Differentially Private Approximation Algorithms

Anupam Gupta Katrina Ligett Frank McSherry Aaron Roth Kunal Talwar

November 12, 2009

Abstract

Consider the following problem: given a metric space, some of whose points are "clients", open a set of at most k facilities to minimize the average distance from the clients to these facilities. This is just the well-studied k-median problem, for which many approximation algorithms and hardness results are known. Note that the objective function encourages opening facilities in areas where there are many clients, and given a solution, it is often possible to get a good idea of where the clients are located. However, this poses the following quandary: what if the identity of the clients is sensitive information that we would like to keep private? *Is it even possible to design good algorithms for this problem that preserve the privacy of the clients?*

In this paper, we initiate a systematic study of algorithms for discrete optimization problems in the framework of differential privacy (which formalizes the idea of protecting the privacy of individual input elements). We show that many such problems indeed have good approximation algorithms that preserve differential privacy; this is even in cases where it is impossible to preserve cryptographic definitions of privacy while computing any non-trivial approximation to even the *value* of an optimal solution, let alone the entire solution.

Apart from the k-median problem, we consider the problems of vertex and set cover, min-cut, k-median, facility location, and Steiner tree, and give approximation algorithms and lower bounds for these problems. We also consider the recently introduced submodular maximization problem, "Combinatorial Public Projects" (CPP), shown by Papadimitriou et al. [PSS08] to be inapproximable to subpolynomial multiplicative factors by any efficient and *truthful* algorithm. We give a differentially private (and hence approximately truthful) algorithm that achieves a logarithmic additive approximation.

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 providing a list of acceptable sites).

The reader may immediately recognize that (ignoring other constraints that may arise) the above problems can be simplistically modeled as instances of well-known combinatorial optimization problems: respectively the minimum cut problem, the k-median problem, and the set cover problem. While the min cut problem is known to be efficiently solvable, the k-median problem is NP hard but admits an efficient constant factor approximation. Similarly, the set cover problem is NP hard and has an efficient logarithmic approximation algorithm. Moreover, good heuristics have been designed for these problems, and hence the above problems may be considered well-studied and solved.

However, in the above scenarios and in many others, the input data (friendship relations, medical history, agents' locations) consists of sensitive information about individuals. Often customer expectations, legal obligations, security concerns, or a desire to gather better quality data necessitate that confidentiality of the private input data is guaranteed. This privacy constraint (formalized below), ignored by the modeling above, is often an important design goal and hence one may prefer an private algorithm that gives slightly suboptimal solutions to a non-private optimal algorithm. Given that the most benign (or even the most beneficial) of actions possibly leaks sensitive information, how should we design algorithms for the above problems? What are the fundamental trade-offs between the utility of these algorithms and the privacy guarantees they give us?

In this paper we initiate a systematic study of the problem of designing algorithms for combinatorial optimization problems under the constraint of *differential privacy*. Informally, the differential privacy definition requires the distribution of outcomes of the computation to not change significantly (in a precise sense) when one individual changes her input data. This is a very strong privacy guarantee: no matter what auxiliary information an adversary may have, it will not be able to learn anything significant about any individual that it couldn't have learnt were the individual not to participate in the database at all. While this guarantees privacy of an individual's sensitive data, it nevertheless allows the computation to respond when many individuals change their data, as any useful computation must do.

This work explores the challenges of producing combinatorial objects by differentially private computation. One interesting feature of these problems is the interplay between efficiency and privacy in imposing approximation constraints. The exponential mechanism [MT07] is a natural approach to attack such search problems. However, as we shall see, while it gives good utility guarantees for some problems such as k-median, it may lead to solutions very far from optimal on many others like minimum cut. A second hurdle to applying the exponential mechanism is its computational inefficiency when the set of candidate outcomes is exponentially large, which is usually the case for combinatorial optimization problems. Both these factors force us to design new algorithms—algorithms that efficiently output good solutions while guaranteeing differential privacy.

1.1 Our Results

We first look at the minimum cut problem. Given a graph $G = (V, E)$, the goal is to partition the vertices into two sets so as to minimize the number of edges going from one set to the other. We consider the edges of the graph as the sensitive inputs and the goal is to output a partition that does not reveal the presence or absence of individual edges. It is not hard to see that we cannot expect to output the minimum cut while guaranteeing privacy. Previous results on differentially private computation [DMNS06] show that one can solve the decision problem well, and output the *value* of the minimum cut up to an additive error of $O(\frac{1}{\epsilon})$ while guaranteeing ϵ-differential privacy.

Our results work for the search problem where a cut itself must be output: for this case, we show that an additive error of $\Omega(\frac{\log n}{\epsilon})$ is necessary to guarantee ϵ-differential privacy. This and other lower bounds in this work are

information theoretic bounds that derive from the privacy constraint and hold irrespective of computational constraints. We then give an (inefficient) differentially-private algorithm that matches this lower bound. Finally, we present a polynomial-time variant of the algorithm which gives the same error bound and satisfies a slight relaxation of differential privacy.

For the k-median problem, we treat the clients' locations as their sensitive information. Once again, we show that any private algorithm must incur a certain additive error, and give an inefficient algorithm that matches the lower bound. We cannot however expect to design a polynomial time algorithm with the same guarantees, as the problem itself is APX-hard even in the absence of privacy constraints. Hence, we instead give an algorithm with both multiplicative and additive overheads that satisfies differential privacy.

Vertex Cover and New Challenges. We next turn our attention to covering problems such as the Vertex Cover problem, in which we must select a small set of vertices that cover an input set of (sensitive) edges. Previous work on this problem had investigated the alternate notion of "functional" privacy, and had concluded that privately approximating even the *value* of the optimal vertex cover within a factor of $n^{1-\xi}$ is not possible under standard complexity assumptions [HKKN01]. Our use of differential privacy allows us to bypass these lower bounds and report a 2-approximation to the value of the optimal cover (with a small *additive* error) in the unweighted case.

What if we want to privately output a vertex cover instead of just its value? Interestingly, covering problems differ in one crucial aspect from the k-median and minimum cut problems: the (private) data imposes hard constraints that must be satisfied by a solution. Put differently, while private data only influences the *objective function* in the k-median problem, the data determines the *constraints* defining the set of feasible solutions in the case of the vertex cover problem. And this hard covering constraint make it impossible to output an explicit yet small vertex cover while preserving privacy: any differentially private algorithm constrained to always output an explicit vertex cover must output $n-1$ vertices, as leaving out two vertices u and v leaks the information that the edge (u, v) is certainly not present, violating differential privacy.

How can we get around this hurdle? We do so by allowing the algorithm to output an orientation for *each of the* $\binom{|V|}{2}$ *possible edges*; the orientation implicitly defines a vertex cover for any subset of edges, without needing to reveal the subset itself. In other words, instead of explicitly outputting a feasible solution, we output a variable for each possible constraint that can be used to satisfy this constraint. In the undercover agent example above, the output enables each agent to know which drop-off location to use. At the same time, it ensures that the agent cannot from this information infer the presence or absence of other agents. The privacy guarantee we give is actually significantly stronger: an arbitrary set of agents can collude and still not be able to infer the location of any other agent from the algorithm's output.

Having resolved the representation issue, we can now turn to the computational question: *given a set of edges E, how do we find an orientation efficiently but privately?* How can we use enough information about the edge set to give us a good approximation, but not so much as to violate privacy? We give an ϵ-differentially private randomized algorithm that outputs an orientation whose expected cost is at most $(2 + 16/\epsilon)$ times the optimum. For the weighted version of the problem, the approximation guarantee weakens slightly to $16(1 + 1/\epsilon)$. In both cases, as the privacy guarantees strengthen (with decreasing ϵ), the approximation guarantees weaken proportionally—this loss is essential, as we show by a lower bound proving that the $1/\epsilon$ loss is natural and necessary, regardless of computational considerations. We give similar upper and lower bounds for Set Cover and uncapacitated facility location problems.

Differential Privacy versus Truthfulness. Our results have potential implications to non-privacy related problems as well, as we discuss in the context of the recent *Combinatorial Public Project* problem, a special case of submodular maximization introduced by Papadimitriou et al. [PSS08]. They showed that the problem can admit either truthfulness or efficient approximation, but not both simultaneously. In contrast to this negative result, we show that under differential privacy (which can be interpreted as an approximate but robust alternative to truthfulness) we can achieve the same approximation factor as the best non-truthful algorithm, plus an additive logarithmic loss.

Finally, we prove a private boosting lemma. Randomized algorithms usually come with guarantees on their performance in expectation, with the understanding that independently repeating the random experiment and picking the best among the solutions can provide similar guarantees with high probability. When working with sensitive inputs, this is not a feasible approach: while each trial may leak a very small amount of information

about an individual, publishing results of repeated trials may reveal individual data. We show how to take a private algorithm that gives bounds in expectation and convert it into one that privately gives a high probability performance guarantee. Moreover, if the original algorithm is polynomial time, then so is the modified one.

Table 1 summarizes the bounds we prove in this paper. For each problem, from left to right, it reports the best known non-private approximation guarantees, our best efficient ϵ-differentially private algorithms, and in each case matching upper and lower bounds for inefficient ϵ-differentially private algorithms. For a few of the efficient algorithms (marked with a †) the guarantees are only for an approximate form of differential privacy, incorporating a failure probability δ, and scaling the effective value of ϵ up by $\ln(1/\delta)$.

	Non-private	Efficient Algorithms	Information Theoretic
Vertex Cover	$2 \times \mathsf{OPT}$ [Pit85]	$(2 + 16/\epsilon) \times \mathsf{OPT}$	$\Theta(1/\epsilon) \times \mathsf{OPT}$
Wtd. Vertex Cover	$2 \times \mathsf{OPT}$ [Hoc82]	$(16 + 16/\epsilon) \times \mathsf{OPT}$	$\Theta(1/\epsilon) \times \mathsf{OPT}$
Set Cover	$\ln n \times \mathsf{OPT}$ [Joh74]	$O(\ln n + \ln m/\epsilon) \times \mathsf{OPT}$	$\Theta(\ln m/\epsilon) \times \mathsf{OPT}$
Wtd. Set Cover	$\ln n \times \mathsf{OPT}$ [Chv79]	$O(\ln n(\ln m + \ln\ln n)/\epsilon) \times \mathsf{OPT}$ †	$\Theta(\ln m/\epsilon) \times \mathsf{OPT}$
Min Cut	OPT [FF56]	$\mathsf{OPT} + O(\ln n/\epsilon)$ †	$\mathsf{OPT} + \Theta(\ln n/\epsilon)$
CPPP	$(1 - 1/e) \times \mathsf{OPT}$ [NWF78]	$(1 - 1/e) \times \mathsf{OPT} - O(k \ln m/\epsilon)$	$\mathsf{OPT} - \Theta(k \ln(m/k)/\epsilon)$
k-Median	$(3 + \gamma) \times \mathsf{OPT}$ [AGK+04]	$6 \times \mathsf{OPT} + O(k^2 \ln^2 n/\epsilon)$	$\mathsf{OPT} + \Theta(k \ln(n/k)/\epsilon)$

Table 1: Summary of Results

Secure Function Evaluation vs. Differential Privacy

Prior work on Secure Function Evaluation (SFE) tells us that in fact the minimum cut in a graph can be computed in a distributed fashion in such a way that computations *reveals nothing that cannot be learnt from the output of the computation*. While this is a strong form of a privacy guarantee, it may be unsatisfying to an individual whose private data can be inferred from the privately computed output. Indeed, it is not hard to come up with instances where an attacker with some limited auxiliary information can infer the presence or absence of specific edges from local information about the minimum cut in the graph. By relaxing the whole input privacy requirement of SFE, differential privacy is able to provide unconditional per element privacy, which SFE need not provide if the output itself discloses properties of input.

1.2 Related Work

Differential privacy is a relatively recent privacy definition (e.g., see [DN04, CDM+05, DMNS06, Dwo06, NRS07, BCD+07, BLR08, KLN+08, Smi08], and see [Dwo08] for an excellent survey), that tries to capture the intuition of individual privacy. Many algorithms in this framework have focused on measurement, statistics, and learning tasks applied to statistical data sets, rather than on processing and producing combinatorial objects. One exception to this is the Exponential Mechanism of [MT07] which allows the selection from a set of discrete alternatives.

The problem of private approximation was introduced by Feigenbaum et al. [FIM+06], where the constraint of *functional privacy* was used to require that two inputs with the same output value (e.g. the size of an optimal vertex cover) must produce the same value under the approximation algorithm. With this constraint, Halevi et al. [HKKN01] show that approximating the value of vertex cover to within $n^{1-\xi}$ is as hard as computing the value itself, for any constant ξ. These hardness results were extended to *search* problems by Beimel et al. [BCNW06], where the constraint is relaxed to only equate those inputs whose sets of optimal solutions are identical. These results were extended and strengthened by Beimel et al. [BHN07, BMNW07].

Nonetheless, Feigenbaum et al. [FIM+06] and others show a number of positive approximation results under versions of the functional privacy model. Halevi et al. [HKKN01] provide positive results in the function privacy setting when the algorithm is permitted to leak few bits (each equivalence class of input need not produce identical output, but must be one of at most 2^b possible outcomes). Indyk and Woodruff also give some positive results for the approximation of ℓ_2 distance and a nearest neighbor problem [IW06]. However, the privacy guarantee in functional privacy is similar to SFE and does not protect sensitive data that can be inferred from the output.

2 Definitions

Differential privacy is a privacy definition for computations run against sensitive input data sets. Its requirement, informally, is that the computation behaves nearly identically on two input data sets that are nearly identical; the probability of any outcome must not increase by more than a small constant factor when the input set is altered by a single element. Formally,

Definition 2.1. *We say 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 $S \subseteq Range(M)$,*

$$\mathbf{Pr}[M(A) \in S] \quad \leq \quad \exp(\epsilon) \times \mathbf{Pr}[M(B) \in S] \; . \tag{2.1}$$

The definition has several appealing properties from a privacy perspective. One that is most important for us is that arbitrary sequences of differentially private computations are also differentially private, with an ϵ parameter equal to the sum of those comprising the sequence. This is true even when subsequent computations can depend on and incorporate the results of prior differentially private computations [DKM+06], allowing us to repeat differentially private steps to improve solutions.

2.1 Approximate Differential Privacy

One relaxation of differential privacy [DKM+06] that occasionally proves easier to satisfy allows a small additive term in the bound:

Definition 2.2. *We say a randomized computation M has* δ-approximate ϵ-differential privacy *if for any two input sets A and B with symmetric difference one, and for any set of outcomes $S \subseteq Range(M)$,*

$$\mathbf{Pr}[M(A) \in S] \quad \leq \quad \exp(\epsilon) \times \mathbf{Pr}[M(B) \in S] + \delta \; . \tag{2.2}$$

The flavor of guarantee is that although not all events have their probabilities preserved, the alteration is only for very low probability events, and is very unlikely to happen. The δ is best thought of as $1/poly(n)$ for a data set containing some subset of n candidate records.

2.2 The Exponential Mechanism

One particularly general tool that we will often use is the exponential mechanism of [MT07]. This construction allows differentially private computation over arbitrary domains and ranges, parametrized by a query function $q(A, r)$ mapping a pair of input data set A (a multiset over some domain) and candidate result r to a real valued "score". With q and a target privacy value ϵ, the mechanism selects an output with exponential bias in favor of high scoring outputs:

$$Pr[\mathcal{E}_q^\epsilon(A) = r] \quad \propto \quad \exp(\epsilon q(A, r)) \; . \tag{2.3}$$

If the query function q has the property that any two adjacent data sets have score within Δ of each other, for all possible outputs r, the mechanism provides $2\epsilon\Delta$-differential privacy. Typically, we would normalize q so that $\Delta = 1$. We will be using this mechanism almost exclusively over discrete ranges, where we can derive the following simple analogue of a theorem of [MT07], that the probability of a highly suboptimal output is exponentially low:

Theorem 2.3. *The exponential mechanism, when used to select an output $r \in R$ gives $2\epsilon\Delta$-differential privacy, letting R_{OPT} be the subset of R achieving $q(A, r) = \max_r q(A, r)$, ensures that*

$$\mathbf{Pr}[q(A, \mathcal{E}_q^\epsilon(A)) < \max_r q(A, r) - \ln(|R|/|R_{\mathsf{OPT}}|)/\epsilon - t/\epsilon] \quad \leq \quad \exp(-t) \; . \tag{2.4}$$

The proof of the theorem is almost immediate: any outcome with score less than $\max_r q(A, r) - \ln(|R|/|R_{\mathsf{OPT}}|)/\epsilon - t/\epsilon$ will have normalized probability at most $\exp(-t)/|R|$; each has weight at most $\exp(\mathsf{OPT} - t)|R_{\mathsf{OPT}}|/|R|$, but is normalized by at least $|R_{\mathsf{OPT}}| \exp(\mathsf{OPT})$ from the optimal outputs. As there are at most $|R|$ such outputs their cumulative probability is at most $\exp(-t)$.

3 Private Min-Cut

Given a graph $G = (V, E)$ the minimum cut problem is to find a cut (S, S^c) so as to minimize $E(S, S^c)$. In absence of privacy constraints, this problem is efficiently solvable exactly. However, outputting an exact solution violates privacy, as we show in Section 3.1. Thus, we give an algorithm to output a cut within additive $O(\log n/\epsilon)$ edges of optimal.

The algorithm has two stages: First, given a graph G, we add edges to the graph to raise the cost of the min cut to at least $4 \ln n/\epsilon$, in a differentially private manner. Second, we deploy the exponential mechanism over all cuts in the graph, using a theorem of Karger to show that for graphs with min cut at least $4 \ln n/\epsilon$ the number of cuts within additive t of OPT increases no faster than exponentially with t. Although the exponential mechanism takes time exponential in n, we can construct a polynomial time version by considering only the polynomially many cuts within $O(\ln n/\epsilon)$ of OPT.

Algorithm 1 The Min-Cut Algorithm

1: **Input:** $G = (V, E)$,ϵ.
2: **Let** $H_0 \subset H_1, \ldots, \subset H_{\binom{n}{2}}$ be arbitrary strictly increasing sets of edges on V.
3: **Choose** index i with probability proportional to $\exp(-\epsilon|\mathsf{OPT}(G \cup H_i) - 8 \ln n/\epsilon|)$.
4: **Choose** cut C with probability proportional to $\exp(-\epsilon Cost(G \cup H_i, C))$.
5: **Output** C.

Our result relies on a result of Karger about the number of near-minimum cuts in a graph [Kar93]

Lemma 3.1 ([Kar93])**.** *For any graph G with min cut C, there are at most $n^{2\alpha}$ cuts of size at most αC.*

By enlarging the size of the min cut in $G \cup H_i$ to at least $4 \ln n/\epsilon$, we ensure that the number of cuts of value $\mathsf{OPT}(G \cup H_i) + t$ is bounded by $n^2 \exp(\epsilon t/2)$. The downweighting of the exponential mechanism will be able to counteract this growth in number and ensure that we select a good cut.

Theorem 3.2. *For any graph G, the expected cost of ALG is at most $\mathsf{OPT} + O(\ln n/\epsilon)$.*

Proof. First, we argue that the selected index i satisfies $4 \ln n/\epsilon < \mathsf{OPT}(G \cup H_i) < \mathsf{OPT}(G) + 12 \ln n/\epsilon$ with probability at least $1 - 1/n^2$. For $\mathsf{OPT} > 8 \ln n/\epsilon$, Equation 2.4 ensures that the probability of exceeding the optimal choice (H_0) by $4 \ln n/\epsilon$ is at most $1 - 1/n^2$. Likewise, for $\mathsf{OPT} < 8 \ln n/\epsilon$, there is some optimal H_i achieving min cut size $8 \ln n/\epsilon$, and the probability we end up farther away than $4 \ln n/\epsilon$ is at most $1 - 1/n^2$.

Assuming now that $\mathsf{OPT}(G \cup H_i) > 4 \ln n/\epsilon$, Karger's lemma argues that the number c_t of cuts in $G \cup H_i$ of cost at most $\mathsf{OPT}(G \cup H_i) + t$ is at most $n^2 \exp(\epsilon t/2)$. As we are assured a cut of size $\mathsf{OPT}(G \cup H_i)$ exist, each cut of size $\mathsf{OPT}(G \cup H_i) + t$ will receive probability at most $\exp(-\epsilon t)$. Put together, the probability of a cut exceeding $\mathsf{OPT}(G \cup H_i) + b$ is at most

$$\begin{aligned}
\mathbf{Pr}[Cost(G \cup H_i, C) > \mathsf{OPT}(G \cup H_i) + b] &\leq \sum_{t>b} \exp(-\epsilon t)(c_t - c_{t-1}) && (3.5)\\
&\leq (\exp(\epsilon) - 1) \sum_{t>b} \exp(-\epsilon t) c_t && (3.6)\\
&\leq (\exp(\epsilon) - 1) \sum_{t>b} \exp(-\epsilon t/2) n^2 && (3.7)
\end{aligned}$$

The sum telescopes to $\exp(-\epsilon b/2)n^2/(\exp(\epsilon/2)-1)$, and the denominator is within a constant factor of the leading factor of $(\exp(\epsilon) - 1)$, for $\epsilon < 1$. For $b = 8 \ln n/\epsilon$, this probability becomes at most $1/n^2$. □

Theorem 3.3. *The algorithm above preserves 2ϵ-differential privacy.*

Remark 3.4. *Note that the first instance of the exponential mechanism in our algorithm runs efficiently (since it is selecting from only $\binom{n}{2}$ objects), but the second instance does not. However, using Karger's algorithm we can efficiently (with high probability) generate all cuts of size at most $\mathsf{OPT} + k \ln n/\epsilon$ for any constant k. If we are willing to tolerate a polynomially small $\delta = O(1/n^k)$ to achieve (ϵ, δ)-privacy, we can select only from such a pre-generated polynomially sized set, and achieve (ϵ, δ)-differential privacy efficiently. We omit the details from this extended abstract.*

3.1 Lower Bounds

Theorem 3.5. *Any ϵ-differentially private algorithm for min-cut must incur an expected additive $\Omega(\ln n/\epsilon)$ cost over OPT, for any $\epsilon \in (\ln n/n^2, 1)$.*

Proof. Consider a random $\ln n/2\epsilon$-regular graph on n vertices. For almost all such graphs, the optimal cuts are exactly those that isolate a single vertex. At least one of these n cuts must have probability at most $1/n$ on this random regular graph. If we consider removing the $\ln n/2\epsilon$ edges incident to that node, the probability of outputting the matching cut can increase to at most $1/n^{1/2}$, leaving near certain probability that some other cut is used. But all other cuts, even with these edges removed, cost at least $\ln n/2\epsilon - 1$. This input has an isolated vertex, and so its optimal cost is zero. □

4 Private k-Median

We next consider a private version of the metric k-median problem: There is a pre-specified set of points V and a metric on them, $d : V \times V \to \mathbb{R}$. There is a (private) set of demand points $D \subseteq V$. We wish to select a set of medians $F \subset V$ with $|F| = k$ to minimize the quantity $\mathsf{cost}(F) = \sum_{v \in D} d(v, F)$ where $d(v, F) = \min_{f \in F} d(v, f)$. Let $\Delta = \max_{u,v \in V} d(u, v)$ be the diameter of the space.

As we show in Section 4.1, any privacy-preserving algorithm for k-median must incur an additive loss of $\Omega(\Delta \cdot k \ln(n/k)/\epsilon)$, regardless of computational constraints. We observe that running the exponential mechanism to choose one of the $\binom{n}{k}$ subsets of medians gives an (computationally inefficient) additive guarantee.

Theorem 4.1. *Using the exponential mechanism to pick a set of k facilities gives an $O(\binom{n}{k}poly(n))$-time ϵ-differentially private algorithm that outputs a solution with expected cost $\mathsf{OPT} + O(k\Delta \log n/\epsilon)$.*

We next give a polynomial-time algorithm that gives a slightly worse approximation guarantee. Our algorithm is based on the local search algorithm of Arya *et al.* [AGK+04]. We start with an arbitrary set of k medians, and use the exponential mechanism to look for a (usually) improving swap. After running this local search for a suitable number of steps, we select a good solution from amongst the ones seen during the local search. The following result shows that if the current solution is far from optimal, then one can find improving swaps.

Theorem 4.2 (Arya et al. [AGK+04])**.** *For any set $F \subseteq V$ with $|F| = k$, there exists a set of k swaps $(x_1, y_1), \ldots, (x_k, y_k)$ such that $\sum_{i=1}^{k}(\mathsf{cost}(F) - \mathsf{cost}(F - \{x_i\} + \{y_i\})) \geq \mathsf{cost}(F) - 5\mathsf{OPT}$.*

Corollary 4.3. *For any set $F \subseteq V$ with $|F| = k$, there exists some swap (x, y) such that*

$$\mathsf{cost}(F) - \mathsf{cost}(F - \{x_i\} + \{y_i\}) \geq \frac{\mathsf{cost}(F) - 5\mathsf{OPT}}{k}.$$

Algorithm 2 The k-Median Algorithm

1: **Input:** V, Demand points $D \subseteq V$, k,ϵ.
2: **let** $F_1 \subset V$ arbitrarily with $|F_1| = k$, $\epsilon' \leftarrow \epsilon/(2\Delta(T+1))$.
3: **for** $i = 1$ to T **do**
4: Select $(x, y) \in F_i \times (V \setminus F_i)$ with probability proportional to $\exp(-\epsilon' \times \mathsf{cost}(F_i - \{x\} + \{y\}))$.
5: **let** $F_{i+1} \leftarrow F_i - \{x\} + \{y\}$.
6: **end for**
7: Select j from $\{1, 2, \ldots, T\}$ with probability proportional to $\exp(-\epsilon' \times \mathsf{cost}(F_j))$.
8: **output** F_j.

Theorem 4.4. *Setting $T = 6k \ln n$ and $\epsilon' = \epsilon/(2\Delta(T+1))$, the k-median algorithm provides ϵ-differential privacy and except with probability $O(1/poly(n))$ outputs a solution of cost at most $6\mathsf{OPT} + O(\Delta k^2 \log^2 n/\epsilon)$.*

Proof. We first prove the privacy. Since the cost function has sensitivity Δ, Step 4 of the algorithm preserves $2\epsilon'\Delta$ differential privacy. Since Step 4 is run at most T times and privacy composes additively, outputting all of the T candidate solutions would give us $(2\epsilon'\Delta T)$ differential privacy. Picking out a good solution from the T candidates costs us another $2\epsilon'\Delta$, leading to the stated privacy guarantee.

We next show the approximation guarantee. By Corollary 4.3, so long as $\mathsf{cost}(F_i) \geq 6\mathsf{OPT}$, there exists a swap (x, y) that reduces the cost by at least $\mathsf{cost}(F_i)/6k$. As there are only n^2 possible swaps, the exponential mechanism ensures through (2.4) that we are within additive $4 \ln n/\epsilon'$ with probability at least $1 - 1/n^2$. When $\mathsf{cost}(F_i) \geq 6\mathsf{OPT} + 24k \ln n/\epsilon'$, with probability $1 - 1/n^2$ we have $\mathsf{cost}(F_{i+1}) \leq (1 - 1/6k) \times \mathsf{cost}(F_i)$.

This multiplicative decrease by $(1 - 1/6k)$ applies for as long as $\mathsf{cost}(F_i) \geq 6\mathsf{OPT} + 24k \ln n/\epsilon'$. Since $\mathsf{cost}(F_0) \leq n\Delta$, and $n\Delta(1 - 1/6k)^T \leq \Delta \leq 24k \ln n/\epsilon'$, there must exist an $i < T$ such that $\mathsf{cost}(F_i) \leq 6\mathsf{OPT} + 24k \ln n/\epsilon'$, with probability at least $(1 - T/n^2)$.

Finally, by applying the exponential mechanism again in the final stage, we select from the F_i scoring within an additive $4 \ln n/\epsilon'$ of the optimal visited F_i with probability at least $1 - 1/n^2$, again by (2.4). Plugging in the value of ϵ', we get the desired result. Increasing the constants in the additive term can drive the probability of failure to an arbitrarily small polynomial. □

4.1 k-Median Lower Bound

Theorem 4.5. *Any ϵ-differentially private algorithm for the k-median problem must incur cost $\mathsf{OPT} + \Omega(\Delta \cdot k \ln(n/k)/\epsilon)$ on some inputs.*

Proof. Consider a point set $V = [n] \times [L]$ of nL points, with $L = \ln(n/k)/10\epsilon$, and a distance function $d((i, j), (i', j')) = \Delta$ whenever $i \neq i'$ and $d((i, j), (i, j')) = 0$. Let M be a differentially private algorithm that takes a subset $D \subseteq V$ and outputs a set of k locations, for some $k < \frac{n}{4}$. Given the nature of the metric space, we assume that M outputs a k-subset of $[n]$. For a set $A \subseteq [n]$, let $D_A = A \times [L]$. Let A be a size-k subset of V chosen at random.

We claim that that $\mathbb{E}_{A,M}[|M(D_A) \cap A|] \leq \frac{k}{2}$ for any ϵ-differentially private algorithm M. Before we prove this claim, note that it implies the expected cost of $M(D_A)$ is $\frac{k}{2} \times \Delta L$, which proves the claim since $\mathsf{OPT} = 0$.

Now to prove the claim: define $\phi := \frac{1}{k}\mathbb{E}_{A,M}[|A \cap M(D_A)|]$. We can rewrite

$$k \cdot \phi = \mathbb{E}_{A,M}[|A \cap M(D_A)|] = k \cdot \mathbb{E}_{i \in [n]}\mathbb{E}_{A \setminus \{i\},M}[\mathbf{1}_{i \in M(D_A)}] \tag{4.8}$$

Now changing A to $A' := A \setminus \{i\} + \{i'\}$ for some random i' requires altering at most $2L$ elements in $D_{A'}$, which by the differential privacy guarantee should change the probability of the output by at most $e^{2\epsilon L} = (n/k)^{1/5}$. Hence

$$\mathbb{E}_{i \in [n]}\mathbb{E}_{A',M}[\mathbf{1}_{i \in M(D_{A'})}] \geq \phi \cdot (k/n)^{1/5}. \tag{4.9}$$

But the expression on the left is just k/n, since there at at most k medians. Hence $\phi \leq (k/n)^{4/5} \leq 1/2$, which proves the claim. □

Corollary 4.6. *Any 1-differentially private algorithm for uniform facility location that outputs the set of chosen facilities must have approximation ratio $\Omega(\sqrt{n})$.*

Proof. We consider instances defined on the uniform metric on n points, with $d(u, v) = 1$ for all u, v, and facility opening cost $f = \frac{1}{\sqrt{n}}$. Consider a 1-differentially private mechanism M when run on a randomly chosen subset A of size $k = \sqrt{n}$. Since OPT is $kf = 1$ for these instances, any $o(\sqrt{n})$-approximation must select at least $\frac{k}{2}$ locations from A in expectation. By an argument analogous to the above theorem, it follows that any differentially private M must output $n/20$ of the locations in expectation. This leads to a facility opening cost of $\Omega(\sqrt{n})$. □

5 Vertex Cover

We now turn to the problem of (unweighted) vertex cover, in which we must select a set of vertices of minimal size so that every edge in the graph is incident to at least one vertex. In the privacy-preserving version of the problem, the private information we wish to conceal is the presence of absence of each edge.

Approximating the Vertex Cover *Size*. As mentioned earlier, the problem of approximating the vertex cover size was used as the polynomial inapproximability result of [HKKN01, BCNW06] under the constraint of *functional* privacy. Despite this, any approaches to approximating the *size* of the optimal vertex cover are immediately compatible with differential privacy. E.g., the size of a maximum matching times two is a 2-approximation, and only changes in value by at most two with the presence or absence of a single edge. Hence, this value plus

Laplace($2/\epsilon$) noise provides ϵ-differential privacy [DMNS06]. (It is important that we use maximum rather than just maximal matchings, since the size of the latter is not uniquely determined by the graph, and the presence or absence of an edge may dramatically alter the size of the solution.)

Interestingly, for *weighted* vertex cover with maximum weight $w_{\max}$ (studied in Section 5.3), we have to add in Lap($w_{\max}/\epsilon$) noise to privately estimate the weight of the optimal solution, which can be much larger than OPT itself. The mechanism in Section 5.3 avoids this barrier by only implicitly outputting the solution and gives us a $O(1/\epsilon)$ multiplicative approximation with ϵ-differential privacy.

Private approximation algorithms for the Vertex Cover *search* problem. As noted in the introduction, any differentially private algorithm for vertex cover that outputs an explicit vertex cover (a subset of the n vertices) must output a cover of size at least $n-1$ with probability 1 on any input, an essentially useless result. This lower bound exists because differential privacy requires that any output that is possible under some input is possible under all inputs. In order to address this challenge, we change our target output slightly, and allow our algorithm to output, rather than a subset of the vertices, an orientation of the edges (including non-existent edges) so as to minimize the size of the set of destinations of existing edges. Such an orientation may be viewed as a privacy-preserving set of instructions that allows for the construction of a good vertex cover in a distributed manner: in the case of the undercover agents mentioned in the introduction, the complete set of active dropoff sites (nodes) is not revealed to the agents, but an orientation on the edges tells each agent which dropoff site to use, if she is indeed an active agent.

5.1 Approximating Solutions: Unweighted Vertex Cover

Many algorithms exist that provide a factor-two approximation to the unweighted vertex cover, such as selecting all endpoints of a maximal matching, or repeatedly adding a random endpoint of a random uncovered edge. Although some are randomized, none that we know of have outputs that can be made to satisfy differential privacy. One simple non-private 2-approximation to vertex cover [Pit85] repeatedly selects an uncovered edge uniformly at random, and includes a random endpoint of the edge. We can view the process, equivalently, as selecting a vertex at random with probability proportional to its uncovered degree. We will take this formulation and mix in a uniform distribution over the vertices, using a weight that will grow as the number of remaining vertices decreases. As mentioned above, our goal is to output an orientation for each edge (regardless of whether it is actually present). The way we will do this will be to output a permutation of the vertex set; we will infer from this that an edge is oriented toward whichever of its endpoints appears first in the permutation.

Let us start from $G_1 = G$, and let G_i be the graph with $n-i+1$ vertices remaining. We will write $d_v(G)$ for the degree of vertex v in graph G. The algorithm ALG in step i chooses from the $n-i+1$ vertices of G_i with probability proportional to $d_v(G_i)+w_i$, for an appropriate sequence $\langle w_i \rangle$. Taking $w_i = (4/\epsilon) \times (n/(n-i+1))^{1/2}$ provides ϵ-differential privacy and a $(2+16/\epsilon)$ approximation factor, the proof of which will follow from the forthcoming Theorem 5.1 and Theorem 5.2.

Algorithm 3 Unweighted Vertex Cover

1: **let** $n \leftarrow |V|$, $V_1 \leftarrow V, E_1 \leftarrow E$.
2: **for** $i = 1, 2, \ldots, n$ **do**
3: **let** $w_i \leftarrow (4/\epsilon) \times \sqrt{n/(n-i+1)}$.
4: **pick** a vertex $v \in V_i$ with probability proportional to $d_{E_i}(v) + w_i$.
5: **output** v. **let** $V_{i+1} \leftarrow V_i \setminus \{v\}$, $E_{i+1} \leftarrow E_i \setminus (\{v\} \times V_i)$.
6: **end for**

Theorem 5.1 (Privacy)**.** *ALG's differential privacy guarantee is* $\max\{1/w_1, \sum_i 2/iw_i\} \leq \varepsilon$ *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 the 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.

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

When A and B differ in exactly one 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 by exactly one, and after this term $m_i(A) = m_i(B)$. The number of nodes is always equal, of course. Letting j be the index in π of the first endpoint of the edge in difference, we can cancel all terms after j and rewrite

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

An edge may have arrived in A, in which case $m_i(A) = m_i(B) + 1$ for all $i \le j$, and each term in the product is at most one; moreover, $d_j(A) = d_j(B) + 1$, and hence the leading term is at most $1 + 1/w_j < \exp(1/w_1)$, which is bounded by $\exp(\epsilon/2)$.

Alternately, an edge may have departed from A, in which case the lead term is no more than one, but each term in the product exceeds one and their product must now be bounded. Note that $m_i(A)+1 = m_i(B)$ for all relevant i, and that by ignoring all other edges we only make the product larger. Simplifying, and using $1 + x \le \exp(x)$, we see

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

The w_i are chosen so that $\sum_i 2/(n-i+1)w_i = (\varepsilon/\sqrt{n}) \sum_i 1/2\sqrt{i}$ is at most ε.

□

Theorem 5.2 (Accuracy)**.** *For all G, $\mathbb{E}[ALG(G)] \le (2 + 2\, avg_{i \le n}\, w_i) \times |OPT(G)| \le (2 + 16/\epsilon)|\mathsf{OPT}(G)|$.*

Proof. Let $OPT(G)$ denote an arbitrary optimal solution to the vertex cover problem on G. The proof is inductive, on the size n of G. For G with $|OPT(G)| > n/2$, the theorem holds. For G with $|OPT(G)| \le n/2$, the expected cost of the algorithm is the probability that the chosen vertex v is incident to an edge, plus the expected cost of $ALG(G \setminus v)$.

$$\mathbb{E}[ALG(G)] \;=\; \mathbf{Pr}[v \text{ incident on edge}] + \mathbb{E}_v[\mathbb{E}[ALG(G \setminus v)]] .$$

We will bound the second term using the inductive hypothesis. To bound the first term, the probability that v is chosen incident to an edge is at most $(2mw_n + 2m)/(nw_n + 2m)$, as there are at most $2m$ vertices incident to edges. On the other hand, the probability that we pick a vertex in $OPT(G)$ is at least $(|OPT(G)|w_n + m)/(nw_n + 2m)$. Since $|OPT(G)|$ is non-negative, we conclude that

$$\mathbf{Pr}[v \text{ incident on edge}] \le (2 + 2w_n)(m/(nw_n + 2m)) \le (2 + 2w_n)\mathbf{Pr}[v \in OPT(G)]$$

Since $\mathbf{1}[v \in OPT(G)] \le |OPT(G)| - |OPT(G \setminus v)|$, and using the inductive hypothesis, we get

$$\begin{aligned} \mathbb{E}[ALG(G)] &\le (2 + 2w_n) \times (|OPT(G)| - \mathbb{E}_v[|OPT(G \setminus v)|]) + (2 + 2 \operatorname*{avg}_{i<n} w_i) \times \mathbb{E}_v[|OPT(G \setminus v)|] \\ &= (2 + 2w_n) \times |OPT(G)| + (2 \operatorname*{avg}_{i<n} w_i - 2w_n) \times \mathbb{E}_v[|OPT(G \setminus v)|] \end{aligned}$$

The probability that v is from an optimal vertex cover is at least $(|OPT(G)|w_i + m)/(nw_i + 2m)$, as mentioned above, and (using $(a+b)/(c+d) \ge \min\{a/c, b/d\}$) is at least $\min\{|OPT(G)|/n, 1/2\} = |OPT(G)|/n$, since $|OPT(G)| < n/2$ by assumption. Thus $\mathbb{E}[|OPT(G \setminus v)|]$ is bounded above by $(1 - 1/n) \times |OPT(G)|$, giving

$$\mathbb{E}[ALG(G)] \;\le\; (2 + 2w_n) \times |OPT(G)| + (2 \operatorname*{avg}_{i<n} w_i - 2w_n) \times (1 - 1/n) \times |OPT(G)| .$$

Simplification yields the claimed results, and instantiating w_i completes the proof. □

Hallucinated Edges. Here is a slightly different way to implement the intuition behind the above algorithm: imagine adding $O(1/\epsilon)$ "hallucinated" edges to each vertex (the other endpoints of these hallucinated edges being fresh "hallucinated" vertices), and then sampling vertices without replacement proportional to these altered degrees. However, once (say) $n/2$ vertices have been sampled, output the remaining vertices in random order. This view will be useful to keep in mind for the weighted vertex cover proof. (A formal analysis of this algorithm is in Appendix A.)

5.2 Vertex Cover Lower Bounds

Theorem 5.3. *Any algorithm for the vertex cover problem that prescribes edge-orientations with ϵ-differential privacy must have an $\Omega(1/\epsilon)$ approximation guarantee, for any $\epsilon \in (\frac{1}{n}, 1]$.*

Proof. Let $V = \{1, 2, \ldots, \lceil \frac{1}{2\epsilon} \rceil\}$, and let M be an ϵ-differentially private algorithm that takes as input a private set E of edges, and outputs an orientation $M_E : V \times V \to V$, with $M_E(u, v) \in \{u, v\}$ indicating to the edge which endpoint to use. Picking two distinct vertices $u \neq v$ uniformly at random (and equating (u, v) with (v, u)), we have by symmetry:

$$\mathbf{Pr}_{u,v}[M_\emptyset((u, v)) \neq u] = \tfrac{1}{2}.$$

Let $\star_u = (V, \{u\} \times (V \setminus \{u\}))$ be the star graph rooted at u. Since $\star_u$ and $\emptyset$ differ in at most $\frac{1}{2\epsilon} - 1 < \frac{1}{\epsilon}$ edges and M satisfies ϵ-differential privacy, we conclude that

$$\mathbf{Pr}_{u,v}[M_{\star_u}((u, v)) \neq u] \geq \tfrac{1}{2e}.$$

Thus the expected cost of M when input a uniformly random $\star_u$ is at least $\frac{1}{2e} \times \lceil \frac{1}{2\epsilon} \rceil$, while $\mathsf{OPT}(\star_u)$ is 1. We can repeat this pattern arbitrarily, picking a random star from each group of $1/\epsilon$ vertices; this results in graphs with arbitrarily large vertex covers where M incurs cost $1/\epsilon$ times the cost. □

5.3 Weighted Vertex Cover

In the weighted vertex cover problem, each vertex V is assigned a weight $w(v)$, and the cost of any vertex cover is the sum of the weights of the participating vertices. One can extend the unweighted 2-approximation that draws vertices at random with probability proportional to their uncovered degree to a weighted 2-approximation by drawing vertices with probability proportional to their uncovered degree divided by their weight. The differentially private analog of this algorithm essentially draws vertices with probability proportional to $1/\epsilon$ plus their degree, all divided by the weight of the vertex; the algorithm we present here is based on this idea.

Define the *score* of a vertex to be $s(v) = 1/w(v)$. Our algorithm involves hallucinating edges: to each vertex, we add in $1/\epsilon$ hallucinated edges, the other endpoints of which are imaginary vertices, whose weight is considered to be ∞ (and hence has zero score). The score of an edge $e = (u, v)$ is defined to be $s(e) = s(u) + s(v)$; hence the score of a fake edge f incident on u is $s(f) = s(u)$, since its other (imaginary) endpoint has infinite weight and zero score. We will draw edges with probability proportional to their score, and then select an endpoint to output with probability proportional to its score. In addition, once a substantial number of vertices of at least a particular weight have been output, we will output the rest of those vertices.

Assume the minimum vertex weight is 1 and the maximum is 2^J. For simplicity, we round the weight of each vertex up to a power of 2, at a potential loss of a factor of two in the approximation. Define the j^{th} weight class V_j to be the set of vertices of weight 2^j. In addition, we will assume that $|V_j| = |V_{j+1}|$ for all weight classes. In order to achieve this, we hallucinate additional fake vertices. We will never actually output a hallucinated vertex. Let N_j denote $|V_j|$.

Algorithm 4 Weighted Vertex Cover

1: **while** not all vertices have been output **do**
2: **pick** an uncovered (real or hallucinated) edge $e = (u, v)$ with probability proportional to $s(e)$.
3: **output** endpoint $u \in e$ with probability proportional to $s(u)$.
4: **while** there exists some weight class V_j such that the number of nodes of class j or higher that we've output is at least $N_j/2 = |V_j|/2$ **do**
5: **pick** the smallest such value of j
6: **output** ("dump") all remaining vertices in V_j in random order.
7: **end while**
8: **end while**

We imagine the i^{th} iteration of the outer loop of the algorithm as happening at *time* i; note that one vertex is output in Step 3, whereas multiple vertices might be output in Step 6. Let $\widetilde{n}_i$ be the sum of the scores of all real vertices not output before time i, and $\widetilde{m}_i$ be the sum of the scores of all real edges not covered before time i.

5.3.1 Privacy Analysis

Theorem 5.4. *The weighted vertex cover algorithm preserves $O(\epsilon)$ differential privacy.*

Proof. Consider some potential output π of the private vertex cover algorithm, and two weighted vertex cover instances A and B that are identical except for one edge $\mathbf{e} = (p, q)$. Let p appear before q in the permutation π; since the vertex sets are the same, if the outputs of both A and B are π, then p will be output at the same time t in both executions. Let v_t be the vertex output in Step 3 at time t in such an execution; note that either $p = v_t$, or p is output in Step 6 after v_t is output.

The probability that (conditioned on the history) a surviving vertex v is output in Step 3 of the algorithm at time i is:

$$\sum_{\text{edges } e} \mathbf{Pr}[\text{pick } e] \cdot \mathbf{Pr}[\text{output } v \mid \text{pick } e] = \sum_{e \ni v} \frac{s(e)}{\widetilde{m}_i + \widetilde{n}_i/\epsilon} \cdot \frac{s(v)}{s(e)} = \frac{(d(v)+1/\epsilon)\cdot s(v)}{\widetilde{m}_i + \widetilde{n}_i/\epsilon}.$$

Since we compare the runs of the algorithm on A and B which differ only in edge $\mathbf{e}$, these will be identical after time t when $\mathbf{e}$ is covered, and hence

$$\frac{\mathbf{Pr}[M(A)=\pi]}{\mathbf{Pr}[M(B)=\pi]} = \frac{(d_A(v_t)+1/\epsilon)s(v_t)}{(d_B(v_t)+1/\epsilon)s(v_t)} \prod_{i \le t} \left(\frac{\widetilde{m}_i^B + \widetilde{n}_i/\epsilon}{\widetilde{m}_i^A + \widetilde{n}_i/\epsilon} \right).$$

Note that if the extra edge $\mathbf{e} \in A \setminus B$ then $d_A(v_t) \le d_B(v_t) + 1$ and $\widetilde{m}_i^B \le \widetilde{m}_i^A$, so the ratio of the probabilities is at most $1 + \epsilon < \exp(\epsilon)$. Otherwise, the leading term is less than 1 and $\widetilde{m}_i^B = \widetilde{m}_i^A + s(\mathbf{e})$, and we get

$$\frac{\mathbf{Pr}[M(A)=\pi]}{\mathbf{Pr}[M(B)=\pi]} \le \prod_{i \le t} \left(1 + \frac{s(\mathbf{e})}{\widetilde{n}_i/\epsilon}\right) \le \exp\left(s(\mathbf{e}) \cdot \epsilon \cdot \sum_{i \le t} \frac{1}{\widetilde{n}_i}\right).$$

Let T_j be the time steps $i \le t$ where vertices in V_j are output in π. $\frac{1}{\widetilde{n}_i}$. Letting 2^{j^*} be the weight of the lighter endpoint of edge $\mathbf{e}$, we can break the sum $\sum_{i \le t} \frac{1}{\widetilde{n}_i}$ into two pieces and analyze each separately:

$$\sum_{i \le t} \frac{1}{\widetilde{n}_i} = \sum_{j \le j^*} \sum_{i \in T_j} \frac{1}{\widetilde{n}_i} + \sum_{j > j^*} \sum_{i \in T_j} \frac{1}{\widetilde{n}_i},$$

For the first partial sum, for some $j \le j^*$, let $\sum_{i \in T_j} \frac{1}{\widetilde{n}_i} = \frac{1}{\widetilde{n}_{i_0}} + \frac{1}{\widetilde{n}_{i_1}} + \ldots + \frac{1}{\widetilde{n}_{i_\lambda}}$ such that $i_0 > i_1 > \ldots > i_\lambda$. We claim that $\widetilde{n}_{i_0} \ge 2^{-j^*} N_{j^*}/2$. Indeed, since $\mathbf{e}$ has not yet been covered, we must have output fewer than $N_{j^*}/2$ vertices from levels j^* or higher, and hence at least $N_{j^*}/2$ remaining vertices from V_{j^*} contribute to $\widetilde{n}_{i_0}$.

In each time step in T_j, at least one vertex of score 2^{-j} is output, so we have that $\widetilde{n}_{i_\ell} \ge 2^{-j^*} N_{j^*}/2 + \ell \cdot 2^{-j}$. Hence

$$\sum_{i \in T_j} \frac{1}{\widetilde{n}_i} \le \frac{1}{2^{-j^*} N_{j^*}/2} + \frac{1}{2^{-j^*} N_{j^*}/2 + 2^{-j}} + \ldots + \frac{1}{2^{-j^*} N_{j^*}/2 + N_j\, 2^{-j}}.$$

Defining $\theta = 2^{-j^*+j} \cdot N_{j^*}/2$, the expression above simplifies to

$$2^j \left(\frac{1}{\theta} + \frac{1}{\theta+1} + \ldots + \frac{1}{\theta+N_j} \right) \le 2^j \ln\left(\frac{\theta+N_j}{\theta}\right) = 2^j \ln\left(1 + \frac{N_j}{\theta}\right).$$

Now using the assumption on the size of the weight classes, we have $N_j \le N_{j^*} \implies N_j/\theta \le 2^{j^*-j+1}$, and hence $\sum_{i \in T_j} \frac{1}{\widetilde{n}_i} \le (j^* - j + 2)2^j$, for any $j \le j^*$. Finally,

$$\sum_{j \le j^*} \sum_{i \in T_j} \frac{1}{\widetilde{n}_i} \le \sum_{j \le j^*} (j^* - j + 2)2^j = O(2^{j^*}).$$

We now consider the other partial sum $\sum_{j > j^*} \sum_{i \in T_j} \frac{1}{\widetilde{n}_i}$. For any such value of i, we know that $\widetilde{n}_i \ge 2^{-j^*} N_{j^*}/2$. Moreover, there are at most $N_{j^*}/2$ times when we output a vertex from some weight class $j \ge j^*$ before we output all of V_{j^*}; hence there are at most $N_{j^*}/2$ terms in the sum, each of which is at most $\frac{1}{2^{-j^*} N_{j^*}/2}$, giving a bound of 2^{j^*} on the second partial sum. Putting the two together, we get that

$$\frac{\mathbf{Pr}[M(A) = \pi]}{\mathbf{Pr}[M(B) = \pi]} \le \exp(s(\mathbf{e}) \cdot \epsilon \cdot O(2^{j^*})) = \exp(O(\epsilon)),$$

using the fact that $s(\mathbf{e}) \le 2 \cdot 2^{-j^*}$, since the lighter endpoint of $\mathbf{e}$ had weight 2^{j^*}. ∎

5.3.2 Utility Analysis

Call a vertex v *interesting* if it is incident on a real uncovered edge when it is picked. Consider the weight class V_j: let $I_j^1 \subseteq V_j$ be the set of interesting vertices output due to Steps 3, and $I_j^2 \subseteq V_j$ be the set of interesting vertices of class j output due to Step 6. The cost incurred by the algorithm is $\sum_j 2^j(|I_j^1| + |I_j^2|)$.

Lemma 5.5. $\mathbb{E}[\sum_j 2^j |I_j^1|] \leq \frac{4(1+\varepsilon)}{\varepsilon}\mathsf{OPT}$

Proof. Every interesting vertex that our algorithm picks in Steps 3 has at least one real edge incident on it, and at most $\frac{1}{\varepsilon}$ hallucinated edges. Conditioned on selecting an interesting vertex v, the selection is due to a real edge with probability at least $1/(1+\frac{1}{\varepsilon})$. One can show that the (non-private) algorithm $\mathcal{A}$ that selects only real edges is a 2-approximation [Pit85]. On the other hand each vertex in I_j^1 can be coupled to a step of $\mathcal{A}$ with probability $\varepsilon/(1+\varepsilon)$. Since we rounded up the costs by at most a factor of two, the claim follows. □

Lemma 5.6. $\mathbb{E}[|I_j^2|] \leq 6\,\mathbb{E}[\sum_{j' \geq j} |I_{j'}^1|]$

Proof. Let t_j denote the time that class j is dumped. Recall that by (5.3.1), we pick a surviving vertex v with probability $\propto (d(v) + \frac{1}{\varepsilon}) \cdot s(v)$ at each step. This expression summed over all uninteresting vertices is $\cup_{j' \geq j} V_{j'}$ is at most $(1/\varepsilon)\sum_{j' \geq j} 2^{-j'} N_{j'} \leq 2^{-j+1} N_j/\varepsilon$. On the other hand, at each step before time t_j, all the interesting vertices in I_j^2 are available and the same expression summed over them is at least $2^{-j}|I_j^2|/\epsilon$. Thus for any $t \leq t_j$, conditioned on outputting a vertex $v_t \in \cup_{j' \geq j} V_{j'}$ in Step 3, the probability that it is interesting is at least $\frac{|I_j^2|2^{-j}/\varepsilon}{(|I_j^2|2^{-j} + 2^{1-j}N_j)/\varepsilon} \geq \frac{|I_j^2|}{3N_j}$ (using $|I_j^2| \leq N_j$). Now since we output $N_j/2$ vertices from $\cup_{j' \geq j} V_{j'}$ in Step 3 before time t_j, we conclude that $\mathbb{E}\big[\sum_{j' \geq j} |I_{j'}^1| \,\big|\, |I_j^2|\big] \geq \frac{N_j}{2} \times \frac{|I_j^2|}{3N_j} = \frac{|I_j^2|}{6}$. Taking expectations completes the proof. □

We can now compute the total cost of all the interesting vertices dumped in Steps 6 of the algorithm.

$$\mathbb{E}[\mathsf{cost}(\textstyle\bigcup_j I_j^2)] = \sum_j 2^j\,\mathbb{E}[|I_j^2|] \leq 6\,\sum_j 2^j\,\sum_{j' \geq j} \mathbb{E}[|I_{j'}^1|] \leq 6\,\sum_{j'} \mathbb{E}[|I_{j'}^1|]\,2^{j'+1} \leq 12\,\cdot \mathbb{E}[\mathsf{cost}(\textstyle\bigcup_j I_j^1)].$$

Finally, combining this calculation with Lemma 5.5, we conclude that our algorithm gives an $O(\frac{1}{\varepsilon})$ approximation to the weighted vertex cover problem.

6 Set Cover

We now turn our attention to private approximations for the Set Cover: as for vertex cover, we do not explicitly output a set cover, but instead output a permutation over the sets in the set system so that each universe element is deemed covered by the first set in this permutation that contains it. Our algorithms for set cover give the slightly weaker (ϵ, δ)-privacy guarantees.

6.1 Unweighted Set Cover

We are given a set system $(U, \mathcal{S})$ and must cover a private subset $R \subset U$. Let the cardinality of the set system be $|\mathcal{S}| = m$, and let $|U| = n$. We first observe a computationally inefficient algorithm.

Theorem 6.1. *The exponential mechanism, when used to pick a permutation of sets, runs in time $O(m!poly(n))$ and gives an $O(\log(em/\mathsf{OPT})/\epsilon)$-approximation.*

Proof. A random permutation, with probability at least $\binom{m}{\mathsf{OPT}}^{-1}$ has all the sets in OPT before any set in OPT^c. Thus the additive error is $O(\log\binom{m}{\mathsf{OPT}}/\epsilon)$. □

The rest of the section gives a computationally efficient algorithm with slightly worse guarantees: this is a modified version of the greedy algorithm, using the exponential mechanism to bias towards picking large sets.

6.1.1 Utility Analysis

At the beginning of iteration i, say there are $m_i = m - i + 1$ remaining sets and $n_i = |R_i|$ remaining elements, and define $L_i = \max_{S \in \mathcal{S}} |S \cap R_i|$, the largest number of uncovered elements covered by any set in $\mathcal{S}$. By a standard argument, any algorithm that always picks sets of size $L_i/2$ is an $O(\ln n)$ approximation algorithm.

Algorithm 5 Unweighted Set Cover

Input: Set system $(U, \mathcal{S})$, private $R \subset U$ of elements to cover, ϵ,δ.
let $i \leftarrow 1$, $R_i = R$. $\epsilon' \leftarrow \epsilon/2\ln(\frac{e}{\delta})$.
while not all sets in $\mathcal{S}$ have been output **do**
  pick a set S from $\mathcal{S}$ with probability proportional to $\exp(\epsilon'|S \cap R_i|)$.
  output set S.
  $R_{i+1} \leftarrow R_i \setminus S$, $i \leftarrow i + 1$.
end while

Theorem 6.2. *The above algorithm achieves an expected approximation ratio of* $O(\ln n + \frac{\ln m}{\epsilon'}) = O(\ln n + \frac{\ln m \ln(e/\delta)}{\epsilon})$.

Proof. As there is at least one set containing L_i elements, our use of the exponential mechanism to select sets combined with Equation 2.4 ensures that the probability we select a set covering fewer than $L_i - 3\ln m/\epsilon$ elements is at most $1/m^2$. While $L_i > 6\ln m/\epsilon$, with probability at least $(1 - 1/m)$ we always select sets that cover at least $L_i/2$ elements, and can therefore use no more than $O(\mathsf{OPT}\ln n)$ sets. Once L_i drops below this bound, we observe that the number of remaining elements $|R_i|$ is at most $\mathsf{OPT} \cdot L_i$. Any permutation therefore costs at most an additional $O(\mathsf{OPT}\ln m/\epsilon')$. □

6.1.2 Privacy

Theorem 6.3. *The unweighted set cover algorithm preserves* (ϵ, δ) *differential privacy for any* $\epsilon \in (0, 1)$, *and* $\delta < 1/e$.

Proof. Let A and B be two set cover instances that differ in some element I. Say that S^I is the collection of sets containing I. Fix an output permutation π, and write $s_{i,j}(A)$ to denote the size of set S_j after the first $i-1$ sets in π have been added to the cover.

$$\begin{aligned}\frac{\mathbf{Pr}[M(A) = \pi]}{\mathbf{Pr}[M(B) = \pi]} &= \prod_{i=1}^{n}\left(\frac{\exp(\epsilon' \cdot s_{i,\pi_i}(A))/(\sum_j \exp(\epsilon' \cdot s_{i,j}(A)))}{\exp(\epsilon' \cdot s_{i,\pi_i}(B))/(\sum_j \exp(\epsilon' \cdot s_{i,j}(B)))}\right) \\ &= \frac{\exp(\epsilon' \cdot s_{t,\pi_t}(A))}{\exp(\epsilon' \cdot s_{t,\pi_t}(B))} \cdot \prod_{i=1}^{t}\left(\frac{\sum_j \exp(\epsilon' \cdot s_{i,j}(B))}{\sum_j \exp(\epsilon' \cdot s_{i,j}(A))}\right)\end{aligned}$$

where t is such that S_{π_t} is the first set containing I to fall in the permutation π. After t, the remaining elements in A and B are identical, and all subsequent terms cancel. Moreover, except for the t^{th} term, the numerators of both the top and bottom expression cancel, since all the relevant set sizes are equal. If A contains I and B does not the first term is $\exp(\epsilon')$ and the each term in the product is at most 1.

Now suppose that B contains I and A does not . In this case, the first term is $\exp(-\epsilon') < 1$. Moreover, in instance B, every set in S^I is larger by 1 than in A, and all others remain the same size. Therefore, we have:

$$\begin{aligned}\frac{\mathbf{Pr}[M(A) = \pi]}{\mathbf{Pr}[M(B) = \pi]} &\leq \prod_{i=1}^{t}\left(\frac{(\exp(\epsilon') - 1) \cdot \sum_{j \in S^I} \exp(\epsilon' \cdot s_{i,j}(A)) + \sum_j \exp(\epsilon' \cdot s_{i,j}(A))}{\sum_j \exp(\epsilon' \cdot s_{i,j}(A))}\right) \\ &= \prod_{i=1}^{t}(1 + (\exp(\epsilon') - 1) \cdot p_i(A))\end{aligned}$$

where $p_i(A)$ is the probability that a set containing I is chosen at step i of the algorithm running on instance A, conditioned on picking the sets $S_{\pi_1}, \ldots, S_{\pi_{i-1}}$ in the previous steps.

By the definition of t, no set containing I has been chosen in the first $t-1$ rounds; since at each round $i \leq t$, the probability of not choosing I is at most $1 - p_i(A)$ (it is even smaller on instance B), the probability that I is not chosen in the first $t-1$ rounds is at most $\prod_{i=1}^{t-1}(1 - p_i(A)) \leq \exp(-\sum_{i=1}^{t-1} p_i(A))$. Setting this quantity to be less than δ, we have that with probability at least $1 - \delta$,

$$\sum_{i=1}^{t-1} p_i(A) \leq \ln \delta^{-1}.$$

Conditioning on this event, we can continue the analysis from above:

$$\frac{\mathbf{Pr}[M(A)=\pi]}{\mathbf{Pr}[M(B)=\pi]} \le \prod_{i=1}^{t} \exp((\exp(\epsilon')-1)p_i(A)) \le \exp(2\epsilon' \sum_{i=1}^{t} p_i(A))$$
$$\le \exp(2\epsilon'(\ln(\frac{1}{\delta}) + p_t(A))) \le \exp(2\epsilon'(\ln(\frac{1}{\delta}) + 1)).$$

Thus except with probability $(1-\delta)$, the affect of I on the distribution is at most $exp(\epsilon)$. □

Corollary 6.4. *For $\epsilon < 1$ and $\delta = 1/\mathrm{poly}(n)$, there is an $O(\frac{\ln n \ln m}{\epsilon})$-approximation algorithm for the unweighted set cover problem preserving (ϵ, δ)-differential privacy.*

6.2 Weighted Set Cover

We are given a set system $(U, \mathcal{S})$ and a cost function $C : \mathcal{S} \to \mathbb{R}$. We must cover a private subset $R \subset U$. W.l.o.g., let $\min_{S\in\mathcal{S}} C(S) = 1$, and denote $\max_{S\in\mathcal{S}} C(S) = W$. Let the cardinality of the set system be $|\mathcal{S}| = m$, and let $|U| = n$.

Algorithm 6 Weighted Set Cover

1: **let** $i \leftarrow 1$, $R_i = R$, $\mathcal{S}_i \leftarrow \mathcal{S}$, $r_i \leftarrow n$, $\epsilon' = \frac{\epsilon}{2\ln(e/\delta)}$, $T = \Theta\big(\frac{\log m + \log\log(nW)}{\epsilon'}\big)$
2: **while** $r_i \ge 1/W$ **do**
3: **pick** a set S from $\mathcal{S}_i$ with probability proportional to $\exp\big(\epsilon'\big(\,|S \cap R_i| - r_i \cdot C(S)\,\big)\big)$
or `halve` with probability proportional to $\exp(-\epsilon' T)$
4: **if** `halve` **then**
5: **let** $r_{i+1} \leftarrow r_i/2$, $R_{i+1} \leftarrow R_i$, $\mathcal{S}_{i+1} \leftarrow \mathcal{S}_i$, $i \leftarrow i+1$
6: **else**
7: **output** set S
8: **let** $R_{i+1} \leftarrow R_i \setminus S$, $\mathcal{S}_{i+1} \leftarrow \mathcal{S}_i - \{S\}$, $r_{i+1} \leftarrow r_i$, $i \leftarrow i+1$
9: **end if**
10: **end while**
11: **output** all remaining sets in $\mathcal{S}_i$ in random order

Let us first analyze the utility of the algorithm. If $R = \emptyset$, the algorithm has cost zero and there is nothing to prove. So we can assume that $\mathsf{OPT} \ge 1$. We first show that (**whp**) $r_i \gtrapprox R_i/\mathsf{OPT}$.

Lemma 6.5. *Except with probability $1/\mathrm{poly}(m)$, we have $r_i \ge \frac{|R_i|}{2\mathsf{OPT}}$ for all iterations i.*

Proof. Clearly $r_1 = n \ge |R_1|/2\mathsf{OPT}$. Since for r_i to fall below $|R_i|/2$, it must be in $(\frac{|R_i|}{2\mathsf{OPT}}, \frac{|R_i|}{\mathsf{OPT}}]$ and be halved in Step 6 of some iteration i. We'll show that this is unlikely: if at some iteration i, $\frac{|R_i|}{2\mathsf{OPT}} \le r_i \le \frac{|R_i|}{\mathsf{OPT}}$, then with high probability, the algorithm will not output `halve` and thus not halve r_i. Since all remaining elements R_i can be covered at cost at most OPT, there must exist a set S such that $\frac{|S\cap R_i|}{C(S)} \ge \frac{|R_i|}{\mathsf{OPT}}$, and hence $|S \cap R_i| \ge C(S) \cdot \frac{|R_i|}{\mathsf{OPT}} \ge C(S) \cdot r_i$.

Hence $u_i(S) := |S\cap R_i| - r_i \cdot C(S) \ge 0$ in this case, and the algorithm will output S with probability at least proportional to 1, whereas it outputs `halve` with probability proportional to $\exp(-\epsilon' T)$. Thus, $\mathbf{Pr}[$ algorithm returns `halve` $] < \exp(-\epsilon' T) = 1/\mathrm{poly}(m \log nW)$. Since there are m sets in total, and r ranges from n to $1/W$, there are at most $m + O(\log nW)$ iterations, and the proof follows by a union bound. □

Let us define a *score* function $u_i(S) := |S \cap R_i| - r_i \cdot C(S)$, and $u_i(\texttt{halve}) := -T$: note that in Step 4 of our algorithm, we output either `halve` or a set S, with probabilities proportional to $\exp(\epsilon' u_i(\cdot))$. The following lemma states that with high probability, none of the sets output by our algorithm have very low scores (since we are much more likely to output `halve` than a low-scoring set).

Lemma 6.6. *Except with probability at most $1/\mathrm{poly}(m)$, Step 4 only returns sets S with $u_i(S) \ge -2T$.*

Proof. There are at most $|\mathcal{S}_i| \le m$ sets S with score $u_i(S) \le -2T$, and so one is output with probability at most proportional to $m\exp(-2T\epsilon)$. We will denote this bad event by $\mathcal{B}$. On the other hand, `halve` is output with probability proportional to $\exp(-T\epsilon)$. Hence, $\mathbf{Pr}[\texttt{halve}]/\mathbf{Pr}[\mathcal{B}] \ge \exp(T\epsilon)/m$, and so $\mathbf{Pr}[\mathcal{B}] \le m/\exp(T\epsilon) \le 1/\operatorname{poly}(m \log nW)$. Again there are at most $m + O(\log nW)$ iterations, and the lemma follows by a trivial union bound. □

We now analyze the cost incurred by the algorithm in each stage. Let us divide the algorithm's execution into *stages*: stage j consists of all iterations i where $|R_i| \in (\frac{n}{2^j}, \frac{n}{2^{j-1}}]$. Call a set S *interesting* if it is incident on an uncovered element when it is picked. Let $\mathcal{I}_j$ be the set of interesting sets selected in stage j, and $C(\mathcal{I}_j)$ be the total cost incurred on these sets.

Lemma 6.7. *Consider stages $1, \ldots, j$ of the algorithm. Let i^* be the index of the last iteration in stage j, and let $S_1, S_2, \ldots, S_{i^*}$ be the sets selected in the first i^* iterations of the algorithm. Then except with probability $1/\operatorname{poly}(m)$, we can bound the cost of the interesting sets in stage j by:*

$$\sum_{j' \le j} C(\mathcal{I}_{j'}) \le 4j\mathsf{OPT} \cdot (1 + 2T).$$

Proof. By Lemma 6.6 all the output sets have $u_i(S_i) \ge -2T$ **whp**. Rewriting, each S_i selected in a round $j' \le j$ satisfies

$$C(S_i) \le \frac{|S_i \cap R_i| + 2T}{r_i} \le \frac{2^{j'+1}\,\mathsf{OPT}}{n}(|S_i \cap R_i| + 2T),$$

where the second inequality is **whp**, and uses Lemma 6.5. Now summing over all rounds $j' \le j$, we get

$$\sum_{j' \le j} C(\mathcal{I}_{j'}) \le \sum_{j' \le j} \frac{2^{j'+1}\,\mathsf{OPT}}{n}\Big(\sum_{i \text{ s.t. } S_i \in \mathcal{I}_{j'}} \big(|S_i \cap R_i| + 2T\big)\Big).$$

Consider the inner sum for any particular value of j': let the first iteration in stage j' be iteration i_0—naturally $R_i \subseteq R_{i_0}$ for any iteration i in this stage. Now, since $S_i \cap R_i \subseteq R_{i_0}$ and $S_i \cap R_i$ is disjoint from $S_{i'} \cap R_{i'}$, the sum over $|S_i \cap R_i|$ is at most $|R_{i_0}|$, which is at most $\frac{n}{2^{j'-1}}$ by definition of stage j'. Moreover, since we are only concerned with bounding the cost of interesting sets, each $|S_i \cap R_i| \ge 1$, and so $|S_i \cap R_i| + 2T \le |S_i \cap R_i|(1 + 2T)$. Putting this together, (6.2) implies

$$\sum_{j' \le j} C(\mathcal{I}_{j'}) \le \sum_{j' \le j} \frac{2^{j'+1}\,\mathsf{OPT}}{n} \times \frac{n}{2^{j'-1}}(1 + 2T) = 4j\,\mathsf{OPT}\,(1 + 2T),$$

which proves the lemma. □

Theorem 6.8 (Utility). *The weighted set cover algorithm incurs a cost of $O(T \log n\,\mathsf{OPT})$ except with probability $1/\operatorname{poly}(m)$.*

Proof. Since the number of uncovered elements halves in each stage by definition, there are at most $1 + \log n$ stages, which by Lemma 6.7 incur a total cost of at most $O(\log n\ \mathsf{OPT} \cdot (1 + 2T))$. The sets that remain and are output at the very end of the algorithm incur cost at most W for each remaining uncovered element; since $r_i < 1/W$ at the end, Lemma 6.5 implies that $|R_i| < 2\mathsf{OPT}/W$ (**whp**), giving an additional cost of at most $2\,\mathsf{OPT}$. □

We can adapt the above argument to bound the expected cost by $O(T \log n\ \mathsf{OPT})$. (Proof in the full version.)

Theorem 6.9 (Privacy). *For any $\delta > 0$, the weighted set cover algorithm preserves (ϵ, δ) differential privacy.*

Proof. We imagine that the algorithm outputs a set named "HALVE" when Step 4 of the algorithm returns `halve`, and show that even this output is privacy preserving. Let A and B be two set cover instances that differ in some element I. Say that S^I is the collection of sets containing I. Fix an output π, and write $u_{i,j}(A)$ to denote the score of π_j (recall this may be `halve`) after the first $i - 1$ sets in π have been selected.

$$\frac{\mathbf{Pr}[M(A) = \pi]}{\mathbf{Pr}[M(B) = \pi]} = \prod_{i=1}^{n}\left(\frac{\exp(\epsilon' \cdot u_{i,\pi_i}(A))/(\sum_j \exp(\epsilon' \cdot u_{i,j}(A)))}{\exp(\epsilon' \cdot u_{i,\pi_i}(B))/(\sum_j \exp(\epsilon' \cdot u_{i,j}(B)))}\right) = \frac{\exp(\epsilon' \cdot u_{t,\pi_t}(A))}{\exp(\epsilon' \cdot u_{t,\pi_t}(B))} \cdot \prod_{i=1}^{t}\left(\frac{\sum_j \exp(\epsilon' \cdot u_{i,j}(B))}{\sum_j \exp(\epsilon' \cdot u_{i,j}(A))}\right)$$

where t is such that S_{π_t} is the first set containing I to fall in the permutation π. After t, the remaining elements in A and B are identical, and all subsequent terms cancel. Moreover, except for the t^{th} term, the numerators of both the top and bottom expression cancel, since all the relevant set sizes are equal. If A contains I and B does not the first term is $\exp(\epsilon')$ and the each term in the product is at most 1.

Now suppose that B contains I and A does not . In this case, the first term is $\exp(-\epsilon') < 1$. Moreover, in instance B, every set in S^I is larger by 1 than in A, and all others remain the same size. Therefore, we have:

$$\frac{\mathbf{Pr}[M(A)=\pi]}{\mathbf{Pr}[M(B)=\pi]} \leq \prod_{i=1}^{t}\left(\frac{(\exp(\epsilon')-1)\cdot\sum_{j\in S^I}\exp(\epsilon'\cdot u_{i,j}(A))+\sum_j \exp(\epsilon'\cdot u_{i,j}(A))}{\sum_j \exp(\epsilon'\cdot u_{i,j}(A))}\right) = \prod_{i=1}^{t}\left(1+(e^{\epsilon'}-1)\cdot p_i(A)\right)$$

where $p_i(A)$ is the probability that a set containing I is chosen at step i of the algorithm running on instance A, conditioned on picking the sets $S_{\pi_1},\ldots,S_{\pi_{i-1}}$ in the previous steps.

By the definition of t, no set containing I has been chosen in the first $t-1$ rounds; since at each round $i \leq t$, the probability of not choosing I is at most $1-p_i(A)$ (it is even smaller on instance B), the probability that I is not chosen in the first $t-1$ rounds is at most $\prod_{i=1}^{t-1}(1-p_i(A)) \leq \exp(-\sum_{i=1}^{t-1} p_i(A))$.

There are two cases: If $\exp(-\sum_{i=1}^{t-1} p_i(A)) \leq \delta$, certainly $\mathbf{Pr}[M(A)=\pi] \leq \mathbf{Pr}[M(B)=\pi]+\delta$, which completes the proof. Otherwise,

$$\sum_{i=1}^{t-1} p_i(A) \leq \ln \delta^{-1}.$$

Continuing the analysis from above,

$$\begin{aligned}\frac{\mathbf{Pr}[M(A)=\pi]}{\mathbf{Pr}[M(B)=\pi]} &\leq \prod_{i=1}^{t}\exp((\exp(\epsilon')-1)p_i(A)) \leq \exp\left(2\epsilon'\sum_{i=1}^{t}p_i(A)\right)\\ &\leq \exp\left(2\epsilon'\left(\ln\left(\frac{1}{\delta}\right)+p_t(A)\right)\right) \leq \exp\left(2\epsilon'\left(\ln\left(\frac{1}{\delta}\right)+1\right)\right).\end{aligned}$$

Thus, in this second case, as desired, the effect of I on the output distribution is at most a multiplicative $\exp(\epsilon)$. □

6.3 Removing the Dependence on W

We can remove the dependence of the algorithm on W with a simple idea. For an instance $\mathcal{I} = (U, \mathcal{S})$, let $\mathcal{S}^j = \{S \in \mathcal{S} \mid C(S) \in (n^j, n^{j+1}]\}$. Let U^j be the set of elements such that the cheapest set containing them is in $\mathcal{S}^j$. Suppose that for each j and each $S \in \mathcal{S}^j$, we remove all elements that can be covered by a set of cost at most n^{j-1}, and hence define S' to be $S \cap (U^j \cup U^{j-1})$. This would change the cost of the optimal solution only by a factor of 2, since if we were earlier using S in the optimal solution, we can pick S' and at most n sets of cost at most n^{j-1} to cover the elements covered by $S \setminus S'$. Call this instance $\mathcal{I}' = (U, \mathcal{S}')$.

Now we partition this instance into two instances $\mathcal{I}_1$ and $\mathcal{I}_2$, where $\mathcal{I}_1 = (\cup_{j \text{ even}} U^j, \mathcal{S}')$, and where $\mathcal{I}_2 = (\cup_{j \text{ odd}} U^j, \mathcal{S}')$. Since we have just partitioned the universe, the optimal solution on both these instances costs at most $2\,\mathsf{OPT}(\mathcal{I})$. But both these instances $\mathcal{I}_1, \mathcal{I}_2$ are themselves collections of *disjoint* instances, with each of these instances having $w_{\max}/w_{\min} \leq n^2$; this immediately allows us to remove the dependence on W. Note that this transformation is based only on the set system $(U, \mathcal{S})$, and not on the private subset R.

Theorem 6.10. *For any $\epsilon \in (0,1)$, $\delta = 1/\operatorname{poly}(n)$, there is an $O(log^2 n(\log m + \log\log n)/\epsilon)$-approximation for the weighted set cover problem that preserves (ϵ, δ)-differential privacy.*

6.4 Lower bounds

Theorem 6.11. *Any ϵ-differentially private algorithm that maps elements to sets must have approximation factor $\Omega(\log m/\epsilon)$, for a set cover instance with m sets and $((\log m)/\epsilon)^{O(1)}$ elements, for any $\epsilon \in (2\log m/m^{\frac{1}{20}}, 1)$.*

Proof. We consider a set system with $|U| = N$ and $\mathcal{S}$ a uniformly random selection of m size-k subsets of U. We will consider problem instances S_i consisting of one of these m subsets, so $\mathsf{OPT}(S_i) = 1$.

Let M be an ϵ-differentially private algorithm that on input $T \subseteq U$, outputs an assignment f mapping each element in U to some set in $\mathcal{S}$ that covers it. The number of possible assignments is at most m^N. The cost on input T under an assignment f is the cardinality of the set $f(T) = \cup_{e \in T} f(e)$.

We say assignment f is good for a subset $T \subseteq U$ if its cost $|f(T)|$ is at most $l = \frac{k}{2}$. We first show that any fixed assignment $f : U \to [m]$, such that $|f^{-1}(j)| \leq k$ for all j, is unlikely to be good for a randomly picked size-k subset T of U.

The number of ways to choose l sets from among those with non-empty $f^{-1}(\cdot)$ is at most $\binom{N}{l}$. Thus the probability that f is good for a random size-k subset is at most $\binom{N}{l}\left(\frac{lk}{N}\right)^k$. Setting $k = N^{1/10}$, and $l = \frac{k}{2}$, this is at most

$$\left(\frac{Ne}{l}\right)^l \left(\frac{lk}{N}\right)^k = \left(\frac{ek^3}{2N}\right)^{k/2} \leq 2^{-k \log N/4}.$$

Let $m = 2^{2\epsilon k}$. The probability that f is good for at least t of our m randomly picked sets is bounded by

$$\binom{m}{t}\left(2^{-k \log N/4}\right)^t \leq 2^{2\epsilon k t} 2^{-tk \log N/4} \leq 2^{-tk \log k/8}.$$

Thus, with probability at most $2^{-Nk \log k/8}$, a fixed assignment is good for more than N of m randomly chosen size-k sets. Taking a union bound over $m^N = 2^{2\epsilon k N}$ possible assignments, the probability that *any* feasible assignment f is good for more than N sets is at most $2^{-Nk \log k/16}$. Thus there exists a selection of size-k sets $S_1, \ldots, S_m$ such that no feasible assignment f is good for more than N of the S_i's.

Let $p_{M(\emptyset)}(S_i)$ be the probability that an assignment drawn from the distribution defined by running M on the the empty set as input is good for S_i. Since any fixed assignment is good for at most N of the m sets, the average value of $p_{M(\emptyset)}$ is at most N/m. Thus there exists a set, say S_1 such that $p_{M(\emptyset)}(S_1) \leq N/m$. Since $|S_i| = k$ and M is ϵ-differentially private, $p_{M(S_1)}(S_1) \leq exp(\epsilon k) p_{M(\emptyset)}(S_1) < \frac{1}{2}$. Thus with probability at least half, the assignment M picks on S_1 is not good for S_1. Since $\mathsf{OPT}(S_1) = 1$, the expected approximation ratio of M is at least $l/2 = \frac{\log m}{4\epsilon}$.

Additionally, one can take s distinct instances of the above problem, leading to a new instance on $s \cdot N$ elements and $s \cdot m$ sets. OPT is now s, while it is easy to check that any private algorithm must cost $\Omega(s \cdot l)$ in expectation. Thus the lower bound in fact rules out additive approximations. □

6.5 An Inefficient Algorithm for Weighted Set Cover

For completeness, we now show that the lower bound shown above is tight even in the weighted case, in the absence of computational constraints. Recall that we are given a collection $\mathcal{S}$ of subsets of a universe U, and a private subset $R \subseteq U$ of elements to be covered. Additionally, we have weights on sets; we round up weights to powers of 2, so that sets in $\mathcal{S}_j$ have weight exactly 2^{-j}. Without loss of generality, the largest weight is 1 and the smallest weight is $w = 2^{-L}$.

As before, we will output a permutation π on $\mathcal{S}$, with the understanding that the cost $cost(R, \pi)$ of a permutation π on input R is defined to be the total cost of the set cover resulting from picking the first set in the permutation containing e, for each $e \in R$.

Our algorithm constructs this permutation in a gradual manner. It maintains a permutation π_j on $\cup_{i \leq j} \mathcal{S}_i$ and a threshold T_j. In step j, given π_{j-1} and T_{j-1}, the algorithm constructs a partial permutation π_j on $\cup_{i \leq j} \mathcal{S}_j$, and a threshold T_j. In each step, we use the exponential mechanism to select an extension with an appropriate base distribution μ_j and score function q. At the end of step L, we get our permutation $\pi = \pi_L$ on $\mathcal{S}$.

Our permutations π_j will all have a specific structure. The weight of the ith set in the permutation, as a function of i will be a unimodal function that is non-increasing until T_j, and then non-decreasing. In other words, π_j contains sets from $\mathcal{S}_j$ as a continuous block. The sets that appear before T_j are said to be in the *bucket*. We call a partial permutation respecting this structure *good*. We say a good permutation π *extends* a good partial permutation π_j if π_j and π agree on their ordering on $\cup_{i \leq j} \mathcal{S}_i$.

We first define the score function that is used in these choices. A natural objective function would be $cost(R, \pi_j) = \min_{\pi \text{ extends } \pi_j} cost(R, \pi)$, i.e. the cost of the optimal solution conditioned on respecting the partial permutation π_j. We use a slight modification of this score function: we force the cover to contain all sets from the

bucket and denote as $\widetilde{cost}(R,\pi)$ the resulting cover defined by R on π. We then define $\widetilde{cost}(R,\pi_j)$ naturally as $\min_\pi$ extends $_{\pi_j}$ $\widetilde{cost}(R,\pi)$. We first record the following easy facts:

Observation 6.12. *For any R, $\min_\pi \widetilde{cost}(R,\pi) = \min_\pi cost(R,\pi) = \mathsf{OPT}$. Moreover, for any π, $\widetilde{cost}(R,\pi) \geq cost(R,\pi)$.*

To get (π_j, T_j) given (π_{j-1}, T_{j-1}), we insert a permutation σ_j of $\mathcal{S}_j$ after the first T_{j-1} elements of π_{j-1}, and choose T_j, where both σ_j and T_j are chosen using the exponential mechanism. The base measure on σ_j is uniform and the base measure on $T_j - T_{j-1}$ is the geometric distribution with parameter $1/m^2$.

Let $\widetilde{cost}(A,(\sigma_j,T_j))$ be defined as $\widetilde{cost}(A,\pi_j) - \widetilde{cost}(A,\pi_{j-1})$, where π_j is constructed from π_{j-1} and (σ_j,T_j) as above. The score function we use to pick (σ_j,T_j) is $score_j(R,(\sigma_j,T_j)) = 2^j\widetilde{cost}(R,(\sigma_j,T_j))$. Thus $Pr[(\sigma_j,T_j)] \propto (1/m^2(T_j - T_{j-1}))\exp(\varepsilon score((\sigma_j,T_j)))$.

Let the optimal solution to the instance contain n_j sets from $\mathcal{S}_j$. Thus $\mathsf{OPT} = \sum_j 2^{-j}n_j$. We first show that $\widetilde{cost}(R,\pi_L)$ is $O(\mathsf{OPT}\log m/\varepsilon)$. By Observation 6.12, the approximation guarantee would follow.

The probability that the n_j sets in OPT fall in the bucket when picking from the base measure is at least $1/m^{3n_j}$. When that happens, $\widetilde{cost}(R,\pi_j) = \widetilde{cost}(R,\pi_{j-1})$. Thus the exponential mechanism ensures that except with probability $1/poly(m)$:

$$\widetilde{cost}(R,\pi_j) \leq \widetilde{cost}(R,\pi_{j-1}) + 4\cdot 2^{-j}\log(m^{3n_j})/\epsilon = \widetilde{cost}(R,\pi_{j-1}) + 12\cdot 2^{-j}n_j\log m/\epsilon$$

Thus with high probability,

$$\begin{aligned}\widetilde{cost}(R,\pi_L) &\leq \widetilde{cost}(R,\pi_0) + 12\sum_j 2^{-j}n_j\log m/\epsilon \\ &= \mathsf{OPT} + 12\mathsf{OPT}\log m/\epsilon\end{aligned}$$

Finally, we analyze the privacy. Let $e \in U$ be an element such that the cheapest set covering U has cost 2^{-j_e}. Let A and B be two instances that differ in element e. It is easy to see that $|\widetilde{cost}(A,(\sigma_j,T_j)) - \widetilde{cost}(B,(\sigma_j,T_j))|$ is bounded by 2^{-j} for all j. We show something stronger:

Lemma 6.13. *For any good partial permutation π_j and any A, B such that $A = B \cup \{e\}$,*

$$|score(A,(\sigma_j,T_j)) - score_j(B,(\sigma_j,T_j))| \leq \begin{cases} 0 & \text{if } j > j_e \\ 2^{j_e-j+1} & \text{if } j \leq j_e \end{cases}$$

Proof. Let π_B be the permutation realizing $\widetilde{cost}(B)$. For $j \leq j_e$, if e is covered by a set in the bucket in π_B, then the cost of π_B is no larger in instance A and hence $\widetilde{cost}(A,\pi_j) = \widetilde{cost}(B,\pi_j)$[1]. In the case that the bucket in π_B does not cover e, then $\widetilde{cost}(A,\pi_j) \leq \widetilde{cost}(A,\pi_B) = \widetilde{cost}(B,\pi_B) + 2^{-j_e} = \widetilde{cost}(B,\pi_j) + 2^{-j_e}$. Since this also holds for π_{j-1}, this implies the claim from $j \leq j_e$.

For $j > j_e$, observe that the first set in π_B that covers e is fully determined by the partial permutation π_j, since the sets in $\cup_{i>j_e}\mathcal{S}_i$ do not contain e. Thus $\widetilde{cost}(A,(\sigma_j,T_j)) = \widetilde{cost}(B,(\sigma_j,T_j))$ and the claim follows. □

Then for any $j \leq j_e$, lemma 6.13 implies that for any (σ_j,T_j), $\exp(\epsilon(score(A,(\sigma_j,T_j)) - score(B,(\sigma_j,T_j)))) \in [\exp(-\epsilon 2^{j_e-j+1}), \exp(\epsilon 2^{j_e-j+1})]$. Thus

$$\frac{\mathbf{Pr}[\sigma_j,T_j|A]}{\mathbf{Pr}[\sigma_j,T_j|B]} \in [\exp(-2^{j-j_e+2}\epsilon), \exp(2^{j-j_e+2}\epsilon)]$$

Moreover, for any $j > j_e$, this ratio is 1. Thus

$$\begin{aligned}\frac{\mathbf{Pr}[\sigma_j,T_j|A]}{\mathbf{Pr}[\sigma_j,T_j|B]} &\in [\Pi_{j\leq j_e}\exp(-2^{j-j_e+2}\epsilon), \Pi_{j\leq j_e}\exp(2^{j-j_e+2}\epsilon)] \\ &\subseteq [\exp(-8\epsilon), \exp(8\epsilon)],\end{aligned}$$

which implies 8ϵ-differential privacy.

[1] We remark that this is not true for the function *cost*, and is the reason we had to modify it to $\widetilde{cost}$.

7 Facility Location

Consider the metric facility location problem: we are given a metric space (V, d), a facility cost f and a (private) set of demand points $D \subseteq V$. We want to select a set of facilities $F \subseteq V$ to minimize $\sum_{v \in D} d(v, F) + f \cdot |F|$. (Note that we assume "uniform" facility costs here instead of different costs f_i for different $i \in V$.) Assume that distances are at least 1, and let $\Delta = \max_{u,v} d(u, v)$ denote the diameter of the space.

We use the result of Fakcharoenphol et al. [FRT04] that any metric space on n points can be approximated by a distribution over dominating trees with expected stretch $O(\log n)$; moreover all the trees in the support of the distribution are rooted 2-HSTs—they have $L = O(\log \Delta)$ levels, with the leaves (at level 0) being exactly $= V$, the internal nodes being all Steiner nodes, the root having level L, and all edges between levels $(i + 1)$ and i having length 2^i. Given such a tree T and node v at level i, let T_v denote the (vertices in) the subtree rooted at v.

By Corollary 4.6, it is clear that we cannot output the actual set of facilities, so we will instead output instructions in the form of an HST $T = (V_T, E_T)$ and a set of facilities $F \subseteq V_T$: each demand $x \in D$ then gets assigned to its ancestor facility at the lowest level in the tree. (We guarantee that the root is always in F, hence this is well-defined.) Now we are charged for the connection costs, and for the *facilities that have at least one demand assigned to them.*

Algorithm 7 The Facility Location Algorithm

1: **Input:** Metric (V, d), facility cost f, demands $D \subseteq V$,ϵ.
2: Pick a random distance-preserving FRT tree T; recall this is a 2-HST with $L = O(\log \Delta)$ levels.
3: **let** $F \leftarrow$ root r.
4: **for** $i = 1$ to L **do**
5: **for** all vertices v at level i **do**
6: **let** $N_v = |D \cap T_v|$ and $\widetilde{N_v} = N_v + \text{Lap}(L/\epsilon)$.
7: **if** $\widetilde{N_v} \cdot 2^i > f$ **then** $F \leftarrow F \cup v$.
8: **end for**
9: **end for**
10: **output** (T, F): each demand $x \in D$ is assigned to the ancestor facility at lowest level in T.

Theorem 7.1. *The above algorithm preserves ϵ-differential privacy and outputs a solution of cost* $\mathsf{OPT} \cdot O(\log n \log \Delta) \cdot \frac{\log \Delta}{\epsilon} \log\left(\frac{n \log^2 \Delta}{\epsilon}\right)$.

For the privacy analysis, instead of outputting the set F we could imagine outputting the tree T and all the counts $\widetilde{N_v}$; this information clearly determines F. Note that the tree is completely oblivious of the demand set. Since adding or removing any particular demand vertex can only change L counts, and the noise added in Step 6 gives us ε/L-differential privacy, the fact that differential privacy composes linearly gives us the privacy claim.

For the utility analysis, consider the "noiseless" version of the algorithm which opens a facility at v when $N_v \cdot 2^i \geq f$. It can be shown that this ideal algorithm incurs cost at most $f + O(\log n \log \Delta) \cdot \mathsf{OPT}$ (see, e.g., [Ind04, Theorem 3]). We now have two additional sources of error due to the noise:

- Consider the case when $N_v \cdot 2^i \geq f > \widetilde{N_v} \cdot 2^i$, which increases the connection cost of some demands in D. However, the noise is symmetric, and so we overshoot the mark with probability at most $1/2$—and when this happens the 2-HST property ensures that the connection cost for any demand x increases by at most a factor of 2. Since there are at most $L = O(\log \Delta)$ levels, the expected connection cost increases by at most a factor of L.
- Consider the other case when $N_v \cdot 2^i < f \leq \widetilde{N_v} \cdot 2^i$, which increases the facility cost. Note that if $N_v \cdot 2^i \geq f/2$, then opening a facility at v can be charged again in the same way as for the noiseless algorithm (up to a factor of 2). Hence suppose that $\widetilde{N_v} - N_v \geq \frac{1}{2}(f/2^i)$, and hence we need to consider the probability p_i of the event that $\text{Lap}(L/\epsilon) > \frac{1}{2}(f/2^i)$, which is just $\frac{L}{\epsilon} \exp(-\frac{f}{2^{i+1}} \frac{\epsilon}{L})$.

  Note that if for some value of i, $f \geq \frac{L\, 2^{i+1}}{\epsilon} \log \frac{L^2 n}{\varepsilon}$, the above probability p_i is at most $1/Ln$, and hence the expected cost of opening up spurious facilities at nodes with such values of i is at most $(1/Ln) \cdot Ln \cdot f = f$. (There are L levels, and at most n nodes at each level.)

For the values of i which are higher; i.e., for which $f < \frac{L\,2^{i+1}}{\epsilon} \log \frac{L^2 n}{\varepsilon}$, we pay for this facility only if there is a demand $x \in D$ in the subtree below v that actually uses this facility. Hence this demand x must have used a facility above v in the noiseless solution, and we can charge the cost f of opening this facility to length of the edge 2^{i+1} above v. Thus the total cost of spurious facilities we pay for is the cost of the noiseless solution times a factor $\frac{L}{\epsilon} \log \frac{L^2 n}{\varepsilon}$.

Thus the expected cost of the solution we generate is at most

$$\mathsf{OPT} \cdot O(\log n \log \Delta) \cdot \frac{\log \Delta}{\epsilon} \log \left(\frac{n \log^2 \Delta}{\epsilon} \right). \tag{7.10}$$

8 Combinatorial Public Projects (Submodular Maximization)

Recently Papadimitriou et al.[PSS08] introduced the Combinatorial Public Projects Problem (CPP Problem) and showed that there is a succinctly representable version of the problem for which, although there exists a constant factor approximation algorithm, no efficient *truthful* algorithm can guarantee an approximation ratio better than $m^{\frac{1}{2}-\epsilon}$, unless $NP \subseteq BPP$. Here we adapt our set cover algorithm to give a privacy preserving approximation to the CPP problem within logarithmic (additive) factors.

In the CPP problem, we have n agents and m resources publicly known. Each agent submits a private non-decreasing and *submodular* valuation function f_i over subsets of resources, and our goal is to select a size-k subset S of the resources to maximize $\sum_{i=1}^{n} f_i(S)$. We assume that we have oracle access to the functions f_i. Note that since each f_i is submodular, so is $\sum_{i=1}^{n} f_i(S)$, and our goal is to produce a algorithm for submodular maximization that preserves the privacy of the individual agent valuation functions. Without loss of generality, we will scale the valuation functions such that they take maximum value 1: $\max_{i,S} f_i(S) = 1$.

Once again, we have an easy computationally inefficient algorithm.

Theorem 8.1. *The exponential mechanism when used to choose k sets runs in time $O(\binom{m}{k} poly(n))$ and has expected quality at least $OPT - O(\log \binom{m}{k}/\epsilon)$.*

We next give a computationally efficient algorithm with slightly worse guarantees. We adapt our unweighted set cover algorithm, simply selecting k items greedily:

Algorithm 8 CPP Problem

Input: A set of M of m resources, private functions $f_1, \ldots, f_n$, a number of resources k, ϵ, δ.
let $M_1 \leftarrow M$, $F(x) := \sum_{i=1}^{m} f_i(x)$, $S_1 \leftarrow \emptyset$, $\epsilon' \leftarrow \frac{\epsilon}{8e \ln(2/\delta)}$.
for $i = 1$ to k **do**
  pick a resource r from M_i with probability proportional to $\exp(\epsilon'(F(S_i + \{r\}) - F(S_i)))$.
  let $M_{i+1} \leftarrow M_i - \{r\}$, $S_{i+1} \leftarrow S_i + \{r\}$.
end for
Output S_{k+1}.

8.1 Utility Analysis

Theorem 8.2. *Except with probability $O(1/poly(n))$, the algorithm for the CPP problem returns a solution with quality at least $(1 - 1/e)\mathsf{OPT} - O(k \log m/\epsilon')$.*

Proof. Since F is submodular and there exists a set S^* with $|S| = k$ and $F(S) = \mathsf{OPT}$, there always exists a resource r such that $F(S_i + \{r\}) - F(S_i) \geq (\mathsf{OPT} - F(S_i))/k$. If we always selected the optimizing resource, the distance to OPT would decrease by a factor of $1 - 1/k$ each round, and we would achieve an approximation factor of $1 - 1/e$. Instead, we use the exponential mechanism which, by (2.4), selects a resource within $4 \ln m/\epsilon'$ of the optimizing resource with probability at least $1 - 1/m^3$. With probability at least $1 - k/m^3$ each of the k selections decreases $\mathsf{OPT} - F(S_i)$ by a factor of $(1 - 1/k)$, while increasing it by at most an additive $4 \ln m/\epsilon'$, giving $(1 - 1/e)\mathsf{OPT} + O(k \ln m/\epsilon')$. □

8.2 Privacy Analysis

Theorem 8.3. *For any $\delta \leq 1/2$, the CPP problem algorithm preserves $(8\epsilon'(e-1)\ln(2/\delta), \delta)$-differential privacy.*

Proof. Let A and B be two CPP instances that differ in a single agent I with utility function f_I. We show that the output set of resources, even revealing the order in which the resources were chosen, is privacy preserving. Fix some ordered set of k resources, $\pi_1, \ldots, \pi_k$ write $S_i = \bigcup_{j=1}^{i-1}\{\pi(j)\}$ to denote the first $i-1$ elements, and write $s_{i,j}(A) = F_A(S_i + \{j\}) - F_A(S_i)$ to denote the marginal utility of item j at time i in instance A. Define $s_{i,j}(B)$ similarly for instance B. We consider the relative probability of our mechanism outputting ordering π when given inputs A and B:

$$\frac{\mathbf{Pr}[M(A)=\pi]}{\mathbf{Pr}[M(B)=\pi]} = \prod_{i=1}^{k}\left(\frac{\exp(\epsilon'\cdot s_{i,\pi_i}(A))/(\sum_j \exp(\epsilon'\cdot s_{i,j}(A)))}{\exp(\epsilon'\cdot s_{i,\pi_i}(B))/(\sum_j \exp(\epsilon'\cdot s_{i,j}(B)))}\right),$$

where the sum over j is over all remaining unselected resources. We can separate this into two products

$$\prod_{i=1}^{k}\left(\frac{\exp(\epsilon'\cdot s_{i,\pi_i}(A))}{\exp(\epsilon'\cdot s_{i,\pi_i}(B))}\right)\cdot\prod_{i=1}^{k}\left(\frac{\sum_j \exp(\epsilon'\cdot s_{i,j}(B))}{\sum_j \exp(\epsilon'\cdot s_{i,j}(A))}\right).$$

If A contains agent I but B does not, the second product is at most 1, and the first is at most $\exp(\epsilon'\sum_{i=1}^{k}(F_I(S_i) - F_I(S_{i-1}))) \leq \exp(\epsilon')$. If B contains agent I, and A does not, the first product is at most 1, and in the remainder of the proof, we focus on this case. We will write $\beta_{i,j} = s_{i,j}(B) - s_{i,j}(A)$ to be the additional marginal utility of item j at time i in instance B over instance A, due to agent I. Thus

$$\begin{aligned}
\frac{\mathbf{Pr}[M(A)=\pi]}{\mathbf{Pr}[M(B)=\pi]} &\leq \prod_{i=1}^{k}\left(\frac{\sum_j \exp(\epsilon'\cdot s_{i,j}(B))}{\sum_j \exp(\epsilon'\cdot s_{i,j}(A))}\right)\\
&= \prod_{i=1}^{k}\left(\frac{\sum_j \exp(\epsilon'\beta_{i,j})\cdot\exp(\epsilon'\cdot s_{i,j}(A))}{\sum_j \exp(\epsilon'\cdot s_{i,j}(A))}\right)\\
&= \prod_{i=1}^{k}\mathbb{E}_i[\exp(\epsilon'\beta_i)],
\end{aligned}$$

where β_i is the marginal utility actually achieved at time i by agent I, and the expectation is taken over the probability distribution over resources selected at time i in instance A. For all $x \leq 1$, $e^x \leq 1 + (e-1)\cdot x$. Therefore, for all $\epsilon' \leq 1$, we have:

$$\begin{aligned}
\prod_{i=1}^{k}\mathbb{E}_i[\exp(\epsilon'\beta_i)] &\leq \prod_{i=1}^{k}E_i[1+(e-1)\epsilon'\beta_i]\\
&\leq \exp((e-1)\epsilon'\sum_{i=1}^{k}E_i[\beta_i]).
\end{aligned}$$

We must therefore show that with high probability, the sum of expected marginal utilities of agent I is small, which we will denote by $C = \sum_{i=1}^{k}E_i[\beta_i]$.

We note that the *realized* total utility of agent i is at most 1: $\sum_{i=1}^{k}\beta_{i,\pi_i} \leq 1$. If C is large, then the realized utility of agent i has a large deviation from its expected value; such realizations π must be rare. By a Chernoff bound, this occurs only with small probability:

$$\mathbf{Pr}[|\sum_{i=1}^{k}\beta_{i,\pi_i} - C| \geq C-1] \leq 2\exp\left(\frac{-(C-1)^2}{3C-1}\right).$$

Setting this probability to be at most δ and solving for C, we find that except with probability δ, $C \leq 8\ln(2/\delta)$. Therefore, with probability $1-\delta$ we have:

$$\frac{\mathbf{Pr}[M(A)=\pi]}{\mathbf{Pr}[M(B)=\pi]} \leq \exp((e-1)\epsilon'\cdot 8\ln\left(\frac{2}{\delta}\right))$$

which proves the theorem.

□

Remark 8.4. *By choosing $\epsilon' = \epsilon/k$, we immediately get ϵ-differential privacy and expected utility at least $\mathsf{OPT} - O(k^2 \ln m/\epsilon)$. This may give better guarantees for some values of k and δ.*

We remark that the k-coverage problem is a special case of the CPP problem. Therefore:

Corollary 8.5. *The CPP algorithm (with* sets *as resources) is an (ϵ, δ)-differential privacy preserving algorithm for the k-coverage problem achieving approximation factor at least $(1 - 1/e)\mathsf{OPT} - O(k \log m \log(2/\delta)/\epsilon)$.*

8.3 Truthfulness

The CPP problem can be viewed as a mechanism design problem when each agent i has a choice of whether to submit his actual valuation function f_i, or to lie and submit a different valuation function f_i' if such a misrepresentation yields a better outcome for agent i. A mechanism is *truthful* if for every valuation function of agents $j \neq i$, and every valuation function f_i of agent i, there is never a function $f_i' \neq f_i$ such that agent i can benefit by misrepresenting his valuation function as f_i'. Intuitively, a mechanism is approximately truthful if no agent can make more than a slight gain by not truthfully reporting.

Definition 8.6. *A mechanism for the CPP problem is γ-truthful if for every agent i, for every set of player valuations f_j for $j \neq i$, and for every valuation function $f_i' \neq f_i$:*

$$E[f_i(M(f_1, \ldots, f_i, \ldots, f_n))] \geq E[f_i(M(f_1, \ldots, f_i', \ldots, f_n))] - \gamma$$

Note that 0-truthfulness corresponds to the usual notion of (exact) truthfulness.

(ϵ, δ)-differential privacy in our setting immediately implies $(2\epsilon + \delta)$-approximate truthfulness.We note that Papadimitriou et al. [PSS08] showed that the CPP problem is inapproximable to an $m^{\frac{1}{2}-\epsilon}$ multiplicative factor by any polynomial time 0-truthful mechanism. Our result shows that relaxing that to γ-truthfulness allows us to give a constant approximation to the utility whenever $\mathsf{OPT} \geq 2k \log m \log(1/\gamma)/\gamma$ for any γ.

8.4 Lower Bounds

Theorem 8.7. *No ϵ-differentially private algorithm for the maximum coverage problem can guarantee profit larger than $\mathsf{OPT} - (k \log(m/k)/20\epsilon)$.*

The proof is almost identical to that of the lower bound Theorem 4.5 for k-median, and hence is omitted.

9 Steiner Forest

Consider the Steiner network problem, where we are given a metric space $M = (V, d)$ on n points, and a (private) subset $R \subseteq V \times V$ of source-sink (terminal) pairs. The goal is to buy a minimum-cost set of edges $E(R) \subset \binom{V}{2}$ such that these edges connect up each terminal pair in R. As in previous cases, we give instructions in the form of a tree $T = (V, E_T)$; each terminal pair $(u, v) \in R$ takes the unique path $P_T(u, v)$ in this tree T between themselves, and the (implicit) solution is the set of edges $E(R) = \bigcup_{(u,v)\in R} P_T(u, v)$.

The tree T is given by the randomized construction of Fakcharoenphol et al. [FRT04], which guarantees that $\mathbb{E}[\mathsf{cost}(E(R))] \leq O(\log n) \cdot \mathsf{OPT}$; moreover, since the construction is oblivious to the set R, it preserves the privacy of the terminal pairs perfectly (i.e., $\epsilon = 0$). The same idea can be used for a variety of network design problem (such as the "buy-at-bulk" problem) which can be solved by reducing it to a tree instance.

10 Private Boosting Theorem

In this section, we show that differentially private mechanisms that give good guarantees in expectation can be repeated privately to boost the probability of a good outcome.

Theorem 10.1 (Private Repetition Theorem)**.** *Let $M : D \to R$ be an ϵ-differentially private mechanism such that for a query function q with sensitivity 1, and a parameter Q, $\mathbf{Pr}[q(A, M(A)) \geq Q] \geq p$ for some $p \in (0, 1)$. Then for any $\delta > 0$, $\epsilon' \in (0, \frac{1}{2})$, there is a mechanism M' which satisfies the following properties:*

- ***Utility:*** $\mathbf{Pr}[q(A, M(A)) \geq Q - \frac{4}{\epsilon'} \log(\frac{1}{\epsilon' \delta p})] \geq (1 - \delta)$.
- ***Efficiency:*** *M' makes $O((\frac{1}{\epsilon' \delta p})^2 \log(\frac{1}{\epsilon' \delta p}))$ calls to M.*
- ***Privacy:*** *M' satisfies $(\epsilon + 8\epsilon')$-differential privacy.*

Proof. Let $T = (\frac{8}{\epsilon' \delta p})^2 \log(\frac{1}{\epsilon' \delta p})$. The mechanism M' runs M on the input A independently $(T + 1)$ times to get outputs $S_1 = \{r_1, \ldots, r_{T+1}\}$. It also adds in $T' = \frac{\sqrt{4T \log T}}{\epsilon'}$ dummy outcomes $S_2 = \{s_1, \ldots, s_{T'}\}$ and selects an outcome from $S_1 \cup S_2$ using the exponential mechanism with privacy parameter ε' and score function

$$\widetilde{q}(A, r) = \begin{cases} \min(Q, q(A, r)) & \text{if } r \in S_1 \\ Q & \text{if } r \in S_2 \end{cases}$$

The efficiency of M' is immediate from the construction. To analyze the utility, note that (2.4) ensures that the exponential mechanism's output r satisfies $\widetilde{q}(A, r) > Q - \frac{4}{\epsilon'} \log(\frac{1}{\epsilon' \delta p})$ with probability $(1 - \frac{\delta}{2})$. Conditioned on the output r satisfying this property, the ratio $\mathbf{Pr}[r \in S_1]/\mathbf{Pr}[r \in S_2]$ is at least $|\{r \in S_1 : q(A, r) \geq Q\}|/|S_2|$. Since the numerator is at least pT in expectation, the probability of r being a dummy outcome is at most $\frac{\delta}{2}$. This establishes the utility property.

We now show the privacy property. For any $r_0 \in R$,

$$\begin{aligned}
\mathbf{Pr}[M'(A) = r_0] &= \sum_{i=1}^{T+1} \mathbf{Pr}[r_i = r_0] \mathbb{E}\left[\frac{\exp(\epsilon' \widetilde{q}(A, r_0))}{\sum_{r \in S_1} \exp(\epsilon' \widetilde{q}(A, r)) + T' \exp(\epsilon' Q)} \,\middle|\, r_i = r_0 \right] \\
&= (T + 1) \cdot \mathbf{Pr}[M(A) \text{ outputs } r_0] \cdot \exp(\epsilon' \widetilde{q}(A, r_0)) \cdot \mathbb{E}\left[\frac{1}{\sum_{r \in S_1} \exp(\epsilon' \widetilde{q}(A, r)) + T' \exp(\epsilon' Q)} \,\middle|\, r_{T+1} = r_0 \right] \\
&= (T + 1) \cdot \mathbf{Pr}[M(A) \text{ outputs } r_0] \cdot \exp(\epsilon' \widetilde{q}(A, r_0)) \cdot \exp(-\epsilon' Q) \cdot \\
&\qquad \mathbb{E}\left[\frac{1}{\sum_{r \in S_1 \setminus \{r_0\}} \exp(\epsilon'(\widetilde{q}(A, r) - Q)) + \exp(\epsilon'(\widetilde{q}(A, r_0) - Q)) + T'} \right] \qquad (10.11)
\end{aligned}$$

where the expectation is also taken over runs $1, \ldots, T$ of M (we've explicitly conditioned on run $(T+1)$ producing r_0).

It is easy to bound the change in the first two terms when we change from input A to a neighboring input B, since M satisfies ϵ-differential privacy, and $\widetilde{q}$ has sensitivity 1. Let $D = D(A)$ denote the denominator in the final expectation; we would like to show that $\mathbb{E}[\frac{1}{D(A)}] \leq \exp(\epsilon) \mathbb{E}[\frac{1}{D(B)}]$ for neighboring inputs A and B. Let $C = \exp(\epsilon'(\widetilde{q}(A, r_0) - Q)) + T'$ denote the constant term in $D(A)$.

First observe that

$$\begin{aligned}
\mathbb{E}[D(A)] &= C + T \cdot \mathbb{E}_{r \in M(A)}[\exp(\epsilon'(\widetilde{q}(A, r) - Q)] \\
&\geq C + T \cdot \exp(-\epsilon') \cdot \mathbb{E}_{r \in M(A)}[\exp(\epsilon'(\widetilde{q}(B, r) - Q)] \\
&\geq C + T \cdot \exp(-2\epsilon') \cdot \mathbb{E}_{r \in M(B)}[\exp(\epsilon'(\widetilde{q}(B, r) - Q)] \\
&\geq \exp(-2\epsilon') \cdot \mathbb{E}[D(B)],
\end{aligned}$$

where the first inequality follows from the sensitivity of q and the second from the ϵ-differential privacy of M. Thus $\mathbb{E}[D(A)]$ is close to $\mathbb{E}[D(B)]$. We now show that $\mathbb{E}[\frac{1}{D(A)}]$ is close to $\frac{1}{\mathbb{E}[D(A)]}$ for each A, which will complete the proof.

The first step is to establish that $D(A)$ is concentrated around its expectation. Since $D = C + \sum_{i=1}^{T} Y_i$, where the Y_i's are i.i.d. random variables in $[0, 1]$, standard concentration bounds imply

$$\mathbf{Pr}[D \geq \mathbb{E}[D] + t] \leq \exp(-2t^2/T); \qquad \mathbf{Pr}[D \leq \mathbb{E}[D] - t] \leq \exp(-2t^2/T);$$

Since $\frac{1}{D} \geq \frac{1}{C}$, we can now estimate

$$\begin{aligned}
\mathbb{E}[\frac{1}{D}] &\leq \frac{\exp(\epsilon')}{\mathbb{E}[D]} + \int_{\frac{\exp(\epsilon')}{\mathbb{E}[D]}}^{\frac{1}{C}} \mathbf{Pr}[\frac{1}{D} \geq y]\mathrm{d}y \\
&\leq \frac{\exp(\epsilon')}{\mathbb{E}[D]} + \int_{C}^{\exp(-\epsilon')\mathbb{E}[D]} \frac{\mathbf{Pr}[D \leq z]}{z^2}\mathrm{d}z \\
&\leq \frac{\exp(\epsilon')}{\mathbb{E}[D]} + \frac{1}{C^2}\int_{C}^{\exp(-\epsilon')\mathbb{E}[D]} \exp(-2(z-\mathbb{E}[D])^2/T)\mathrm{d}z \\
&\leq \frac{\exp(\epsilon')}{\mathbb{E}[D]} + \frac{(\exp(-\epsilon')\mathbb{E}[D] - C)}{C^2}\exp(-(\epsilon'\mathbb{E}[D])^2/T) \\
&\leq \frac{\exp(\epsilon')}{\mathbb{E}[D]} + \frac{1}{T^2}
\end{aligned}$$

since $\mathbb{E}[D] > C > \frac{\sqrt{4T\log T}}{\epsilon'}$, $\mathbb{E}[D] < 2T$, and $C > 1$. Thus $\mathbb{E}[\frac{1}{D}] \leq \frac{\exp(2\varepsilon')}{\mathbb{E}[D]}$.

Similarly,

$$\begin{aligned}
\mathbb{E}[\frac{1}{D}] &\geq \frac{\exp(-\epsilon')}{\mathbb{E}[D]} - \int_{0}^{\frac{\exp(-\epsilon')}{\mathbb{E}[D]}} \mathbf{Pr}[\frac{1}{D} \leq y]\mathrm{d}y \\
&\geq \frac{\exp(-\epsilon')}{\mathbb{E}[D]} - \int_{\exp(\epsilon')\mathbb{E}[D]}^{\infty} \frac{\mathbf{Pr}[D \geq z]}{z^2}\mathrm{d}z \\
&\geq \frac{\exp(-\epsilon')}{\mathbb{E}[D]} - \frac{\exp(-2\epsilon')}{\mathbb{E}[D]^2}\int_{\exp(\epsilon')\mathbb{E}[D]}^{\infty} \exp(-2(z-\mathbb{E}[D])^2/T)\mathrm{d}z \\
&\geq \frac{\exp(-\epsilon')}{\mathbb{E}[D]} - \frac{\exp(-2\epsilon')}{\mathbb{E}[D]^2}\sqrt{T} \\
&\geq \frac{\exp(-\epsilon')}{\mathbb{E}[D]} - \frac{\epsilon'}{\mathbb{E}[D]},
\end{aligned}$$

so that $\mathbb{E}[\frac{1}{D}] \geq \frac{\exp(-3\epsilon')}{\mathbb{E}[D]}$.

Thus $\mathbb{E}[\frac{1}{D(A)}] \leq \exp(7\epsilon')\mathbb{E}[\frac{1}{D(B)}]$ for neighboring inputs A and B. Now using this fact in expression (10.11) for $\mathbf{Pr}[M'(A) = r_0]$ above, we conclude that M' satisfies $(\epsilon + 8\epsilon')$-differential privacy. □

References

[AGK+04] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[BCD+07] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 273–282. ACM Press New York, NY, USA, 2007.

[BCNW06] A. Beimel, P. Carmi, K. Nissim, and E. Weinreb. Private approximation of search problems. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 119–128. ACM New York, NY, USA, 2006.

[BHN07] A. Beimel, R. Hallak, and K. Nissim. Private Approximation of Clustering and Vertex Cover. *Theory of Cryptography Conference*, 4392:383, 2007.

[BLR08] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the fourtieth annual ACM symposium on Theory of computing*, pages 609–618. ACM New York, NY, USA, 2008.

[BMNW07] A. Beimel, T. Malkin, K. Nissim, and E. Weinreb. How Should We Solve Search Problems Privately? In *CRYPTO*, volume 4622, page 31. Springer, 2007.

[CDM+05] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Theory of Cryptography Conference*, pages 363–385, 2005.

[Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, pages 233–235, 1979.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: privacy via distributed noise generation. In *Advances in cryptology—EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Comput. Sci.*, pages 486–503. Springer, Berlin, 2006.

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *Proceedings of the 3rd Theory of Cryptography Conference*, pages 265–284, 2006.

[DN04] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Advances in cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Comput. Sci.*, pages 528–544. Springer, Berlin, 2004.

[Dwo06] Cynthia Dwork. Differential privacy. In *Automata, languages and programming. Part II*, volume 4052 of *Lecture Notes in Comput. Sci.*, pages 1–12. Springer, Berlin, 2006.

[Dwo08] C. Dwork. Differential Privacy: A Survey of Results. In *Theory and Applications of Models of Computation–TAMC 2008*, volume 4978 of *Lecture Notes In Computer Science*, pages 1–19, 2008.

[FF56] L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

[FIM+06] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure multiparty computation of approximations. *ACM Trans. Algorithms*, 2(3):435–472, 2006.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004.

[HKKN01] S. Halevi, R. Krauthgamer, E. Kushilevitz, and K. Nissim. Private approximation of NP-hard functions. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 550–559. ACM New York, NY, USA, 2001.

[Hoc82] D.S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555, 1982.

[Ind04] P. Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing*, pages 373–380. ACM New York, NY, USA, 2004.

[IW06] P. Indyk and D. Woodruff. Polylogarithmic Private Approximations and Efficient Matching. *Theory of Cryptography*, 3876:245, 2006.

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9:256–278, 1974.

[Kar93] David R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993)*, pages 21–30, New York, 1993. ACM.

[KLN+08] Shiva Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *Proceedings of the 49th annual Symposium on Foundations of Computer Science*, 2008.

[MT07] F. McSherry and K. Talwar. Mechanism Design via Differential Privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103, 2007.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC'07—Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 75–84. ACM, New York, 2007.

[NWF78] GL Nemhauser, LA Wolsey, and ML Fisher. An analysis of approximations for maximizing submodular set functionsI. *Mathematical Programming*, 14(1):265–294, 1978.

[Pit85] L. Pitt. A simple probabilistic approximation algorithm for vertex cover. Technical report, Yale University, 1985.

[PSS08] C. Papadimitriou, M. Schapira, and Y. Singer. On the Hardness of Being Truthful. In *Foundations of Computer Science, 2008. FOCS'08. 49th Annual IEEE Symposium on*, 2008.

[Smi08] A. Smith. Efficient, Differentially Private Point Estimators. *Arxiv preprint arXiv:0809.4794*, 2008.

A Unweighted Vertex Cover Algorithm: An Alternate View

In this section, we consider a slightly different way to implement the vertex cover algorithm. Given a graph $G = (V, E)$, we mimic the randomized proportional-to-degree algorithm for αn rounds ($\alpha < 1$), and output the remaining vertices in random order. That is, in each of the first αn rounds, we select the next vertex i with probability proportional to $d(i) + 1/\epsilon$: this is equivalent to imagining that each vertex has $1/\epsilon$ "hallucinated" edges in addition to its real edges. (It is most convenient to imagine the other endpoint of these hallucinated edges as being fake vertices which are always ignored by the algorithm.)

When we select a vertex, we remove it from the graph, together with the real and hallucinated edges adjacent to it. This is equivalent to picking a random (real or hallucinated) edge from the graph, and outputting a random real endpoint. Outputting a vertex affects the real edges in the remaining graph, but does not change the hallucinated edges incident to other vertices.

Privacy Analysis. The privacy analysis is similar to that of Theorem 5.1: imagine the weights being $w_i = 1/\epsilon$ for the first αn rounds and $w_i = \infty$ for the remaining rounds, which gives us $2\sum_{i=(1-\alpha)n}^{n} \frac{1}{iw_i} \leq \epsilon$ ($\frac{2\alpha}{1-\alpha}$)-differential privacy.

Utility Analysis. To analyze the utility, we couple our algorithm with a run of the non-private algorithm $\mathcal{A}$ that at each step picks an arbitrary edge of the graph and then picks a random endpoint: it is an easy exercise that this an 2-approximation algorithm.

We refer to vertices that have non-zero "real" degree at the time they are selected by our algorithm as *interesting vertices*: the cost of our algorithm is simply the number of interesting vertices it selects in the course of its run. Let I_1 denote the number of interesting vertices it selects during the first αn steps, and I_2 denote the number of interesting vertices it selects during its remaining $(1 - \alpha)n$ steps, when it is simply ordering vertices randomly. Clearly, the total cost is $I_1 + I_2$.

We may view the first phase of our algorithm as selecting an edge at random (from among both real and hallucinated ones) and then outputting one of its endpoints at random. Now, for the rounds in which our algorithm selects a real edge, we can couple this selection with one step of an imagined run of $\mathcal{A}$ (selecting the same edge and endpoint). Note that this run of $\mathcal{A}$ maintains a vertex cover that is a subset of our vertex cover, and that once our algorithm has completed a vertex cover, no interesting vertices remain. Therefore, while our algorithm continues to incur cost, $\mathcal{A}$ has not yet found a vertex cover.

In the first phase of our algorithm, every interesting vertex our algorithm selects has at least one real edge adjacent to it, as well as $1/\epsilon$ hallucinated edges. Conditioned on selecting an interesting vertex, our algorithm had selected a real edge with probability at least $\epsilon' = 1/(1 + 1/\epsilon)$. Let R denote the random variable that represents the number of steps $\mathcal{A}$ is run for. $E[R] \leq 2\mathsf{OPT}$ since $\mathcal{A}$ is a 2-approximation algorithm. By linearity of expectation:

$$2\mathsf{OPT} \geq \mathbb{E}[R] \geq \epsilon' \cdot E[I_1] \tag{A.12}$$

We now show that most of our algorithm's cost comes from the first phase, and hence that I_2 is not much larger than I_1.

Lemma A.1.

$$\mathbb{E}[I_1] \geq \ln\left(\frac{1}{1-\alpha}\right) \cdot \mathbb{E}[I_2]$$

Proof. Consider each of the αn steps of the first phase of our algorithm. Let n_i denote the number of interesting vertices remaining at step i. Note that $\{n_i\}$ is a non-increasing sequence. At step i, there are n_i interesting vertices and $n - i + 1$ remaining vertices. Note that the probability of picking an interesting vertex is strictly greater than $n_i/(n - i + 1)$ at each step. We may therefore bound the expected number of interesting vertices picked in the first phase:

$$\mathbb{E}[I_1] > \sum_{i=1}^{\alpha n} \frac{\mathbb{E}[n_i]}{n-i+1} \geq \mathbb{E}[n_{\alpha n}] \sum_{i=(1-\alpha)n}^{n} \frac{1}{i} \geq \ln\left(\frac{1}{1-\alpha}\right) \cdot \mathbb{E}[n_{\alpha n}]$$

Noting that $\mathbb{E}[I_2] \leq \mathbb{E}[n_{\alpha n}]$ completes the proof. □

Combining the facts above, we get that

$$\frac{\mathbb{E}[\mathsf{cost}]}{\mathsf{OPT}} \leq \frac{2}{\epsilon'}\left(1 + \frac{1}{\ln(1-\alpha)^{-1}}\right). \tag{A.13}$$