Lecture notes: Beyond worst case analysis – Models

Given the hardness of approximation results we have seen, it is natural to ask: are some problems simply hopeless in terms of getting approximation guarantees? Today we will discuss cases in which we can still prove some guarantees, as long as the inputs are not "worst-case".

Getting around hardness results

The PCP theorem allows us to understand the approximability of various optimization problems, as we have seen. For problems such as Independent-Set, it allows us to obtain extremely strong inapproximability results. Even for *natural* problems such as graph partitioning, constant factor approximations are not expected to be possible in the worst case.

While this is great from the point of view of theoretical understanding, it suggests that when instances of these problems arise in practice, heuristics are all one can obtain, and there is no hope for guarantees. Recently, researchers have started to repair this rather pessimistic viewpoint. The hardness results do not say that *all* instances of a problem are difficult. Indeed, the reductions often give rise to very structured instances – the kind that we are unlikely to encounter in practice.

Can we formalize this intuition? For different problems, can we come up with heuristic algorithms that *provably* work well on *realistic* instances? Another question we could ask is, can we come up with heuristics that work well on *most* instances?

This line of questioning has come to be known as *beyond worst case analysis*, and is an active research area. Explaining the *practical success* of heuristics is another goal of this research. For many natural problems (e.g., graph partitioning, and several problems in machine learning), known hardness results say that achieving good approximation ratios is impossible, while *useful* solutions can be obtained in practice via heuristics (deep learning is a classic example).

We will study four broad approaches. We also refer to Tim Roughgarden's lecture notes for a deep dive into each of these topics: *http://theory.stanford.edu/∼tim/f14/f14.html*

Generative models

The first and perhaps the most "classic" alternative to worst case analysis is *average case analysis*, in which one tries to analyze the performance (running time/approximation quality) of an algorithm assuming data is drawn from a distribution over inputs (uniform random being the simplest example). A classic example is the average case running time of quicksort (which is $O(n \log n)$ as opposed to $O(n^2)$ in the worst case).

However, most interesting problems become trivial if we restrict to a *uniformly random* distribution over inputs. For instance, consider the question of finding the size of the maximum independent set in a graph. If G is drawn indepdently from $G(n, 1/2)$ (every edge is chosen independently with probability $1/2$), then it turns out that the largest indepdent set has size $2 \log n$ with high probability. Thus outputting $2 \log n$ as the answer is trivially a very good approximation (with high probability). Similarly, if we consider a natural way of constructing *random* instances of 3-SAT, in which we pick a certain number m of clauses *randomly* (i.e., each clause is picked uniformly from the set of all 3-literal clauses over a set of n variables, independently of

Getting around hardness results continued on next page...

the other clauses), then it turns out that there is a sharp threshold τ such that if $m > \tau$, then the formula is unsatisfiable w.h.p., and if $m < \tau$, the formula is satisfiable w.h.p. This makes the question of checking satisfiability trivial (w.h.p.)

Thus, for many problems, coming up with interesting *probabilistic models* for data is an art. The goal is to come up with a model for data that is *representative* of real life instances. Ideally, we would like algorithms that work well for the model to also work on real data. The algorithmic task is to design *robust* algorithms for data generated from these models. We will study a couple of such models in the next lecture.

Smoothed analysis

For some problems, heuristics *always* seem to do quite well in practice. The classic example is the simplex algorithm for linear programming. While instances for which the algorithm takes exponential time are known, it turns out that in nearly all applications, simplex algorithm competes with, or even performs better than, sophisticated interior point methods. Is there a theoretical *explanation* for this?

In their celebrated paper, Spielman and Teng came up with a new formalism to explain this, called *smoothed analysis*. They proved that *hard* instances for the simplex method are extremely rare, in the following sense. Suppose we denote by $T(I)$ the running time of the simplex algorithm on the instance I. Spielman and Teng considered *perturbations* of instances by a quantity σ (for an instance of linear programming, imagine perturbing every equation that defines a constraint by a Gaussian of variance σ^2, independently at random). Let us call the perturbed instance $\widetilde{I}$. Their main result then says (informally):

> For *every* instance I, the running time of a σ-perturbation $\widetilde{I}$ satisfies
>
> $$T(\widetilde{I}) = \text{poly}(n/\sigma)f(1/\delta),$$
>
> with probability at least $1 - \delta$. The function $f()$ is also at most polynomial (but could often be much better, even logarithmic).

The key point in the result is the dependence on σ. Even for fairly small perturbations ($1/n^2$, which could just be measurement errors in the coefficients of the LP), this suggests that the perturbed instance is *easy* for the simplex algorithm.

Another way to think about the result above, is as saying that the *bad instances* are rare, and even in a small ball around a bad instances, we mostly find good instances. (Thus, in a sense, the bad instances are not *robust*.) We will refer to the introductory portion of the paper of Spielman and Teng for more exposition. In the next lecture, we will see a concrete example.

Stability of the optimum

In many problems (specifically of the clustering type), instances that arise in practice have a *ground truth* solution that optimizes the objective. This solution is often *unique* in a strong sense, and also *stable*. Let us formalize these two notions.

- **Uniqueness.** Consider an optimal solution S to an instance I of some optimization problem. We say that S is *robustly unique*, if any solution S' that is close to S in objective value is also close to S in terms of some natural metric between solutions.

- **Stability.** An optimal solution S to an instance I of an optimization problem is said to be stable if it remains optimal even if we perturb I.

Formal definitions of the two notions (what is 'close'? how to perturb? etc.) can be found in the lecture notes linked above. For various clustering objectives, if we know that there exists a stable, or a robustly unique optimum, it turns out that we can develop algorithms with much better approximation guarantees.

Bi-criteria approximations

We will discuss this formally next week, but here's an example that illustrates the basic idea. The *max k-coverage* problem is the following (is is very similar to the set cover problem we saw early on in the course). We are given a set of people P, and each person is an expert on a subset of topics T. The goal is to find k people who collectively are experts on the most number of topics. Alternately, given a collection of sets, we want to pick k of them so as to maximize the union. (Note the difference to SetCover).

Suppose we denote by OPT_k the best possible objective value obtained from picking k people. It turns out approximating OPT_k to a factor better than $1 - 1/e \approx 0.63..$ is NP-hard – i.e., finding a set of k people with objective value $> 0.64 \cdot \text{OPT}_k$ is hard. Now, suppose we are allowed to pick *a few more than* k people. Can we end up with an objective value say $0.99 \cdot \text{OPT}_k$?

It turns out this is possible, and algorithms of this kind are called bi-criteria approximations. In this case, it turns out that one can pick $\approx k \log(1/\epsilon)$ people, and obtain an objective value $(1 - \epsilon)\text{OPT}_k$. This is sometimes referred to as *resource augmentation*.