TTIC 31290: Machine Learning for Algorithm Design (Fall 2025)
Avrim Blum and Dravyansh Sharma

Lecture 3: 10/07/25 Lecturer: Avrim Blum

Outline for today

- "Beyond Worst-Case Analysis"
- The Semi-Random model
- Case study: 3-coloring
- Smoothed Analysis

1 Beyond Worst-Case Analysis

We saw in the last two classes some of the classic approaches for theoretically analyzing hard optimization problems. However, in many cases, worst-case analysis is too pessimistic, and average-case analysis is too optimistic/unrealistic. For instance, random graphs all pretty much look the same. The term "Beyond Worst-Case Analysis" has become used for the development and study of models that go beyond these classic ones. Today, we'll look at two models that interpolate between average-case and worst-case analysis: semi-random models and smoothed analysis.

2 The Semi-Random model

In the semi-random graph model, the input is created by an adversary, but each of the adversary's decisions about whether to include an edge or not is then flipped with some probability p. For $p = 0$ we have a fully adversarial model and for $p = 1/2$ we get a random graph model. There are several variations on this model, for instance we can imagine that we first flip all the coins to create an initial random graph and then an adversary can add additional edges if it wants to (this is called a "monotone model"). The exact formulation that makes sense will depend on the problem.

Here, as a case study, we'll look at a semi-random model for 3-Coloring, which is where it was first defined and studied [B90]. But it's also been used for Independent Set / Vertex Cover, clustering, and other problems.

3 Case study: 3-coloring

Let's consider an n-node 3-colorable graph created according to the following semi-random process:

1. Nodes are arbitrarily partitioned into three color classes (red, blue, green).

2. For each pair (u, v) of nodes from different color classes, an adversary decides whether to include the edge (u, v) or not; with probability p, this decision is reversed. [Equivalently: an adversary chooses $p_{uv} \in [p, 1-p]$ and the edge is included with probability p_{uv}]

2'. An alternative version of step 2: we first put in each allowable edge with probability p, and then an adversary can add extra allowable edges if it wants to ("allowable" means an edge between nodes of different colors). Technically, this is "more adversarial" than step 2, but sometimes it is easier to think about.

Question: would our previous algorithm for the random $\mathcal{G}(n, p, 3)$ model (where we assign vertices based on how many neighbors they have in common) still work here? Answer: no.

So, we need a more robust algorithm, that's not so sensitive to the precise distribution. It turns out the best algorithms for this problem (that work for the lowest values of p) use a technique called Semi-Definite Programming. But here we'll analyze a simpler algorithm that works for any constant p, and in fact for any $p > n^{-1/3+\epsilon}$ for any constant $\epsilon > 0$.

3.1 An algorithm for 3-coloring in the semi-random model

Before running the algorithm, first try to 2-color the graph. If that succeeds, we're done. Otherwise, run the algorithm below.

Algorithm Two-Step

For each pair of vertices u, v (think of u as a blue node and v as a green node) try to 3-color the graph as follows:

1. Let $S = N(u) \cap N(v)$ be their neighbors in common.
2. Let $T = N(S)$ be the neighborhood of S.
3. If T is 2-colorable and $V \setminus T$ is an independent set, then color T blue and green, color $V \setminus T$ red, and halt.

Theorem 1. *Algorithm Two-Step will find a 3-coloring with high probability for any $p > n^{-1/3+\epsilon}$ for any constant $\epsilon > 0$.*

Proof. Let "red" be the color with the most vertices according to the arbitrary partition into colors in Step 1 of the semi-random model, and we may assume without loss of generality that there is at least one blue node u and at least one green node v. Fix one such pair of nodes; notice that S will consist only of red nodes, and so T will consist only of blue and green nodes. We will argue that with high probability over the randomness in Step 2 of the semi-random model, T will contain *all* the blue and green nodes, and so the algorithm will succeed.

To do this analysis, we may assume that in Step 2 of the semi-random model, each allowable edge is placed in the graph with probability exactly p. [Can you see why? It's perhaps easier to see with the harder alternative version Step 2' above.]

Now, there are at least $n/3$ red nodes, and each such node belongs to S independently with probability p^2. So the expected size of S is at least $np^2/3$, and by Chernoff bounds, the probability that S has size less than $np^2/6$ is at most $e^{-\mathbf{E}[S]/8}$, which is exponentially small in n for our given value of p (this isn't the bottleneck).

Now, let's look at the neighborhood of S, and assume that indeed $|S| \geq np^2/6$. Notice that so far we have only conditioned on the edges out of u and v. This means that for every other blue or green node w, the probability that w is *not* in the neighborhood of S is

$$(1-p)^{|S|} \leq e^{-p|S|} \leq e^{-np^3/6} = e^{-\Omega(n^{3\epsilon})} = o(1/n).$$

So, by the union bound, with high probability *every* blue and green node w is in T. □

4 Smoothed Analysis

Smoothed analysis was developed by Spielman and Teng in this paper [ST00] with the goal of explaining why the Simplex algorithm works well in practice even though it can take exponential time in the worst case. Similar to the semi-random model, smoothed analysis involves taking an adversarial input and adding a small amount of random noise. What's different is that the noise here is real-valued, rather than binary, and typically added to *all* entries, not just *allowable* entries. For example, for linear programming, to maximize $\mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$, we would add small amounts of Gaussian noise to every entry in A and $\mathbf{b}$, even if this LP came from (say) a network flow problem, where zeroes "should" be truly zero.

A number of problems have been analyzed in the smoothed analysis model, including linear programming, the knapsack problem, and the Traveling Salesman problem. We'll see this model popping up again when we get to online algorithms for data-driven algorithm design, where we'll use something very similar to get around worst-case lower bounds. For now, see Anupam Gupta's lecture notes (below) for more information.

Additional Resources

- [B90] A. Blum, "Some Tools for Approximate 3-Coloring", FOCS 1990. `https://home.ttic.edu/~avrim/Papers/colortools.pdf`
- [ST01] D. Speilman and S. Teng, "Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time", STOC 2001. `https://dl.acm.org/doi/10.1145/990308.990310`
- U. Feige, "Introduction to Semi-Random Models", Chapter 9 in Beyond Worst-Case Analysis. `https://www.wisdom.weizmann.ac.il/~feige/mypapers/bwca.pdf`
- Smoothed analysis lecture notes by Anupam Gupta: `https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15850-f20/www/notes/lec26.pdf`