CS787: Advanced Algorithms

Topic: Differentially Private Approximation Algorithms **Presenter(s):** Balasubramanian Sivan, Tyson Williams

Instead of stating a new problem and presenting related results, we revisit existing problems and consider their privacy implications.

17.8.1 Introduction

Consider the following problems:

- Assign people using a social network such as Facebook to one of two servers so that most pairs of friends are assigned to the same server.
- Open a certain number of HIV treatment centers so that the average commute time for patients is small.
- Open a small number of drop-off centers for undercover agents so that each agent is able to visit some site convenient to her (each agent providing a list of acceptable sites).

These are all well known combinatorial optimization problems: respectively the minimum cut problem, the k-median problem, and the set cover problem. The minimum cut problem has a polynomial time algorithm, the k-median problem has a constant factor approximation algorithm, and the set cover problem has a logarithmic approximation algorithm. Although many would consider these problems well-studied and solved, the examples above should make it clear that privacy can be an important attribute of an algorithm. The input to these algorithms is friendship relations, medical history, and agents' locations. Since privacy was never considered during the design of these algorithms, it is notsurprising to learn that these algorithms fail to preserve privacy.

17.8.2 Differential Privacy

In order to prove that any algorithm *is* or *is not* a privacy preserving algorithm, we need a precise definition of privacy. Our notion of privacy is called *differential privacy*, which guarantees privacy to agents at an individual level.

17.8.2.1 Intuitive Definition

Deterministic algorithms are rarely privacy preserving ones. Therefore, we must make the algorithm randomized. Thus for any input, we have a distribution over outcomes and not a single outcome. Intuitively, the definition of differential privacy says that the distribution of outcomes of the computation does not change significantly when one individual changes her input data. A "change in input data" includes (1) that natural meaning where someone originally supplied input

i but now supplies input i', but it also includes the situations when (2) someone did not originally supply input but does so now, (3) someone originally did supply input but does not supply now.

17.8.2.2 Formal Definition

We say that a randomized computation M has ϵ-differential privacy if for any two input sets A and B with symmetric difference one, and for any set of outcomes $S \subseteq \text{Range}(M)$,

$$\Pr[M(A) \in S] \leq e^{\epsilon} \cdot \Pr[M(B) \in S].$$

Smaller values of ϵ give more privacy and $\epsilon = 0$ is completely private.

A direct consequence of this definition of differential privacy is that any output that is possible under some input is possible under all inputs.

17.8.3 Other Notions of Privacy

Before the introduction of differential privacy, other metrics for privacy existed, but some are too strong and some are too weak.

Prior work on Secure Function Evaluation tells us that min-cut can be computed in a distributed fashion in such a way that computations reveal nothing that cannot be learned from the output of the computation. This can be a strong notion of privacy, but, it is very weak when the output of a problem reveals much about the input. Such is the case for the minimum cut problem. What differential privacy provides is an unconditional per element privacy.

Another popular notion of privacy is that of functional privacy, which requires that two inputs with the same output value must have the same output under the approximation algorithm also. This requirement turned out to be so strong that, Halevi et al. [2] proved that, even approximating the value of vertex cover to within a factor of $n^{1-\epsilon}$ is as hard as the vertex cover problem itself.

17.8.4 Vertex Cover

The vertex cover problem should need no introduction since we have already studied three different ways to achieve a 2-approximation:

1. Both vertices of a maximum matching.
2. Rounding the LP solution of an ILP relaxation.
3. LP primal-dual algorithm.

Before discussing the benefit gained by applying differential privacy to this problem, we should first understand why vertex cover problem and covering problems in general are somewhat different, in terms of preserving privacy, from other problems like min-cut, k-median.

17.8.4.1 Challenges in vertex cover

The input of a vertex cover problem is a graph $G = (V, E)$. Now think about the set cover problem from the introduction. If each set contains exactly two elements, then we can model that problem as a graph where each set is an edge and the elements of the sets are vertices. This means that the existence or non-existence of an edge corresponds to the existence or non-existence of a secret agent. The private information that needs to be preserved is the existence of a secret agent.

While the private data for min-cut (edges) and k-median (vertices) determine only the cost of the objective function, the private data for vertex cover (i.e. edges) determines the set of feasible solutions itself for vertex cover. To see this, suppose that there is no secret agent sitting between the vertices u and v. If the set of vertices returned by the algorithm does not include u or v, then it has leaked the information that there is no secret agent between the vertices u and v. Any solution that can retain the privacy of existence of every single edge must output a vertex cover containing $|V| - 1$ vertices ([2]) which is useless. Thus, while outputting the value of vertex cover is not very difficult, outputting the vertex cover itself explicitly causes this kind of a problem.

17.8.4.2 Permutation Output

Since a vertex cover of size $O(|V|)$ can be arbitrarily far from the optimal solution, we must find a different way of outputting the vertex cover by reevaluating what we actually need from the output of the vertex cover algorithm. There is a secret agent on each edge and each secret agent needs a location to drop off secret packages. Therefore, all we really need is an orientation on each edge so that each secret agent knows to which drop-off box her secret packages should go. Even if an edge does not contain a secrete agent, we should specify an orientation on that edge as well. A permutation of the vertices can capture this information. If u appears before v in the permutation, then the orientation points toward u. In this way, a permutation gives each secret agent the necessary information about their drop-off location while preserving the existence or non-existence of a secret agent on every edge.

17.8.4.3 The Differentially Private Approximation Algorithm for Unweighted Vertex Cover

Gupta et al. [1] provide differentially private approximation algorithms for a number of combinatorial optimization problems including vertex cover, set cover, min-cut, k-median, and Steiner tree. We now discuss their differentially private approximation algorithm for the unweighted vertex cover problem.

As mentioned before, there are several algorithms that yield a 2-approximation for the vertex cover problem. However, none of these are differentially private. Instead of modifying any of these algorithms, we will modify the 2-approximation from [3]. Pitt's algorithm is a randomized non-private 2-approximation algorithm for unweighted vertex cover. It repeatedly selects an uncovered edge uniformly at random and selects a random end-point of that edge. This process can be seen equivalently as selecting a vertex at random with a probability proportional to its uncovered degree. The algorithm we discuss now is a slight modification of this algorithm, where we will pick a vertex with a probability proportional to the sum of its uncovered degree and a weight whose value we

discuss below.

We start with the graph $G_1 = G$ and denote by G_i the graph with $n - i + 1$ vertices remaining. Let $d_v(G)$ denote the degree of vertex v in G.

Algorithm 1: Private algorithm for unweighted vertex cover
ALG($G = (V, E)$)
(1) Let $n = |V|, V_1 = V, E_1 = E$.
(2) **for each** $i = 1, 2, \ldots, n$
(3) Let $w_i = (4/\epsilon) \cdot \sqrt{n/(n - i + 1)}$.
(4) Pick a vertex $v \in V_i$ with a probability proportional to $d_v(G_i) + w_i$.
(5) Output v.
(6) Let $V_{i+1} = V_i - \{v\}, E_{i+1} = E_i - (\{v\} \times V_i), G_{i+1} = (V_{i+1}, E_{i+1})$.

17.8.5 Privacy and accuracy analysis

We prove theorems about *ALG* which guarantee privacy and the approximation factor claimed.

Theorem 17.8.5.1 *(Privacy). ALG's differential privacy guarantee is*

$$max\{1/w_1, \sum_i 2/(i \cdot w_i)\} \leq \epsilon$$

for the settings of w_i above.

Proof: For any two sets of edges A and B, and any permutation π, let d_i be the degree of i^{th} vertex in the permutation π and let m_i be the remaining edges, both ignoring edges incident to the first $i - 1$ vertices in π. This method of calculating degree gives us the "uncovered" degree as opposed to the overall degree. We prove here that

$$\frac{\Pr[ALG(A) = \pi]}{\Pr[ALG(B) = \pi]} \leq \exp(\epsilon) \tag{17.8.5.1}$$

Proving (17.8.5.1) is enough to establish ϵ-differential privacy because, for any set of permutations S that could have been output, we have

$$\Pr[ALG(A) \in S] = \sum_{\pi \in S} \Pr[ALG(A) = \pi] \leq e^{\epsilon} \sum_{\pi \in S} \Pr[ALG(B) = \pi] = e^{\epsilon} \Pr[ALG(B) \in S]$$

So to prove (17.8.5.1), note that

$$\frac{Pr[ALG(A) = \pi]}{Pr[ALG(B) = \pi]} = \prod_{i=1}^{n} \frac{(w_i + d_i(A))/((n - i + 1)w_i + 2m_i(A))}{(w_i + d_i(B))/((n - i + 1)w_i + 2m_i(B))}$$

For a permutation π to be output, we need n events to happen, namely the vertex $\pi(i)$ is output in the i^{th} position for all $i = 1, 2, \ldots, n$. Let us say for simplicity that $\pi(i) = i$. The ith event, which is that i^{th} vertex is i, has has a probability $\frac{w_i + d_i(A)}{(n-i+1)w_i + 2m_i(A)}$. This is so because the degrees

of the remaining edges sum up to twice the remaining edges which $2m_i(A)$ and the weights of the remaining $(n-i+1)$ vertices, all of which are equal to w_i add up to $(n-i+1)w_i$.

When A and B differ in exactly on edge, $d_i(A) = d_i(B)$ for all i except the first endpoint incident to the edge in the difference. Until this term $m_i(A)$ and $m_i(B)$ differ exactly by one, and after this term $m_i(A) = m_i(B)$. If j is the index in π of the first endpoint of the edge in difference, we can cancel all terms after j and rewrite

$$\frac{Pr[ALG(A)=\pi]}{Pr[ALG(B)=\pi]} = \frac{w_j + d_j(A)}{w_j + d_j(B)} \prod_{i \le j} \frac{(n-i+1)w_i + 2m_i(B)}{(n-i+1)w_i + 2m_i(A)}$$

There are two cases now, namely $m_i(A) = m_i(B) + 1$ and $m_i(B) = m_i(A) + 1$. In the former case, each term in the above product is less than 1. Also, since $d_j(A) = d_j(B) + 1$, the leading term is at most

$$1 + 1/w_j < \exp(1/w_j) < \exp(1/w_1) = \exp(\epsilon/4)$$

In the later case, the leading term is less than one, but each term in the product exceeds one and we can bound them as follows

$$\begin{aligned}
\prod_{i \le j} \frac{(n-i+1)w_i + 2m_i(B)}{(n-i+1)w_i + 2m_i(A)} &= \prod_{i \le j} \frac{(n-i+1)w_i + 2m_i(A) + 2}{(n-i+1)w_i + 2m_i(A)} \\
&\le \prod_{i \le j} \frac{(n-i+1)w_i + 2}{(n-i+1)w_i + 0} \\
&= \prod_{i \le j} \left(1 + \frac{2}{(n-i+1)w_i}\right) \\
&\le \exp\left(\sum_{i \le j} \frac{2}{(n-i+1)w_i}\right)
\end{aligned}$$

Since we chose our w_i's as $w_i = (4/\epsilon) \cdot \sqrt{n/(n-i+1)}$, we have

$$\sum_i \frac{2}{(n-i+1)w_i} = (\epsilon/\sqrt{n}) \sum_i \frac{1}{2\sqrt{i}} \le \epsilon$$

■

Theorem 17.8.5.2 *For all G, $E[ALG(G)] \le (2 + 2avg_{i \le n} w_i) \times |OPT(G)| \le (2 + 16/\epsilon)|OPT(G)|$*

Proof: We prove this theorem by induction on the number of vertices n. If $|OPT(G)| \ge n/2$, we already have a two approx. So assume $|OPT(G)| < n/2$. Let v be the first vertex picked by ALG.

$$E[ALG(G)] = \Pr[v \text{ is incident on some edge}] + E_v[E[ALG(G - \{v\})]] \tag{17.8.5.2}$$

Recall that in step i, we pick each vertex with a probability proportional to $d_v(G_i) + w_i$. Thus, even if the degree is zero, there is a non-zero probability that a vertex gets picked. So in step 1, the probability that we pick a vertex that is incident on some edge is

$$\Pr[v \text{ is incident on some edge}] \le \frac{2mw_1 + 2m}{nw_1 + 2m} \tag{17.8.5.3}$$

The denominator is the sum of the weights and the degrees of all vertices. This quantity sums up to nw_1 (weights) and $2m$ (degree). The numerator is the sum of the degrees and weights of the non-zero degree vertices. The degrees of non-zero degree vertices sum up to $2m$. The number of non-zero degree vertices is at most $2m$ and thus the weights sum up to at most $2mw_1$. We would like to relate this quantity to $OPT(G)$. So we calculate,

$$\Pr[v \in OPT(G)] \geq \frac{|OPT(G)|w_1 + m}{nw_1 + 2m} \tag{17.8.5.4}$$

The explanation for (17.8.5.4) is similar to that for (17.8.5.3). Combining (17.8.5.3) and (17.8.5.4), we have

$$\begin{aligned}\Pr[v \text{ is incident on some edge}] &\leq (2w_1 + 2)\frac{m}{nw_1 + 2m} \\ &\leq (2 + 2w_1)\Pr[v \in OPT(G)] \\ &\leq (2 + 2w_n)\Pr[v \in OPT(G)] \end{aligned} \tag{17.8.5.5}$$

We now proceed to estimate $\Pr[v \in OPT(G)]$.

$$\begin{aligned}\Pr[v \in OPT(G)] &= \sum_{u \in OPT(G)} \Pr[u \text{ is picked}] \\ &= \sum_{u \in G} I[u \in OPT(G)] \times \Pr[u \text{ is picked}] \\ &\leq \sum_{u \in G} [|OPT(G)| - |OPT(G - \{u\})|] \Pr[u \text{ is picked}] \\ &= OPT(G) - E_v[OPT(G - \{v\})] \end{aligned} \tag{17.8.5.6}$$

where $I[u \in OPT(G)]$ is the indicator random variable which takes a value 1 if $u \in OPT(G)$ and zero otherwise. In the inequality above we used the fact that $I[u \in OPT(G)] \leq |OPT(G)| - |OPT(G - \{u\})|$. This is so because, whenever $u \in OPT(G)$, $|OPT(G)| - |OPT(G - \{u\})| = 1$. Combining (17.8.5.2), (17.8.5.5) and (17.8.5.6), and using the induction hypothesis, we have,

$$\begin{aligned}E[ALG(G)] &\leq (2 + 2w_n)[|OPT(G)| - E_v[|OPT(G - \{v\})|]] \\ &\quad + (2 + 2 \underset{i<n}{avg}\ w_i) E_v[|OPT(G - \{v\})|] \\ &= (2 + 2w_n)|OPT(G)| \\ &\quad + (2 \underset{i<n}{avg}\ w_i - 2w_n) \times E_v[OPT(G - \{v\})] \end{aligned} \tag{17.8.5.7}$$

We now proceed to bound $E_v[|OPT(G - \{v\})|]$ by first noting that,

$$\Pr[v \in OPT(G)] \geq \frac{|OPT(G)|w_1 + m}{nw_1 + 2m} \geq \frac{|OPT(G)|}{n}$$

where the last inequality follows from the fact that $\frac{a+b}{c+d} \geq min\{a/c, b/d\}$ and that $\frac{|OPT(G)|}{n} < 1/2$. Thus,

$$\begin{aligned} E_v[|OPT(G - \{v\})|] &\leq \Pr[v \in OPT(G)](|OPT(G)| - 1) + \Pr[v \notin OPT(G)](|OPT(G)|) \\ &= |OPT(G)| - \Pr[v \in OPT(G)] \\ &\leq |OPT(G)| - \frac{|OPT(G)|}{n} \end{aligned} \tag{17.8.5.8}$$

The first inequality above has the same reasoning as the one given for indicator random variable. Combining (17.8.5.7) and (17.8.5.8), we have,

$$\begin{aligned} E[(ALG(G)] &\leq (2 + 2w_n)|OPT(G)| + (2 \underset{i<n}{avg}\ w_i - 2w_n) \times (1 - 1/n) \times |OPT(G)| \\ &= (2 + 2 \underset{i \leq n}{avg}\ w_i) \times |OPT(G)| \end{aligned}$$

■

References

[1] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private approximation algorithms. *CoRR*, abs/0903.4510, 2009.

[2] Shai Halevi, Robert Krauthgamer, Eyal Kushilevitz, and Kobbi Nissim. Private approximation of np-hard functions. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 550–559, New York, NY, USA, 2001. ACM.

[3] L. Pitt. A simple probabilistic approximation algorithm for vertex cover. Technical report, Yale University, 1985.