Outline:

- Introduction to smoothed analysis
- 2-OPT heuristic for TSP

In this lecture and the next, we discuss a 'beyond worst-case analysis' paradigm for the design of algorithms– called smoothed analysis. Our presentation draws heavily from the excellent lecture notes of Roughgarden [Rou17].

1 Introduction

Smoothed analysis was originally introduced in a seminal paper of Spielman and Teng [ST04] to analyze the performance of the simplex algorithm, an extremely well-known and practical algorithm for linear programming. It is known that the simplex algorithm takes exponential time in the worst case. Such worst-case instances, however, do not arise in practice, where the simplex algorithm and its variants form the dominant methods for linear programming (and preferred over the polynomial-time ellipsoid algorithm). There have been many attempts to explain this. In their breakthrough paper Spielman and Teng gave a plausible theoretical justification for the superior performance of the simplex algorithm in practice. They introduced the notion of smoothed analysis, as a tool to analyze algorithms and heuristics that goes beyond the worst-case performance. The high level approach is to study the performance of the algorithm on randomly perturbed instances; in the case of simplex, they show that for *any input instance* x, the expected running time of the simplex algorithm on an input x' obtained by a random perturbation of x is polynomial, where the expectation is over the perturbation. For the smoothed analysis model, we need to make some assumptions about the perturbations allowed.

Perturbation assumption: Each point in an input instance, which is problem specific, can be thought of as being perturbed according to a distribution with parameter σ. The perturbation distribution is assumed to not be "spiky". Intuitively, we want a "smooth" looking perturbation.

Example of a perturbation: Consider the uniform distribution $f(x) = \frac{1}{\sigma}$ for $x \in (0, \sigma)$. The points in the input are perturbed by a random number $r(\sigma)$ drawn from this distribution, where σ can be viewed as the maximum perturbation.

Given an input $x \in \mathbb{R}^n$ and an algorithm A, we let $c(A, x)$ denote the cost of A on input x. For the purpose of this lecture, we assume the cost is the running time of the algorithm. Now, instead of running A on x, consider perturbing x by $r(\sigma)$ and running A on $x + r(\sigma)$. The smoothed complexity of algorithm A is defined as,

$$S(n) = \sup_{x \text{ s.t. } |x|=n} \mathbb{E}_{r(\sigma)}[c(A, x + r(\sigma))]$$

A problem is said to admit a smoothed polynomial time algorithm, if there exists an algorithm with smoothed complexity which is polynomial in the input size (i.e. n) and $\frac{1}{\sigma}$.

The motivation for smoothed analysis is that worst-case analysis, while being extremely useful in classifying algorithms and problems according to their efficiency, can be very pessimistic in terms of analyzing algorithms that perform well in practice but may not be efficient in the worst case. On the other hand, average case analysis can be overly optimistic in that the distributional assumptions can be rather strong and may not reflect ”typical” inputs. Let us observe how smoothed analysis can be viewed as an interpolation between average case analysis and worst case analysis: if $\sigma \to 0$, then smoothed analysis tends to worst case analysis, while if $\sigma \gg 0$, then smoothed analysis essentially mimics average case analysis over an input distribution determined by the perturbation since the input instances are essentially defined by the perturbation distribution.

2 Local search for TSP

An accessible example of how smoothed analysis can be employed is the analysis of the well known 2-OPT heuristic for solving the travelling salesperson problem (TSP) on the plane [ERV16]. We consider TSP under the ℓ_1 norm. We are given n points in the plane $\mathbb{R}^2$ and in the ℓ_1 metric such that $d(x, y) = |x_1-y_1|+|x_2-y_2|$ for any $x, y \in \mathbb{R}^2$. The objective is to find the minimum cost tour connecting the n points. The local search algorithm for this problem is extremely simple.

The Local Search 2-OPT heuristic: Given any tour, consider swapping a pair of edges $\{(u, v), (x, y)\}$ with $\{(u, y), (v, x)\}$ if it improves the cost of the tour, i.e. if $d(u, v) + d(x, y) - d(u, y) - d(v, x) > 0$. If no such pair exists, return the current tour.

Our focus is on the running time of this local search algorithm, and in particular the number of iterations to convergence since the cost of checking whether such a pair exists is clearly polynomial (enumerate over all $\binom{n}{4}$ combinations of swaps). We prove the following theorem.

Theorem 1. The expected number of iterations taken by the 2-OPT heuristic for TSP when the points are in $\mathbb{R}^2$ and the distances are given by the standard ℓ_1 metric, is $O(\frac{n^5 \log n}{\sigma})$ where the probability density function $f(x)$ of the perturbation function satisfies $f(x) \leq \frac{1}{\sigma}$ for all x in the domain.

Theorem 1 basically states that if the input points are slightly perturbed, then in expectation the number of iterations taken by local search is polynomial in n and $\frac{1}{\sigma}$. Note that we are not analyzing the length of the tour; instead, we are analyzing how long the local search will take to terminate. For ease of exposition, we assume that all the n points are in $[0, 1] \times [0, 1]$; this is without loss of generality since we can translate and scale the points by the largest coordinate on each axis. We begin with an observation about the length of a tour.

Observation: The maximum length of any tour, and in particular, the initial tour is bounded by $2n$, since the ℓ_1 distance between any 2 points in $[0, 1] \times [0, 1]$ is bounded by 2. On the other hand, if the maximum distance between any two points is at least 1, then the length of the any tour is at least 1.

Perturbation assumption: Let $f(x)$ denote the density function of the perturbation. We assume that $f(x) \leq \frac{1}{\sigma}$ where σ is a distributional parameter. Each coordinate of a point is randomly perturbed by a quantity drawn at random from $[0, \sigma]$, i.e. for a point $u = (u_1, u_2)$, the perturbed point is $u' = (u_1 + r(\sigma), u_2 + r(\sigma))$ where the perturbations in each coordinate are independent.

Claim 1. Under the above perturbation assumption, it holds for any $x \in \mathbb{R}$, $\Pr[x + r(\sigma) \leq x + \epsilon] \leq \frac{\epsilon}{\sigma}$.

Proof. Follows directly from the perturbation assumption. □

Our proof will quantify the probability that an iteration leads to a large improvement in cost; many such iterations provide an upper bound on the number of iterations. We quantify the 'goodness' of a swap by measuring the change in the cost in the resulting tour. We say a swap is ϵ-bad if it reduces the length of the tour by less than ϵ. The following lemma precisely captures the expected impact of perturbations on the improvement by a swap.

Lemma 1. For a given iteration, the probability that there exists an ϵ-bad swap is at most $O(\frac{\epsilon n^4}{\delta})$.

Proof. Fix any swap $\{(u, v), (x, y)\}$ where (u, v) is swapped with (u, y) and (x, y) with (v, x). The change in the cost is given by the following.

$$\begin{aligned}\Delta(C) &= d(u, v) + d(x, y) - d(u, y) - d(v, x) \\ &= |u_1 - v_1| + |u_2 - v_2| + |x_1 - y_1| + |x_2 - y_2| + |u_1 - y_1| + \quad |u_2 - y_2| + |v_1 - x_1| + |v_2 - x_2|.\end{aligned}$$

Observe that the coefficient corresponding to every coordinate in the expanded expression is in $\{-2, 0, 2\}$. This gives 3^8 possible combinations corresponding to all possible coefficient values of the coordinates in the expression. Fix one of the combinations, and a particular coordinate say u_1: what is the probability that $\Delta(C) \in (0, \epsilon)$? Let $\Delta(C)_{\overline{u_1}}$ denote the contribution of all the terms excluding u_1. Since the coefficient of u_1 is at most 2, the probability that $\Delta(C)_{\overline{u_1}} + 2u_1 \in (0, \epsilon)$ is bounded by $\frac{\epsilon}{2\sigma}$ (by Claim 1. Taking a union bound over all possible $\binom{n}{4}$ swaps completes the proof. □

We now prove Theorem 1 by analyzing the expected number of iterations. Note that if none of the swaps is ϵ-bad, then the number of iterations is given by $\frac{2n}{\epsilon}$, since each iteration would decrease the objective value by at least ϵ, and the worst case cost of the tour is $2n$ as observed earlier. On the other hand, the worst case number of iterations is $n!$–the number of possible tours on n points. The idea is to consider different ranges of ϵ and consider the expected number of iterations for each range. Each range is of the form $[\frac{1}{2^{i+1}} + 1, \frac{1}{2^i}]$ for $i \in [0, \log n!]$. The number of iterations if $\epsilon \in [\frac{1}{2^{i+1}} + 1, \frac{1}{2^i}]$ is bounded by $\frac{2n}{1/2^{i+1}} = 2n \cdot 2^{i+1}$. Thus, the expected number of iterations whenever $\epsilon \in [\frac{1}{2^{i+1}} + 1, \frac{1}{2^i}]$ is bounded by $2n \cdot 2^{i+1} \cdot \frac{n^4}{\sigma 2^{i+1}} = O(\frac{n^5}{\sigma})$. The number of possible ranges are $\log(n!) = O(n \log n)$, so that the total number of iterations taken by the local search heuristic is $O(\frac{n^6 \log n}{\sigma})$.

We provide an alternative proof utilizing the fact that for a discrete random variable X taking non-negative values, we have that $E[X] = \sum_{i \geq 1} \Pr[X \geq i]$. Let Y denote the number of iterations for the local search heuristic. Then, we have that,

$$\begin{aligned}E[Y] &= \sum_{i=1}^{n!} \Pr[Y \geq i] \\ &= \sum_{i=1}^{n!} \Pr[\exists \left(\frac{2n}{i}\right) \text{-bad swap}] \\ &= \sum_{i=1}^{n!} O\left(\frac{2n^5}{i\sigma}\right) \\ &= O\left(\frac{n^5 \log n}{\sigma}\right)\end{aligned}$$

References

[ERV16] Matthias Englert, Heiko Röglin, and Berthold Vöcking. Smoothed analysis of the 2-opt algorithm for the general tsp. *ACM Trans. Algorithms*, 13(1), sep 2016.

[Rou17] T. Roughgarden. Smoothed analysis of local search. Lecture notes, available from `https://timroughgarden.org/w17/l/l17.pdf`, 2017.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, may 2004.