CIS 800/002 The Algorithmic Foundations of Data Privacy November 3, 2011

Lecture 15

Lecturer: Aaron Roth *Scribe: Aaron Roth*

Private Combinatorial Optimization: Min Cut

We have now exhausted much of what I know about the problem of private query release, so it is time to move onto a new topic: combinatorial optimization. Here we will be concerned with solving *search* problems, where the goal is to not only release the numeric objective value of some optimization problem, but also the solution itself which leads to that objective value. The field of approximation algorithms generally concentrates on finding near-optimal solutions in polynomial time to NP-hard optimization problems. In this setting, the only reason that we cannot achieve exact solutions is our own feeble computational limitations. When we require that our algorithms satisfy differential privacy as well, however, we also have an information theoretic constraint preventing us from returning exact solutions. Therefore, when privacy is a constraint, it makes sense to study approximation algorithms even for problems which are polynomial time solvable. That is what we will do today.

Definition 1 *An undirected graph $G = (V, E)$ is defined by a* vertex set *V of size $|V| = n$ and an* edge set *$E \subset V \times V$. If $(u, v) \in E$ We say that there is an edge between vertex u and vertex v if $(u, v) \in E$. For undirected graphs, we consider (u, v) to be equivalent to (v, u).*

An important question when formulating a privacy problem is what is the granularity of privacy that we wish to protect. We will consider *edge privacy*, meaning we will view V as publicly known, and view the edge set E as the private database. That is, if we have two edge sets E, E' differing in a single edge: $|E \triangle E'| \leq 1$, defining two graphs $G = (V, E)$ and $G' = (V, E')$, and an algorithm $A : V \times (2^V) \to R$, the constraint of (ϵ, δ)-privacy would require that for all $S \subseteq R$: $\Pr[A(G) \in S] \leq \exp(\epsilon) \Pr[A(G') \in S] + \delta$. We could also choose to try and protect *vertex privacy*, which would insist that algorithms be insensitive to the addition or removal of a vertex, as well as all (possibly $n - 1$) edges adjacent to it. This would be a much stronger privacy guarantee, but less would be possible.

In general, a combinatorial optimization problem is defined by a feasible set and an objective function.

Definition 2 *An optimization problem $P = (\mathcal{F}, Q)$ is an* feasible set *of solutions together with an objective function $Q : \mathcal{F} \to \mathbf{R}$. For a problem P, we write $OPT(P) = \min_{x \in \mathcal{F}} Q(x)$.*

We can now define the min-cut problem.

Definition 3 *A cut is a subset of vertices $S \subseteq V$. The value of a cut in a graph G, written $C_G(S, S^C)$ is defined to be the number of edges crossing from S to $S^C = V \setminus S$:*

$$C_G(S, S^C) = |\{(u, v) \in E : u \in S, v \notin S\}|$$

The min-cut problem is: given a graph $G = (V, E)$, return a cut $S \subseteq V$ minimizing the quantity $C_G(S, S^C)$. That is, the feasible set is $\mathcal{F} = \{S \subseteq V\}$, and the objective function is $Q(S) = C_G(S, S^C)$. There are many classical algorithms to compute this quantity exactly in polynomial time, if privacy is not a concern. In the non-private setting, approximation algorithms typically give multiplicative approximation guarantees, but in private settings, additive approximations will be necessary.

Definition 4 *An algorithm A is an $(T(\epsilon), F(\epsilon))$-private approximation algorithm for a class of optimization problems $\mathcal{P}$, if it is ϵ-differentially private, and for any $P = (\mathcal{F}, Q) \in \mathcal{P}$, with high probability:*

$$Q(A(P)) \leq T(\epsilon) OPT(P) + F(\epsilon)$$

Observe that the *value* of the min-cut is a 1-sensitive function, so we can easily output its value with only constant additive error. But we want to actually output the cut itself.

We will make use of a by now standard fact about graph cuts, first proven by Karger.

Theorem 5 (Karger's Min-Cut Theorem) *For any graph $G = (V, E)$ with $|V| = n$ and min-cut size C, and for any $\alpha \geq 1$ there are at most $n^{2\alpha}$ cuts of size at most $\alpha \cdot C$. Moreover, given a graph, all of these cuts can be found efficiently.*

If you haven't seen this theorem before, it is worth reading about – its really cool! Karger gives a randomized "contraction" algorithm that is very simple, and an analysis that shows that when you run it on a graph, any fixed cut of size at most αC will be output with probability at least $n^{-2\alpha}$ – and hence there can be at most $n^{2\alpha}$ of them! This gives a fast algorithm for finding min-cuts as well, but we won't cover that here...

The idea of our algorithm will be simple, but for a few twists. We will seek to simply select the smallest cut using the exponential mechanism. We will be able to prove, using Karger's theorem, that the exponential mechanism gives a good result, so long as the true min-cut in the graph is not too small. But what if it is? We'll first change the graph to increase the size of the min-cut in a privacy preserving way. Finally, we have to sample from the exponential mechanism efficiently, so we'll sample just from the set of small cuts (which we can generate with Kargers algorithm), and show that this still preserves (ϵ, δ) privacy. So here is the algorithm:

Min-Cut$(G = (V, E), \epsilon)$

Let $H_0 \subset H_1 \subset H_2 \subset \ldots \subset H_{\binom{n}{2}}$ be an arbitrary strictly increasing sequence of edge sets on V.

Choose index i with probability proportional to $\exp(-\epsilon|\text{OPT}((V, E \cup H_i)) - \frac{16 \ln n}{\epsilon}|)$.

Choose a subset $S \subseteq V$ with probability proportional to $\exp(-\epsilon \cdot C_{(V,E \cup H_i)}(S, S^C))$

Output the cut S.

We will prove the following utility theorem:

Theorem 6 *For any graph G, let Min-Cut$(G, \epsilon) = S$. Then with probability at least $1 - O(1/n^2)$: $C_G(S, S^C) \leq OPT(G) + O(\frac{\log n}{\epsilon})$. i.e. Min-Cut is a $(T(\epsilon), F(\epsilon))$-approximation algorithm for $T(\epsilon) = 1$ and $F(\epsilon) = O(\frac{\log n}{\epsilon})$.*

Proof We will prove this using two lemmas. First we will argue that with high probability, the algorithm chooses an index i such that: $\text{OPT}((V, E \cup H_i))$ is in the right range.

Lemma 7 *With probability at least $1 - 1/n^2$ we have:*

$$\frac{8 \ln n}{\epsilon} \leq OPT((V, E \cup H_i)) \leq OPT(G) + \frac{8 \ln n}{\epsilon}$$

Proof First suppose $\text{OPT}(G) < 16 \ln(n)/\epsilon$. Then there exists some i such that $\text{OPT}((V, E \cup H_i)) - \frac{16 \ln n}{\epsilon} = 0$. Recall our utility theorem for the exponential mechanism. For any range R, output r, and sensitivity-1 cost score q we have:

$$\Pr[q(r) \geq \text{OPT} + \frac{2}{\epsilon}(\log(R) + t)] \leq e^{-t}$$

Here, $\log(R) \leq 2 \log(n)$, and setting $t = 2 \log(n)$ as well we have:

$$\Pr[|\text{OPT}((V, E \cup H_i)) - \frac{16 \ln n}{\epsilon}| \geq \frac{8}{\epsilon} \log n] \leq \frac{1}{n^2}$$

Now, suppose $\text{OPT}(G) \geq 16 \ln(n)/\epsilon$. Now we know that $\text{OPT}((V, E \cup H_0)) = \text{OPT}(G)$, and so applying the utility theorem again we get:

$$\Pr[|\text{OPT}((V, E \cup H_i)) - \frac{16 \ln n}{\epsilon}| \geq (\text{OPT}(G) - \frac{16 \ln n}{\epsilon}) + \frac{8}{\epsilon} \log n] \leq \frac{1}{n^2}$$

■

Lemma 8 *If* $OPT((V, E \cup H_i)) \geq \frac{8 \ln n}{\epsilon}$ *and* $\epsilon < 1$ *then:*

$$\Pr[C_{(V,E\cup H_i)}(S, S^C) \geq OPT((V, E \cup H_i)) + b] \leq \frac{1}{n^2}$$

for $b = O(\log n/\epsilon)$

Proof Let c_t denote the number of cuts of size $\mathrm{OPT}((V, E \cup H_i)) + t$. Note that a cut of size $\mathrm{OPT}((V, E \cup H_i)) + t$ will be output with probability at most $\exp(-\epsilon t)$. Therefore we have:

$$\begin{aligned}
\Pr[C_{(V,E\cup H_i)}(S, S^C) \geq \mathrm{OPT}((V, E \cup H_i)) + b] &\leq \sum_{t \geq b} \exp(-\epsilon t) \cdot (c_t - c_{t-1}) \\
&= \left(\sum_{t \geq b} (\exp(-\epsilon t) - \exp(-\epsilon(t+1))) c_t \right) - \exp(-\epsilon b) c_{b-1} \\
&\leq \sum_{t \geq b} (\exp(-\epsilon t) - \exp(-\epsilon(t+1))) c_t \\
&= (1 - \exp(-\epsilon)) \sum_{t \geq b} \exp(-\epsilon t) c_t \\
&\leq (1 - \exp(-\epsilon)) \sum_{t \geq b} \exp(-\epsilon t) n^{t\epsilon/4 \ln n} \\
&= (1 - \exp(-\epsilon)) \frac{\exp(-\frac{3}{4}(b-1) \cdot \epsilon)}{\exp(\frac{3}{4}\epsilon) - 1} \\
&= O\left(\exp(-(b-1) \cdot \epsilon)\right)
\end{aligned}$$

where the last line follows because if $\epsilon < 1$, then $(1 - \exp(-\epsilon))/(\exp(\frac{3}{4}\epsilon) - 1) = \Omega(1)$. Plugging in $b = O(\log n/\epsilon)$ completes the proof. ■

Finally, putting the two pieces together, we get that except with probability at most $2/n^2$, we have:

$$C_G(S, S^C) \leq C_{(V,E\cup H_i)}(S, S^C) \leq \mathrm{OPT}((V, E \cup H_i)) + O\left(\frac{\log n}{\epsilon}\right) \leq \mathrm{OPT}(G) + O\left(\frac{\log n}{\epsilon}\right)$$

■

Finally, lets consider the efficiency of the algorithm. We run the exponential mechanism twice. The first run is selecting among a universe of only $\binom{n}{2}$ elements, and so runs efficiently. The second run is selecting among a set of 2^n cuts, and so does not in obviously run efficiently. But here is a trick:

There was nothing special about the failure probability $1/n^2$ in our analysis: in fact, we could have proven that for any constant c, the algorithm outputs a cut with error $O(\log n/\epsilon)$ with probability at least $1/n^c$. Moreover, using Kargers algorithm, we can actually generate all cuts of size $c \cdot \mathrm{OPT}(G)$ in time n^{2c}, and so for any graph with min-cut size at least $\log n/\epsilon$, we can generate all cuts of size $\mathrm{OPT}(G) + O(\log n/\epsilon)$ in time polynomial in n (Because there can be at most n^{2c} cuts of size $c \log n/\epsilon$). So what our algorithm will do is first generate all cuts of size $c \log n/\epsilon$, and then run only over this polynomially sized domain. This algorithm now runs in polynomial time: what are its privacy guarantees?

Consider a modified version of our exponential time Min-Cut algorithm, which reports "Fail" whenever it outputs a cut that has error worse than $O(\log n/\epsilon)$, which occurs with probability at most (say) $n^- c$. We can couple our modified polynomial time algorithm with this exponential time algorithm, and observe that their output distributions have statistical distance at most $n^- c$. Therefore, because the exponential time algorithm was $(\epsilon, 0)$-differentially private, the polynomial time algorithm must be (ϵ, δ)-differentially private for $\delta = n^{-c}$.

Bibliographic Information The min-cut algorithm is from "Differentially Private Combinatorial Optimization", 2010 by Gupta, Ligett, McSherry, Roth, and Talwar.