SPARSI: Partitioning Sensitive Data amongst Multiple Adversaries

Theodoros Rekatsinas
University of Maryland
thodrek@cs.umd.edu

Amol Deshpande
University of Maryland
amol@cs.umd.edu

Ashwin Machanavajjhala
Duke University
ashwin@cs.duke.edu

ABSTRACT

We present SPARSI, a novel theoretical framework for partitioning sensitive data across multiple non-colluding adversaries. Most work in privacy-aware data sharing has considered disclosing summaries where the aggregate information about the data is preserved, but sensitive user information is protected. Nonetheless, there are applications, including online advertising, cloud computing and crowdsourcing markets, where detailed and fine-grained user data must be disclosed. We consider a new data sharing paradigm and introduce the problem of *privacy-aware data partitioning*, where a sensitive dataset must be partitioned among k untrusted parties (*adversaries*). The goal is to maximize the utility derived by partitioning and distributing the dataset, while minimizing the total amount of sensitive information disclosed. The data should be distributed so that an adversary, without colluding with other adversaries, cannot draw additional inferences about the private information, by *linking* together multiple pieces of information released to her. The assumption of no collusion is both reasonable and necessary in the above application domains that require release of private user information. SPARSI enables us to formally define privacy-aware data partitioning using the notion of *sensitive properties* for modeling private information and a *hypergraph* representation for describing the interdependencies between data entries and private information. We show that solving privacy-aware partitioning is, in general, NP-hard, but for specific information disclosure functions, good approximate solutions can be found using *relaxation* techniques. Finally, we present a local search algorithm applicable to generic information disclosure functions. We conduct a rigorous performance evaluation with real-world and synthetic datasets that illustrates the effectiveness of SPARSI at partitioning sensitive data while minimizing disclosure.

1. INTRODUCTION

The landscape of online services has changed significantly in the recent years. More and more sensitive information is released on the Web and processed by online services. The most common paradigm to consider are people who rely on online social networks to communicate and share information with each other. This leads to a diverse collection of voluntarily published user data. Online services such as Web search, news portals, and recommendation and e-commerce systems, collect and store this data in their effort to provide high-quality personalized experiences to a heterogeneous user base. Naturally, this leads to increased concerns related to an individual's privacy and the possibility of private personal information being aggregated by untrusted third-parties such as advertisers.

A different application domain that is increasingly popular is crowdsourcing markets. Tasks, typically decomposed into microtasks, are submitted by users to a crowdsourcing market and are fulfilled by a collection of workers. The user needs to provide each worker with the necessary data to accomplish each microtask. However, this data may contain information that is sensitive and care must be taken not to disclose any more sensitive information than minimally needed to accomplish the task. Consider, for example, the task of labeling a dataset with information about the location of different individuals to be used as input to a machine learning algorithm. Since the cost of hand-labeling the dataset is high, submitting this task to a crowdsourcing market provides an inexpensive alternative. However, the dataset might contain sensitive information about the trajectories the individuals follow and the structure of the social network they form. Hence, we must perform a clever partitioning of the dataset to the different untrusted workers to avoid disclosing sensitive information. Observe that, under this paradigm, the sensitive information in the dataset is not necessarily associated with a particular data entry.

Similarly with the rise of cloud computing, increasing volumes of private data are being stored and processed on untrusted servers in the cloud. Even if the data is stored and processed in an encrypted form, an adversary may be able to infer some of the private information by aggregating, over a period of time, the information that is available to her (e.g., password hashes of users, workload information). This has led security researchers to recommend splitting data and workloads across systems or organizations to remove such points of compromise [27, 7].

In all applications presented above, a party, called *publisher*, is required to distribute a collection of data (e.g., user information) to many different third parties. The utility in sharing data results either from the improved quality of personalized services or from the cost reduction in fulfilling a decomposable task. The sensitive information is often not limited to the identity of a particular entity in the dataset (e.g., a user using a social network based service), but rather arises from the combination of a set of data items. It is these sets we would like to partition across different adversaries.

1.1 Illustrative Examples

We next use two real-world examples to show how utility can be obtained via partitioning the data, how sensitive information is disclosed and why there is no incentive for the adversaries to collude.

Proceedings of the VLDB Endowment, Vol. 6, No. 13

EXAMPLE 1. *Consider a scenario where users share their locations via check-ins at different time instances either with location-based data aggregators like location-based social networks (e.g., Brightkite [1]) and Gowalla [2]) or location-based recommendation systems, powered by companies like Google, Yelp and Yahoo. User location data is of particular interest to advertisers, as analyzing it can provide them with a detailed profile of the user. Using such data allows advertisers to devise efficient personalized marketing strategies. Hence, they are willing to pay large amounts of money to the data publisher for user information. On the other hand, the user receives personalized services (e.g., notifications for friends near-by) or recommendations for locations of interest.*

The utility for publishing this data comes from both the money advertisers are willing to pay and the quality of the recommendation the user receives. Sharing data with different third-parties yields different amounts of utility, since advertisers may be interested only in a particular subset of user check-ins. Also, recommendation quality may vary across different systems. For example, Google offers more accurate recommendations for sparsely populated regions while Yelp is more accurate for urban areas.

However, analyzing the locations of multiple users collectively can reveal information about the friendship links between users, thus, revealing the structure of the social network. It was recently shown [4] that simple trajectory similarity techniques can be very accurate at predicting the existence of a friendship link between two users. Thus, the disclosure of sensitive information can be computed using the probability that a friendship link exists given the revealed data. Disclosing the structure of the social network might not be desirable, as it can be used for viral marketing purposes, which may drive users away from using the services described above. Thus, a natural tradeoff exists between publishing user data and receiving high monetary utility, versus keeping this data private to ensure the popularity of the social network. We note that there is no incentive for advertisers and location-based service providers to collude due to conflicting monetary interests. Further, such collusion will likely run afoul of the two-party agreements between the publisher and the service providers.

This example shows how an adversary may infer some sensitive information that is not explicitly mentioned in the dataset but is related to the provided data and can be inferred only when particular entries of the dataset are collectively analyzed. Not revealing all those entries together to an adversary prevents disclosure of the sensitive information. We further exemplify this using a crowdsourcing application.

EXAMPLE 2. *Consider outsourcing medical transcription data. Medical transcription is a vital part of healthcare operations. Accurate, timely transcription of daily notes such as operating room reports, and radiology reports is essential for communication among healthcare providers, and has become important in the development of electronic health records. Due to cost-saving policies healthcare providers often outsource their transcription [5] by dividing the task into micro-tasks submitted to multiple workers.*

Utility is obtained by outsourcing the task that needs to be completed, and can be modeled by a function that takes into account both the quality and cost for each worker. Medical records contain confidential information and if all fields in a record are revealed to the same worker, highly sensitive information is disclosed. However, if each record is partitioned in a way that different workers are responsible for transcribing different fields of it, no sensitive information is disclosed. Patients, healthcare providers and doctors cannot be linked with confidential information, such as a disease or a particular treatment. Disclosure of sensitive information can be modeled using a step function that captures the fact that if certain fields are disclosed together sensitive information is revealed.

In scenarios as the one described above, the probability that adversaries who may collude will be assigned to the same task is minuscule due to the large number of anonymous available workers. Moreover, collusion is regulated by the HIPPAA's privacy regulations [5], which prevents workers from hiring subcontractors and sharing the data with other parties.

The second example illustrates how distributing a dataset allows one to achieve a particular task, while minimizing the disclosure of sensitive information.

1.2 Problem Definition and Contributions

Motivated by applications such as the ones presented above, we introduce the problem of *privacy-aware partitioning* of a dataset, where our goal is to partition a dataset among k untrusted parties and to maximize either user's utility, or the third parties' utilities, or a combination of those. Further, we would like to do this while minimizing the total amount of sensitive information disclosed.

Most of the previous work has either considered sharing *privacy-preserving summaries* of data, where the aggregate information about the population of users is preserved, or has bypassed the use of personal data and its disclosure to multiple advertisers [25, 22, 15]. These approaches focus on worst-case scenarios assuming arbitrary collusion among adversaries. Therefore, all adversaries are combined and treated as a single adversary. However, this strong threat model does not allow publishing of fine-grained information needed in many applications.

Other approaches have explicitly focused on online advertising, and have developed specialized systems that limit information disclosure by storing sensitive information on the user's side [14, 26]. While effective, the proposed techniques are not applicable to other data partitioning scenarios, like crowdsourcing or cloud computing. Finally, Krause et al. have studied how the disclosure of a subset of the attributes of a data entry can allow access to fine-grained information [18]. While considering the utility-disclosure tradeoff, their work does not take into account the interdependencies across different data entries and assumes a single adversary (third party).

In this work we propose SPARSI, a new framework that allows us to formally reason about leakage of sensitive information in scenarios such as the ones presented above, namely, setups where we are given a dataset to be partitioned among a set of non-colluding adversaries in order to obtain some utility. We consider a generalized form of utility that captures both the utility that each adversary obtains by receiving part of the data and the user's personal utility derived by fulfilling a task. We elaborate more on this generalization in the next section. This raises a natural tradeoff between maximizing the overall utility while minimizing information disclosure. We provide a formal definition of the privacy-aware data partitioning problem, as an optimization of the aforementioned tradeoff.

While non-collusion results in a somewhat weaker threat model, we argue that it is a reasonable and practical assumption in a variety of scenarios, including the ones discussed above. In setups like online advertising there is no particular incentive for adversaries to collude, due to conflicting monetary interests. Collusion in crowdsourcing and cloud computing scenarios is prevented by the corresponding privacy laws and two-party agreements. In crowdsourcing scenarios the probability that adversaries who may collude will be assigned to the same task is minuscule due to the large number of anonymous available workers. Attempts to collude can often be detected easily, and the possibility of strict penalization (by the crowdsourcing market) provides additional disincentive to collude.

Finally, in most of these situations the assumption of no collusion is a necessary one for accomplishing the task.

The main contributions of this paper are as follows:

- We introduce the problem of *privacy-aware data partitioning* across multiple adversaries, and analyze its complexity. To our knowledge this is the first work that addresses the problem of minimizing information leakage when partitioning a dataset across multiple adversaries.
- We introduce SPARSI, a rigorous framework based on the notion of *sensitive properties* that allows us to formally reason about how information is leaked and the total amount of information disclosure. We represent the interdependencies between data and sensitive properties using a *hypergraph* and we show how the problem of privacy-aware partitioning can be cast as an optimization problem.
- We analyze the problem for specific families of information disclosure functions, including step and linear functions, and show how good solutions can be derived by using *relaxation* techniques. Furthermore, we propose a set of algorithms, based on a generic *greedy randomized local search* algorithm, for obtaining approximate solutions to this problem under generic families of utility and information disclosure functions.
- Finally, we demonstrate how, using SPARSI, one can distribute user-location data, like in Example 1, to multiple advertisers while ensuring that almost no sensitive information about potential user friendship links is revealed. We experimentally verify the performance of the proposed algorithms for both synthetic and real-world datasets. We compare the proposed greedy local search algorithm against approaches tailored to specific disclosure functions, and show that it is capable of producing solutions that are close to the optimal.

2. SPARSI FRAMEWORK

In this section we start by describing the different components of SPARSI. More precisely, we show how one can formally reason about the sensitive information contained in a dataset by introducing the notion of *sensitive properties*. Then, we show how to model the interdependencies between data entries and sensitive properties rigorously, and how to reason about the leakage of sensitive information in a principled manner.

2.1 Data Entries and Sensitive Information

Let D denote the dataset to be partitioned among different adversaries. Moreover, let A denote the set of adversaries. We assume that D is comprised of data entries $d_i \in D$ that disclose minimal sensitive information if revealed alone. To clarify this consider Example 1 (Section 1.1) where each data entry is the check-in location of a user. The user is sharing this information voluntarily with the social network service in exchange for local advertisement services, hence, this entry is assumed not to disclose sensitive information. In Example 2, the data entries to be published are the fields of the prescriptions. Observe that if the disease field is revealed in isolation, no information is leaked about affected individuals.

However, revealing several data entries together discloses sensitive information. We define a *sensitive property* to be a property that is related to a subset of data entries but not explicitly represented in the data set, and that can be inferred if the data entries are collectively analyzed. Let P denote the set of sensitive properties that are related to data entries in D. To formalize this abstract notion of indirect information disclosure, we assume that each sensitive property $p \in P$ is associated with a variable (either numerical or categorical) V_p with true value v_p^*. Let $D_p \subset D$ be the smallest

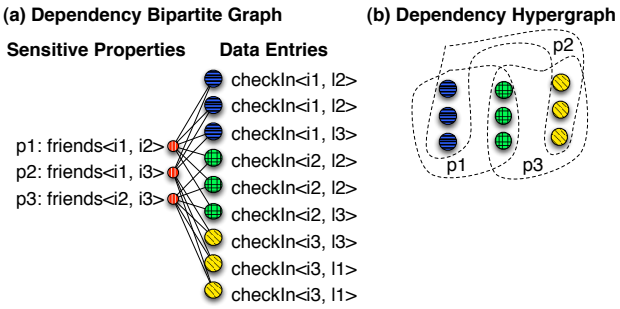

Figure 1: A dependency graph between data entries and sensitive properties.

set of data entries from which an adversary can infer the true value v_p^* of V_p with high confidence, if all entries in D_p are revealed to her. We assume that there is a unique such D_p corresponding to each property p. We say that data entries in D_p disclose information about property $p \in P$ and that information disclosure can be modeled as a function over D_p (see Section 2.2).

We assume that sensitive properties are specified by an expert and the dependencies between data entries in D and properties in P, via sets D_p, $\forall p \in P$, are represented as an undirected bipartite graph, called a *dependency graph*. Returning to the example applications presented above we have the following: In Example 1 the sensitive properties correspond to the friendship links between users, and the associated datasets D_p correspond to the check-in information of the pairs of users participating in friendship links. In Example 2, the sensitive properties correspond to the links between a patient's id and a particular disease, or a doctor's id and particular medication. In general, it has been shown that data mining techniques can be used to determine the dependencies between data items and sensitive properties [21].

Let $\mathcal{G}_d$ denote such a dependency graph. $\mathcal{G}_d$ has two types of nodes, i.e., nodes P that correspond to sensitive properties and nodes D that correspond to data entries. An edge connects a data entry $d \in D$ with a property $p \in P$ only if d can potentially disclose information about p. Alternatively, we can use an equivalent *hypergraph* representation, that is easier to reason about in some cases. Converting the dependency graph $\mathcal{G}_d$ into an equivalent *dependency hypergraph* is simply done by mapping each property node into a hyperedge. Figure 1 shows an example of a bipartite graph and its equivalent hypergraph, corresponding to the social-network scenario presented in Example 1 (Section 1.1). Since our goal is not to disclose any information about friendship links in the social network, each sensitive property correspond to a friendship link in the network. The dependencies between check-ins and friendship links are captured by the edges in the bipartite graph.

2.2 Information Disclosure

We model the information disclosed to an adversary $a \in A$ using a vector valued function $f_a : \mathcal{P}(D) \rightarrow [0,1]^{|P|}$, which takes as input the subset of data entries published to an adversary, and returns a vector of disclosure values; one per sensitive property. That is, $f_a(S_a)[i]$ denotes the information disclosed to adversary $a \in A$ about the ith property when a has access to the subset S_a of data entries. We assume that information disclosure takes values in $[0, 1]$, with 0 indicating no disclosure and 1 indicating full disclosure. Generic disclosure functions, including posterior beliefs, and distribution distances can be naturally represented by SPARSI.

We define the overall disclosure function f as an aggregate of the disclosure functions of all adversaries. Before presenting the formal definition, we define the *assignment set*, given as input to f.

DEFINITION 1 (ASSIGNMENT SET). *Let x_{da} be an indicator variable set to 1 if data entry $d \in D$ is published to adversary $a \in A$. We define the assignment set $\mathcal{S}$ to be the set of all variables x_{da}, i.e., $\mathcal{S} = \{x_{11}, \cdots, x_{1|A|}, \cdots, x_{|D||A|}\}$, and the assignment set S_a corresponding to an adversary $a \in A$ to be the set of indicator variables corresponding to adversary a, i.e., $\mathcal{S}_a = \{x_{1a}, x_{2a}, \cdots, x_{|D|a}\}$.*

Worst Disclosure. The overall disclosure can be expressed as:

$$f_\infty(\mathcal{S}) = max_{a \in A}(\|f_a(\mathcal{S}_a)\|_\infty) \quad (1)$$

Using the infinity norm accounts for the worst case disclosure across properties. Thus, full disclosure of at least one sensitive property suffices to maximize the information leakage. This function is indifferent to the total number of sensitive properties that are fully disclosed in a particular partitioning and gives the same score to all that have at least one fully disclosed property.

Average Disclosure. Considering the sensitive information associated with each sensitive property in isolation is natural in many real-world scenarios like Examples 1 and 2. However, there are cases where quantifying the amount of disclosed information requires reasoning over sensitive properties in a collective manner. We specifically consider the average function and introduce a variation of the overall disclosure function that considers the total disclosure per adversary. We replace the infinity norm in the equation above with the L_1 norm:

$$f_{L_1}(\mathcal{S}) = max_{a \in A}(\frac{\|f_a(\mathcal{S}_a)\|_1}{|P|}) \quad (2)$$

Observe that both Equation (1) and Equation (2) consider the maximum over the disclosure across adversaries, i.e., they can be written as:

$$f(\mathcal{S}) = max_{a \in A} f'_a(\mathcal{S}_a) \quad (3)$$

where $f'_a(\mathcal{S}_a) = \|f_a(\mathcal{S}_a)\|_\infty$ or $f'_a(\mathcal{S}_a) = \frac{\|f_a(\mathcal{S}_a)\|_1}{|P|}$.

For example, consider publishing single nucleotide polymorphisms (SNP) genotyping microarray data. One should avoid disclosing the identity of an individual whose trace is present in the data, as well as, any information related to particular genetic traits of that individual. Recent work has shown how one can determine whether individuals are in a complex genomic DNA mixture by analyzing SNP genotyping microarray data [16]. However, one also needs to hide any information related to particular genetic traits. To capture both kinds of sensitive information, sensitive properties need to model how SNP data entries are connected with different SNP groups related to different genetic traits. An adversary can infer the identity of an individual based on a function over all the sensitive properties (here groups of SNPs) rather than each individual sensitive property. Thus, it makes sense to also consider measures of overall disclosure that are not just worst case over the disclosures of each individual sensitive property.

2.3 Overall Utility

Let u denote the utility derived by partitioning the dataset across multiple adversaries. We have that $u : \mathcal{P}(D \times A) \rightarrow \mathbb{R}$, where $\mathcal{P}(D \times A)$ denotes the powerset of possible data-to-adversary assignments. Next we present how this utility function can be generalized to capture the adversaries' and publisher's utility.

As discussed before, an adversary's utility is obtained by acquiring part of the dataset D (see Example 1) and the publisher's utility is derived by fulfilling a particular task that requires partitioning the data (see Example 2). Under many real world examples these two different kinds of utility can be unified under a single utility. Consider Example 1. Typically, advertisers pay higher amounts for data that maximize their individual utility. Thus, maximizing the utility of each individual advertiser maximizes the utility (maybe monetary) of the data publisher as well.

Based on this observation we unify the two types of utilities under a single formulation based on the utility of adversaries. Intuitively, we would expect that the more data an adversary receives, the less the observation of a new, previously unobserved, data entry would increase the gain of the adversaries. This notion of *diminishing returns* is formalized by the combinatorial notion of *submodularity* and is shown to hold in many real-world scenarios [23, 18]. More formally, a set function $G : 2^V \rightarrow \mathbb{R}$ mapping subsets $A \subseteq V$ into the real numbers is called *submodular* [6], if for all $A \subseteq B \subseteq V$, and $v' \in V \setminus B$, it holds that $G(A \cup \{v'\}) - G(A) \geq G(B \cup \{v'\}) - G(B)$, i.e., adding v' to a set A increases G more than adding v' to a superset B of A. F is called *nondecreasing*, if for all $A \subseteq B \subseteq V$ it holds that $G(A) \leq G(B)$.

Let u_a be a set function that quantifies the utility of each adversary a. As mentioned above, we assume that u_a is a nondecreasing submodular function. For convenience we will occasionally drop the nondecreasing qualification in the remainder of the paper. Let U_A denote the set of all utility functions for a given set of adversaries A. Let u denote the overall utility associated with publishing the data (i.e., without considering the cost). The function u can be defined as an aggregate function of all utilities $u_a \in U_A$. For example the overall utility may be defined as a linear combination, i.e., a weighted sum, of all functions in U_A, following the form:

$$u(\mathcal{S}) = \sum\nolimits_{a \in A} w_a u_a(\mathcal{S}_a) \quad (4)$$

where $\mathcal{S}$ and $\mathcal{S}_a$ are defined in Definition 1. Because all functions in U_A are submodular, u will also be submodular, since it is expressed as a linear combination of submodular functions [10].

An example of a submodular function u_a is an additive function. Assume that each data entry in $d \in D$ has some utility w_{da} for an adversary $a \in A$. We have that $u_a(\mathcal{S}_a) = \sum_{d \in D} w_{da} x_{da}$, where x_{da} is an indicator variable that takes value 1 when data entry d is revealed to adversary a and 0 otherwise. For the remainder of the paper we will assume that utility u is normalized so that $u \in [0, 1]$.

3. PRIVACY-AWARE DATA PARTITIONING

In this section, we describe two formulations of the *privacy-aware partitioning* problem. We show how both can be expressed as maximization problems that are, in general, NP-hard to solve. We consider a dataset D that needs to be partitioned across a given set of adversaries A. We assume that the functions to compute the overall utility and information disclosure are given. Let these functions be denoted by u and f respectively.

In many cases publishing the data to adversaries also incurs a monetary cost (see Example 2). Ideally, we wish to maximize the utility while minimizing the disclosure and the cost; however, there is a natural tradeoff between the two optimization goals. A traditional approach is to set a requirement on information disclosure and cost while optimizing the utility. Accordingly we can define the following optimization problem.

DEFINITION 2 (DISCBUDGET). *Let D be a set of data entries, A be a set of adversaries, and τ_I be a budget on information disclosure. This formulation of the privacy-aware partitioning*

problem finds a data entry to adversary assignment set $\mathcal{S}$ that maximizes $u(\mathcal{S})$ under constraint $f(\mathcal{S}) \leq \tau_I$. More formally we have the following optimization problem:

$$\begin{aligned} \underset{\mathcal{S}\in\mathcal{P}(D\times A)}{\textit{maximize}} \quad & u(\mathcal{S}) \\ \textit{subject to} \quad & f(\mathcal{S}) \leq \tau_I, \\ & \sum\nolimits_{a=1}^{k} x_{da} \leq t, \forall d \in D, \\ & x_{da} \in \{0,1\}. \end{aligned} \tag{5}$$

where x_{da} and $\mathcal{S}$ are defined in Definition 1 as before and $t \geq 1$ is the maximum number of adversaries to whom a particular data entry can be published. The upper bound t is used to model cases where the number of assignments per data entry needs to be restricted due to cost, e.g., monetary cost in crowdsourcing or cloud-computing applications.

This optimization problem already captures our desire to reduce the information disclosure while increasing the utility. However, the optimization objective is agnostic to information disclosure. Considering only the utility in the objective, does not differentiate among solutions with the same utility that satisfy the disclosure constraint but have significantly different disclosures, i.e., there is no preference for solutions with smaller disclosure. To overcome this, we consider a different formulation of the privacy-aware data partitioning problem where we seek to maximize the difference between the utility and the information disclosure functions. We consider the Lagrangian relaxation of the previous optimization problem. We assume that both functions are measured using the same unit. We have the following:

DEFINITION 3 (TRADEOFF). *Let D be a set of data entries, A be a set of adversaries, and τ_I be a budget on information disclosure. This formulation of the privacy-aware partitioning problem finds a data entry to adversary assignment $\mathcal{S}$ that maximizes the tradeoff between the overall utility and the overall information disclosure, i.e., $u(\mathcal{S})+\lambda(\tau_I - f(\mathcal{S}))$, where λ is a nonnegative weight. More formally we have the following optimization problem:*

$$\begin{aligned} \underset{\mathcal{S}\in\mathcal{P}(D\times|A|)}{\textit{maximize}} \quad & u(\mathcal{S}) + \lambda(\tau_I - f(\mathcal{S})) \\ \textit{subject to} \quad & f(\mathcal{S}) \leq \tau_I, \\ & \sum\nolimits_{a=1}^{k} x_{da} \leq t, \forall d \in D, \\ & x_{da} \in \{0,1\}. \end{aligned} \tag{6}$$

where x_{da} and $\mathcal{S}$ are defined in Def. 1 and t is the maximum number of adversaries to whom a data entry can be published.

In both cases we modelled the cost using an upper bound on the number of assignments for each data entry. Both formulations can be extended to incorporate more sophisticated linear cost functions. For example, one can consider an additive cost model, defined as:

$$c(\mathcal{S}) = \sum\nolimits_{a\in A} \sum\nolimits_{d\in\mathcal{S}_a} c_{da} \tag{7}$$

where c_{da} denotes the cost of assigning data item d to adversary a and $\mathcal{S}$ and $\mathcal{S}_a$ are defined in Definition 1. In the case of TRADEOFF, the objective retains its structural properties, e.g., submodularity, since the cost function has a modular form [6]. We focus our discussion on Definitions 2 and 3 for clarity. We prove that both DISCBUDGET and TRADEOFF formulations of the privacy-aware data partitioning problem are NP-hard by reducing them to the problem of maximizing a submodular function under uniform matroid constraints [20].

THEOREM 1. *Both* DISCBUDGET *and* TRADEOFF *formulations of the privacy-aware data partitioning problem are, in general, NP-hard to solve.*

PROOF. The problem is in NP since the size of every feasible solution is polynomially bounded in the size of the given instance and the objective can be computed in polynomial time. We now show it is NP-hard. Consider the problem of maximizing a submodular function u over a set of items S under k uniform matroid constraints, where each matroid $M_i, \forall i \in [k]$ has rank r_i. We now reduce this NP-hard problem [20] to computing a solution for our problem.

We create the following instance of the privacy-aware data partitioning. Fix the set of adversaries A to contain a single adversary, i.e., $|A| = 1$, a set of data entries D such that $D = S$, and a set of sensitive properties P over the data entries in D, such that $|P| = k$ and the i-th property contains exactly $r_i + 1$ items from D. Let $D_i \subset D$ be the set of data entries associated with property $i \in P$. Moreover, let the utility function be u, and, let the information disclosure f be a step function of the following form: If all the data entries corresponding to a particular sensitive property $p \in P$ are revealed to the same adversary the disclosure is 1 otherwise 0. Based on this form of f our objective is equivalent to maximizing utility alone. In particular we can replace the upper disclosure threshold constraint in both formulations with the following linear constraint $\sum_{d\in D_i} x_{da} < |D_i|, \forall i \in P$. We also set $t = 1$. Clearly, this is a valid instance for privacy-aware partitioning. Notice that each of the linear constraints above can be expressed as a uniform matroid constraint. In fact the i-th constraint will correspond to a matroid of rank $|D_i| - 1 = r_i$. It is easy to see that a solution of this instance of the privacy-aware data partitioning problem is a valid solution for the problem of maximizing the submodular function u under k uniform matroid constraints. This completes the reduction. □

In the remainder of the paper we describe efficient heuristics for solving the partitioning problem – we present approximation algorithms for specific information disclosure functions in Section 4, and a greedy local-search heuristic for the general problem in Section 5. Moreover, we present an empirical comparison between DISCBUDGET and TRADEOFF in Section 6, showing that TRADEOFF yields solutions comparable to DISCBUDGET in terms of utility, but with significantly smaller disclosure. Due to space constraints, henceforth, we will only focus on the TRADEOFF formulation. All of our algorithms also work for the DISCBUDGET formulation (with minimal modifications).

4. ANALYZING SPECIFIC CASES OF INFORMATION DISCLOSURE

In this section, we present approximation algorithms when the information disclosure function takes the following special forms: 1) step functions, 2) linearly increasing functions. The utility function is assumed to be submodular.

4.1 Step Functions

Information disclosure functions that correspond to a step function can model cases when each sensitive property $p \in P$ is either fully disclosed or fully private. A natural application of step functions is the crowdsourcing scenario shown in Example 2. When certain fields of a medical transcription, e.g., name and diagnosis, or gender and the zip code and birth date, are revealed together, the corresponding sensitive property is disclosed. We now describe such functions formally. Let $D_p \subset D$ be the set of data entries associated with property $p \in P$. Property p is fully disclosed only if

D_p is published in its entirety to an adversary. This can be modeled using a set of step functions $f_a \in F$:

$$f_a(D_a)[p] = \begin{cases} 1 & \text{if } D_p \subseteq D_a \\ 0 & \text{if } D_p \not\subseteq D_a \end{cases}$$

Observe that information disclosure is minimized (and is equal to 0) when no adversary gets all the elements in D_p, for all p. For step functions we consider worst case disclosure, since ideally we do not want to fully disclose any property.

Considering DISCBUDGET and TRADEOFF formulations separately is not meaningful for step functions. Since disclosure can only take the extreme values $\{0, 1\}$, the maximum disclosure threshold, τ_I, should be set to 0 in TRADEOFF, as full disclosure of a property always penalizes the utility. One can reformulate the problem and seek for solutions that maximize the utility under the constraint that information disclosure is 0, i.e., no property exists such that all its data entries are published to the same adversary.

Given these families of information disclosure functions and a submodular utility function, both formulations of privacy-aware data partitioning can be represented as an *integer program (IP)*:

$$\begin{aligned} \underset{\mathcal{S}\in\mathcal{P}(D\times|A|)}{\text{maximize}} \quad & u(\mathcal{S}) \\ \text{subject to} \quad & \sum\nolimits_{d\in D_p} x_{da} < |D_p|, \forall p \in P, \forall a \in A, \\ & \sum\nolimits_{a=1}^{k} x_{da} \leq t, \forall d \in D, \\ & x_{da} \in \{0, 1\}. \end{aligned} \tag{8}$$

where t is the maximum number of adversaries to whom a particular data entry can be published.

The first constraint enforces that there is no full disclosure of a sensitive property. The partitioning constraint enforces that a data entry is revealed to no more than t adversaries. Solving the optimization problem in (8) corresponds to maximizing a submodular function under linear constraints.

For additive utility functions ($u = \sum_{a\in A}\sum_{d\in D} w_{da}x_{da}$), Equation (8) becomes an integer linear program, that can be approximately solved in polynomial time in two steps. First, one can solve a linear relaxation of Equation (8), where x_{da} is some fraction in $[0, 1]$. The resulting fractional solution can be converted into an integral solution using a *rounding strategy*.

The simplest rounding strategy, called *randomized rounding* [24], works as follows: assign data entry d to an adversary a with probability equal to $\hat{x}_{da}$, where $\hat{x}_{da}$ is the fractional solution to the linear relaxation. The value of the objective function for the derived integral solution is in expectation equal to the optimal value of the objective achieved by the linear relaxation. Moreover, randomized rounding preserves all constraints in expectation. A different rounding scheme, called *dependent rounding* [11], ensures that constraints are satisfied in the integral solution with probability 1. For an overview of randomized rounding techniques for budgeted problems we refer the reader to the work by Doerr et al. [6]

One can solve the general problem with worst-case approximation guarantees by leveraging a recent result on submodular maximization under multiple linear constraints by Kulik et al. [19].

THEOREM 2. *Let the overall utility function u be a nondecreasing submodular function. One can find a feasible solution to the optimization problem in (8) with expected approximation ratio of $(1-\epsilon)(1-e^{-1})$, for any $\epsilon > 0$, in polynomial time.*

PROOF. This holds by Theorem 2.1 of Kulik et al. [19]. □

To obtain this approximation ratio, Kulik et al. introduce a framework that first obtains an approximate solution for a continuous relaxation of the problem, and then uses a non-trivial combination of a randomized rounding procedure with two enumeration phases, one on the most *profitable elements*, and the other on the 'big' elements, i.e., elements with high cost. This combination enables one to show that the rounded solution can be converted to a feasible one with high expected profit. We refer the reader to Kulik et al. for a detailed description of the algorithm.

4.2 Linearly Increasing Functions

In this section, we consider linearly increasing disclosure functions. Linear disclosure functions can naturally model situtations where each data entry independently affects the likelihood of disclosure. In particular, if normalized log-likelihood is used as a measure of information disclosure, the disclosure of a property p takes the following additive form:

$$f_a(\cdot)[p] = \sum\nolimits_{d\in D_p} a_{dp}x_{da} \tag{9}$$

where a_{dp} is a weight associated with the information that is disclosed about property p when data d is revealed to an adversary.

We can rewrite the TRADEOFF problem as:

$$\begin{aligned} \underset{\mathcal{S}\in\mathcal{P}(D\times A)}{\text{maximize}} \quad & u(\mathcal{S}) + \lambda(\tau_I - \max_{a\in A}(f'_a(\mathcal{S}_a))) \\ \text{subject to} \quad & f(\mathcal{S}) \leq \tau_I, \\ & \sum\nolimits_{a=1}^{k} x_{da} \leq t, \forall d \in D, \\ & x_{da} \in \{0, 1\}. \end{aligned} \tag{10}$$

When the utility function is additive, the above problem is an integer linear program, and hence can be solved by rounding the LP relaxation as explained in the previous section.

For general submodular $u(\cdot)$, the objective is not submodular anymore, since it cannot be expressed as the difference between a submodular and a supermodular (i.e., $\max_{a\in A}(f'_a(\mathcal{S}_a))$) function. The maximum of additive information disclosure functions is not supermodular [10]. Unlike the case of step functions, we cannot use the result of Kulik et al. [19] to get an approximate solution.

Nevertheless, we can compute approximate solutions by considering the following max-min formulation of the problem:

$$\begin{aligned} \underset{\mathcal{S}\in\mathcal{P}(D\times A)}{\text{maximize}} \quad & \min_{a\in A}(u(\mathcal{S}) + \lambda(\tau_I - f'_a(\mathcal{S}_a))) \\ \text{subject to} \quad & f(\mathcal{S}) \leq \tau_I, \\ & \sum\nolimits_{a=1}^{k} x_{da} \leq t, \forall d \in D, \\ & x_{da} \in \{0, 1\}. \end{aligned} \tag{11}$$

Since the overall utility function is a nondecreasing submodular function, and the disclosure for each adversary is additive, the objective now is a max-min of submodular functions. For worst-case disclosure, the optimization objective can be rewritten as:

$$\underset{\mathcal{S}\in\mathcal{P}(D\times A)}{\text{maximize}} \quad \min_{a\in A, p\in P}(u(\mathcal{S}) + \lambda(\tau_I - f_a(\mathcal{S}_a)[p])) \tag{12}$$

and, for average disclosure (Equation (2)), it can be written as:

$$\underset{\mathcal{S}\in\mathcal{P}(D\times A)}{\text{maximize}} \quad \min_{a\in A}(u(\mathcal{S}) + \lambda(\tau_I - \frac{1}{|P|}\sum\nolimits_{p\in P} f_a(\mathcal{S}_a)[p]))$$

The above max-min problem formulation is closely related to the *max-min fair allocation* problem [13] for both types of information disclosure functions. The main difference between Problem (11) and the max-min fair allocation problem is that data items may be assigned to multiple advrsaries. In the max-min fair allocation problem a data item is assigned at most once. If $t \leq 1$ then the

two problems are equivalent, and thus, one can provide worst case guarantees on the quality of the approximate solution. The problem of max-min fair allocation was studied by Golovin [13] and Khot and Ponnuswami [17]. Let n denote the total number of data entries (*goods* in the max-min fair allocation problem) and m denote the number of adversaries (*buyers* in the max-min fair allocation problem). The first two papers focus on additive functions and give algorithms achieving an $(n - m + 1)$-approximation and a $(2m + 1)$-approximation respectively, while the third one gives a $\mathcal{O}(n^{\frac{1}{2}} m^{\frac{1}{4}} \log n \log^{\frac{3}{2}} m)$-approximation. Those algorithms could be directly used if $t \leq 1$. The local search heuristic that we present next, on the other hand, can be used in either case.

5. A LOCAL SEARCH HEURISTIC

So far we studied specific families of disclosure functions to derive worst-case guaranties for the quality of approximate solutions. In this section we present two greedy heuristic algorithms suitable for any general disclosure function. We still require the utility function to be submodular. Our heuristics are based on hill climbing and the *Greedy Randomized Adaptive Local Search Procedure* (GRASP) [9]. Local search heuristics are known to perform well when maximizing a submodular objective function [10].

5.1 Overall Algorithm

Our algorithm proceeds in two phases (Algorithm 1). The first phase, which we call *construction*, constructs a data-to-adversary assignment matrix M_{ini} by greedily picking assignments that maximize the specified objective function $G(\cdot)$, e.g., the tradeoff between utility and information disclosure for the TRADEOFF formulation, while ensuring that all disclosure and assignment constraints in the problem formulation are satisfied. The second phase, called *local-search*, searches for a better solution in the neighborhood of the M_{ini}, by changing one assignment of one data item at a time if it improves the objective function, resulting in an assignment M. The construction algorithm may be randomized; hence, the overall algorithm is executed r times, and the best solution $M_{opt} = \text{argmax}_{\{M_1,\ldots,M_r\}} G(M_i)$ is returned as the final solution.

Algorithm 1 Overall Algorithm

Input: A: set of adversaries; G: objective function; r: number of repetitions; t: max. adversaries per data item; τ_I: max. disclosure
Output: M_{opt}: a data-to-adversary assignment matrix
for all $i = 1 \rightarrow r$ **do**
  $M_\emptyset \leftarrow$ empty assignment , $g_{opt} \leftarrow G(M_\emptyset)$
  $\langle M_{ini}, g_{ini} \rangle \leftarrow \textsf{CONSTRUCTION}(G, A, t)$;
  $\langle M, g \rangle \leftarrow \textsf{LOCALSEARCH}(G, A, t, M_{ini}, g_{ini})$;
  if $g > g_{opt}$ **then**
    $M_{opt} \leftarrow M$; $g_{opt} \leftarrow g$;
return M_{opt};

5.2 Construction Phase

The construction phase (Algorithm 2) starts with an empty data-to-adversary assignment matrix and greedily adds a new $\langle d, a \rangle$ assignment to the mapping M if it improves the objective function. This is achieved by iteratively performing two steps. The algorithm first computes a set of candidate assignments S. For any data item d (which does not already have t assignments), and any adversary a, $\langle d, a \rangle$ is a candidate assignment if it does not appear in M.

Second, the algorithm picks the next best assignment from the candidates (using PICKNEXTBEST, Algorithm 3). We consider two methods for picking the next best assignment – GREEDY and GRASP. The GREEDY strategy picks $\langle d^\star, a^\star \rangle$ that maximizes the objective $G(M \cup \langle d^\star, a^\star \rangle)$. On the other hand, GRASP identifies a set S_n of top n assignments that have the highest value for the objective $g_{\langle d,a \rangle} = G(M \cup \langle d, a \rangle)$, such that $g_{\langle d,a \rangle}$ is greater than the current value of the objective g. Note that S_n can contain less than n assignments. The GRASP strategy picks an assignment $\langle d^\star, a^\star \rangle$ at random from S_n. Both strategies return NULL if $\langle d^\star, a^\star \rangle$ does not improve the value of the objective function. The construction stops when no new assignment can improve the objective function. The run time complexity of the construction phase is $O(t \cdot |A| \cdot |D|^2)$. There are $O(t \cdot |D|)$ iterations, and each iteration may have a worst case running time of $O(|D| \cdot |A|)$. PICKNEXTBEST permits a simple parallel implementation.

Algorithm 2 CONSTRUCTION

Input: G: objective function; A: set of adversaries; t: max. adversaries per data item; τ_I: max. disclosure
Output: $\langle M, g \rangle$: data-to-adversary assignment, objective value
$maxIterations \leftarrow t \cdot |D|$
Initialize: $M \leftarrow$ empty assignment
for all $i \in [1, maxIterations]$ **do**
  // Compute a set of candidate assignments
  $D_M \leftarrow$ data entries assigned to $< t$ adversaries in M;
  Let $S \leftarrow D_M \times A - M$
  // Pick the next best assignment that improves the objective
  $\langle d, a \rangle \leftarrow \textsf{PICKNEXTBEST}(M, g, S, G)$
  if $\langle d, a \rangle$ is NULL **then**
    break; *// No new assignments improve the objective*
  Assign the selected data entry d to the selected adversary a;
return $\langle M, G(M) \rangle$;

Algorithm 3 PICKNEXTBEST

Input: G: objective function; M: current assignment; g: current value of objective, S: possible new assignments
Output: new assignment $\langle d^\star, a^\star \rangle$, or NULL
GREEDY:
  $\langle d^\star, a^\star \rangle \leftarrow \text{argmax}_{\langle d,a \rangle \in S} G(M \cup \langle d, a \rangle)$
GRASP:
  Pick the top-n assignments S_n having the highest value for $g_{\langle d,a \rangle} = G(M \cup \langle d, a \rangle)$ from S, and having $g_{\langle d,a \rangle} > g$ and $f_{\langle d,a \rangle} < \tau_I$.
  $\langle d^\star, a^\star \rangle$ is drawn uniformly at random from S_n
if $G(M \cup \langle d, a \rangle) > g$ **then**
  return $\langle d, a \rangle$
else
  return NULL

5.3 Local-Search Phase

The second phase employs local search (Algorithm 4) to improve the initial assignment M_{ini} output by the construction phase. In this phase, the data items are considered exactly once in some (random) order. Given the current assignment M, for each data item, a set of neighboring assignments N_d (including M) are considered by (i) removing an assignment to an adversary a in M, (ii) modifying the assignment from adversary a to an adversary a' (that d was not already assigned to in M), and (iii) adding a new assignment (if d is not already assigned to t adversaries in M). Next, the neighboring assignment in N_d with the maximum value for the objective M_{opt} is picked. The next iteration considers the data item succeeding d (in the ordering) with M_{opt} as the current assignment.

Algorithm 4 LOCALSEARCH

Input: G: objective function; A: set of adversaries; t: max. assignments per data item; M: current assignment; g: current objective value
Output: $\langle M_{opt}, g_{opt} \rangle$: the new assignment, the corresponding objective value
for all $d \in D$ **do**
  $A_d \leftarrow$ the set of adversaries to whom data item d is assigned (according to current assignment M);
  // Construct a set of neighboring assignments
  $N_d \leftarrow \{M\}$.
  if ($|A_d| < t$) **then** $N_d \leftarrow N_d \cup \{M \cup \{\langle d, a' \rangle\} | \forall a' \notin A_d\}$;
  for each adversary $a \in A_d$ **do**
    $N_d \leftarrow N_d \cup \{M - \{\langle d, a \rangle\}\}$
    $N_d \leftarrow N_d \cup \{M - \{\langle d, a \rangle\} \cup \{\langle d, a' \rangle\} | \forall a' \notin A_d\}$;
  // Pick the neighboring assignment with maximum objective
  // such that the upper disclosure is satisfied
  $M_{opt} \leftarrow \text{argmax}_{M' \in N_d} G(M')$
  $M \leftarrow M_{opt}$
return $\langle M_{opt}, G(M_{opt}) \rangle$;

We found that making a second pass of the dataset in the local search phase does not improve the value of the objective function. The run time complexity of the local-search phase is $O(t \cdot |A| \cdot |D|)$.

5.4 Extensions

The construction phase (Algorithm 2) runs in quadratic time in the size of D. This is because in each iteration, the PICKNEXTBEST subroutine computes a *global* maximum assignment across all data-adversary pairs. While this approach makes the algorithm more effective in avoiding local minima it reduces its scalability due to its quadratic cost. The quadratic complexity of the algorithm can be limiting when partitioning large datasets.

To improve scalability, we propose using a *local* myopic approach during construction. Instead of considering all possible (data,adversary) pairs when constructing the list of candidate assignments (see Ln. 8 in Algorithm 2), one can consider a single data entry d and populate the set of candidate assignments S using only (data,adversary) pairs that contain d. We fix a total ordering of the data entries $\mathcal{O}$, and perform t iterations of the following:

- Consider the next data item d in $\mathcal{O}$. Let M be the current assignment.
- Construct S as $(\{d\} \times A) - M$.
- Pick the next best assignment in S that improves the objective function using Algorithm 3 (GREEDY or GRASP).
- Update the current assignment M, and proceed with the next data entry in $\mathcal{O}$.

We compare the performance of the two versions of the proposed algorithmic framework in Section 6. The run time complexity of the myopic-construction phase is $O(t \cdot |A| \cdot |D|)$.

5.5 Discussion

The correctness of the algorithms stems directly from the fact that both construction and local search ensure that each data item is assigned to no more that t adversaries, and that the upper bound on disclosure is satisfied. The algorithms are generic enough to handle arbitrary utility and disclosure functions, as well as, constraints on cost. As mentioned before we focus on the problem formulations presented in Section 3 for clarity.

6. EXPERIMENTS

In this section we present an empirical evaluation of SPARSI. The main questions we seek to address are: (1) how the two versions of the privacy-aware partitioning problem – TRADEOFF and DISCBUDGET– compare with each other, and how well they exploit the presence of multiple adversaries with respect to disclosure and utility, (2) how the different algorithms perform in optimizing the overall utility and disclosure, and (3) how practical SPARSI is for distributing real-world datasets across multiple adversaries.

We empirically study these questions using both real and synthetic datasets. First we discuss the experimental methodology, and then we describe the data and results that demonstrate the effectiveness of our framework on partitioning sensitive data. The evaluation is performed on an Intel(R) Core(TM) i7 3.7 GHz/64bit/32GB machine. SPARSI is implemented in MATLAB, using MOSEK, a commercial optimization toolbox. The local-search procedures used for the real-world data experiments are implemented in Java.

Algorithms: We evaluate the following algorithms:

- RAND+: Each data entry is assigned to exactly t adversaries. The probability of assigning a data entry to an adversary is proportional to the corresponding utility weight w_{da}. We run the random partitioning a 100 times, and select the data-to-adversary assignment that maximizes the objective function. Random has been shown to be a highly effective heuristic for extensions of the max-cut problem [12], and is thus a natural heuristic to solve this problem.
- LP: We solve the LP relaxation of the optimization problems for step (Section 4.1) and linear (Section 4.2) disclosure functions. We generate an integral solution from the resulting fractional solution using naive randomized rounding (see Section 4.1), where the constraints are satisfied in expectation. Moreover, we perform a second pass over the derived integral solution to guarantee that the cardinality constrains are satisfied. This is a naive, yet effective, rounding scheme because the fractional solutions we get are close to the integral ones. More sophisticated rounding techniques can be used [6]. We run the rounding 100 times and select the data-to-adversary assignment with maximum value of objective.
- ILP: We solve the exact ILP algorithm for step and linear disclosure functions.
- GREEDY: Algorithm 1 with GREEDY strategy for picking a candidate
- GRASP: Algorithm 1 with GRASP strategy for picking the candidate assignments using $n = 5$ and $r = 10$.
- GREEDYL: Local myopic variant of Algorithm 1 (see Section 5.4 with GREEDY strategy for picking a candidate).
- GRASPL: Local myopic variant of Algorithm 1 (see Section 5.4 with GRASP strategy for picking candidates) using $n = 3$ and $r = 10$.

Evaluation: We use the following metrics for evaluation: (1) the total utility u corresponding to the final assignment, (2) the information disclosure f for the final assignment and (3) the tradeoff between utility and disclosure, given by $u + \lambda(\tau_I - f)$. While utility and disclosure take values in $[0, 1]$, the tradeoff can be greater than 1. We evaluate the different algorithms using different step and linear information disclosure functions for TRADEOFF.

For all experiments we set $\lambda = 1$ and assume an additive utility function of the form $u_a(\mathcal{S}_a) = \frac{\sum_{d \in D} w_{da} x_{da}}{\sum_{d \in D} \sum_{\text{top-t}(A) \in A} w_{da}}$, where x_{da} is an indicator variable that takes value 1 when data entry d

is revealed to adversary a and 0 otherwise, and $\mathsf{top-t(A)}$ returns the top t adversaries with respect to weights w_{da}. Observe that the normalization used corresponds to the maximum total utility a valid data-to-adversary assignment may have, when ignoring disclosure. Using this value ensures that the total utility and the quantity $\tau_I - f$ have the same scale $[0, 1]$. For convenience we fixed the upper information disclosure to $\tau_I = 0$. Finally, for $RAND+$ we perform 10 runs and report the average, while for LP we perform the rounding procedure 10 times and report the average. The corresponding standard errors are shown as error bars in the plots below.

6.1 Real Data Experiments

First we examine how SPARSI can be applied to real world domains. We consider a social networking scenario as discussed in Example 1. We desire to distribute the check-in information to advertisers, while minimizing the information we disclose for the structure of the network. We use the *Brighkite* dataset published by Cho et al. [4]. This dataset was extracted from Brightkite, a former location-based social networking service provider where users shared their locations by checking-in.

Each check-in entry contains information about the id of the user, the timestamp and location of the check-in. In addition the dataset contains the friendship network of users. The original dataset contains 4.5 million check-ins, $58,228$ users and $214,078$ edges. We run experiments both on the full dataset and on a sample, denoted as BK-full and BK-sample respectively. For BK-sample we subsample the dataset and extract $365,907$ check-ins. The corresponding friendship network contains $3,266$ nodes and $2,935$ edges.

Utility Weights: We start by modeling the utility. Recall that each check-in contains information about location. We assume a total number of k advertisers, and that each adversary is interested in check-in entries that occurred in a certain geographical area. Given an adversary $a \in A$, we draw w_{da} from a uniform distribution $\mathcal{U}(0.8, 1)$ for all entries $d \in D$ that satisfy the location criteria of the adversary, and we set $w_{da} = 0.1$ otherwise. We simulate this process by performing a random partitioning of the location ids across adversaries. We assume an additive utility function.

Sensitive Properties and Information Disclosure: The sensitive property, here, is the structure of the social network. We want to minimize the information leaked about the existence of any friendship link among users. Each friendship link can be associated with a sensitive property. Now, we examine how check-in entries leak information about the presence of a friendship link. Cho et al. [4] proved that there is a strong correlation between the trajectory cosine similarity and the existence of a friendship link for two users. Because of this strong correlation we assume that the information leakage for a sensitive property, i.e., the link between a pair of users, is equal to the trajectory similarity.

Let $D_a \subset D$ be the check-in data published to adversary $a \in A$, and U denote the set of users referenced in D_a. Given a sensitive property $p = e(u_i, u_j),\ u_i, u_j \in U, i \neq j$ we have that the information disclosure for p is:

$$f(D_a)[p] = \mathsf{CosineSimilarity}(D_a(u_i), D_a(u_j)) \qquad (13)$$

where $D_a(u_i)$ and $D_a(u_j)$ denote the set of check-in data for users u_i and u_j respectively. We aggregate the given check-ins based on their unique pairs of users and locations. We extract $15,661$ and $98,058$ data entries for the subsampled and entire dataset respectively. The entries contain the user id, the location and the number of times that user visited that particular location. Cosine similarity is computed over these counts.

Results: We aim to minimize the information leaked about all edges in the entire network. We consider the average case information disclosure, and we solve the corresponding TRADEOFF problem with $\tau_I = 0.5$ and $t = 2$. Due to cosine similarity we are limited to using RAND+ and the local-search heuristics. We run experiments for $|A| \in \{2, 3, 5, 7, 10\}$.

Table 1: Runtime and percentage of unpublished data for GREEDYL, GRASPL, GREEDY and GRASP for BK-sample.

Alg.	Metric	k=2	k=3	k=5	k=7	k=10
GREEDYL	runtime (sec.)	87.6	82.8	100	42	14.5
	unpub. data (%)	0	0	0	0	0
GRASPL	runtime (sec.)	307	301	280	145	41.5
	unpub. data (%)	0	0	0	0	0
GREEDY	runtime (sec.)	2761	3095	3726	4323	4665
	unpub. data (%)	6.4	0.4	0	0	0
GRASP	runtime (sec.)	8181	10095	12495	14588	18296
	unpub. data (%)	6.5	0.6	0	0	0

First we consider BK-sample. We evaluate the performance of RAND+, GREEDYL, GRASPL, GREEDY and GRASP. The corresponding utility and disclosure are shown in Figure 2. GRASP and GREEDY outperform RAND+ for all adversaries, since they return solutions with utility close to one and almost zero disclosure. Only for $k = 2$, RAND+ returns a solution with higher utility but the corresponding *average* disclosure is above 0.6. Recall that we compute the average over multiple runs of RAND+, where we consider the best solution reported over multiple executions.

Next we focus on the myopic versions of the proposed local-search procedure, i.e., GREEDYL and GRASPL. When the number of adversaries is small the corresponding utility is significantly lower than the one returned by RAND+, GREEDY and GRASP. This is expected since both algorithms give particular emphasis to minimizing information disclosure due to the tradeoff formulation of the optimization objective. As the number of adversaries increases, both algorithms can exploit the structure of the problem better, and offer solutions with higher utility values than RAND+ and comparable values to GREEDY and GRASP. Notice, that the disclosure is again close to zero.

For completeness we report the runtime and percentage of non-published data entries (i.e., number of data entries not published divided by the total number of data entries in the dataset) for the different local-search algorithms. The results are shown in Table 1. GREEDYL and GRASPL are significantly faster than GREEDY and GRASP. This is expected due to their linear complexity, versus the quadratic complexity of GREEDY and GRASP. We observe that when the number of adversaries is small, GREEDY and GRASP do not publish all the data. Nevertheless, the percentage of non-published entries is significantly small. Their myopic alternatives, i.e., GRASPL and GREEDYL, publish everything but that leads to higher disclosure and lower utility for small number of adversaries. We see that the myopic versions of the proposed local-search procedures are more time efficient and return solutions of comparable quality with the corresponding global versions.

Due to their quadratic complexity GREEDY and GRASP are not efficient for large datasets like BK-full. However, their myopic variations, GREEDYL and GRASPL (see Section 5.4) are very efficient due to their linear complexity. Due to this, we only evaluate the performance of RAND+, GRASPL and GREEDYL on BK-full. Again, we consider the TRADEOFF version of the problem with $\tau_I = 0.5$ and $t = 2$. The corresponding utility and disclosure are shown in Figure 3. We observe similar performance for the local-search procedures. The runtime for GREEDYL ranges from 1017 to 4459 sec., while for GRASPL it ranges from 3683 to 13175 sec. Finally, we report the objective value (i.e., $u + (\tau_I - f)$) for the solutions after running the construction phase only and after running both the construction and local-search phase. The corresponding

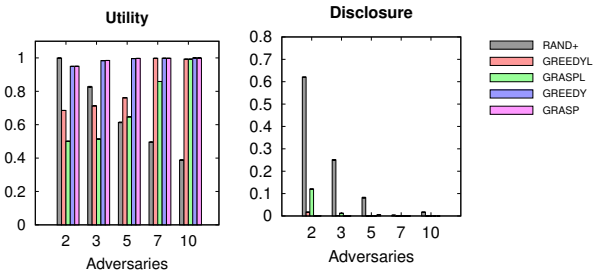

Figure 2: Utility and disclosure for BK-sample. The proposed local-search algorithms clearly outperform RAND+.

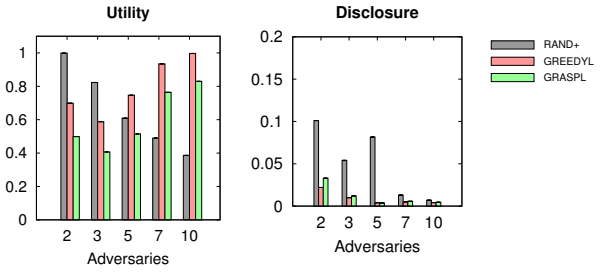

Figure 3: Utility and disclosure for BK-full. The proposed local-search algorithms clearly outperform RAND+.

Table 2: Objective value for construction only versus construction with localsearch for GREEDYL and GRASPL. The results correspond to BK-full.

Objective Value						
Alg.	Metric	k=2	k=3	k=5	k=7	k=10
GREEDYL	cstr.	1.276	1.067	0.878	0.823	0.829
	cstr.+ loc. srch	1.276	1.068	0.886	0.895	0.829
GRASPL	cstr.	1.066	0.885	0.733	0.691	0.743
	cstr.+ loc.srch	1.066	0.886	0.734	0.785	0.743

results are shown in Table 2. We observe that in some cases (i.e., for $k = 5$ and $k = 7$) there is an improvement in the quality of the solution. For example for $k = 7$ we see 8% and 13% improvement for GREEDYL and GRASPL respectively. Similar results were observed for BK-sample and synthetic data but are not reported due to space limitations.

6.2 Synthetic Data Experiments

Next, we use synthetically generated data to understand the properties of different disclosure functions and the performance of the proposed algorithms better. There are two data-related components in our framework: (1) a hypergraph that describes the interactions between data entries and sensitive properties (see Section 2.1), (2) a set of weights w_{da} representing the utility received when data entry $d \in D$ is published to adversary $a \in A$. The synthetic data are generated as follows. First, we set the total number of data entries $|D| \in \{50, 100, 200, 300, 500\}$, the total number of sensitive properties $|P| \in \{5, 10, 50, 100\}$, and the total number of adversaries $|A| \in \{2, 3, 5, 7, 10\}$.

For simplicity we consider an additive utility function as shown in Equation (4). Despite being a simple modeling choice, this utility function clearly illustrates the efficiency of the proposed techniques. Next, we describe the scheme we used to generate the utility weights w_{da}. There are two particular properties that need to be satisfied by such a scheme. The first one is that assigning any entry to an adversary should induce some minimum utility, since it allows us to fulfil the task under consideration (see Example 2). The second one is that there are cases where certain data items should induce higher utilities when assigned to specific adversaries, e.g., some workers may have better accuracy than others in crowdsourcing, or some advertisers may pay more for certain types of data.

The utility weights need to satisfy the aforementioned properties. To achieve this, we first choose a minimum utility value $u_{\min}$ from a uniform distribution $\mathcal{U}(0, 0.1)$. Then, we iterate over all possible data-to-adversary assignments and set the corresponding weight w_{da} to a value drawn from a uniform distribution $\mathcal{U}(0.8, 1)$ with probability p_u, or to $u_{\min}$ with probability $1 - p_u$. For our experiments we set the probability p_u to 0.4. Notice that both properties are satisfied. Finally, weights are scaled down by dividing them by the number of adversaries $|A|$.

Next, we describe how we generate a random hypergraph $H = (X, E)$, with $|X| = |D|$ and $|E| = |P|$, describing the interaction between data entries and sensitive properties. To create H we simply generate an equivalent bipartite dependency graph G (see Section 2.1) and convert that to the equivalent dependecy hypergraph. We iterate over the possible data to sensitive property pairs and insert the corresponding edge to G with probability p_f. For our experiments we set p_f to 0.3.

We examine the behavior of the proposed algorithms under several scenarios, varying the properties of the dataset, the number of adversaries and the family of disclosure functions considered.

Step Functions: First, we consider step functions. Under this family of disclosure functions, both DISCBUDGET and TRADEOFF correspond to the same optimization problem (see Section 4.1). Assuming feasibility of the optimization problem, information disclosure will always be zero. Thus, considering the total utility alone is sufficient for comparing the different algorithms.

First, we fix the number of data entries in the dataset to be $|D| = 500$ and consider values of $|P|$ in $\{5, 10, 50, 100\}$. Figure 4 shows the utility derived by the data-to-adversary assignment corresponding to different algorithms for $|P| = 50$. As depicted, all algorithms that exploit the structure of the dependency graph (i.e., LP, GREEDY, GREEDYL, GRASP and GRASPL) outperform RAND+. In most cases, LP, GREEDYL, GREEDY and GRASP were able to find the optimal solution that the ILP reported.

GRASPL finds solutions with non-optimal utilities, that are still better than RAND+. Comparing GRASP and GRASPL, we conjecture that while randomization is helpful in the case of non-myopic local-search it leads to worse solutions (with respect to utility) when keeping a myopic view on the optimization objective. The reported numbers correspond to no information disclosure, while missing values correspond to full disclosure of at least one sensitive property. For instance, the LP failed to find a valid solution for $k = 2$. Similar results were observed for different values of $|D|$ and $|P|$.

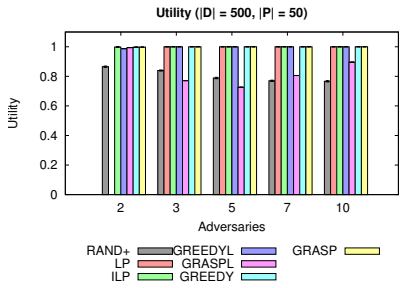

Figure 4: Utility for step disclosure functions. Reported numbers correspond to no information disclosure; missing values to full disclosure.

All local-search algorithms explicitly satisfy the upper disclosure constraint. To enforce this, the algorithm may avoid publishing some data. In our experiments, however, all algorithms assign all data items to at least one adversary. The experiments above show that GREEDYL, GREEDY and GRASP are viable alternatives for the case of step functions.

Linear Functions: Figure 5 compares the optimal ILP solution to DISCBUDGET and TRADEOFF on a synthetic instance of the problem by setting $|D| = 50$ and $|P| = 10$, and we run ILP for $|A| = \{2, 3, 5, 7, 10\}$, setting the maximum allowed disclosure to $\tau_I = 0.9$ and $t = 2$. We consider both worst and average disclosure. For both versions of the optimization problem the disclosure is decreasing as the number of adversaries increases. However, TRADEOFF exploits the presence of multiple adversaries better, to reduce disclosure while maintaining high utility.

Subsequently, we evaluate RAND+, LP, GREEDYL, GRASPL, GREEDY and GRASP on TRADEOFF with $\tau_I = 0.6$. We do not report any results for the ILP since for $|D| > 50$, it was not able to terminate in reasonable time. First, we fix the number of properties to $|P| = 50$ and consider instances with $|D| = \{100, 200, 300, 500\}$, and consider worst case disclosure. The performance of the algorithms for worst case disclosure and $|D| = 500$ is shown in Figure 6. As shown, LP, GREEDY and GRASP outperform RAND+ both in terms of utility and disclosure. RAND+ performs poorly as it returns solutions with higher disclosure but lower utility than the LP. Furthermore, we see that the performance gap between the local-search algorithms and RAND+ keeps increasing as the number of adversaries increases. This is expected as the proposed heuristics consider the structure of the underlying dependency graph, and can exploit the presence of multiple adversaries to achieve higher utility and lower disclosure.

We see that solutions derived using the LP approximation have maximum utility, while solutions derived using the proposed local-search algorithms have minimal disclosure. As presented in Figure 6 using the myopic construction returns solutions with low utility. This is expected since the algorithm does not maintain a global view of the data-to-adversary assignment. Observe that randomization improves the quality of the solution with respect to utility, when the global-view construction is used (see GREEDY and GRASP). Again, like in Section 6.1, when the myopic construction is used, randomization gives solutions of lower quality.

The worst disclosure is an indicator of the overall performance of the proposed algorithms. However, it does not provide us with detailed feedback about the information disclosure across all properties for the different algorithms. To understand this better, we measure the total number of properties that exceed a particular disclosure level. We present the corresponding plots for $|D| = 500$, $|P| = 50$, $k = 2$ and $k = 7$ in Figure 7. As shown, the proposed search-algorithms can exploit the presence of multiple adversaries effectively to minimize disclosure. Comparing Figures 7(a) and 7(b) we observe that the number of properties for which disclosure exceeds a particular threshold reaches zero for a significantly smaller thresholds for $k = 7$.

Figure 8 illustrates the percentage of unpublished data items for $|D| = 500$, $|P| = 100$, with τ_I taking values in $\{0.2, 0.4, 0.6, 0.8\}$, and considering 2 adversaries. We observe that LP, GREEDYL, and GRASPL retain the same level of unpublished items for different disclosure thresholds, while GRASP and GREEDY are able to find solutions where all data points are published to the adversaries, as the disclosure threshold increases. For $k > 2$ the number of unpublished data items for all algorithms drops to zero.

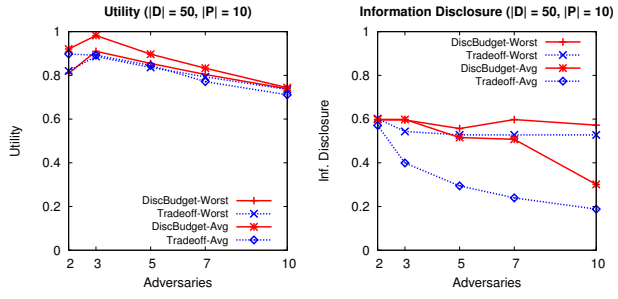

Figure 5: The (a) utility and (b) disclosure when solving DISCBUDGET and TRADEOFF optimally. TRADEOFF reduces disclosure more effectively while maintaining high utility.

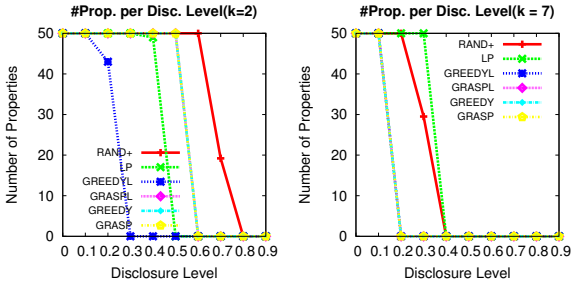

Figure 7: Properties exceeding a particular disclosure level for $|D| = 500$ and $|P| = 50$.

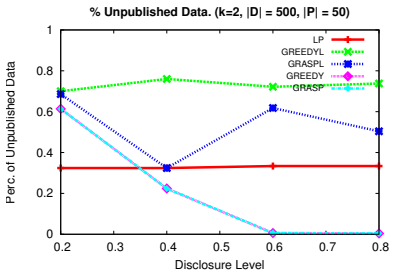

Figure 8: Unpublished data items for different disclosure levels.

7. RELATED WORK

There has been much work on the problem of publishing sensitive datasets [3, 8]. Information disclosure is characterized by a privacy definition, expressed either as syntactic constraints on the output dataset (e.g., k-anonymity [25] or ℓ-diversity [22]), or as constraints on the publishing algorithm (e.g., ϵ-differential privacy [8]). Each privacy definition is associated with a privacy level (k, ℓ, ϵ, etc.) that represents a bound on the information disclosure. Typical algorithmic techniques for data publishing, which include generalization, or coarsening of values, suppression, output perturbation, and sampling, attempt to maximize the utility of the published data given some level of privacy (i.e., a bound on the disclosure). Krause et al. [18] consider the trade-off between utility and disclosure, considering general submodular utility and supermodular disclosure functions. They formulate a submodular optimization problem, and present efficient algorithm for solving it. However, all the above techniques assume publishing data to a single adversary. Even under the presence of multiple parties, prior work makes a worst-case assumption of arbitrary collusion. In this paper, we formulate the novel problem of multiple non-colluding adversaries, and develop near-optimal algorithms for trading-off utility for information disclosure in this setting.

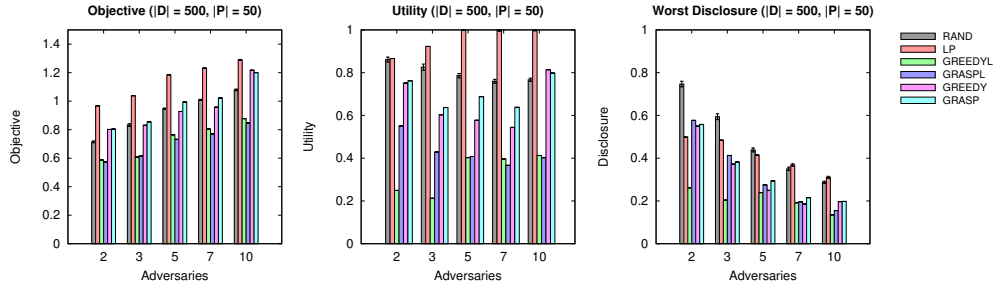

Figure 6: Tradeoff objective, utility and disclosure for linear functions considering worst disclosure. LP, GREEDY and GRASP outperform RAND+. LP returns the maximum utility, while the local-search heuristics return the minimum disclosure.

8. CONCLUSIONS AND FUTURE WORK

Increasing amounts of sensitive information are released on the Web and processed by online services, naturally raising concerns related to privacy in domains where detailed and fine-grained information must be published. Motivated by applications like online advertising and crowd-sourcing markets, we introduced the problem of privacy-aware data partitioning, namely, the problem of splitting a sensitive dataset amongst untrusted parties. We presented SPARSI, a theoretical framework that allows us to formally define the problem as an optimization of the tradeoff between the utility derived by publishing the data and the maximum information disclosure incurred to any single adversary. We proved that solving it is NP-hard and presented a performance analysis of different approximation algorithms for a variety of synthetic and real-world datasets, and demonstrated how SPARSI can be applied in the domain of online advertising. Our algorithms were able to partition user-location data to multiple advertisers while ensuring that almost no sensitive information about potential friendship links of these users can be inferred by any advertiser.

Our research has raised several interesting research directions. To our knowledge, this is the first work that leverages the presence of multiple adversaries to minimize information disclosure while maximizing utility. While we provided worst case guarantees for several families of disclosure functions, an interesting future direction is to examine if rigorous guarantees can be provided for other widely-used information disclosure functions like information gain. Finally, it is of particular interest to consider how the proposed framework can be extended to interactive scenarios where data is published to adversaries based on their queries, or in streaming data where the partitioning must be done in an online manner.

Acknowledgements: We would like to thank the anonymous reviewers for their comments. This work was supported by the National Science Foundation under Grants No. 0916736 and No. 1253327 and a gift from Google.

9. REFERENCES

[1] Brightkite: A location based social network. `http://en.wikipedia.org/wiki/Brightkite`.

[2] Gowalla: A location based social network. `http://en.wikipedia.org/wiki/Gowalla`.

[3] B.-C. Chen, D. Kifer, K. Lefevre, and A. Machanavajjhala. Privacy-preserving data publishing. *Foundations and Trends in Databases*, 2(1-2):1–167, 2009.

[4] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, pages 1082–1090, 2011.

[5] M. Davino. Assessing privacy risk in outsourcing. *American Health Information Management Association*, vol. 75, 2004.

[6] B. Doerr, M. Künnemann, and M. Wahlström. Randomized rounding for routing and covering problems: experiments and improvements. In *SEA*, 2010.

[7] Y. Duan, N. Youdao, J. Canny, and J. Zhan. P4p: Practical large-scale privacy-preserving distributed computation robust against malicious users abstract. In *Proceedings of the 19th USENIX Security Symposium*, page 14, 2010.

[8] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

[9] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[10] S. Fijishige. *Submodular functions and optimization*. Annals of Discrete Mathematics. Elsevier, 2005.

[11] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360, 2006.

[12] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[13] D. Golovin. Max-min fair allocation of indivisible goods. Technical report, CMU, 2005.

[14] S. Guha, B. Cheng, and P. Francis. Privad: practical privacy in online advertising. In *NSDI*, page 13, 2011.

[15] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. *CoRR*, 2010.

[16] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genet*, 4, 2008.

[17] S. Khot and A. K. Ponnuswami. Approximation algorithms for the max-min allocation problem. In *APPROX/RANDOM*, 2007.

[18] A. Krause and E. Horvitz. A utility-theoretic approach to privacy and personalization. In *AAAI*, pages 1181–1188, 2008.

[19] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, pages 545–554, 2009.

[20] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *STOC*, pages 323–332, 2009.

[21] T. Li and N. Li. Injector: Mining background knowledge for data anonymization. In *ICDE*, pages 446 – 455, 2008.

[22] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[23] A. Marshall. *Principles of Economics*. 1890.

[24] P. Raghavan and C. Tompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[25] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[26] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *NDSS*, 2010.

[27] K. Zhang, X. Zhou, Y. Chen, X. Wang, and Y. Ruan. Sedic: privacy-aware data intensive computing on hybrid clouds. In *CCS*, pages 515–526, 2011.