CS264: Beyond Worst-Case Analysis
Lecture #10: Planted and Semi-Random Graph Models*

Tim Roughgarden†

October 22, 2014

1 Preamble

Lectures #6–8 proposed several different "stability conditions" on problem instances, of various NP-hard clustering and graph partitioning problems, under which the exact recovery of the optimal solution is possible in polynomial time. These stability conditions were reasonably natural, in that there was a plausible narrative about why "real-world" instances might tend to satisfy them (at least approximately).[1]

Last lecture (Lecture #9), we gave a sufficient condition for the computationally efficient recovery of sparse solutions to underdetermined linear systems. The condition was technical — that the kernel of the constraint matrix A is an almost Euclidean subspace — but was justified by the (omitted) proof that random matrices satisfy the condition with high probability (for many different distributions on matrices).

Today we continue our ongoing study of the polynomial-time exact recovery of "planted" or "ground truth" solutions. We'll study randomized models (i.e., input distributions) that share some spirit with the stability conditions of Lectures #6–8, in that the optimal solution tends to "stick out." The goals are to design and analyze polynomial-time algorithms that recover the optimal solution with high probability (over the input distribution), and to understand how far the optimal solution to an NP-hard problem has to stick out before exact recovery is possible in polynomial time.

†Department of Computer Science, Stanford University, 474 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

[1]There has not been much empirical work on the extent to which these conditions hold in practice, however.

2 Probabilistic Planted Models

2.1 Erdös-Renyi Random Graphs

We review *Erdös-Renyi random graphs* as a starting point. Recall that a random graph from $\mathcal{G}(n,p)$ is a simple undirected graph on n nodes, where each of the $\binom{n}{2}$ possible edges is present independently with probability p. The special case of $p = \frac{1}{2}$ is the uniform distribution over all n-vertex graphs. There are at least two reasons why the $\mathcal{G}(n,p)$ model is typically not very useful for informing the design of graph algorithms. First, as with most pure average-case analysis frameworks, the data model is too specific; second, as we show below in two different examples, it fails to meaningfully differentiate between different algorithms.

Example 2.1 (Minimum Bisection) The *graph bisection* problem is the same as the minimum cut problem, except that the two sides of the graph are additionally constrained to have equal size. That is, the input is an undirected graph $G = (V,E)$ with an even number of vertices and the goal is to identify the cut $(S, \bar{S})$ with $|S| = |\bar{S}|$ that has the fewest number of crossing edges.

Assume for simplicity that the edge probability p is a constant. Then for every bisection $(S, \bar{S})$ of a set of n nodes, the expected number of crossing edges in a random graph $G \in \mathcal{G}(n,p)$ is $pn^2/4$. A straightforward application of the Chernoff bound (see Exercise #37) shows that, with high probability, the number of edges crossing *every* bisection is this same quantity, up to a $1 \pm o(1)$ factor. Thus even an algorithm that computes a *maximum* bisection is an almost optimal algorithm for computing a minimum bisection!

Example 2.2 (Maximum Clique) In the *maximum clique* problem, the goal (given an undirected graph) is to identify the largest subset of vertices that are mutually adjacent. In a random graph in the $\mathcal{G}(n, \frac{1}{2})$ model, the size of the maximum clique is very likely to be $\approx 2\log_2 n$.[2] To see heuristically why this is true, note that for an integer k, the expected number of cliques on k vertices in a random graph of $\mathcal{G}(n, \frac{1}{2})$ is exactly

$$\binom{n}{k} 2^{-\binom{k}{2}} \approx n^k 2^{-k^2/2},$$

which is 1 precisely when $k = 2\log_2 n$. That is, $2\log_2 n$ is roughly the largest k for which we expect to see at least one k-clique.

On the other hand, there is no known polynomial-time algorithm that computes, with high probability, a clique significantly larger than $\approx \log_2 n$ in a random graph from $\mathcal{G}(n, \frac{1}{2})$. And trivial heuristics — like starting with an arbitrary vertex and repeatedly adding an arbitrary vertex that is adjacent to everything already chosen — already obtain the $\log_2 n$ bound with high probability (see Problem #19). Thus the Erdös-Renyi model fails to distinguish between different efficient heuristics for the Maximum Clique problem.[3]

[2] A canonical application of the "second moment method" [2] shows that this random variable is unbelievably concentrated: there as an integer $k \approx 2\log_2 n$ such that almost every graph has maximum clique size either k or $k+1$.

[3] The computational complexity of finding a maximum clique in a random graph is highly unclear. Note

2.2 Planted Random Graph Models

Recall our primary motivation for our stability and sparsity conditions in Lectures #6–9: we are often only interested in inputs that have an obviously meaningful solution, which we identify with being "clearly optimal" in some sense. *Planted graph models* are a nice family of probabilistic models in which such a "clearly optimal" solution exists with high probability. Such models can be viewed as a randomized analog of the stability conditions studied in Lectures #6–8. We follow the elegant formalism of McSherry [15]; see also the "stochastic block model" [8].

The Model. There are n vertices, partitioned into t clusters $S_1, \ldots, S_t$. For each pair i, j of clusters (possibly with $i = j$) there is a parameter $p_{ij} \in (0, 1)$, with $p_{ij} = p_{ji}$ for every i, j. A random graph is sampled by independently including each edge with endpoints in S_i, S_j with probability p_{ij}.

The Goal. Given a random graph from the above model, we strive to reverse engineer the S_i's with high probability. Note that there needs to be nontrivial separation between certain p_{ij}'s; otherwise this goal is impossible due to inherent ambiguity in the sample.

Our first example is a planted version of the minimum bisection problem (recall Example 2.1), and it was originally proposed by Bui et al. [5].[4] There were several follow-up papers in the 1980s.

Example 2.3 (Planted Bisection) There are two clusters ($t = 2$) satisfying $|S_1| = |S_2| = n/2$. The parameters p_{11}, p_{22} are equal, say to p. The parameter p_{12}, denoted q, is less than p. Intuitively, the problem gets easier as $p - q$ grows larger. The question is then: for how small a gap $p - q$ is polynomial-time recovery of the planted bisection possible?

We return to the graph bisection problem when we discuss semirandom models below.

Our second example is a planted version of the maximum clique problem (recall Example 2.2) and was first suggested by Karp [10].

Example 2.4 (Planted Clique) There are again two clusters, with $|S_1| = k$ and $|S_2| = n - k$. We set $p_{11} = 1$ so that S_1 is a k-clique. The parameters p_{12} and p_{22} are set to a common value p. The question is then: for a given p ($\frac{1}{2}$, say), how big does k need to be before the planted clique can be recovered in polynomial time?

The planted clique problem is clearly easy when k is really huge, like $k = n - O(1)$. Kucera [11] observed that it is easy (for $p = \frac{1}{2}$) even when $k = \Omega(\sqrt{n \log n})$. To see this, think about generating a random sample of the planted maximum clique problem in the following way: first take a sample from the usual Erdös-Renyi $\mathcal{G}(n, \frac{1}{2})$ model; then choose k vertices at

that the problem can be solved in quasi-polynomial (i.e., $n^{O(\log n)}$) time by brute-force search, with high probability. It is conjectured to be a hard problem.

[4] Strictly speaking, the model in [5] is slightly different from the one considered here: it considers a random graph with a given number of edges and a given minimum bisection size.

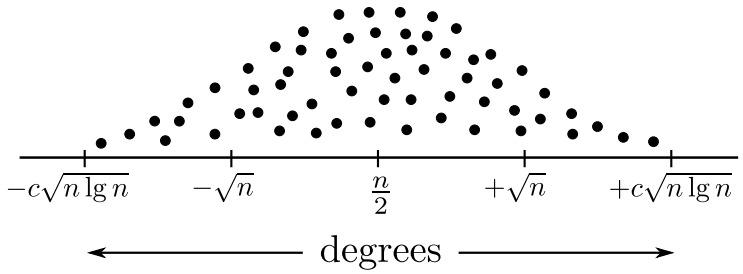

Figure 1: Degree distribution for $\mathcal{G}(n, 1/2)$, before planting the clique. A Chebyshev and union bound argument shows that the spread is $O(\sqrt{n \lg n})$ with high probability. If $k == \Omega(\sqrt{n \lg n})$ then the planted clique will consist of the k nodes with the highest degrees.

random and "fill them in" to make them a clique. After the first step, the expected degree of each node is $(n-1)/2$, and Chebyshev's inequality implies sharp concentration: with high probability, *all* node degrees are $\frac{n}{2} \pm c\sqrt{n \log n}$ for a suitable constant c (Figure 1). Filling in the k-clique boosts the degree of those k nodes by roughly $k/2$ each, without affecting the degrees of nodes outside the clique — in Figure 1, the cliques nodes all jump $k/2$ positions to the right. Thus, if $k > 4c\sqrt{n \log n}$, the clique nodes are the k nodes of the graph with the largest degrees (with high probability); the clique is then obviously recoverable in linear time.

Alon et al. [1] used more sophisticated techniques to achieve a slight improvement: polynomial-time recovery of a planted clique with $p = \frac{1}{2}$ and $k = \Omega(\sqrt{n})$. Closing the gap between the upper bound of $k \approx \sqrt{n}$ and the lower bound of $k \approx \log n$ for the polynomial-time solvability of the planted clique problem is a major open question.[5]

3 Semirandom Models

Semirandom graph modes are a very nice idea of Blum and Spencer [3]; the results we discuss are from Feige and Kilian [6]. The goal of these models is to have as robust a data model as possible, subject to preserving the existence of a "planted" or "clearly optimal" solution. Like upcoming lectures on pseudorandom data and smoothed analysis, this is a "hybrid model" that blends together the probabilistic and adversarial approaches.

3.1 Definition and Discussion

We explain the model in particular for the minimum bisection and maximum clique problems. Nature and an adversary conspire to produce a hard input as follows. First, nature chooses a random instance from the corresponding planted model (Examples 2.3 and 2.4). Second,

[5]Intractability of the planted clique problem with $k = o(\sqrt{n})$ is increasingly being used as a novel hardness assumption. There is even a (theoretical) cryptosystem based on it [9]. For example, the problem is used in [7] to identify a barrier to finding certain approximate Nash equilibria in two-player games.

the adversary can make the sparse regions sparser (by removing edges) and the dense regions denser (by adding edges) in an arbitrary way. So in the minimum bisection problem, the adversary can delete any edges across the planted bisection and add any edges that do not cross this bisection. In the maximum clique problem, the adversary can remove any edges that are not inside the planted clique.[6]

An easy but key observation is that the planted solution survives arbitrary actions of the adversary. In fact, since the adversary can intuitively only make the planted solution "more optimal," one might wonder if the semirandom model is really any harder than the planted one. There is no known formal equivalence or separation between any planted problem and its semirandom counterpart. But there are certainly algorithms that work in a planted model but not in the corresponding semirandom one. For example, the "top k degrees" algorithm above that recovers the planted maximum clique with $k = \Omega(\sqrt{n \log n})$ fails miserably in the semirandom model, even when k is linear in n: the adversary can sparsify the edges between the clique nodes and the other nodes, lowering the degrees of the clique nodes to right around $n/2$ (Exercise #38).

The benefit of the semirandom model over the planted one is that it encourages more "robust" solutions — algorithms that are not overfitted to a particular data model. In the planted model, an algorithm can take advantage of two properties of the input: (i) there is a clearly optimal solution (a plausible property of many "real instances"); and (ii) the input shares lots of non-trivial and useful properties with completely random instances, such as highly predictable node degrees (a questionable property of "real instances"). The first advantage was the motivating point behind the planted model, while the second was only a side effect of our modeling choices. The semirandom model offers an algorithm only the first advantage, and is thus more faithful to our original goals.

3.2 Case Study: Graph Bisection

Recall the planted bisection problem (Example 2.3). Assume that p, q are constants with

$$p - q \geq c\sqrt{\frac{\log n}{n}}, \tag{1}$$

where c is a sufficiently large constant. Calculations show that this separation is necessary for the intended planted bisection (S_1, S_2) to be the minimum bisection with high probability [5].[7]

Feige and Kilian [6] prove the following cool theorem.

Theorem 3.1 ([6]) *Under assumption* (1)*, there is a polynomial-time algorithm that computes the minimum bisection in the semirandom model, with high probability.*

[6] See [12, 13] for recent proposals of semi-random graph models with an even stronger adversary.

[7] The results of this section continue to hold for p, q as small as $\Theta((\log n)/n)$. For recent results where p, q are very small (meaning $O(1/n)$), and only approximate recovery is possible, see [14, 16, 17].

Their approach is to use semidefinite programming. Given a graph $G = (V, E)$, we define a relaxation (i.e., a lower bound) of the minimum bisection problem as follows. We adopt the linear objective function

$$\sum_{(i,j) \in E \,:\, i<j} \frac{1 - x_{ij}}{2}, \tag{2}$$

and maximize it subject to the linear constraints

$$\sum_{i,j \in V} x_{ij} = 0 \tag{3}$$

and

$$x_{ii} = 1 \text{ for all } i \in V, \tag{4}$$

and also the possibly bizarre-sounding constraint that the $V \times V$ matrix X of the x_{ij}'s is symmetric and positive semidefinite.[8] This mathematical program can be solved in polynomial time (up to an arbitrarily small additive error term), using either the ellipsoid method or modern interior-point techniques.[9]

But what good is it? Let $b(G)$ denote the number of edges crossing the minimum bisection of G and let $h(G)$ denote the optimal solution to the semidefinite program above. We claim that $h(G) \leq b(G)$. To see this, let (S_1, S_2) be an optimal bisection of G and let $s \in \{\pm 1\}^V$ denote the corresponding characteristic vector. Form the matrix $X = ss^T$, which is clearly symmetric and positive semidefinite (and rank one, even). The constraint (4) on the diagonal of X clearly holds. Since $|S_1| = |S_2|$, the constraint (3) also holds. Finally, observe that the objective function value of this X is precisely $b(G)$. The optimal value $h(G)$ can only be less than this.

Since computing a minimum bisection is NP-hard and computing $h(G)$ can be done in polynomial time, we expect that $h(G) < b(G)$ for many graphs G. The main technical lemma in Feige and Kilian [6], which is inspired by Boppana [4], is that *the relaxation is exact in the planted model*.

Lemma 3.2 *For a random graph in the planted (non-semirandom) bisection model, under assumption* (1)*, $h(G) = b(G)$ with high probability.*

The proof of Lemma 3.2 is quite technical and we won't discuss it. It involves "guessing and checking" a solution to the dual semidefinite program that has objective function value $b(G)$. See next lecture for an analogous (and less technical) "dual certificate" argument in the context of a linear programming relaxation of a decoding problem; many "exact recovery" results to-date follow this general paradigm.

[8] Recall there are many equivalent definitions of such matrices: the (full) set of real eigenvalues are all nonnegative; the quadratic form $y^T X y$ is nonnegative for all y; or X has a "square root" U with $X = UU^T$.

[9] A good first-cut rule of thumb is that minimization mathematical programs are polynomial-time solvable if and only if both the objective function and the feasible region are convex. Observe that the set of symmetric and positive semidefinite matrices is indeed convex, so we should not be surprised that this mathematical program is tractable.

Unlike previous algorithms for planted models, the semidefinite programming approach extends automatically to the semirandom model (and is therefore a "robust algorithm" in some sense). The proof of this uses the following claim: if $\hat{G}$ is obtained from G be adding a single edge, then

$$h(G) \leq h(\hat{G}) \leq h(G) + 1. \tag{5}$$

We leave the verification of (5) as an exercise; the key point is that constraint (4) and the positive semidefinite constraint force all x_{ij}'s to have magnitude at most 1 (Exercise #39).

Proof of Theorem 3.1: Begin with a random sample G_0 from the planted model. By Lemma 3.2, $h(G_0) = b(G_0)$ with high probability. The adversary adds or deletes edges one at a time, yielding a sequence $G_0, \ldots, G_t$. We claim that, by induction, $b(G_i) = h(G_i)$ for every i.

First, when the adversary adds an edge not in the planted bisection, $b(G_i) = b(G_{i-1})$. By (5), $h(G_i) \geq h(G_{i-1}) = b(G_{i-1}) = b(G_i)$. But since $h(G) \leq b(G)$ for every graph, we must have $b(G_i) = h(G_i)$.

When the adversary removes an edge of the planted bisection, $b(G_i) = b(G_{i-1}) - 1$. By (5), $h(G_i) \geq h(G_{i-1}) - 1 = b(G_{i-1}) - 1 = b(G_i)$. But since $h(G) \leq b(G)$ for every graph, we must have $b(G_i) = h(G_i)$. ■

The proof above shows how to compute the *value* $b(G)$ of the minimum bisection in polynomial time with high probability (by computing $h(G)$ instead). We leave as an exercise the task of using this subroutine to reconstruct the planted bisection itself in polynomial time (Exercise #40).

References

[1] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hiddin clique in a random graph. *Random Structures & Algorithms*, 13(3-4):457–466, 1998.

[2] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, 2008. Third edition.

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

[4] R. B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 280–285, 1987.

[5] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987. Preliminary version in *FOCS '84*.

[6] U. Feige and J. Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63(4):639–671, 2001. Preliminary version in *FOCS '98*.

[7] E. Hazan and R. Krauthgamer. How hard is it to approximate the best Nash equilibrium? In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 720–727, 2009.

[8] P. W. Holland, K. Lasket, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[9] A. Juels and M. Peinado. Hiding cliques for cryptographic security. *Designs, Codes, and Cryptography*, 20(3):269–280, 2000.

[10] R. M. Karp. The probabilistic analysis of some combinatorial search algorithms. In J. F. Traub, editor, *Algorithms andComplexity: New Directions and Recent Results*, pages 1–19. Academic Press, 1976.

[11] L. Kucera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57(2-3):193–212, 1995.

[12] K. Makarychev, Y. Makarychev, and A. Vijayaraghavan. Approximation algorithms for semi-random partitioning problems. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 367–384, 2012.

[13] K. Makarychev, Y. Makarychev, and A. Vijayaraghavan. Constant factor approximation for balanced cut in the PIE model. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 41–49, 2014.

[14] L. Massoulie. Community detection thresholds and the weak Ramanujan property. arXiv:1311.3085, 2013.

[15] F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 529–537, 2001.

[16] E. Mossel, J. Neeman, and A. Sly. Stochastic block models and reconstruction. arXiv:1202.1499, 2012.

[17] E. Mossel, J. Neeman, and A. Sly. A proof of the block model threshold conjecture. arXiv:1311.4115, 2013.