Fall 2014 Randomized Algorithms Sep 24, 2014

Lecture 1

Prof. Friedrich Eisenbrand *Scribes: Alfonso Cevallos*

A randomized algorithm is an algorithm that has access to a source of (pseudo-)random bits, and bases its actions accordingly. In other words, when taking decisions it "flips a coin" several times. Consequently, for a fixed input the running time and the output become random variables, so we use statistical tools to measure its performance.

There are several reasons to study randomized algorithms. Sometimes they provide the most elegant and intuitive solution to a problem, or may provide additional structural insight. And as we shall see, for some problems they offer a performance that is provably unachievable with a deterministic algorithm.

Min-Cut Problem and Karger's algorithm

We start off with our first example. We are given a simple connected graph $G = (V, E)$ with $|V| = n$, and positive edge weights $w : E \to \mathbb{R}_{>0}$. For a vertex subset $U \subset V$, its cut-set $\delta(U)$ is the collection of edges that connect U to its complement $V \setminus U$. That is, $\delta(U) = \{e \in E \ : \ |e \cap U| = 1\}$. Its weight is $w(\delta(U)) = \sum_{e \in \delta(U)} w(e)$. The objective is to find a non-empty proper vertex subset U whose cut-set is of minimum weight.

Karger's algorithm is based on the *contraction operation*, which is defined as follows. For an edge $e \in E$, contracting it means erasing e, merging its endpoints into a new vertex, and replacing any resulting double edges by a single edge whose weight is the sum of weights of the previous edges. Notice that any contraction reduces the number of vertices by exactly 1.

The algorithm is: While $|V| > 2$, select an edge e at random, with probability equal to $\frac{w(e)}{w(E)}$, and contract it. After $n-2$ rounds, we end up with 2 supernodes, which represent a vertex partition $\{U, V \setminus U\}$. Output $\delta(U) = \delta(V \setminus U)$.

For the analysis, suppose $\delta(U')$ is the minimum-weight cut-set (we fix a particular optimal solution if it is not unique). We will prove that the probability to output exactly $\delta(U')$ is not too small. We say that $\delta(U')$ "survives" the i-th execution round if none of its edges is contracted during this round, and it is easy to see that the output will be $\delta(U')$ iff it survives all rounds. Recall now the basic graph theoretical identity $\sum_{v \in V} \deg(v) = 2|E|$, obtained by a double-counting argument. The same argument also gives the corresponding weighted

version: $2w(E) = \sum_{v \in V} w(\delta(v))$. Moreover, for any vertex v, by minimality of the cut-set $\delta(U')$ we know that $w(\delta(U')) \leq w(\delta(v))$, so we get the inequality

$$2w(E) = \sum_{v \in V} w(\delta(v)) \geq n \cdot w(\delta(U')). \tag{1}$$

Now, for $1 \leq i \leq n-2$, if P_i is the probability that $\delta(U')$ survives the i-th round, conditioned on the event that it has survived all previous rounds, then clearly $P_i = 1 - \frac{w(\delta(U'))}{w(E(G_i))}$, where G_i is the state of the graph at the beginning of the round (with $G_1 = G$ and $|V(G_i)| = n - i + 1$). As all "surviving" cut-sets preserve their weights, $\delta(U')$ is still minimal in G_i, and the inequality (1) applied to graph G_i gives $\frac{w(\delta(U'))}{w(E(G_i))} \leq \frac{2}{n-i+1}$. Therefore $P_i \geq 1 - \frac{2}{n-i+1} = \frac{n-i-1}{n-i+1}$. Finally, the probability that $\delta(U')$ survives all rounds (and is output) is $\prod_{i=1}^{n-2} P_i \geq \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} = \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$.

Corollary 1 *The number of minimum-weight cut-sets of an n-vertex graph G is at most $\binom{n}{2}$.*

It is left as an exercise to check if this upper-bound is tight. This corollary shows how a randomized algorithm can provide some structural insight into the problem. Now, what if we repeat the algorithm above k times? The probability that the optimal solution $\delta(U')$ is NOT output in any of the k runs is at most $(1 - \frac{2}{n^2})^k \leq \exp(-\frac{2k}{n^2})$. Therefore, it is enough to run the algorithm $O(n^2)$ times to get the exact solution with arbitrarily high constant probability.

Linear Programming and Seidel's algorithm

For a given matrix $A \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, a linear programming (LP) instance is of the form $\max\{c^T x : \ Ax \leq b, \ x \in \mathbb{R}^n\}$. The vector inequality stands for coordinate-wise inequalities, so each row in A corresponds to a linear constraint. It is an open question whether there is a *strongly polynomial* algorithm for solving LPs. This means, an algorithm whose running time is polynomial in n and m, and where the binary encoding lengths of all numbers involved remain polynomial in n and m. So far, all known algorithms for LP are at best *weakly polynomial*, i.e., they are polynomial only when compared to the largest binary enconding length of the input numbers.

We can assume, w.l.o.g., that A has full column-rank, because if it is not the case, there is a non-singular matrix $U \in \mathbb{R}^{n \times n}$ such that $AU = [A'|0]$, where A' is of full column-rank, and we rewrite the problem as follows. Let $U^T c =: d = \binom{d_1}{d_2}$ and $U^{-1} x =: y = \binom{y_1}{y_2}$. Then $c^T x = c^T U U^{-1} x = d^T y = d_1^T y_1 + d_2^T y_2$; and $Ax = AUU^{-1}x = [A'|0]y = A'y_1$. If $A'y_1 \leq b$ is unfeasible, then so is the original problem. Else, if $d_2 \neq 0$, the problem is unbounded (as there are no constraints over the vector y_2). Else, we can rewrite the problem as $\max\{d_1^T y_1 : \ A'y_1 \leq b\}$.

It can also be assumed, w.l.o.g., that on any feasible point there are at most n constraints that are tight (i.e., the LP is non-degenerate), and it can be proved as well that if the LP is feasible and bounded, then the optimum value is achieved in a vertex of the polytope defined by the constraints (where *exactly* n constraints are tight).

Next, we describe Seidel's algorithm for LP, denoted SeidelLP, which runs in expected linear time in m, when the dimension n is fixed. We assume the input instance to be feasible and bounded, and we consider only the case $n = 2$, but the general case can the easily derived by induction. Let H denote the set of m constraints, which for $n = 2$ correspond to lines in the plane. The algorithm is:

- If $H = 2$, return the intersection point of the two lines. (Assume $H > 2$);
- Pick a constraint $h \in H$ uniformly at random, and find by recursion $y = \text{SeidelLP}(H \setminus h)$;
- If y satisfies h, return y;
- Else: We know h must be tight, i.e., the optimum solution lies on (the line defined by) h. Find the intersection points of all other constraint lines with h. If we parametrize now the line h by a real variable, the constraints consist of a list of numerical upper and lower bounds. In linear time we can find the optimal point.

For the analysis of the algorithm, let $T(m)$ denote its expected running time on an instance with m constraints. The time-consuming portions of the algorithm are the last step, which runs in linear time, and the recursive calls. Notice that there are two tight constraints for the optimum vertex, and the algorithm makes a recursive call unless it finds a tight constraint, with probability $\frac{2}{m}$. Hence, $T(m) \leq T(m-1) + \frac{2}{m}O(m) = T(m-1) + O(1)$, and therefore $T(m) = O(m)$.

We can extend this algorithm for any fixed n, and prove by induction that its expected running time is still $O(m)$. However, the dependancy on n will be exponential (it is left as an exercise to prove it).

Volume estimation of convex bodies

Let $B_n \subset \mathbb{R}^n$ be the n-dimensional unit ball centered in the origin. A set $K \subset B_n$ is called a *convex body* if it is full-dimensional, compact and convex. By full-dimensional, we mean that there is $x \in K, \epsilon > 0$ such that the n-dimensional ball with center in x and radius ϵ is contained in K. In an instance of the problem, an adversary fixes a convex body $K \subset B_n$ that is initially unkown to us. The only way to gain information about K is via a black-box *membership oracle*, which for any input point $x \in \mathbb{R}^n$, in polynomial time outputs truthfully whether $x \in K$. The goal of the problem is to approximate the

volume of K.

This is an example of a problem where randomization is provably necessary. We present a hardness-of-approximation result for deterministic algorithms, and in a future lecture we will see how a randomized algorithm does the job. But first we need to following lemma.

Lemma 2 *Given any m points $x_1, \cdots, x_m \in B_n$, we denote by Conv$\{x_1, \cdots, x_m\}$ their convex hull. Then Vol(Conv$\{x_1, \cdots, x_m\}$) $\leq \frac{m}{2^n}$ Vol(B_n).*

Proof For $1 \leq i \leq m$, let B^i be the ball with center $\frac{x_i}{2}$ and radius $\frac{\|x_i\|}{2}$. Hence B^i is defined in such a way that x_i and 0 are antipodal points on its surface, and any point Q is in B^i iff the angle $0Qx_i$ is $\geq \frac{\pi}{2}$. Notice also that $\mathrm{Vol}(B^i) \leq 2^{-n}\mathrm{Vol}(B_n)$, because the radius of B^i is $\leq \frac{1}{2}$. We claim that $\mathrm{Conv}\{x_1, \cdots, x_m\} \subset \cup_{i=1}^m B^i$, whence the lemma follows immediately.

Assume, otherwise, that there is a point $Q \in \mathrm{Conv}\{x_1, \cdots, x_m\}$ such that $Q \notin B^i$ for each $1 \leq i \leq m$. Then, for each i, the angle $0Qx_i$ is $< \frac{\pi}{2}$. So if P is the hyperplane containing Q and perpendicular to segment $0Q$, all points x_i are at a non-zero distance from P, and on the same side as the origin. Hence P is separating Q from $\mathrm{Conv}\{x_1, \cdots, x_m\}$, which leads to a contradiction. ■

Theorem 3 *There does not exist a deterministic, polynomial-time, constant-approximation algorithm for the problem of volume estimation of convex bodies.*

Proof More concretely, what we need to prove is that for any positive constants c, k, N, and any deterministic algorithm ALG that makes at most cn^k queries to the membership oracle, there is an instance $K \subset B_n$ of the problem such that, if ALG(K) is the volume estimation output by the algorithm, then either $\mathrm{ALG}(K) < \frac{1}{N}\mathrm{Vol}(K)$ or $\mathrm{ALG}(K) > N\mathrm{Vol}(K)$.

Consider such fixed constants c, k, N, and algorithm ALG, and let n be big enough so that $2^n > N^2 cn^k$. Now, let $x_1, \cdots, x_m$ ($m \leq cn^k$) be the query points of ALG for the instance $K = B_n$, and define $K' = \mathrm{Conv}\{x_1, \cdots, x_m\}$. As ALG is deterministic, it will query exactly the same points (and obtain the same results from the oracle) for both K and K', so it necessarily outputs the same estimation $\mathrm{ALG}(B_n) = \mathrm{ALG}(K')$.

From the previous lemma, together with our choice of n, we obtain that $\mathrm{Vol}(K') \leq \frac{m}{2^n}\mathrm{Vol}(B_n) \leq \frac{cn^k}{2^n}\mathrm{Vol}(B_n) < \frac{1}{N^2}\mathrm{Vol}(B_n)$. Therefore, either $\mathrm{ALG}(K') > N\mathrm{Vol}(K')$ or $\mathrm{ALG}(B_n) < \frac{1}{N}\mathrm{Vol}(B_n)$ ■