 3-color its vertices, if possible. A legal 3-coloring is a partition of V into three sets of vertices, referred to as color classes, where the subgraph induced on each color class is an independent set.

Distributional. Given parameters n and $0 < p < 1$, first, a graph $G(V, E')$ is generated as an Erdos-Renyi $G_{n,p}$ random graph (where there are n vertices, and every edge is present independently with probability p). Thereafter, every vertex independently at random is associated with one of the colors, c_1, c_2 or c_3. This association is referred to as the planted 3-coloring. All monochromatic edges (edges whose endpoints are associated with the same color) are removed. The resulting graph is the input graph $G(V, E)$. The goal is to output a legal 3-coloring. If p is sufficiently large (e.g., $p \geq \frac{2 \log n}{n}$ suffices) then with high probability the planted 3-coloring is the unique legal 3-coloring of $G(V, E)$.

Semi-random. Given parameters n and $0 < p < 1$, the input graph $G(V, E)$ is generated as follows. First, a graph $G(V, E')$ is generated as a $G_{n,p}$ random graph. This completes the random component of the construction.

Thereafter, an adversary observes the random graph and associates with each vertex one of the three colors, c_1, c_2 or c_3, with the only restriction being that every color class is of size $\frac{n}{3}$ (rounded up or down to an integer value). All monochromatic edges (with respect to this planted 3-coloring) are removed. The goal is to output a legal 3-coloring for the resulting input graph $G(V, E)$. Here, even if p is fairly large (but not larger than $\frac{1}{3}$), $G(V, E)$ may have legal 3-colorings that differ from the planted one.

Our final example (for this section) gives a semi-random model that is a variation on a distributional planted model. In the underlying distributional planted model, with high probability the planted solution is the unique optimal solution. In contrast, in the semi-random model the planted solution might be far from optimal. The algorithmic goal in the semi-random setting is to find a solution that is as good as the planted one.

Maximum independent set (MIS).

Worst case. The input is an arbitrary graph $G(V, E)$. The goal is to output a set $S \subset V$ of maximum cardinality that induces an independent set (namely, $(u, v) \notin E$ for every $u, v \in S$).

Distributional. Given parameters n, k and p (where $k < n$ are positive integers and $0 < p < 1$), first, a graph $G(V, E')$ is generated as a $G_{n,p}$ random graph. Thereafter, a random set $S \subset V$ of k vertices is turned into an independent set by removing from E' all edges of the form (u, v) for $u, v \in S$. This S is referred to as the *planted independent set*. The resulting graph $G(V, E)$ is the input graph, and the goal is to output an independent set of maximum size. If k is sufficiently large (e.g., for $p = \frac{1}{2}$ it suffices to take $k = 3 \log n$) then with high probability the unique independent set of maximum size is S.

Semi-random. Given parameters n, k and p, the input graph $G(V, E)$ (with $|V| = n$) is generated as follows. First, a graph with a planted independent set S is generated exactly as in the distributional model. This completes the random component of the construction. Thereafter, all edges within $V \setminus S$ are removed. Finally, an adversary observes the random graph (whose only edges are between S and $V \setminus S$) and may add to it arbitrary edges of its choice, as long as none of the added edges has both endpoints inside S. Hence the adversary has complete control over the subgraph induced on $V \setminus S$, and acts as a monotone adversary with respect to the edges between S and $V \setminus S$. The goal is to output an independent set of size at least k in the resulting input graph $G(V, E)$. Note that regardless of the edges added by the adversary, S itself is a feasible solution, but depending on the edges (not) added by the adversary, there might be other feasible solutions.

1.2 Why study semi-random models?

A semi-random model involves a random component and an adversarial component. As we have seen in Section 1.1.1 (and we will see additional examples later), there are many different roles that we can delegate to the adversary when constructing the input instance. Below we discuss some of the considerations involved in proposing a semi-random model. As semi-random models are often conceived as refinements of distributional models, we shall also discuss some of the motivations for studying distributional models, with emphasis on those motivations that apply also to semi-random models.

In the discussion that follows, it will be convenient to distinguish between two classes of distributional models. We shall refer to one class as that of *oblivious random models*, and to the other as *planted models*.

In oblivious random models, the input is generated by a random process that is independent of the optimization problem that one is interested in solving. The distributional model given for 3SAT in Section 1.1.1 is an example of an oblivious random model, and the model applies without change to other constraint satisfaction problems such as 3AND or *not-all-equal* 3SAT. An example of an oblivious random model for graph problems is the Erdos-Renyi $G_{n,p}$ random graph model. This model applies to any graph problem, though the range of interest for the parameter p might depend on the optimization problem of interest. For example, for max clique one might choose $p = \frac{1}{2}$, for Hamiltonicity one might choose $p = \frac{\log n}{n}$, whereas for 3-coloring one might choose $p = \frac{4}{n}$.

In planted models, the input is generated by a random process that depends on the optimization problem that one is interested in. For example, the planted distributional models considered in Section 1.1.1 for the three graph problem, min bisection, 3-coloring and maximum independent set, are different from each other. In planted models, the planted solution is often the unique optimal solution.

1.2.1 Average case analysis

Distributional models are sometimes assumed to represent input instances that may occur in practice. The extent to which one can defend such assumptions depends on the setting.

In some settings, distributional models exactly capture the set of interesting input instances. Most notably, this happens in settings related to cryptography, in which participants in a cryptographic scheme are instructed to generate various inputs to the scheme at random. For example, in public key cryptographic schemes such as RSA, a participant is instructed to generate its public key by privately picking two large random primes p and q, computing $n = pq$, and publishing n as the public key. Factoring algorithms that are tailored for this specific distributional setting have far reaching consequences for the underlying cryptographic scheme.

In some settings, distributional models are conjectured to be a good approximation of reality. For example, studies in statistical physics often involve random graph models that have a geometric nature: the vertices of the graph lie in a low dimensional space, and only vertices that are geometrically close to each other may be connected by edges. The random aspect of the model can either be in the location of the vertices, or in the choice of which of the possible edges are indeed edges.

However, in many settings, the relation between distributional models and typical instances that occur in practice is less clear. For example, in the study of social networks, one often considers distributions over graphs that are generated by a random process such as preferential attachment. These distributions may capture some of the typical aspects of social networks (such as typical degree sequences), though for any given social network of interest, there may be other important aspects (such as the relative frequencies of various small subgraphs) that are not captured well by the underlying distributional model.

Oblivious random models typically exhibit a multitude of nearly optimal solutions, with pairs of nearly optimal solutions that are very different from each other. As argued in Chapters 5 and 6, in some settings, an optimal solution to an optimization problems is most useful when it is basically unique, and of significantly better value than those solutions that significantly differ from it. In these cases, planted models can serve as the basis for average case analysis of instances of interest, as the planted solution are often the unique optimal solutions. But also here, the planted models capture only some aspects of "interesting" input instances, and may not capture other aspects.

Summarizing the discussion above, distributional models, whether oblivious or planted, are sometimes meant to represent average case analysis, but in many cases there is a major difficulty of characterizing the "right" distributional model for the set of problems that may appear in practice. In such settings, it is important that algorithms that are designed for inputs generated by the distributional model will be robust, and work well also for inputs that are generated by other processes. A way of addressing this concern is through the use of semi-random models. In these models, the exact distribution of input instances is not known to the algorithm, as some aspects of the input instance are left to the discretion of an adversary. Consequently, algorithms designed for semi-random models avoid the danger of "over-fitting" to a particular input distribution, and they are expected to be more robust than algorithms that are designed for distributional models. This is one of the major reasons for introducing semi-random models such as the monotone adversary (e.g., for min bisection in Section 1.1.1). In Section 1.3 we will see examples of how such models direct the design of algorithms towards algorithms that are more robust.

Another contribution of semi-random models is in providing refinements of *average case analysis*, clarifying what it is that one actually averages over. In semi-random models, some aspects of the input instance are chosen by an adversary, and

the random distribution is over some other aspects. For example, in smoothed models (e.g., for 3SAT in Section 1.1.1), the algorithm needs to work well on average not only when the average is taken as a global average over all input instances, but also when one takes a local average around any particular tentative input instance, regardless of what this tentative input instance is. Another example is random order models for online algorithms (see Chapter 11) where an adversary may select a worst case instance, and only the order of arrival is random.

1.2.2 Recovering a signal contaminated by noise

Randomness in generation of input instances can sometimes be thought of as representing "noise" that makes finding an otherwise obvious solution more difficult. For example, for error correction of encoded messages transmitted over a noisy channel, the source of difficulty in decoding is because of errors introduced into the encoded message by the noisy channel. In the absence of noise, decoding the transmitted message is trivial. With noise, decoding typically involves two aspects. One is information theoretic – does the noisy message contain sufficient information in order to uniquely recover the transmitted message (with high probability)? The other is algorithmic, designing an efficient algorithm for recovering encoded messages from their noisy received message.

Noise is often modelled as being random. For example, in a binary symmetric channel (BSC) each transmitted bit is flipped independently with probability $p < \frac{1}{2}$. However, it is also reasonable to model noise as being semi-random. For example, one may assume that each transmitted bit i is flipped with probability $p_i \leq p$ rather than exactly p, where the value of p_i is determined by an adversary. If a decoding algorithm works in the former model but not in this latter model, this may be a sign of "over-fitting" the algorithm to the model. Noise is also often modelled as being fully adversarial. In Hamming's model, within a block of bits, the total fraction of bits flipped is at most p, but an adversary may decide which bits are flipped. Decoding in the information theoretic sense is more difficult in the Hamming model than in the BSC model (the transmission rate under which unique decoding is possible is smaller in the Hamming model). Also, decoding in the algorithmic sense appears to be more difficult in the Hamming model. This last statement is supported by the observation that for every $p < \frac{1}{2}$ and $p' < p$, if block sizes are sufficiently large, every decoding algorithm for the Hamming model with p fraction of errors works also in the BSC model when the error probability is p'.

In analogy to the coding setting, planted models can also be viewed as representing an ideal object contaminated by noise. Under this view, the goal is typically to recover the ideal object. Solving an optimization problem associated with the object may serve as a means towards this end, but is not the goal by itself. This view is different from that taken in most of this chapter, where the goal is typically

to solve an optimization problem, and a solution is accepted even if it does not correspond to the planted object.

As an example, for the MIS problem, the ideal object is an independent set of size k in an otherwise complete graph. This ideal object is contaminated by noise, where noise corresponds to removing some of the edges of the graph. If every edge is removed independently with the same probability (in analogy to independent noise), one gets the standard distributional model for planted independent set (but with the goal of finding the planted independent set, rather than that of finding a maximum independent set). Semi-random models for MIS correspond to models in which the noise is not independent, and this makes recovering the ideal object more difficult.

1.2.3 A model for worst case instances

To make progress (e.g., achieve a better approximation ratio) in the design of algorithms for a difficult computational problem, it is useful to have an understanding of which are the most difficult instances of the problem. For some problems, distributional models are conjectured to produce input instances which are essentially as difficult as worst case instances. For example, it is conjectured that 3SAT on random 3CNF formulas with dn clauses (for some large constant d – such formulas are unlikely to be satisfiable) are essentially as difficult to refute as adversarially chosen 3CNF formulas with dn clauses. For some other problems, such as the dense k-subgraph problem (given an input graph and a parameter k, find the induced subgraph on k vertices with the highest average degree), distributional models appear to capture the limitations of currently known algorithms, and progress on distributional instances played a key role in improving the approximation ratio also for worst case instances (see Bhaskara et al. (2010)).

There are problems whose approximability is not well understood (examples include sparsest cut, unique games) and (natural) distributional models produce instances on which known algorithms perform much better than the best approximation ratios known for worst case instances. In such cases, it is instructive to consider semi-random instances, and try to extend the good performance of algorithms to the semi-random instances (as done in Kolla et al. (2011)). Success in this effort may suggest that known algorithmic approaches might also suffice in order to handle worst case instances (even though we might be lacking the analysis to support this), whereas failure may help clarify what aspects of input instances are those that create difficulties for currently known algorithms.

1.2.4 NP-hardness

The theory of NP-completeness has great value in informing us that certain problems do not have polynomial time algorithms (unless P=NP), and hence that we

should not waste efforts in trying to design polynomial time algorithms for them (unless we are seriously trying to prove that P=NP). This theory has been extended to proving NP-hardness of approximation results. This plays a key role in directing research on approximation algorithms to those problems (such as sparsest cut) for which there still is hope for substantial improvements, and away from problems (such as max-3SAT) for which there is only negligible room for improvements. Unfortunately, the theory of NP-completeness has not been successfully extended (so far) to distributional problems, and hence it is difficult to judge whether our failure to find good algorithms for a distributional problem (in those cases where we fail) is because there really is no good algorithm for handling the instances generated by the distribution, or because we are not using the right algorithmic tools for the distributional problem. This makes it difficult to classify which distributional problems are easy and which are hard.

One of the advantages of semi-random models is that their adversarial component offers us possibilities for proving NP-hardness results. Consequently, it is not rare that for semi-random models, for certain ranges of parameters we have polynomial time algorithms, and we also have NP-hardness results that explain why the algorithmic results do not extend to other ranges of the parameters. Hence research on algorithms for semi-random models can be guided by the theory of NP-completeness towards problems where there is hope to make progress, and away from problems for which progress is hopeless. This aspect is missing in research on algorithms for distributional problems.

1.3 Some representative work

In this section we shall present some key insights that emerged in the study of semi-random input models. In doing so, we shall provide some historical perspective of how these ideas developed (though not necessarily in historical order).

1.3.1 Preliminary results on semirandom models

Blum and Spencer (1995) (following earlier work by Blum (1990)) motivated and introduced several semi-random models for the k-coloring problem. One of these models, referred to in their work as the *colorgame* model, is a *monotone adversary* model for k coloring. In this model, the set of vertices is partitioned into k equal sized color classes. Thereafter for every pair of vertices u, v in different color classes, an edge (u, v) is introduced independently with probability p. The edges introduced in this stage are referred to as *random edges*. Finally, the adversary may introduce arbitrary additional edges between color classes, referred to as *adversarial edges*. The goal is to design polynomial time algorithms that k-color the resulting graph,

for a wide range of values of k and p. As in all semi-random models, the coloring algorithm is not told which edges are random and which are adversarial.

For $k = 3$ Blum and Spencer propose the following algorithm. Let $N(v)$ denote the set of neighbors of a vertex v. Two vertices u and v are said to be *linked* if the subgraph induced on $N(u) \cap N(v)$ includes at least one edge. Observe that in every legal 3-coloring, two linked vertices must both be colored by the same color, because in every legal coloring their common neighborhood requires at least two colors. Consequently, two linked vertices u and v may be *merged*, namely, replaced by a single vertex w, with $N(w) = N(u) \cup N(v)$. The new graph is 3-colorable if and only if the original graph is 3-colorable. Any two vertices that were linked in the original graph are also linked in the new graph, but there may be vertices that were not linked in the original graph and become linked in the new graph. Repeatedly merging linked vertices whenever possible (in an arbitrary order – all orders give the same final outcome), the algorithm is successful if the final resulting graph is a triangle. In this case the graph has a unique 3-coloring: for every vertex t of the triangle, the set of vertices that were merged in order to give t forms a color class. Observe that the algorithm is monotone in the following sense: if it is successful for a graph G, then it also successful for every 3-colorable graph G' that can be obtained from G by adding edges to G. This follows because any sequence of merge operations that is performed in G can also be performed in G'. The only edge addition that can prevent a merge between linked vertices u and v is to add the edge (u, v), but this is not allowed because the resulting graph will not be 3-colorable.

Blum and Spencer proved that when $p > n^{-0.6+\epsilon}$ there is high probability (over the choice of the random edges, regardless of the choice of the adversarial edges) that the algorithm indeed 3-colors the graph. At this low edge density, initially most pairs of vertices do not have any common neighbors and hence cannot possibly be linked, and the crux of the proof is in showing that as the algorithm progresses, more pairs of vertices become linked. The algorithm can be adapted to k-coloring of k-colorable semi-random graphs (two vertices are linked if their common neighborhood contains a K_{k-1}), though the required value of p increases to $n^{-\delta_k+\epsilon}$, for $\delta_k = \frac{2k}{k(k+1)-2}$.

Blum and Spencer also considered an *unbalanced* k-colorable semi-random model in which the sizes of different color classes can differ significantly, and showed an NP-hardness result for coloring such graphs.

Theorem 1.1 (Blum and Spencer (1995)) *For every $k \geq 4$ and every $\epsilon > 0$, if $p \leq n^{-\epsilon}$ then it is NP-hard to k-color graphs that are produced by the monotone adversary unbalanced semi-random model for k-coloring.*

Proof We sketch the proof for $k = 4$. Suppose that $p = n^{-3\epsilon}$ for some $0 < \epsilon < 1$. Let H be an arbitrary graph on $3n^{\epsilon}$ vertices for which one wishes to find a 3-coloring. This problem is NP-hard, but can be reduced to the problem of 4-coloring a semi-random graph with unbalanced color classes. This is done by creating a graph G^* that is composed of a disjoint union of H and an independent set I of

size $n - 3n^\epsilon$, and connecting every vertex $u \in H$ and $v \in I$ by an edge (u, v). Every 4-coloring of G^* must 3-color H, and moreover, deriving the 3-coloring for H from the 4-coloring of G^* can be done in polynomial time. Hence if 3-coloring H is NP-hard, so is 4-coloring G^*.

However, G^* can be obtained with high probability as the outcome of the unbalanced semi-random 4-coloring model. Suppose for simplicity of the presentation that the three color classes of H are of equal size. Then consider the unbalanced 4-coloring semi-random model with one "large" color class of size $n-3n^\epsilon$ and three "small" color classes, each of size n^ϵ. With high probability, all random edges in the construction of the input graph will have at least one of their endpoints in the large color class, and no edges between the small color classes. If this high probability event happens, then the monotone adversary can add between the three small color classes a set of edges that make the subgraph induced on them isomorphic to H, and also add all missing edges between the large color class and each of the small color classes, and this results in the graph G^*. As we argued that it is NP-hard to 4-color G^*, it is NP-hard to 4-color graphs in the unbalanced semi-random 4-coloring model. □

1.3.2 Planted clique/MIS with a monotone adversary

In this section we shall discuss algorithms for a semi-random model for the maximum independent (MIS) set problem. The model and associated algorithms can easily be adapted to the clique problem as well, due to the fact that a set S of vertices forms a clique in G if and only if it forms an independent set in the complement graph $\bar{G}$

The following is a standard distributional model $G_{n,k,\frac{1}{2}}$ for MIS, often referred to as planted MIS, or hidden MIS (and in analogy, planted/hidden clique – see Chapter 8). One first generates a random $G_{n,\frac{1}{2}}$ graph G'. In G', one selects a set S of k vertices at random, and removes all edges within S. The result is the input graph G. The goal is to design a polynomial time algorithm that with high probability (over the choice of G) solves the MIS problem. For sufficiently large k (k slightly above $2 \log n$ suffices), there is high probability (over the random choice $G \in G_{n,k,\frac{1}{2}}$) that S is the unique maximum independent set in G, and in this case the goal of solving MIS coincides with a goal of finding S.

When $k \geq c\sqrt{n \log n}$ for a sufficiently large constant c, the vertices of S are (almost surely) simply those of lowest degree in G. When $k \geq c\sqrt{n}$, recovering S (with high probability) is more challenging, but there are several known algorithms that manage to do so. Perhaps the simplest of these is the following algorithm of Feige and Ron (2010). The highest degree vertices in the (residual) graph are removed from the graph in an iterative fashion, until only an independent set remains. Feige and Ron prove that with high probability this independent set I is a relatively

large subset of S. Moreover, S can be recovered by adding to I those vertices not connected to any vertex in I.

Alon et al. (1998) developed a *spectral* algorithm for recovering S. It is easier to present their algorithm in the planted clique model rather than planted MIS. It is well known that for the adjacency matrix A of a random $G_{n,\frac{1}{2}}$ graph, almost surely the largest eigenvalue satisfies $\lambda_1(A) \simeq \frac{n}{2}$, whereas all other eigenvalues are not larger than roughly $\sqrt{n}$. A standard argument based on *Rayleigh quotients* implies that planting a clique of size $k > c\sqrt{n}$ (for sufficiently large constant c) in a random graph should create an eigenvalue of value roughly $\frac{k}{2}$. Hence for the input graph G, we expect its adjacency matrix A_G to satisfy $\lambda_2(A_G) \simeq \frac{k}{2} > \sqrt{n}$. Alon et al. (1998) proved that with high probability, the set K of k largest entries in the eigenvector with eigenvalue λ_2 have an overlap of size at least $\frac{5k}{6}$ with set S. Iteratively removing from K pairs of vertices that do not form an edge results in a clique K' of size at least $\frac{2k}{3}$. It is not hard to prove that necessarily $K' \subset S$, and that all other vertices of S are precisely those vertices that are neighbors with all vertices of K'.

To evaluate the robustness of these algorithmic techniques, the distributional $G_{n,k,\frac{1}{2}}$ model for MIS can be extended into a semi-random model by introducing a monotone adversary. The adversary, who has unbounded computational power, may observe G, and add to it edges of his choice, provided that S remains an independent set. This gives the semi-random graph $\hat{G}$. Observe that if S is a MIS (the unique MIS, respectively) in G, then necessarily S is a MIS (the unique MIS, respectively) in $\hat{G}$ as well. The goal is to design a polynomial time algorithm that with high probability (over the choice of $G \in G_{n,k,\frac{1}{2}}$, for every $\hat{G}$ that may be generated from G) finds S.

The iterative algorithm that is based only on degrees of vertices can easily be fooled by the adversary (who in particular has the power to make all vertices of S have substantial higher degree than all the remaining vertices). Likewise, the spectral algorithm can also be fooled by the adversary, and it too will not find S in $\hat{G}$. However, with additional machinery, the spectral algorithm can be salvaged. An algorithm that does work in the semi-random model is based on semi-definite programming (SDP). At a high level, one may think of SDPs as a technique that combines spectral techniques with linear programming. This is because SDPs involve two types of constraints: spectral (requiring a certain matrix to have no negative eigenvalues), and linear (as in linear programming).

We present here the algorithm of Feige and Krauthgamer (2000) for the semi-random MIS model. It is based on the ϑ function of Lovasz (which will be defined shortly). Given a graph G, $\vartheta(G)$ can be computed (up to arbitrary precision) in polynomial time, and it provides an upper bound (that might be far from tight) on $\alpha(G)$ (the size of the maximum independent set in G). The key technical lemma in Feige and Krauthgamer (2000) is the following.

Lemma 1.2 *Let $k \geq c\sqrt{n}$ for sufficiently large c. For $G \in G_{n,k,\frac{1}{2}}$, with probability at least $1 - \frac{1}{n^2}$ (over choice of G) it holds that $\vartheta(G) = \alpha(G)$.*

Though Lemma 1.2 is stated for $G \in G_{n,k,\frac{1}{2}}$, it applies also for $\hat{G}$ generated by the semi-random model. This is because ϑ is a monotone function – adding edges to G can only cause ϑ to decrease. But ϑ cannot decrease below $\alpha(\hat{G})$, and hence equality is preserved.

Given Lemma 1.2, finding S in $\hat{G}$ is easy. The failure probability is small enough to ensure that with high probability, for every vertex $v \in S$ it holds that $\vartheta(\hat{G} \setminus v) = k - 1$, and for every vertex $v \notin S$ it holds that $\vartheta(\hat{G} \setminus v) = k$ (here $\hat{G} \setminus v$ refers to the graph obtained from $\hat{G}$ by removing vertex v and all its incident edges). This gives a polynomial time test that correctly classifies every vertex of $\hat{G}$ as either in S or not in S. As we shall see, in fact it holds that all vertices can be tested simultaneously just by a single computation of $\vartheta(\hat{G})$.

Let us now provide some details about the contents of Lemma 1.2. The ϑ function has many equivalent definitions. One of them is the following. An *orthonormal representation* of $G(V, E)$ associates with each vertex $i \in V$ a unit vector $x_i \in R^n$, such that x_i and x_j are orthogonal ($x_i \cdot x_j = 0$) whenever $(i, j) \in E$. Maximizing over all orthonormal representations $\{x_i\}$ of G and over all unit vectors d (d is referred to as the *handle*) we have

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

The optimal orthonormal representation and the associated handle that maximize the above formulation for ϑ can be found (up to arbitrary precision) in polynomial time by formulating the problem as an SDP (details omitted). To see that $\vartheta(G) \geq \alpha(G)$, observe that for any independent set S the following is a feasible solution for the SDP: choose $x_i = d$ for all $i \in S$, and choose all remaining vectors x_j for $j \notin S$ to be orthogonal to d and to each other. Observe also that ϑ is indeed monotone as explained above (adding edges to G adds constraints on the orthonormal representation, and hence the value of ϑ cannot increase).

Now we explain how Lemma 1.2 can be used in order to recover the planted independent set S. Applying a union bound over less than n subgraphs, the lemma implies that with probability at least $1 - \frac{1}{n}$ (over the choice of $G \in G_{n,k,\frac{1}{2}}$) $\vartheta(G') = \alpha(G') = \alpha(G) - 1$ for every subgraph G' that can be obtained from G by removing a single vertex of S. The above equalities imply that for every vertex $i \in S$, it holds that in the optimal SDP solution $d \cdot x_i \geq 1 - \frac{1}{2n}$. Otherwise, by dropping i from G without changing the SDP solution we get that $\vartheta(G \setminus \{i\}) > \vartheta(G) - 1 + \frac{1}{2n} > \alpha(G) - 1$, contradicting the equality above (with $G' = G \setminus \{i\}$). No vertex $i \notin S$ can have $d \cdot x_j \geq 1 - \frac{1}{2n}$, as together with the contribution of the vertices from S, the value of $\vartheta(G)$ would exceed $|S| = \alpha(G)$, contradicting Lemma 1.2. We thus conclude that with high probability (over the choice of $G \in G_{n,k,\frac{1}{2}}$), for $\hat{G}$ generated in the

semi-random model, the vertices of S are precisely those that have inner product larger than $1 - \frac{1}{2n}$ with the handle d.

We now explain how Lemma 1.2 is proved. Its proof is based on a dual (equivalent) formulation of the ϑ function. In this formulation, given a graph $G(V, E)$

$$\vartheta(G) = \min_M [\lambda_1(M)]$$

where M ranges over all n by n symmetric matrices in which $M_{ij} = 1$ whenever $(i, j) \notin E$, and $\lambda_1(M)$ denotes the largest eigenvalue of M. As a sanity check, observe that if G has an independent set S of size k, the minimum of the above formulation cannot possibly be smaller than k, because M contains a k by k block of 1 entries (a Rayleigh quotient argument then implies that $\lambda_1(M) \geq k$). Given $G \in G_{n,k,\frac{1}{2}}$ with an independent set S of size k, Feige and Krauthgamer (2000) construct the following matrix M. As required, M is symmetric, and $M_{i,j} = 1$ for all vertices i, j for which $(i, j) \notin E$ (including the diagonal of M). It remains to set the values of $M_{i,j}$ for pairs of vertices i, j for which $(i, j) \in E$ (which can happen only if at least one of i or j is not in S). This is done as follows. If both i and j are not in S, then $M_{ij} = -1$. If $i \notin S$ and $j \in S$ then $M_{i,j} = -\frac{k - d_{i,S}}{d_{i,S}}$, where $d_{i,S}$ is the number of neighbors that vertex i has in the set S. This value of M_{ij} roughly equals -1, and is chosen so that $\sum_{j \in S} M_{ij} = 0$ for every $i \notin S$. Finally, if $i \in S$ and $j \notin S$, then symmetry of M dictates that $M_{ij} = M_{ji}$. For this matrix M, the vector $v_S \in \{0, 1\}^n$, which has entries of value 1 at coordinates that correspond to vertices of S and 0 elsewhere, serves as an eigenvector of eigenvalue k. Feige and Krauthgamer (2000) prove that with high probability (over choice of G) it holds that this matrix M has no eigenvalue larger than k. This establishes that $\vartheta(G) = k$. The same M applies also to any graph $\hat{G}$ derived from G by a monotone adversary, because adding edges to G only removes constraints imposed on M.

Summarizing, the spectral algorithm of Alon et al. (1998) can find the planted independent set in the distributional model $G_{n,p,\frac{1}{2}}$. The way to extend it to the semi-random model is by use of semi-definite programming, based on computing the ϑ function. More generally, a useful rule of thumb to remember is that semidefinite programming can often serve as a robust version of spectral algorithms.

Another advantage of the SDP approach, implicit in the discussion above, is that it not only finds the planted independent set, but also certifies its optimality: the solution to the dual SDP serves as a proof that $\hat{G}$ does not contain any independent set of size larger than k.

1.3.3 Refutation heuristics

In Section 1.3.2 we presented algorithms that search for solutions in various random and semi-random models. Once a solution is found, the algorithm terminates. A complementary problem is that of determining that an input instance does not

have any good solutions. For example, when attempting to verify that a given hardware design or a given software code meets its specification, one often reduces the verification task to that of determining satisfiability of a Boolean formula. A satisfying assignment for the Boolean formula corresponds to a bug in the design, and the absence of satisfying assignments implies that the design meets the specifications. Hence one would like an algorithm that certifies that no solution (satisfying assignment, in this case) exists. Such algorithms are referred to as *refutation algorithms*.

For NP-hard problems such as SAT, there are no polynomial time refutation algorithms unless P=NP. Hence it is natural to consider random and semi-random models for refutation tasks. However, refutation tasks involve a difficulty not present in search tasks. NP-hard problems do not possess polynomial size witnesses for their non-satisfiability (unless NP = coNP). Consequently it is not clear what a refutation algorithm should be searching for, and what evidence a refutation algorithm can gather that would ensure that the input instance cannot possibly have a solution.

Recall the distributional model for 3SAT presented in Section 1.1.1. In that model, the input is a random 3CNF formula ϕ with n variables and m clauses, and the goal is to determine whether it is satisfiable. Standard use of Chernoff bounds and a union bound over all possible assignments shows that when $m > cn$ (for some sufficiently large constant c) then almost surely ϕ is not satisfiable. Hence, if we trust that the formula was indeed generated according the the distributional model, and are willing to tolerate a small probability of error, then a refutation algorithm can simply output *not satisfiable*, and will with high probability (over choice of ϕ) be correct. However, this approach is not satisfactory for multiple reasons, one of which being that it provides no insights as to how to design refutation algorithms in practice. Consequently, we shall be interested in algorithms that for a given distributional model D have the following properties.

1. For every input formula ϕ, the algorithm A correctly determines whether ϕ is satisfiable or not.
2. With high probability (over choice of $\phi \in D$), the algorithm A produces its output in polynomial time.

We can completely trust the output of such an algorithm A. However, on some instances, A might run for exponential time, and we might need to terminate A before obtaining an answer. If most inputs generated by D are not satisfiable, then it is appropriate to refer to A as a *refutation heuristic*.

Before addressing refutation heuristics for SAT, it is useful to consider refutation heuristics for a different NP-hard problem, that of MIS. Consider the $G_{n,\frac{1}{2}}$ distributional model for MIS, and fix $k = \frac{n}{5}$. We refer to graphs G for which $\alpha(G) \geq k$ as *satisfiable*. For this setting we offer the following refutation heuristic, based on the ϑ function discussed in Section 1.3.2.

Refutation heuristic for MIS. Compute $\vartheta(G)$. If $\vartheta(G) < k$ output *not satisfiable*. If $\vartheta(G) \geq k$ use exhaustive search to find the maximum independent set in G. If its size is at least k output *satisfiable*, and if its size is less than k output *not satisfiable*.

The output of the refutation heuristic is always correct because $\vartheta(G) \geq \alpha(G)$ for every graph G. For most input graphs G generated from $G_{n,\frac{1}{2}}$ the algorithm runs in polynomial time, because for such graphs $\vartheta(G) = O(\sqrt{n})$ with high probability (an indirect way of proving this is by combining Lemma 1.2 with monotonicity of the ϑ function), and ϑ can be computed up to arbitrary precision in polynomial time.

The above refutation heuristic extends without change to $G_{n,p}$ models with $p \geq \frac{c}{n}$ for a sufficiently large constant c, because also for such graphs $\vartheta(G) < \frac{n}{5}$ with high probability. See Coja-Oghlan (2005).

Given that we have a refutation heuristic for MIS we can hope to design one for 3SAT as well, by reducing 3SAT to MIS. However, the standard "textbook" reductions from 3SAT to MIS, when applied to a random 3SAT instance, do not give a random $G_{n,p}$ graph. Hence the refutation heuristic for MIS might not terminate in polynomial time for such graphs. This difficulty is addressed by Friedman et al. (2005), who design a different reduction for 3SAT to MIS. They also design a simpler reduction from 4SAT to MIS, and this is the reduction that we choose to explain here.

We consider a random 4CNF formula ϕ with $m = cn^2$ clauses, for large enough c. Partition ϕ it into three subformulas. ϕ^+ contains only those clauses in which all literals are positive, ϕ^- contains only those clauses in which all literals are negative, and ϕ' contains the remaining clauses. We completely ignore ϕ', and construct two graphs, G^+ based on ϕ^+, and G^- based on ϕ^-. We describe the construction of G^+, and the construction of G^- is similar.

The vertex set V of G^+ contains $\binom{n}{2}$ vertices, where each vertex is labeled by a distinct pair of distinct variables. For every clause in ϕ^+ (that we assume contains 4 distinct variables), put an edge in G^+ between the vertex labeled by the first two variables in the clause and the vertex labeled by the last two variables in the clause.

Lemma 1.3 *If ϕ is satisfiable, then at least one of the two graphs G^+ and G^- has an independent set of size at least $\binom{n/2}{2} \simeq |V|/4$.*

Proof Consider an arbitrary satisfying assignment for ϕ, let S^+ be the set of variables assigned to true, and let S^- be the set of variables assigned to false. Consider the set of $\binom{|S^-|}{2}$ vertices in G^+ labeled by pairs of vertices from S^-. They must form an independent set because ϕ cannot have a clause containing only variables from S^- in which all literals are positive. Likewise, G^- has an independent set of size at least $\binom{|S^+|}{2}$. As $\max[|S^+|, |S^-|] \geq n/2$, the proof follows. □

Observe that if ϕ is random then both G^+ and G^- are random graphs, each with

roughly $m/16 \simeq c|V|/8$ edges, and hence average degree roughly $c/4$. (Clarification: the exact number of edges in each of the graphs is not distributed exactly as in the $G_{n,p}$ model. However, given the number of edges, the locations of the edges are random and independent, exactly as in the $G_{n,p}$ model. This suffices for the bounds of Coja-Oghlan (2005) on the ϑ function to apply.) For large enough c, the refutation Heuristic for MIS will with high probability take only polynomial time to certify that neither G^+ nor G^- have independent sets larger than $|V|/5$, and thus establish that ϕ cannot have a satisfying assignment.

The refutation heuristic for 4SAT can be extended to kSAT, refuting random kCNF formulas for all k, provided that $m > cn^{k/2}$. Doing so for even values of k is fairly straightforward. The extension to odd k (including $k = 3$, 3SAT) is significantly more difficult. For some of the latest results in this respect, see Allen et al. (2015) and references therein.

It is an open question whether there are refutation heuristics that can refute random 3CNF formulas with significantly fewer than $n^{3/2}$ clauses. The answer to this question may have implications to the approximability of various NP-hard problems such as min-bisection and dense k-subgraph (see Feige (2002) for details), as well as to problems in statistics and machine learning (see for example Daniely et al. (2013)).

1.3.4 Monotone adversary for locally optimal solutions

Recall the semi-random model for the MIS problem presented in Section 1.1.1. That model, referred here as the FK model (as it was introduced by Feige and Kilian (2001)) is more challenging than the model presented in Section 1.3.2, as the monotone adversary has complete control on the subgraph induced on $V \setminus S$. That subgraph might contain independent sets larger than S, and hence S need not be the maximum independent set in G. Consequently, there is no hope of developing algorithms that solve MIS in the FK model, as the solution might lie within $V \setminus S$, and the graph induced on $V \setminus S$ might be a "worst case" instance for MIS. Likewise, recovering S unambiguously is also not a feasible task in this model, because the adversary may plant in $V \setminus S$ other independent sets of size k that are statistically indistinguishable from S itself. Consequently, for simplicity, we set the goal in this model to be that of outputting one independent set of size at least k. However, we remark that the algorithms for this model meet this goal by outputting a list of independent sets, one of which is S. Hence the algorithms might not be able to tell which of the independent sets that they output is S itself, but they do find S.

The FK model attempts to address the following question: what properties of an independent set make finding the independent set easy? Clearly, being the largest independent set in a graph is not such a property, as MIS is NP-hard. Instead, the FK model offers a different answer which can be phrased as follows: if the independent set S is a strong local maximum, then S can be found. The term

strong local maximum informally means that for every independent set S' in G, either $|S' \cap S|$ is much smaller than $|S|$, or the size $|S'|$ is much closer to $|S' \cap S|$ than to $|S|$. The strong local optimality of S is implied (with high probability) by the random part of the FK model, and adding edges to G (by the monotone adversary) preserves the property of being a strong local minimum.

Another motivation for the FK model comes from the graph coloring problem. Every color class is an independent set, but need not be the largest independent set in the graph. Algorithms for finding independent sets in the FK model easily translate to graph coloring algorithms in various random and semi-random models for the graph coloring problem.

Algorithms for the FK model are based on semi-definite programming. However, Lemma 1.2 need not hold in this model. The subgraph induced on $V \setminus S$ can cause the ϑ function to greatly exceed k – this is true even if this subgraph does not contain any independent set larger than k. Consequently, in the FK model, the algorithm presented in Section 1.3.2 need not find neither S, nor any other independent set in G of size at least k.

Feige and Kilian (2001) make more sophisticated use of semidefinite programming, and in a certain regime for the parameters of the FK model, they obtain the following result.

Theorem 1.4 (Feige and Kilian (2001)) *Let $k = \alpha n$, and let $\epsilon > 0$ be an arbitrary positive constant. Then in the FK model (in which $|S| = k$, edges between S and $V \setminus S$ are introduced independently with probability p, and the adversary may add arbitrary edges $(u, v) \notin S \times S$) the following results hold:*

- *If $p \geq (1+\epsilon)\frac{\ln n}{\alpha n}$ then there is a random polynomial time algorithm that with high probability outputs an independent set of size k.*
- *If $p \leq (1-\epsilon)\frac{\ln n}{\alpha n}$ then the adversary has a strategy such that unless $NP \subset BPP$, every random polynomial time algorithm fails with high probability to output an independent set of size k.*

The algorithm in the proof of Theorem 1.4 has five phases which are sketched below (with most of the details omitted).

1. Make repeated use of the ϑ function to extract from the graph $t \leq O(\log n)$ sets of vertices $S_1, \ldots, S_t$, with the property that most vertices of S are among the extracted vertices.
2. Make repeated use of the random hyperplane rounding technique of Goemans and Williamson (1995) so as to find within each set S_i a relatively large independent set I_i.
3. It can be shown that with high probability, there will be *good* indices $i \in [t]$ for which $|I_i \cap S| \geq \frac{3}{4}|I_i|$. "Guess" (by trying all possibilities – there are only polynomially many of them) which are the good indices. Take the union of the corresponding I_i, and remove a maximal matching from the corresponding

induced subgraph. The resulting set I of vertices that remains forms an independent set (due to the maximality of the matching). Moreover, as every matching edge must contain at least one vertex not from S, it follows that (for the correct guess) most of the vertices of I are from S.

4. Setting up a certain matching problem between I and $V \setminus I$, identify a set of M vertices to remove from I, resulting in $I' = I \setminus M$. It can then be shown that $I' \subset S$.

5. Consider the subgraph induced on the non-neighbors of I' (this subgraph includes I' itself), find in it a maximal matching, and remove the vertices of the matching. This gives an independent set, and if it is larger than I', it replaces I'. It can be shown that this new I' maintains the invariant that it is a subset of S. Repeat this process until there is no further improvement in the size of I'. If at this point $|I'| \geq k$, then output I'.

In phase 3 the algorithm tries out polynomially many guesses, and several of them may result in outputting independent sets of size at least k. Feige and Kilian (2001) prove that when $p \geq (1+\epsilon)\frac{\ln n}{\alpha n}$, there is high probability that the planted independent set S is among those output by the algorithm. However, when $p \leq (1-\epsilon)\frac{\ln n}{\alpha n}$, the monotone adversary has a strategy that may cause the algorithm to fail. The algorithm does manage to complete the first three phases and to find a fairly large independent set, but of size somewhat smaller than k. The difficulty is in the fourth and fifth phases of the algorithm. This difficulty arises because there is likely to be a small (but not negligible) set of vertices $T \subset (V \setminus S)$ that has no random edge to S. The adversary may then choose a pattern of edges between T and S that on the one hand makes the largest independent set in $S \cup T$ be S itself, and on the other hand makes it difficult for the algorithm to determine which vertices of I (the result of the third phase) belong to T. These vertices prevent extending I to a larger independent set. Moreover, these considerations can be used to derive the NP-hardness result stated in the second part of Theorem 1.4, along lines similar to those used in the proof of Theorem 1.1.

We end this section with an open question.

Question: *What is the smallest value of k (as a function of n) such that an independent set of size k can be efficiently found in the FK model when $p = \frac{1}{2}$?*

McKenzie et al. (2018) show that an algorithm based on semidefinite programming works when $k \geq \Omega(n^{2/3})$. In analogy to the results stated in Section 1.3.2, one may hope to design an algorithm that works for $k \geq \Omega(\sqrt{n})$, though such an algorithm is not known at the moment, and neither is there a hardness result that suggests that no such algorithm exists.

1.3.5 Separable semi-random models

In Sections 1.3.2 and 1.3.4 we discussed semi-random graph models in which some of the edges in the input graph are generated at random, and others are generated by an adversary. Hence when generating an input instance, both the random decisions and the adversarial decisions refer to the same aspect of the input instance, to the edges. In this and subsequent sections we discuss classes of semi-random models that we refer to as *separable*. In these models, certain aspects of the input instance are random, and certain other aspects are adversarial. Such models help clarify which aspects of a problem contribute to its computational difficulty.

Recall the 3SAT semi-random model of Section 1.1.1, with n variables, m clauses, and probability p of flipping the polarity of a variable. When setting $p = \frac{1}{2}$, it provides a conceptually simple separable model for 3SAT. One may think of a 3CNF formula as having two distinct aspects: one is the choice of variables in each clause, and the other is the polarity of each variable. In the distributional model, both the choice of variables and the choice of their polarities are random. In the separable semi-random model, the choice of variables is left to the complete discretion of the adversary, whereas given the set of variables in each clause, the polarities of variables are set completely at random (each variable appearance is set independently to be positive with probability $\frac{1}{2}$ and negative with probability $\frac{1}{2}$). As in the distributional model for 3SAT, when $m > cn$ (for some sufficiently large constant c) then almost surely the resulting input formula is not satisfiable. As discussed in Section 1.3.3, when $m > cn^{3/2}$, there are refutation heuristics for the distributional model. As stated, these heuristics do not apply to the separable semi-random model. To appreciate some of the difficulties, observe that for the heuristic described in Section 1.3.3 for refuting 4SAT, the graphs G^+ and G^- referred to in Lemma 1.3 will not be random in the semi-random model. Nevertheless, if one allows a modest increase in the number of clauses to $m \geq cn^{3/2}\sqrt{\log \log n}$, then there are ways of adapting the known refutation heuristics for 3SAT to the semi-random model (see Feige (2007)). This suggests that the key aspect that is required for efficient refutation of random 3CNF formulas (with sufficiently many clauses) is randomness in the polarity of the variables. Randomness in the choice of variables does not seem to play a significant role. To test this conclusion, it makes sense to study also a complementary separable semi-random model for 3SAT, in which the choice of variables in each clause is random, whereas the choice of their polarities is adversarial. We do not know if the known refutation heuristics can be adapted to this other separable semi-random model.

1.3.6 Separable models for unique games

An instructive use of separable semi-random models is provided by Kolla et al. (2011). They consider instances of *unique games*. A unique game instance is speci-

fied by a graph $G(V, E)$ with n vertices, a set $[k]$ of labels, and for every $(u, v) \in E$ – a permutation π_{uv} on $[k]$. Given an assignment of a label $L(v) \in [k]$ to each vertex $v \in V$, the value of the game is the fraction of edges (u, v) for which $L(v) = \pi_{uv}(L(u))$. One seeks an assignment of labels that maximizes the value of the game. This problem is NP-hard, and the *unique games conjecture* (UGC) of Khot (2002) states that for every $\epsilon > 0$, there is some k such that it is NP-hard to distinguish between unique games of value at least $1 - \epsilon$ and unique games of value at most ϵ. Due to its many consequences for hardness of approximation, much effort has been spent both in attempts to prove and in attempts to refute the UGC. Such efforts could presumably be guided towards promising avenues if we knew how to design instances of unique games with value $1 - \epsilon$ for which no known algorithm can find a solution of value greater than ϵ. Given n, k and ϵ, the design of such unique games instance involves four aspects:

1. A choice of input graph $G(V, E)$.
2. A function $L : V \to [k]$ assigning labels to the vertices, and a choice of permutations π_{uv} that cause the value of the assignment to be 1.
3. A choice of $\epsilon|E|$ edges E' to corrupt.
4. A choice of alternative permutations π'_{uv} for $(u, v) \in E'$ (where possibly $L(v) \neq \pi'_{uv}(L(u))$).

If an adversary controls all aspects of the input instance, then we get a worst case unique games instance. There are four separable semi-random models that weaken the adversary in a minimal way. Namely, for each model three of the above aspects are controlled by the adversary, where the remaining one is random. One may ask which of these semi-random models generates a distribution over inputs on which UGC might be true. Somewhat surprisingly, Kolla et al. (2011) prove that none of them do (if the input graph has sufficiently many edges).

Theorem 1.5 (Kolla et al. (2011)) *For arbitrary $\delta > 0$, let k be sufficiently large, let $\epsilon > 0$ (the fraction of corrupted edges) be sufficiently small, and suppose that the number of edges in G is required to be at least $f(k, \delta)n$ (for some explicitly given function f). Then there is a randomized polynomial time algorithm that given an instance generated in any one of the above four separable semi-random models, finds with high probability a solution of value at least $1 - \delta$.*

The probability in the above theorem is taken both over the random choices made in the generation of the semi-random input instance, and over the randomness of the algorithm.

For lack of space we do not sketch the proof of Theorem 1.5. However, we do wish to point out that when only the third aspect (the choice of E') is random, the adversary is sufficiently strong to foil all previously known approaches for approximating unique games. To handle this case, Kolla et al. (2011) introduce the so called *crude* SDP, and develop techniques that exploit its solutions in order to find

approximate solutions for unique games. One of the goals of semi-random models is to bring about the development of new algorithmic techniques, and the separable model for unique games has served this purpose well.

1.3.7 The hosted coloring framework

We discuss here two separable semi-random models for 3-coloring. Recall the 3-coloring distributional model of Section 1.1.1. The key parameter there is p – the probability with which edges are introduced between vertices of different color classes. When p is constant, the 3-coloring can be recovered using the fact that for graphs of degree $\Omega(n)$, 3-coloring can be solved in polynomial time even on worst case instances. (Here is a sketch of how this can be done. A greedy algorithm finds a dominating set S of size $O(\log n)$ is such a graph. "Guess" (that is, try all possibilities) the true color of every vertex in S. For each vertex not in S, at most two possible colors remain legal. Hence the problem of extending the correct 3-coloring of S to the rest of the graph can be cast as a 2SAT problem, and 2SAT is solvable in polynomial time.) As p decreases, finding the planted coloring becomes more difficult. In fact, if there is an algorithm that finds the planted 3-coloring (with high probability) for $p = p_0$, then the same algorithm can be applied for every $p_1 > p_0$, by first sub-sampling the edges of the graph, keeping each edge with probability $\frac{p_0}{p_1}$.

Blum and Spencer (1995) design a combinatorial algorithm that finds the planted 3-coloring (w.h.p.) when $p \geq n^{\epsilon-1}$ for $\epsilon > 0$. Their algorithm is based on the following principle. For every two vertices u and v one computes the size of the intersection of the distance r neighborhood of u and the distance r neighborhood of v, where $r = \Theta(\frac{1}{\epsilon})$ and r is odd. For some threshold t that depends on p and r, it holds with high probability that vertices u and v are in the same color class if and only if the size of the intersection is above t. For example, if $p = n^{-0.4}$ one can take $r = 1$ and $t = \frac{n}{2}p^2$, because vertices of the same color classes are expected to have $p^2\frac{2n}{3}$ common neighbors, whereas vertices of different color classes are expected to have only $p^2\frac{n}{3}$ common neighbors.

Alon and Kahale (1997) greatly improved over the results of Blum and Spencer (1995). They designed a spectral algorithm (based on the eigenvectors associated with the two most negative eigenvalues of the adjacency matrix of G) that finds the planted 3-coloring (w.h.p.) whenever $p \geq \frac{c \log n}{n}$ (for sufficiently large constant c). Moreover, enhancing the spectral algorithm with additional combinatorial steps, they also manage to 3-color the input graph (w.h.p.) whenever $p \geq \frac{c}{n}$ (for sufficiently large constant c). At such low densities, the planted 3-color is no longer the unique 3-coloring of the input graph (for example, the graph is likely to have isolated vertices that may be placed in any color class), and hence the algorithm does not necessarily recover the planted 3-coloring (which is statistically indistinguishable from many other 3-colorings of the graph).

David and Feige (2016) introduced the hosted 3-coloring framework for the 3-coloring problem. In their models, there is a class $\mathcal{H}$ of host graphs. To generate an input graph G, one first selects a graph $H \in \mathcal{H}$, and then plants in it a balanced 3-coloring (by partitioning the vertex set into three roughly equal parts, and removing all edges within each part). The resulting graph G is given as input to a polynomial time algorithm that needs to 3-color G. The distributional 3-coloring model is a special case of the hosted 3-coloring framework, in which $\mathcal{H}$ is the class of $G_{n,p}$ graphs, a member $H \in \mathcal{H}$ is chosen at random, and then a balanced 3-coloring is planted at random. Other models within the hosted 3-coloring framework may assign parts (or even all, if the class $\mathcal{H}$ is sufficiently restricted) of the graph generation process to the discretion on an adversary.

In one separable semi-random model within the framework, $\mathcal{H}$ is the class of d-regular spectral expander graphs. Namely, for every graph $H \in \mathcal{H}$, except for the largest eigenvalue of its adjacency matrix, every other eigenvalue has absolute value much smaller than d. A graph $H \in \mathcal{H}$ is chosen by an adversary, and the planted 3-coloring is chosen at random. David and Feige (2016) show that the 3-coloring algorithm of Alon and Kahale (1997) can be modified to apply to this case. This shows that random planted 3-colorings can be found even if the host graph is chosen by an adversary, provided that the host graph is an expander.

In another separable semi-random model within the framework, a host graph H is chosen at random from $\mathcal{H} = G_{n,p}$, but the planted balanced 3-coloring is chosen by an adversary, after seeing H. Somewhat surprisingly, David and Feige (2016) show that for a certain range of values for p, corresponding to the random graph having average degree somewhat smaller than $\sqrt{n}$, 3-coloring the resulting graph is NP-hard. We explain here the main idea of the NP-hardness result (substantial additional work is required in order to turn the following informal argument into a rigorous proof). Let $\mathcal{Q}$ be a carefully chosen class of graphs on which 3-coloring is NP-hard. First one shows that given any 3-colorable graph $Q \in \mathcal{Q}$ on n^{ϵ} vertices, if p is sufficiently large ($p \geq n^{-2/3}$ is required here), then H is likely to contain many copies of Q. The computationally unbounded adversary can find a copy of Q in H, and plant in H a 3-coloring that leaves this copy of Q unmodified (by having the planted coloring agree on Q with some existing 3-coloring of Q). Moreover, if p is not too large ($p \leq n^{-1/2}$ is required here), the planting can be made in such a way that Q becomes separated from the rest of H (due to the fact that edges that are monochromatic under the planted 3-coloring are removed from H). Any algorithm that 3-colors G can infer from it in polynomial time a 3-coloring for Q, because it is easy to find Q within G. As 3-coloring Q was assumed to be difficult, so is 3-coloring G.

In general, results in the hosted 3-coloring framework help clarify which aspects of randomness in the planted coloring model are the key to successful 3-coloring algorithms.

1.4 Open problems

As is evident from Section 1.3, there are many different semi-random models. We presented some of them, and some others are discussed more extensively in other chapters of this book. Some of the more creative models, such as the PIE model of Makarychev et al. (2014), were not discussed due to lack of space.

We also attempted to provide an overview of some of the algorithmic techniques that are used in handling semi-random models. Further details can be found in the references. Moreover, we provided brief explanations as to how hardness results are proved in semi-random model. We believe that having hardness results (and not just algorithms) is a key component in building a complexity theory for semi-random models.

There are many open questions associated with distributional and semi-random models. Some were mentioned in previous sections. Here we list a few more. The first two refer to improving the parameters under which algorithms are known to work. Similar questions can be asked for other problems. The other two questions relate to less standard research directions.

- Recall that Alon and Kahale (1997) design a 3-coloring for the distributional model for 3-coloring, provided that $p \geq cn$ for a sufficiently large constant c. Can the algorithm be extended to hold for all p? In this context, it is worth mentioning that for a different NP-hard problem, that of Hamiltonicity, there is a polynomial time algorithm that works in the $G_{n,p}$ model for all values of p. That is, regardless of the value of p, with high probability over the choice of the input random graph G, if G is not Hamiltonian then the algorithm provides a witness for this fact (the witness is simply a vertex of degree less that 2), whereas if the graph is Hamiltonian the algorithm produces a Hamiltonian cycle (using the extension-rotation technique). See Bollobás et al. (1987) for details.
- In Theorem 1.5 (concerning unique games), can one remove the requirement that the number of edges is sufficiently high?
- Consider the following semi-random model for the MIS problem. First an adversary selects an arbitrary n vertex graph $H(V, E)$. Thereafter, a random subset $S \subset V$ of size k is made into an independent set by removing all edges induced by S, thus making S a random planted independent set. The resulting graph G (but not H) and the parameter k (size of S) are given as input, and the task is to output an independent set of size at least k. Is there a polynomial time algorithm that with high probability (over the random choice of S) outputs an independent set of size at least k? Is there a proof that this problem is NP-hard?
- Recall that there are refutation heuristics for 3SAT when the random 3CNF formula has more than $n^{3/2}$ clauses. The following questions may serve as a first step towards refuting sparser formulas.

  Given an initial random 3CNF formula ϕ with n^{δ} clauses, one can set the po-

larities of all variables to be positive, and then the resulting formula is satisfiable. The question is how one should set the polarities of the variables so that the resulting formula ϕ' can be certified in polynomial time to be not satisfiable. When $\delta > \frac{3}{2}$, this can be done by setting the polarities at random, as then the refutation heuristic of Section 1.3.3 can be used. For $\frac{7}{5} < \delta < \frac{3}{2}$, with high probability over the choice of ϕ, there are settings of the polarities (not necessarily by a polynomial time procedure) under which refutation can be achieved in polynomial time. (Hint: break ϕ' into a prefix with random polarities, and a suffix whose polarities form a 0/1 string that encodes the refutation witness of Feige et al. (2006) for the prefix.) For $\delta < \frac{7}{5}$, it is an open question whether there is any setting of the polarities (whether done in polynomial time or exponential time) under which polynomial time refutation becomes possible.

Acknowledgments

The work of the author is supported in part by the Israel Science Foundation (grant No. 1388/16). I thank Ankur Moitra, Tim Roughgarden and Danny Vilenchik for their useful comments.

References

Allen, Sarah R., O'Donnell, Ryan, and Witmer, David. 2015. How to Refute a Random CSP. Pages 689–708 of: *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*.

Alon, Noga, and Kahale, Nabil. 1997. A Spectral Technique for Coloring Random 3-Colorable Graphs. *SIAM J. Comput.*, **26**(6), 1733–1748.

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

Bhaskara, Aditya, Charikar, Moses, Chlamtac, Eden, Feige, Uriel, and Vijayaraghavan, Aravindan. 2010. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. Pages 201–210 of: *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC*.

Blum, Avrim. 1990. Some Tools for Approximate 3-Coloring (Extended Abstract). Pages 554–562 of: *31st Annual Symposium on Foundations of Computer Science, Volume II*.

Blum, Avrim, and Spencer, Joel. 1995. Coloring Random and Semi-Random k-Colorable Graphs. *J. Algorithms*, **19**(2), 204–234.

Bollobás, Béla, Fenner, Trevor I., and Frieze, Alan M. 1987. An algorithm for finding Hamilton cycles in a random graph. *Combinatorica*, **7**(4), 327–341.

Coja-Oghlan, Amin. 2005. The Lovász Number of Random Graphs. *Combinatorics, Probability & Computing*, **14**(4), 439–465.

Daniely, Amit, Linial, Nati, and Shalev-Shwartz, Shai. 2013. More data speeds up training time in learning halfspaces over sparse vectors. Pages 145–153 of: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*

David, Roee, and Feige, Uriel. 2016. On the effect of randomness on planted 3-coloring models. Pages 77–90 of: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016.*

Feige, Uriel. 2002. Relations between average case complexity and approximation complexity. Pages 534–543 of: *Proceedings on 34th Annual ACM Symposium on Theory of Computing.*

Feige, Uriel. 2007. Refuting Smoothed 3CNF Formulas. Pages 407–417 of: *48th Annual IEEE Symposium on Foundations of Computer Science FOCS.*

Feige, Uriel, and Kilian, Joe. 2001. Heuristics for Semirandom Graph Problems. *J. Comput. Syst. Sci.*, **63**(4), 639–671.

Feige, Uriel, and Krauthgamer, Robert. 2000. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms*, **16**(2), 195–208.

Feige, Uriel, and Ron, Dorit. 2010. Finding hidden cliques in linear time. Pages 189–204 of: *21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10).*

Feige, Uriel, Kim, Jeong Han, and Ofek, Eran. 2006. Witnesses for non-satisfiability of dense random 3CNF formulas. Pages 497–508 of: *47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006.*

Friedman, Joel, Goerdt, Andreas, and Krivelevich, Michael. 2005. Recognizing More Unsatisfiable Random k-SAT Instances Efficiently. *SIAM J. Comput.*, **35**(2), 408–430.

Goemans, Michel X., and Williamson, David P. 1995. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, **42**(6), 1115–1145.

Khot, Subhash. 2002. On the power of unique 2-prover 1-round games. Pages 767–775 of: *Proceedings on 34th Annual ACM Symposium on Theory of Computing.*

Kolla, Alexandra, Makarychev, Konstantin, and Makarychev, Yury. 2011. How to Play Unique Games Against a Semi-random Adversary: Study of Semi-random Models of Unique Games. Pages 443–452 of: *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011.*

Makarychev, Konstantin, Makarychev, Yury, and Vijayaraghavan, Aravindan. 2014. Constant factor approximation for balanced cut in the PIE model. Pages 41–49 of: *Symposium on Theory of Computing, STOC 2014.*

McKenzie, Theo, Mehta, Hermish, and Trevisan, Luca. 2018. *A New Algorithm for the Robust Semi-random Independent Set Problem.*