Constrained Obfuscation of Relational Databases

Erez Shmueli[a,b], Tomer Zrihen[a,b], Ran Yahalom[b], Tamir Tassa[c]

[a]*Department of Information Systems Engineering, Ben-Gurion University, Be'er Sheva, Israel*
[b]*Deutsche Telekom Laboratories, Ben-Gurion University, Be'er Sheva, Israel*
[c]*Department of Mathematics and Computer Science, The Open University, Ra'anana, Israel*

Abstract

The need to share data often conflicts with privacy preservation. Data obfuscation attempts to overcome this conflict by modifying the original data while optimizing both privacy and utility measures. In this paper we introduce the concept of Constrained Obfuscation Problems (COPs) which formulate the task of obfuscating data stored in relational databases. The main idea behind COPs is that many obfuscation scenarios can be modeled as a data generation process which is constrained by a predefined set of rules. We demonstrate the flexibility of the COP definition by modeling several different obfuscation scenarios: Production Data Obfuscation for Application Testing (PDOAT), anonymization of relational data, and anonymization of social networks. We then suggest a general approach for solving COPs by reducing them into a set of Constrained Satisfaction Problems (CSPs). Such reduction enables the employment of the well-studied CSP framework in order to solve a wide range of complex rules. Some of the resulting CSPs may contain a large number of variables, which may make them intractable. In order to overcome such intractability issues, we present two useful heuristics that decompose such large CSPs into smaller tractable sub-CSPs. We also show how the well-known ℓ-diversity privacy measure can be incorporated into the COP framework in order to evaluate the privacy level of COP solutions. Finally, we evaluate the new method in terms of privacy, utility and execution time.

Keywords: Obfuscation, Privacy Preserving Data Publishing, Diversity, Constrained Satisfaction Problems

1. Introduction

Organizations are increasingly required to share data for different purposes, e.g., for data mining, market researching and testing. In many cases this data contains sensitive private information which should not be exposed to unauthorized users. Data obfuscation attempts to overcome the conflict between the need to share data and the requirement to preserve privacy [5]. In a general obfuscation scenario, the data holder modifies the original data before publishing it in order to preserve privacy. At the same time, the obfuscation process must also preserve the utility of the data, which may be compromised due to those modifications. Thus, obfuscation methods attempt to modify the original data while optimizing both privacy and utility measures. The difficulty of this task highly depends on the inherent trade-off between the required levels of privacy and utility: higher levels of privacy usually require more modifications to the original data and those typically imply reduced utility.

In this paper we introduce the concept of Constrained Obfuscation Problems (COPs) which formulate the task of obfuscating data stored in relational databases, subject to some obfuscation constraints. The main idea behind COPs is simple though flexible as explained next. In relational databases, privacy and/or utility requirements can usually be represented by rules defined over subsets of attributes. Therefore their obfuscation can be seen as generating new data that comply with those predefined rules (or constraints). We demonstrate the flexibility and generality of the COP definition by modeling several different obfuscation scenarios:

- **Production Data Obfuscation for Application Testing (PDOAT).** Testing is a cardinal stage in the life-cycle of every information system. Testing cannot be performed without data which is usually stored in databases. Clearly, the simplest way to provide this data to a test environment is to copy it from the production environment. However, production data often contains private information which should not be exposed in a non-privileged test environment (such as an external, sometimes even foreign, testing group). Thus, the production data must be obfuscated before it is copied to the test environment so that private information is masked from the end-user (i.e. the tester). However, the task of obfuscating production data is non-trivial since it must also preserve certain characteristics (rules) of the original production data in order for the obfuscated data to be useful for testing. Failing to do so may compromise the effectiveness of the testing process. In PDOAT, the goal is to preserve utility, by satisfying all rules, while maximizing the obtained privacy level.

- **Anonymization of relational data.** Data mining is a common scenario in which data needs to be published. In many cases, such data includes private information about individuals which must not be disclosed. If individuals can be uniquely identified in the data then their private information may be disclosed. In order to avoid such identification in published data, uniquely identifying attributes like names and social security numbers are usually removed. However, this does not ensure the privacy of individuals in the data, since some of the remaining attributes, known as quasi-identifiers, may be used to link the published data with an external repository, and thus allow re-identification [44]. Several models were proposed in the literature as counter-measures for such linking attacks. The most basic one is k-anonymity [44, 45]. In that model, the quasi-identifiers are generalized so that each combination of quasi-identifier values appears at least k times in the relation. In this way, each tuple is indistinguishable from at least $k-1$ other tuples with respect to the quasi-identifiers, and thus privacy is preserved to some extent. The stronger model of ℓ-diversity was proposed as an enhancement to k-anonymity [38]. In that model it is required that each group of indistinguishable tuples has at least ℓ well represented sensitive values, in the sense that no sensitive value appears in more than $1/\ell$ of those tuples. Both of these models define the required level of privacy; the goal is to find an anonymization which complies with the required privacy condition and maximizes utility (in the sense that it minimizes information loss due to generalizations).

- **Anonymization of social networks.** Social networks are structures that describe a set of individuals and the relations between them. Social networks appear in many scenarios and they are of interest to researchers from disciplines such as sociology, psychology, or epidemiology. However, the data in such networks cannot be released as is, since it might contain sensitive information. A naïve anonymization of the network, in the sense of removing identifying attributes like names or social security numbers from the data, is insufficient, as shown in

[4]. Therefore, it is needed to anonymize the data prior to its publication in order to address the need to respect the privacy of the individuals whose sensitive information is included in the data. Many models of anonymity were proposed in the literature in recent years, e.g. [9, 10, 16, 29, 37, 46, 54, 60, 61]. Each of those models can be described as a COP. In Section 3 we discuss one of the first models that were proposed in this context — k-degree anonymity [37]. In that model, the goal is to make a minimal number of changes to the edge set in the graph, typically by adding edges, so that the graph becomes k-degree anonymous in the sense that for any vertex in the graph there exist at least $k-1$ other vertices having the same degree as that vertex.

After introducing and demonstrating the concept of COPs, we suggest a generic approach for solving a given COP by reducing it to a set of Constrained Satisfaction Problems (CSPs). This approach was first introduced in our previous work [59]. Following the formal definition in [43], a CSP is defined by a set of variables, $\{x_1, x_2, \ldots, x_n\}$ and a set of constraints, $\{c_1, c_2, \ldots, c_m\}$. Each variable x_i has a non-empty domain D_i of possible values. Each constraint c_i involves some subset of the variables and specifies the allowable combinations of values for that subset. A solution to the CSP is an assignment $\{x_1 = v_1, x_2 = v_2, \ldots, x_n = v_n\}$ of values to all variables that does not violate any constraint. (Such an assignment is called complete and consistent). Briefly, our reduction represents the database cells as CSP variables and translates the COP rules into CSP constraints on these variables. Harnessing the power of the well-established CSP framework allows our approach to cope with a wide variety of CSP constraints (and consequently obfuscation rules) that are supported by this framework. However, this reduction from COPs into CSPs may produce CSPs with a staggering number of variables and constraints which may raise intractability issues. In order to deal with such issues, we suggest two useful heuristics — *Incremental Generation (IG)* and *Local Modification (LM)*. The *LM* heuristic starts with a preliminary solution and then it iteratively applies local modifications to its values. The *IG* heuristic, on the other hand, arranges the variables in an ordered sequence of subsets of variables, and then it assigns values to each subset of variables in its turn, where each assignment is based on variables that were already assigned values in a previous round.

Next, we show how the well-known ℓ-diversity privacy measure [38] can be incorporated into the COP framework in order to evaluate the privacy level of COP solutions. The usual assumption in Privacy Preserving Data Publishing is that the original relation includes four types of attributes [12]: (a) explicit identifiers (ID) — attributes that uniquely identify an individual (e.g. `S.S.N.`); (b) quasi-identifiers (QID) — attributes that do not offer unique identification but their combination might yield unique identification by means of linking attacks (e.g., `zipcode`, `age`, `gender`); (c) sensitive data (S) — personal attributes of private nature, such as health condition or financial data; and (d) other attributes (O) — attributes that are non-sensitive, on one hand, and cannot be used for identification on the other hand. A COP solution respects ℓ-diversity if an adversary is unable to link any individual with any sensitive value with confidence greater than $1/\ell$.

Finally, we empirically evaluate our method in three of the aforementioned scenarios. In the PDOAT scenario, we compare the two heuristics LM and IG. We show that IG better preserves privacy than LM, however, as opposed to LM, it does not guarantee that a solution will always be found. Moreover, we show that a minor variation of LM benefits from the advantages of the two heuristics. We show that despite the fact that CSP is NP-hard in general, our heuristics allow the solving process to complete in a reasonable time even for large input relations; in addition, we show that concurrency can be used to reduce the execution time even further. In contrast, when our heuristics were not used, the standard CHOCO CSP solver was unable to complete its execution even for much smaller relations.

In the k-anonymity scenario, the results of the evaluation show that the new generic method can achieve much better utility than a dedicated method for k-anonymization (Mondrian). In some of the experiments, the utility in the output of our method was two orders of magnitude better than the utility in the output of Mondrian. This staggering improvement in terms of utility is coupled with larger runtimes. However, in all experiments the runtime of our method was feasible. In addition, since anonymization is a process that is executed on tables that contain data that was accumulated during very long periods of time (months and even years), and its output is intended for research and analysis, the execution time of the process is less important than the utility of the output. Hence, a method that runs in few minutes or hours but issues outputs with high utility is preferable over a method that runs in few seconds but issues outputs with much smaller utility.

The results in the ℓ-diversity scenario were similar. The new method achieves much better utility than the dedicated Mondrian algorithm, at the price of higher yet feasible runtimes.

Our contributions can be summarized as follows:

(1) We introduce the COP concept which constitutes a general framework for modeling and analyzing many obfuscation scenarios. To the best of our knowledge, this is the first time that such a framework is proposed. We demonstrate the generality of the COP definition by modeling several obfuscation problems.
(2) We propose an approach for solving any COP by reducing it into a set of CSPs. This allows us to harness the power of the well-studied CSP framework and support a wide variety of complicated COP rules.
(3) We suggest two useful heuristics, namely *Incremental Generation (IG)* and *Local Modification (LM)*, in order to deal with intractability issues that may arise from this approach.
(4) We show how the well-known ℓ-diversity privacy measure can be incorporated into the COP framework in order to evaluate the privacy level of COP solutions.

The two main advantages of the proposed COP framework over existing obfuscation methods are the following:

- Existing obfuscation methods are either confined to a certain application domain and their applicability is thus limited, or are general but incapable of handling complicated rules. (See a detailed discussion of existing obfuscation methods in Section 2.) In contrast, the COP framework is both general and capable of solving a wide range of complex rules. This allows our method to be used even in cases where a dedicated algorithm does not exist (e.g., PDOAT). In cases where a dedicated algorithm does exist (e.g., k-anonymity or ℓ-diversity), our method may even achieve better results than those of the dedicated algorithm, as we demonstrate in our evaluation (Section 6).

- In contrast to existing obfuscation methods, the COP framework allows us to focus on the required result in terms of privacy and utility, and not on the details of an algorithm to achieve it. This is equivalent to the difference between declarative programming and imperative programming: Imperative programming languages (e.g., Java) let the programmer specify the steps that are required in order to solve a particular task. On the other hand, declarative programming languages (e.g., SQL) follow a completely different approach in which the programmer focuses on providing the "what", leaving the "how" up to interpretation. Moreover, privacy problems are usually formalized by specifying the constraints with which the output must adhere to. Hence, a declarative approach that is based on "what" the output should be like seems more natural than an imperative approach that needs to decide "how" to reach that goal. Such a declarative approach may be easier to use, may reduce the risk of erroneous programming, and may result in shorter development times.

The remainder of this paper is organized as follows. In Section 2 we review related work. Section 3 gives a formal definition of COPs and demonstrates how the k-anonymization and PDOAT scenarios can be modeled accordingly. Section 4 shows how the ℓ-diversity privacy measure can be incorporated into the new framework. In Section 5 we present a generic method for reducing any COP into a set of solvable CSPs. Section 6 describes our evaluation and Section 7 concludes and suggests future research directions.

2. Related Work

Data generalization and data perturbation are the two main techniques being used for data obfuscation. Generalization replaces values of specific description, typically the quasi-identifier attributes, with a less specific description. Perturbation distorts the data by adding noise, aggregating values, swapping values or generating synthetic data based on some properties of the original data.

Generalization is widely used in Privacy Preserving Data Publishing (PPDP) and Privacy Preserving Data Mining (PPDM). PPDP is a developing research field that is targeted at developing models to enable publishing data so that privacy is preserved while data distortion is minimized. In the past years, several models and algorithms were suggested in PPDP, most of them evolved from the basic model of k-anonymity [45]. Algorithms for k-anonymization include those of [23, 24, 33, 35, 47, 53], that are designed for obtaining minimum distortion, without making assumptions on the type of data mining to be performed on the published data, and [6, 22, 26, 32, 34], that are geared towards improving classification results. Variations and alternatives of k-anonymity were also studied. For example, [38] proposed the model of ℓ-diversity to address attacks that are based on lack of diversity in the sensitive data; t-closeness [36] was suggested in order to maintain the distribution of the sensitive data; [41] proposed a privacy model similar to t-closeness called average privacy risk; [51, 52] proposed to limit the confidence of inferring a sensitive property for a group of individuals; and [58] proposed the notion of personalized anonymity. For a detailed survey about PPDP see [21]. PPDM, which is closely related to PPDP, was initiated in 2000 by [2]. PPDM algorithms aim at anonymizing data towards its release for specific data mining goals. For example, if the data needs to be used for learning a classifier, the corresponding PPDM algorithm will aim at achieving anonymity while incurring a minimal loss of accuracy in the resulting classifier. In PPDP, on the other hand, the purposes of the published data are unknown and it needs to be anonymized using utility measures that are not targeted to specific data mining tasks. For detailed surveys about PPDM see [1, 50].

Many measures were suggested in the literature for the cost of generalization. Among the commonly used ones are the Loss Metric measure [32] and the Entropy measure [25]. The recent study [34] shows that reduced information loss, as measured by either the Loss Metric or the Entropy Metric, translates also to enhanced accuracy when using the anonymized tables to learn classification models. In section 6 we will use these two measures in order to compare obfuscation algorithms in terms of utility.

One of the means of perturbing the original data is by generating synthetic data. A number of data generation methods have been proposed in the literature, each focusing on aspects such as data dependencies or fast generation of a large amount of data. In [11], the authors present a C-like Data Generation Language (DGL) which allows the generation of data using the definition of its data types, tuples, iterators and distributions. However, dependencies in data (such as foreign keys) must be explicitly handled by the user. In [31], the authors introduce the data dependency graph and describe a modified topological sort used to determine the order in which data is generated. However, it is unclear how this method can handle: (1) cycles in the graph; (2) rules that involve several fields in which no field is explicitly represented as a formula of the others, for example, $X^2 + Y^2 = 1$. In [30] the authors suggest a parallel synthetic data generator (PSDG) designed to generate industrial sized data sets quickly using cluster computing. PSDG depends on an XML based synthetic data description language (SDDL) which codifies the manner in which generated data can be described and constrained. However, PSDG suffers from the same limitations of [31]. In [8], the authors suggest a system to generate query aware data. The system is given a query and produces data that is tailored to the specific query.

However it is not very useful for general purpose generation of test data for ad hoc testing. In [27], a hardware-based technique to quickly generate large databases on multi-processor systems is described. The authors focus on low-level implementation details (such as process spawning and table partitioning strategies) to generate data using a dedicated hardware architecture. In [17], the authors focus on generating data for performance tests. More specifically, they try to generate data which preserves the query optimizer characteristics of the original data. The generation process must preserve three data characteristics: consistence, monotony and dependence between attributes. The authors of [55] propose an automated system for generating a synthetic database for testing database applications. Generation is based on a general location model which they learn by extracting various characteristics from schema constraints, read-only base tables that store information about the database, and the production data itself, when it is available. The characteristics learned from the production data include statistical information (such as distributions and element counts) or patterns in the data derived via data-mining. They also analyze the strength of their method in terms of preventing the disclosure of an individual's identity as well as the disclosure of a value of a certain confidential attribute of that individual. To summarize, as dependencies and rules defined over attributes become more complicated, none of the existing generation methods is general enough to handle them.

PPDP and PPDM algorithms are usually tailored to ensure some privacy measure (e.g. k-anonymity, ℓ-diversity) and thus the obfuscated data trivially satisfies this measure. However, for other obfuscation algorithms, privacy may be measured in retrospect. The authors of [40] propose to use reversibility for the categorization and comparison of obfuscation methods, with respect to their resilience to reverse engineering. The authors of [49] propose a risk estimation tool which analyzes a given disclosure policy, calculates the associated risk for a given dataset and provides the user with relevant suggestions to decrease the risk if it exceeds their expectations. They employ an entropy-based method to derive the *Global Risk* measure which estimates both the rareness of identifier-attribute combinations as well as the rareness of the occurrence value per sensitive attribute. As another example, this time from the realm of social networks, the studies in [9, 10] suggest to randomize the original data of a social network and then to measure the privacy level that those randomized versions provide for the data subjects. If the measured level of privacy is insufficient, the randomization process is repeated with higher randomization parameters.

Commercial obfuscation tools (e.g. [15, 13, 14, 28]) are also available. These provide a means of enforcing common rules, usually a set of predefined integrity rules (such as identity and reference rules)[19] or simple statistical aggregate rules (such as maintaining the average value of some attribute). However, application-specific rules are enforced via obfuscation routines that are developed ad hoc by the user. Although these tools offer powerful scripting capabilities which can be used with such routines, a framework that allows translating a set of application-specific constraints into a set of obfuscating routines without user intervention is clearly missing. Such a framework would drastically decrease the amount of implementation effort required by the user.

Other obfuscation techniques besides generalization and perturbation exist. For example, Anatomy [57] uses a special grouping mechanism in order to release all the quasi-identifier and sensitive values directly in two separate tables. In this paper we will focus on the generalization and perturbation techniques, but our method equally applies to other obfuscation techniques.

3. Constrained Obfuscation Problems

Obfuscation of relational databases generally involves a set of rules (each defined over a subset of attributes) with which the obfuscated data must comply. The relational nature of such databases gives rise to two types of rules. *Global rules* define the allowed combination of values for all tuples. For example, a rule requiring that two attributes constitute a composite key means that the pairs of values in all tuples must be different. *Non-global* rules define the allowed combination of values for each tuple independently of the other tuples. For example, a rule requiring that one attribute is larger than another attribute means that for each tuple, the value corresponding to the first attribute is larger than that of the second attribute. In this section we introduce and formally define the concept of obfuscating data stored in relational databases while adhering to a given set of rules. For convenience, Table 2 in the Appendix summarizes the main notations that we introduce hereinafter. We also demonstrate how four different obfuscation problems can be modeled according to these definitions: Three of these problems are concerned with obfuscation of relational databases (namely, PDOAT, k-anonymity, and ℓ-diversity); the fourth one deals with obfuscation of graph data (k-degree anonymity).

3.1. Formal problem definition

The setting for any relational database obfuscation problem includes the application domain in which the problem arises and two universes: (1) the source universe, denoted by U, which contains all relations that are valid in the application domain and (2) the target universe, denoted by U^*, containing all possible obfuscations of relations in U.

In other words, U includes all legal relations that could occur in the application domain. Those relations may need to undergo obfuscation before being published. The universe U^* includes all legal obfuscated versions of relations in U.

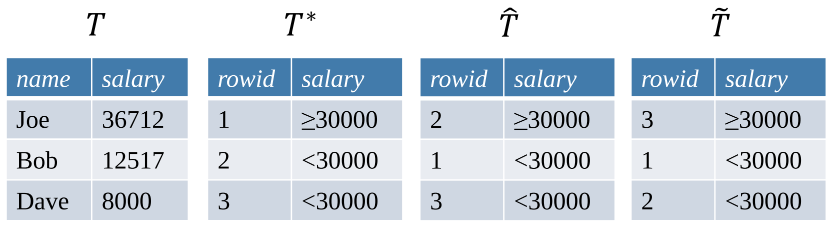
Relations in U are defined over a set of attributes $A = \{A_1, \ldots, A_m\}$; each attribute A_j has a domain D_j in which it may take values. Therefore, every relation T in U consists of tuples from $D = D_1 \times \cdots \times D_m$. Since the order of tuples in T is insignificant, every relation T in U is therefore some set of tuples from D. Hence, U (the collection of all legal relations) is a collection of subsets of D, i.e., $U \subseteq 2^D$.[1] Similarly, relations in U^* are defined over a set of attributes $A^* = \{A_1^*, \ldots, A_{m^*}^*\}$ with domains $D_1^*, ..., D_{m^*}^*$ and contain tuples from $D^* = D_1^* \times ... \times D_{m^*}^*$, i.e. $U^* \subseteq 2^{D^*}$.

Given a relation $T \in U$, we let $U_T^* \subseteq U^*$ denote the collection of all obfuscated relations into which T may be transformed. Namely, every relation in U_T^* is a possible obfuscation of T, but on the other hand, any relation in $U^* \setminus U_T^*$ cannot be the obfuscation of T. The obfuscation process is said to be *constrained* because the relations in U_T^* must comply with a given set of rules. Some of those rules are general, in the sense that they define the legal structure of an obfuscated relation T^*, independently of the original relation T. Other rules define the required consistency conditions between T and T^*. More formally, given a relation $T \in U$ let $R_T^* = \{R_1^*, \ldots, R_q^*\}$ be the set of rules defined on A^* which characterize the legal obfuscations of T. Then, U_T^* is the collection of all relations in 2^{D^*} that satisfy the set of rules R_T^*. In light of the above, we can define the target universe as $U^* = \bigcup_{T \in U} U_T^*$.

Next, we define the notion of Constrained Obfuscation Problems:

Definition 3.1. *Given a relation $T \in U$, a* Constrained Obfuscation Problem (COP) *is the task of finding a relation $T^* \in U_T^*$.*

[1]Depending on the associated application domain, some relations in 2^D may not be valid (as we shall see in Section 3.2).



T

name	salary
Joe	36712
Bob	12517
Dave	8000

T^*

rowid	salary
1	≥30000
2	<30000
3	<30000

$\hat{T}$

rowid	salary
2	≥30000
1	<30000
3	<30000

$\tilde{T}$

rowid	salary
3	≥30000
1	<30000
2	<30000

Figure 1: A relation T and its obfuscated relation T^*.

Stated differently, the task of obfuscating a relation $T \in U$ involves finding a relation $T^* \in U^*$ that conforms to the rules in R^*_T.

Example 3.1. *Consider a source universe U which has two attributes: A_1=name and A_2=salary with domains D_1=$\{$all strings$\}$ and D_2=$\{$all positive integers$\}$. Furthermore, consider a target universe U^* which has two attributes A^*_1=rowid and A^*_2=salary class with domains D^*_1=$\{$all positive integers$\}$ and D^*_2=$\{$<30000, ≥30000$\}$. The rules which characterize the legal obfuscations are the following:*

- *R^*_1: the number of tuples in the obfuscated relation equals the number of tuples in the original relation;*
- *R^*_2: the values in the attribute rowid are all different; and*
- *R^*_3: the value in the attribute salary class is <30000 if the corresponding value in the original salary attribute is lower than 30000 or ≥30000 otherwise.*

Here, R^*_2 is a general rule, independent of T, that defines a condition that must be satisfied by all obfuscated relations in U^*. R^*_1 and R^*_3, on the other hand, are T-dependent rules of compatibility: they define conditions that T^* must satisfy in order to be a legal obfuscation of the relation T. An example of a relation $T \in U$ and its obfuscation T^* are shown in Figure 1.
□

COPs can be solved using an obfuscation algorithm as defined next:

Definition 3.2. *An obfuscation algorithm $\mathcal{A}$ takes a relation $T \in U$ as input and outputs a relation $T^* \in U^*_T$ with probability $P^{\mathcal{A}}_T(T^*)$.*

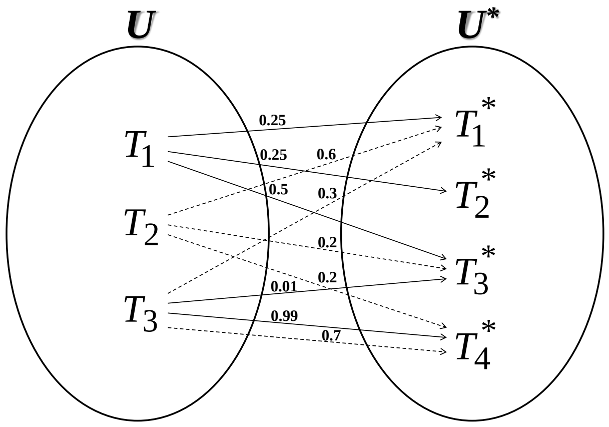


Figure 2: Probability distributions induced by two obfuscation algorithms $\mathcal{A}_1$ (solid arrows) and $\mathcal{A}_2$ (dashed arrows).

In other words, given a relation $T \in U$, $\mathcal{A}$ induces a probability distribution on U^*, denoted by $P_T^{\mathcal{A}}$, where for each $T^* \in U^* \setminus U_T^*$, $P_T^{\mathcal{A}}(T^*) = 0$.[2]

Example 3.2. *Assume that in Example 3.1 we add the condition that the values in the attribute rowid must be limited to the range $\{1, \ldots, n\}$, where n is the number of tuples in T and in T^*. Then T in Figure 1 has two additional obfuscated forms, $\hat{T}$ and $\tilde{T}$, as shown in Figure 1. If the obfuscation algorithm chooses uniformly at random one of the relations in $\{T^*, \hat{T}, \tilde{T}\}$ as the obfuscation of T, then $P_T^{\mathcal{A}}(T^*) = P_T^{\mathcal{A}}(\hat{T}) = P_T^{\mathcal{A}}(\tilde{T}) = \frac{1}{3}$, while $P_T^{\mathcal{A}}(T') = 0$ for all other relations $T' \in U^*$.*
□

Example 3.3. *Figure 2 illustrates probability distributions induced by two obfuscation algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. The blue and red values appearing on top of the arrows from T_i to T_j^* indicate the probabilities $P_{T_i}^{\mathcal{A}_1}(T_j^*)$ and $P_{T_i}^{\mathcal{A}_2}(T_j^*)$, respectively. For clarity, arrows corresponding to zero valued probabilities are omitted.*

[2]Consider the three random variables X_U, X_{U^*} and X_A, whose possible values are the relations of U, the relations of U^* and all possible obfuscation algorithms, respectively. We can think of $P_T^{\mathcal{A}}$ as the conditional probability distribution of X_{U^*} given X_U and X_A, i.e. $P_T^{\mathcal{A}}(T^*) = \Pr[X_{U^*} = T^* \mid X_U = T \wedge X_A = \mathcal{A}]$.

Assuming that $U^*_{T_1} = \{T^*_1, T^*_2, T^*_3\}$ and $U^*_{T_2} = U^*_{T_3} = \{T^*_1, T^*_3, T^*_4\}$, we make the following two observations about obfuscation algorithms:

1. *An algorithm may not necessarily be able to obfuscate all relations in U. In our case, $\mathcal{A}_1$ cannot obfuscate T_2 while $\mathcal{A}_2$ cannot obfuscate T_1. (In Section 5.4 we discuss a concrete example of an obfuscation algorithm that is incapable of obfuscating some relations.)*
2. *Different algorithms may induce different distributions on the same U^*_T. In our case, the distribution that $\mathcal{A}_1$ induces on $U^*_{T_3}$ is: $P^{\mathcal{A}_1}_{T_3}(T^*_1) = 0$, $P^{\mathcal{A}_1}_{T_3}(T^*_3) = 0.01$, $P^{\mathcal{A}_1}_{T_3}(T^*_4) = 0.99$, whereas the distribution that $\mathcal{A}_2$ induces on $U^*_{T_3}$ is $P^{\mathcal{A}_2}_{T_3}(T^*_1) = 0.3$, $P^{\mathcal{A}_2}_{T_3}(T^*_3) = 0$, $P^{\mathcal{A}_2}_{T_3}(T^*_4) = 0.7$.*

□

3.2. Modeling PDOAT as a COP

In some obfuscation scenarios, there exists a set of rules $R = \{R_1, R_2, ..., R_k\}$ that are defined on A; namely, the original data is already constrained. In such cases, U is a proper subset of 2^D because relations in 2^D that do not comply with R are invalid and therefore are not included in U. One such obfuscation scenario is PDOAT, in which the original production data is constrained by a set of rules R which must also be maintained by the obfuscated relation in order to be useful for testing. The PDOAT scenario can be modeled as a COP with the following properties:

1. $A^* = A$.
2. $D^* = D$.
3. $R^*_T = R \bigcup \{R^*_1\}$ where
   (a) R^*_1: $|T^*| = |T|$.

Namely, T^* must be a relation that obeys the same structural rules as T (those are the rules in R). Since those are general rules, the only rule in R^*_T that depends on T is the rule (iii)(a) which requires T^* to have the same number of tuples as T. There is no rule in R^*_T that depends specifically on the values of T. This property will be used in Section 5.5.

As discussed in Section 2, available obfuscation algorithms for PDOAT COPs are incapable of handling a combination of complicated rules. In Section 5 we present a generic yet powerful method for solving COPs.

3.3. Modeling anonymization problems as COPs

Recall that in the k-anonymization scenario, the usual practice is to remove the identifier attributes (ID) and generalize the quasi-identifier attributes (QID) so that the size of each QID block (i.e. a set of tuples in T^* that all have the same QID values) is at least k. All other attributes are kept unchanged. This scenario can be modeled as a COP with the following properties:

1. $A^* = A \backslash ID$. (The identifier attributes are removed; all other attributes are retained.)
2. $\forall A_i \in QID$, $D_i^* = 2^{D_i}$. (In the quasi-identifier attributes, specific values will be replaced by subsets of values.)
3. $\forall A_i \in A \setminus (ID \bigcup QID)$, $D_i^* = D_i$. (The form of all other attributes will remain the same.)
4. $R_T^* = \{R_1^*, R_2^*, R_3^*, R_4^*\}$ where:
   (a) R_1^*: $|T^*| = |T|$. (Namely, each tuple in T has a corresponding obfuscated tuple in T^*.)
   (b) R_2^*: each quasi-identifier value in T^* contains the corresponding value in T. (The quasi-identifier values are properly generalized.)
   (c) R_3^*: all other values in T^* equal the corresponding values in T.
   (d) R_4^*: any quasi-identifier tuple in T^* appears at least k times. (This is the desired privacy condition of k-anonymity.)

Property (ii) allows any type of generalization. However, some attributes are usually subjected to more specific forms of generalization. For example, numerical attributes are usually generalized to intervals of values, while categorical attributes are typically generalized using taxonomy trees. Such restricted forms of generalizations can be achieved via additional rules. In our experimental evaluation, Section 6, we use generalization by taxonomy trees.

While COPs such as this one can be solved using the generic method that we present in Section 5, k-anonymization is a well-studied problem which has many dedicated algorithms. In Section 6 we compare our generic method to the dedicated Mondrian algorithm [35].

As another example, we consider the problem of computing an ℓ-diverse anonymization of a given relation. In its most basic form, the condition of ℓ-diversity requires that in any QID block, no sensitive value appears in more than $1/\ell$ of the tuples in that block. To achieve that goal, we only need to replace rule R_4^* above with the above rule of ℓ-diversity.

In either examples of k-anonymity or ℓ-diversity, the goal is to comply with the desired privacy property while incurring minimal loss of information. The rules that we described above concentrate only on compliance with the privacy condition. They do not address the goal of achieving that privacy target while maintaining maximal utility. To do so, we should define for each obfuscated table $T^* \in U_T^*$ the corresponding information loss. (For example, a very basic measure of information loss could be a count of the number of entries in T^* that differ from the corresponding entries in T. More accurate measures, however, were defined in the anonymity literature; see a survey in [26].) Then, we may augment the CSP with a cost function that equals the information loss in T^* with respect to T, and the target would be to find a solution to the CSP that minimizes that cost function. The usual terminology in the study of CSPs refers to the regular constraints as *hard* constraints, while the cost function is called a *soft* constraint. These names illustrate that hard constraints have to be satisfied, while the soft constraint only expresses a preference of solutions of a lower cost over solutions with a higher cost. (We note that the problem of finding a k-anonymization of a given table with minimal information loss is NP-hard [25, 39]; hence, even dedicated algorithms for k-anonymization find a k-anonymization with a low information loss, but not necessarily one with minimal information loss.)

3.4. Modeling graph anonymization problems as COPs

A graph G can be represented as a relation T in the following manner. Each tuple in the relation T corresponds to a vertex in the graph G. The attributes in a given tuple are the vertex id, its degree, and the list of its neighbors. Assume that the maximal degree in the graph is W. Then the graph can be represented as a relation T over $W + 2$ attributes, $ID, Deg, N_1, \ldots, N_W$. The domain of each of those attributes is the set of nonnegative integers. The rules R with which every such graph-describing relation must comply are as follows:

1. R_1: The values in the attribute ID are all distinct.
2. R_2: The values in the attribute Deg are less than or equal to W.
3. R_3: If the value of Deg in a given tuple is d, then the values of attributes $N_1, \ldots, N_d$ in that tuple are distinct vertex IDs that differ from that tuple's ID.

4. R_4: Let d_i denote the degree of the vertex in the ith tuple. Then if j appears in one of the attributes $N_1, \ldots, N_{d_i}$ of tuple i, then i appears in one of the attributes $N_1, \ldots, N_{d_j}$ of tuple j. (This rule formulates the symmetry of the adjacency relation between vertices.)

The model of k-degree anonymity requires to anonymize a given graph G into another graph G^* such that two properties hold: (a) The edge set of G is a subset of the edge set of G^*; (b) each degree in G^* is shared by at least k vertices in G^*. Hence, the obfuscated relation T^* that describes the k-degree anonymized graph G^* must comply with the above four general rules, R_1, R_2, R_3, R_4, as well as with the following rules:

1. R_1^*: $|T| = |T^*|$.
2. R_2^*: Let d_i and d_i^* denote the degrees in the ith tuples in T and T^*, respectively. Then for all i, $d_i^* \geq d_i$.
3. R_3^*: For all i, the values of attributes $N_1, \ldots, N_{d_i}$ of tuple i in T equal the values of those attributes in tuple i in T^*. (Namely, the set of edges adjacent to any given vertex in G^* is a superset of the set of edges adjacent to that vertex in G.)
4. R_4^*: For each i, there exist at least $k-1$ indices $i_1, \ldots, i_{k-1}$ such that $d_i^* = d_{i_1}^* = \cdots = d_{i_{k-1}}^*$. (This is the k-degree anonymity condition.)

Rules R_1^*, R_2^* and R_3^* specify the required compatibility between T^* and T. The rule R_4^* specifies the required compliance with the property of k-degree anonymity.

All of the above rules define the required compliance with the k-degree anonymity privacy model. In order to select a k-degree anonymization with maximal utility, namely, an anonymized graph with a minimal number of added edges, we may augment the CSP with a cost function that equals $\sum_i (d_i^* - d_i)$.

4. Measuring Privacy

In order to evaluate the ability of an obfuscation algorithm to preserve privacy, or to compare between different obfuscation algorithms, a measure of privacy is required. In this section we show how the well-known ℓ-diversity privacy measure can be incorporated into our COP framework.

Definition 4.1. *We say that an obfuscated relation respects ℓ-diversity if an adversary is unable to link any individual with any sensitive value with confidence greater than* $1/\ell$.

In Section 4.1 we give a precise meaning for the confidence in which an adversary can link any individual with any sensitive value. Then, in Section 4.2, we explain how the level ℓ of diversity can be practically estimated.

4.1. Defining the level ℓ of diversity

The definition of the level ℓ of diversity that is associated with obfuscating a relation T into T^* relies upon the following assumptions:

1. A-priori, all relations in U are equally probable as being the original relation T. (This assumption is known as the random worlds model [3] and it is commonly used to reason about adversaries.)
2. The adversary knows the following information:
   (a) The obfuscation algorithm $\mathcal{A}$.
   (b) The published obfuscated relation T^* that was output by $\mathcal{A}$ for the input T.
   (c) For every relation $\overline{T} \in U$, the probability $P^{\mathcal{A}}_{\overline{T}}(T^*)$ that it was converted by $\mathcal{A}$ into T^*.
   (d) The exact set of individuals that are represented in T and their corresponding QID. Namely, the adversary knows the projection of the relation T onto the ID and QID attributes, which we denote by $BK := \Pi_{ID,QID}(T)$. (The same assumption was made in [47, 53, 58].)

Given the obfuscated relation T^* and the adversarial background knowledge BK, let $W_{T^*} = \{\overline{T} \mid \overline{T} \in U \wedge T^* \in U^*_{\overline{T}} \wedge \Pi_{ID,QID}(\overline{T}) = BK\}$ denote the set of all relations that could be obfuscated into T^*. The adversary, who observes the obfuscated relation T^* and knows BK, may limit the set of "possible worlds" to W_{T^*}; namely, only relations in W_{T^*} are suspected as being the original relation T that was obfuscated into T^*, while all other relations in $U \setminus W_{T^*}$ are ruled out. Clearly, the original relation T is in the collection W_{T^*}. Furthermore, given the obfuscation algorithm $\mathcal{A}$ and $P^{\mathcal{A}}_{\overline{T}}(T^*)$ for every relation $\overline{T} \in U$, the adversary may associate a probability $\hat{P}^{\mathcal{A}}_{T^*}(\overline{T})$ with each possible world $\overline{T} \in W_{T^*}$ as being the original relation T.[3] That probability is proportional to the probability that $\mathcal{A}$ obfuscated $\overline{T}$ into T^*, and is given by

$$\hat{P}^{\mathcal{A}}_{T^*}(\overline{T}) = \frac{P^{\mathcal{A}}_{\overline{T}}(T^*)}{\sum_{T' \in W_{T^*}} P^{\mathcal{A}}_{T'}(T^*)} \tag{1}$$

Example 4.1. *Consider the universes U and U^* and the obfuscation algorithm $\mathcal{A} := \mathcal{A}_1$ that were described in Example 3.3 and Figure 2. Assume that the original relation was $T = T_1$. The obfuscation algorithm can obfuscate it into one of three relations in U^*, namely, T^*_1, T^*_2, or T^*_3, with probabilities $P^{\mathcal{A}}_T(T^*_1) = 0.25$, $P^{\mathcal{A}}_T(T^*_2) = 0.25$, $P^{\mathcal{A}}_T(T^*_3) = 0.5$. Let us assume that $T^* := T^*_3$ was selected by $\mathcal{A}$. By assumption (ii), the adversary, who knows $\mathcal{A}$ and T^*, can compute $P^{\mathcal{A}}_{\overline{T}}(T^*)$ for all relations $\overline{T} \in U$. Those probabilities are illustrated by the labels of the blue arrows that are pointing to $T^* := T^*_3$ in Figure 2, i.e., $P^{\mathcal{A}}_{T_1}(T^*) = 0.5$, $P^{\mathcal{A}}_{T_3}(T^*) = 0.01$, while $P^{\mathcal{A}}_{T_2}(T^*) = 0$. Hence, W_{T^*} in this case consists of just two possible worlds: T_1 and T_3. Their probabilities as being the original relation are therefore $\hat{P}^{\mathcal{A}}_{T^*}(T_1) = \frac{0.5}{0.5+0.01} \approx 0.98$ and $\hat{P}^{\mathcal{A}}_{T^*}(T_3) = \frac{0.01}{0.5+0.01} \approx 0.02$. Hence, if the adversary observes the obfuscated relation T^*, and he knows that algorithm $\mathcal{A} = \mathcal{A}_1$ was utilized, both T_1 and T_3 could be the original relation, but T_1 is much more likely to be the true preimage.* $\square$

Let D_{ID} and D_S denote the domains of the identifier and sensitive attributes, respectively. Fixing a pair $(id, s) \in D_{ID} \times D_S$ of an identifier id and a sensitive value s, we let

$$W^{id,s}_{T^*} = \{\overline{T} \mid \overline{T} \in W_{T^*} \wedge (id, s) \in \Pi_{ID,S}(\overline{T})\}$$

denote the set of all relations in W_{T^*} that contain a tuple that has this pair of values. If we add the probabilities $\hat{P}^{\mathcal{A}}_{T^*}(\overline{T})$ over all possible worlds in $W^{id,s}_{T^*}$, we get the value $p_{id,s} := \sum_{\overline{T} \in W^{id,s}_{T^*}} \hat{P}^{\mathcal{A}}_{T^*}(\overline{T})$ which denotes the confidence that a relation which contains the pair (id, s) was obfuscated into T^*. Finally, let $p_{\max} := \max_{(id,s) \in D_{ID} \times D_S} p_{id,s}$ denote the maximal confidence in which the

[3]Consider the three random variables X_U, X_{U^*} and X_A, whose possible values are the relations of U, the relations of U^* and all possible obfuscation algorithms, respectively. We can think of $\hat{P}^{\mathcal{A}}_{T^*}$ as the conditional probability distribution of X_U given X_{U^*} and X_A, i.e. $\hat{P}^{\mathcal{A}}_{T^*}(\overline{T}) = \Pr[X_U = \overline{T} \mid X_{U^*} = T^* \wedge X_A = \mathcal{A}]$.

adversary may link any identifier id with any sensitive value s. Then, in view of Definition 4.1, T^* respects ℓ-diversity if $p_{\max} \leq 1/\ell$.

Example 4.2. *Following Example 3.3, assume that the identifier and sensitive attributes in relations T_1, T_2 and T_3 are as given in Figure 3. Further, assume that we obfuscated the relation T_1 using $\mathcal{A}_1$ and published the obfuscated relation T_3^*.*

In view of our assumptions, the adversary is capable of limiting the set of possible worlds to two relations: $W_{T_3^*} = \{T_1, T_3\}$. Hence, he may infer that the original relation included the names Joe, Bob and Dave. By observing the obfuscated relation T_3^* he may also infer that the original relation had three sensitive values, a, b, and c. Hence, for each of the possible nine pairs (id, s) of a name and a sensitive value he may compute the set $W_{T_3^*}^{id,s}$:

- $W_{T_3^*}^{Joe,a} = \{T_1, T_3\}$,
- $W_{T_3^*}^{Bob,b} = \{T_1\}$,
- $W_{T_3^*}^{Bob,c} = \{T_3\}$,
- $W_{T_3^*}^{Dave,b} = \{T_3\}$,
- $W_{T_3^*}^{Dave,c} = \{T_1\}$,
- $W_{T_3^*}^{id,s} = \emptyset$ *otherwise.*

In Example 4.1 we showed that $\hat{P}_{T_3^*}^{\mathcal{A}_1}(T_1) \approx 0.98$ and $\hat{P}_{T_3^*}^{\mathcal{A}_1}(T_3) \approx 0.02$. Therefore, the linkage confidences for each of the possible pairs of a name and a sensitive value are

- $p_{Joe,a} = 0.98 + 0.02 = 1$,
- $p_{Bob,b} = 0.98$,
- $p_{Bob,c} = 0.02$,
- $p_{Dave,b} = 0.02$, *and*
- $p_{Dave,c} = 0.98$.

T_1

ID	*S*
Joe	a
Bob	b
Dave	c

T_2

ID	*S*
Harry	a
Dan	b
Greg	c

T_3

ID	*S*
Joe	a
Bob	c
Dave	b

Figure 3: The content of the relations T_1, T_2, and T_3.

The maximal linkage confidence is therefore $p_{\max} = p_{Joe,a} = 1$. Hence, T_3^* respects only 1-diversity (which is the lowest possible level of diversity), since the adversary can make at least one certain linkage: viewing T_3^*, the adversary is confident that in the original relation, T, the identifier Joe was linked with the sensitive value a.
□

Note that if the obfuscation algorithm is based on homogeneous generalization (see a discussion on homogeneous and non-homogeneous generalization in [53]), then the above calculation can be simplified. In such a case, ℓ-diversity is respected if within each QID block B of T^* (i.e. a set of tuples in T^* which all have the same QID values), there is no sensitive value that appears more than $\frac{|B|}{\ell}$ times.

4.2. Estimating ℓ

The process of computing $p_{\max}$ relies on our assumption that we can compute for each relation $\overline{T} \in W_{T^*}$ the probability $\hat{P}^{\mathcal{A}}_{T^*}(\overline{T})$ that it was obfuscated into T^*. This assumption is sometimes unrealistic: it might be impossible to enumerate W_{T^*}, or we may not know $P^{\mathcal{A}}_{\overline{T}}(T^*)$ for all $\overline{T} \in W_{T^*}$. A more realistic assumption is that there exists an algorithm $\mathcal{A}^R$ that acts as a reverse engineering mechanism for $\mathcal{A}$:

Definition 4.2. *An algorithm $\mathcal{A}^R$ is called a reverse engineering algorithm for an obfuscation algorithm $\mathcal{A}$ if, given any $T^* \in U^*$ as input, it outputs a relation $\overline{T} \in W_{T^*}$ with probability $\hat{P}^{\mathcal{A}}_{T^*}(\overline{T})$.*

(In Section 5 we shall see examples for such algorithms.) By executing $\mathcal{A}^R$ repeatedly, we can obtain a *multiset* M_{T^*} of relations from W_{T^*}. Assuming that we execute $\mathcal{A}^R$ t times, namely, until we get t relations in M_{T^*} (some of which could be repeated, as M_{T^*} is a multiset), we can compute an estimate

	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.025$	$\alpha = 0.2$	$\alpha = 0.01$
$d = 0.05$	403	510	624	664	788
$d = 0.02$	2516	3184	3899	4147	4925

Table 1: Minimal value of t for different values of d and α.

for $p_{id,s}$, denoted $\tilde{p}_{id,s}$, as the number of relations in M_{T^*} that contain the pair (id, s), divided by t. After computing the estimated confidence levels, $\tilde{p_{id,s}}$, we may proceed to estimate the diversity of the release as the inverse of $\tilde{p}_{\max} := \max_{(id,s)\in D_{ID}\times D_S} \tilde{p}_{id,s}$.

Clearly, in the process described above, the quality of our estimation of $p_{\max}$ grows with t. A question that arises is which value of t suffices? To that end, we define for each $id \in \Pi_{ID}(T)$ a random variable X_S^{id} over the sensitive attribute domain D_S, where $\Pr(X_S^{id} = s) = p_{id,s}$, for all $s \in D_S$. For simplicity, if we assume that these random variables are independent, we can use Table 1, which was taken from [48], to determine the minimal value of t that guarantees that all $\tilde{p}_{id,s}$ ($id \in \Pi_{ID}(T)$ and $s \in D_S$) have a maximum deviation of d from $p_{id,s}$ with probability at least $1 - \alpha$. As a consequence, using such a value of t guarantees that also $|\tilde{p}_{\max} - p_{\max}| \leq d$ with probability at least $1 - \alpha$.

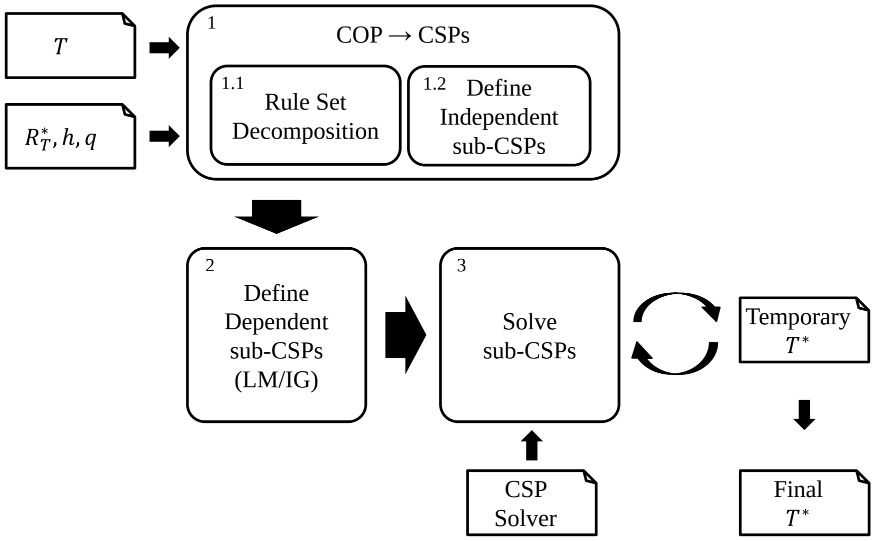
5. Solving COPs

In this section, we propose a generic method for solving a COP by reducing it into a set of CSPs and solving those CSPs using a CSP solver. Since the CSP framework supports a large variety of constraints and offers a rich collection of efficient algorithms for satisfying them, such a reduction provides a powerful means of solving any COP.

The main stages of the proposed method are illustrated in Figure 4:

1. Stage 1: The COP is reduced into a single CSP which is then decomposed into a set of independent sub-CSPs.
2. Stage 2: Large sub-CSPs that were produced in Stage 1 are further decomposed into dependent sub-CSPs. (Those sub-CSPs are dependent in the sense that they should be solved in some order, since the solution of some sub-CSPs may depend on the solution of other sub-CSPs.)
3. Stage 3: All sub-CSPs are solved using a CSP solver and the solutions are stored in T^* which then contains a publishable version of T.

In the following subsections, we describe in details each of those stages.



T
R*_T, h, q
1
COP → CSPs
1.1 Rule Set Decomposition
1.2 Define Independent sub-CSPs
2
Define Dependent sub-CSPs (LM/IG)
3
Solve sub-CSPs
Temporary T*
CSP Solver
Final T*

Figure 4: Outline of the proposed method for solving COPs.

5.1. Reducing COPs into CSPs

We propose the following reduction for deriving a CSP from a given COP. First, a CSP variable is defined for each cell in T^*. We denote the variable that corresponds to the cell in tuple i of attribute A_j in T^* by $x_{i,j}^{T^*}$. Letting N^* be the number of tuples in the obfuscated relation T^* and m^* be the number of attributes in T^* (see Section 3.1), we let $X = \{x_{i,j}^{T^*} : 1 \leq i \leq N^*, 1 \leq j \leq m^*\}$ denote the set of all variables. Next, each rule is transformed into its corresponding set of constraints. As opposed to the rules which are defined over attributes, the CSP constraints are defined over variables. Each global rule r will produce a single global constraint involving $N^* \times |A^r|$ variables, where A^r is the set of attributes that r involves. For example, consider a rule requiring that an attribute A_j is unique. This rule will be translated into a single N^*-ary "all different" [7] constraint: $allDifferent(x_{1,j}^{T^*}, x_{2,j}^{T^*}..., x_{N^*,j}^{T^*})$. Alternatively, each non-global rule r will produce N^* non-global constraints, each involving $|A^r|$ variables. For example, consider a non-global rule which requires that the values of attribute A_j are greater than the corresponding values of attribute A_k. This rule will be translated into N^* binary "greater than" [7] constraints: $\{gt(x_{1,j}^{T^*}, x_{1,k}^{T^*}), gt(x_{2,j}^{T^*}, x_{2,k}^{T^*}), ..., gt(x_{N^*,j}^{T^*}, x_{N^*,k}^{T^*})\}$.

It is often possible to decompose a CSP into a set of disjoint sub-CSPs, in the sense that each variable and each constraint of the original CSP belongs to exactly one sub-CSP. Such a decomposition is desirable since it results in separate simpler sub-problems that can be solved independently. A common way to apply such decomposition is to find the different connectivity components in the constraint graph [43]:

Definition 5.1. *The constraint graph of a given CSP is a graph in which each variable is represented by a vertex and two vertices are connected if there exists a constraint that contains both variables.*

Each connectivity component in the constraint graph corresponds to a disjoint sub-CSP. Although we can apply such a decomposition for COP-derived CSPs, the special characteristics of COPs allows us to modify our reduction in a way that it generates the post-decomposition sub-CSPs without the need of applying a CSP decomposition algorithm on the constraint graph. Before this is further explained, we note the following observation. Assume that we applied a standard CSP decomposition algorithm on the constraint graph of a COP-derived CSP. Then, any two disjoint sub-CSPs satisfy exactly one of the following two conditions, regarding the sets of attributes and tuples of their variables:

1. The two sets of attributes are disjoint.
2. The two sets of attributes are identical, but the two sets of tuples are disjoint. Furthermore, each of the two sets of tuples contains exactly one tuple.

The above observation implies a method for generating the post-decomposition sub-CSPs. First, we decompose the set of rules R_T^* into disjoint subsets of rules by finding the different connectivity components of the rule graph:

Definition 5.2. *The rule graph of a given COP is a graph in which each attribute is represented by a vertex and two vertices are connected if there exists a rule that contains both attributes.*

Namely, rules in different subsets relate to disjoint subsets of attributes. This first step of decomposition is described in Figure 4 as Stage 1.1. Second, for each disjoint rule subset that contains only non-global rules, we generate N^* independent sub-CSP problems corresponding to the N^* different tuples, while for each of the remaining rule subsets we generate a single sub-CSP problem. This second step is described in Figure 4 as Stage 1.2.

The latter type of sub-CSP problems, i.e. those that originate from a rule subset that contains at least one global rule, may have a large number of variables. As a very common example, consider a sub-CSP resulting solely from the $allDifferent$ constraint, that was previously mentioned, and its N^* associated variables. Solving such a sub-CSP is impractical in many cases due to the overwhelming number of variables and may thus require special heuristics.

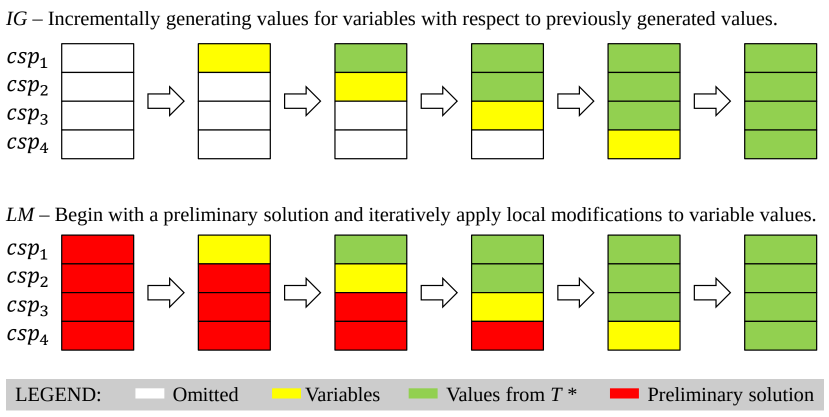
5.2. Defining the dependent CSPs

In general, it is reasonable to assume that for any solver s, there exists some manageable number of variables q_s for which any CSP with more than q_s variables cannot be handled by that solver. Sub-CSPs that originate from a rule subset that contains at least one global rule may have a large number of variables which may exceed q_s. In this section we suggest a method for decomposing such a large sub-CSP (LCSP hereinafter) into a set of dependent sub-CSPs with disjoint subsets of variables of sizes no larger than than q_s. In what follows, X_{LCSP} is the set of variables in the LCSP, A^{LCSP} is the set of attributes to which those variables correspond, and C_{LCSP} is the set of constraints in the LCSP.

The dependent sub-CSP which is generated in the jth iteration of this decomposition process will be denoted csp_j. The set of variables of csp_j, denoted X_{csp_j}, consists of all variables in X_{LCSP} which correspond to tuples of indices between $q(j-1)+1$ and $\min(qj, N^*)$, for some parameter q which is yet to be determined. In other words, each X_{csp_j}, except possibly for the last one, contains $q \cdot |A^{LCSP}|$ variables that are taken from q consecutive tuples. If we choose $q \leq \frac{q_s}{|A^{LCSP}|}$ then obviously $|X_{csp_j}| \leq q_s$, as required in order for csp_j to be manageable.

Note that for each LCSP we could optimally select $q = \lfloor \frac{q_s}{|A^{LCSP}|} \rfloor$, which is the value of q that keeps all corresponding dependent sub-CSPs manageable and minimizes their number. However, for simplicity, we used the same value of q for all LCSPs. As we show later in Section 6, the value of q must be carefully chosen because it influences both performance and privacy.

The set of constraints of csp_j, denoted by C_{csp_j}, is then defined in two steps: (1) C_{csp_j} is initialized with the non-global constraints in C_{LCSP}, denoted by C_{NG}, whose variables belong to X_{csp_j} (note that all variables of a non-global constraint appear in the same tuple); (2) a new global constraint c^* is derived from each $c \in C_{LCSP} \setminus C_{NG}$ and added to C_{csp_j}. We propose two heuristics for deriving c^*:



IG – Incrementally generating values for variables with respect to previously generated values.
csp_1
csp_2
csp_3
csp_4
LM – Begin with a preliminary solution and iteratively apply local modifications to variable values.
csp_1
csp_2
csp_3
csp_4
LEGEND: ⬜ Omitted 🟨 Variables 🟩 Values from *T* * 🟥 Preliminary solution

Figure 5: Heuristics for decomposing a large sub-CSP into dependent sub-CSPs.

- The *Local Modifications (LM)* heuristic derives c^* from c by labeling all variables in $X_c \setminus X_{csp_j}$ (where X_c is the set of variables in the constraint c) as constants whose values are taken from the corresponding cells of T^* during the future stage of solving the CSPs. The label is a pointer to the cell represented by the variable and it is used only to obtain the actual value stored in T^* at the time when csp_j needs to be solved.

- The *Incremental Generation (IG)* heuristic derives c^* from c in two steps: (1) all variables in $X_c \setminus \bigcup_{\ell \leq j} X_{csp_\ell}$ are omitted; (2) all variables in $\bigcup_{\ell < j} X_{csp_\ell}$ are labeled as constants (similarly to LM).

Note that either of the above heuristics transform c into a new global constraint c^* in which the set of variables is $X_c \cap X_{csp_j}$; therefore, c^* has at most q_s variables, whence it is manageable. The names for the two heuristics are self-explanatory: LM begins with an initial solution which is stored in T^*, and then it iteratively applies local modifications to its values; IG, on the other hand, incrementally generates values based on previously generated values. Algorithm 1 and Figure 5 illustrate the two heuristics.

5.3. Solving the CSPs

Solving each CSP requires a CSP solver.[4] Each one of the independent sub-CSPs that were produced in Stage 1 can be solved independently, even

[4]There are many available CSP packages (both commercial and academic) from which a solver can be chosen. Deciding which package to use must be carefully considered because

Algorithm 1 defineDependentCSPs($LCSP$,N^*,q,h)

Input:
$LCSP = \langle X_{LCSP}, C_{LCSP} \rangle$: a large sub-CSP to be decomposed.
N^*: the number of tuples in T^*.
q: the preferable number of tuples for each sub-CSP.
h: the chosen heuristic, either LM or IG.

Output:
$LCSP^{dep}$: the set of dependent sub-CSPs into which $LCSP$ was decomposed.

```
1: $LCSP^{dep} \leftarrow \emptyset$;
2: $firstTuple, j \leftarrow 1$;
3: $C_{NG} \leftarrow \{c \mid c \in C_{LCSP} \wedge c$ is a non-global constraint$\}$;
4: while ($firstTuple < N^*$)
5:     $lastTuple \leftarrow \min\{firstTuple + q - 1, N^*\}$;
6:     $X_{csp_j} \leftarrow \{x \mid x \in X_{LCSP} \wedge firstTuple \leq$tuple of $x \leq lastTuple\}$;
7:     $C_{csp_j} \leftarrow \{c \mid c \in C_{NG} \wedge X_c \subseteq X_{csp_j}\}$;
8:     for all ($c \in C_{LCSP} \setminus C_{NG}$)
9:         $c^* \leftarrow c$;
10:        if $h = LM$
11:            label all $x \in X_{c^*} \setminus X_{csp_j}$ as constants from $T^*$;
12:        else if $h = IG$
13:            $X_{c^*} \leftarrow X_c \setminus \bigcup_{\ell > j} X_{csp_\ell}$;
14:            label all $x \in \bigcup_{\ell < j} X_{csp_\ell}$ as constants from $T^*$;
15:        endif
16:        $C_{csp_j} \leftarrow C_{csp_j} \bigcup \{c^*\}$;
17:    end for
18:    $csp_j \leftarrow \langle X_{csp_j}, C_{csp_j} \rangle$;
19:    $LCSP^{dep} \leftarrow LCSP^{dep} \bigcup \{csp_j\}$;
20:    $firstTuple \leftarrow lastTuple + 1$;
21:    $j \leftarrow j + 1$;
22: end while
23: return $LCSP^{dep}$;
```

simultaneously on different machines. It has been our experience that, when possible, solving fewer problems with a larger number of variables outperforms solving more problems with a smaller number of variables. Thus, at this stage we merge groups of q independent sub-CSPs into a single CSP. (It should be noted that the value of q here could be optimized by grouping the maximal number of independent sub-CSPs which results in a single manageable CSP. However, for the sake of simplicity we chose in this stage in our implementation the same value of q that was used also in decomposing LCSPs to dependent sub-CSPs, see Section 5.2.)

Each of the resulting CSPs is then formulated in terms of the solver, solved and the results are stored in T^* (this is done by the *solve* procedure used in Algorithm 2). Since these CSPs are independent, the order in which they are solved is irrelevant. In contrast to the independent sub-CSPs, the dependent sub-CSPs that were produced in Stage 2 must be iteratively solved in the same order in which they were defined. Remember that some of the actual values from T^* that define any c^* in csp_j (for $j > 1$) are only generated in iterations $\ell < j$, whence it is impossible to solve csp_j before solving csp_ℓ for all $\ell < j$.

5.4. LM vs. IG

LM guarantees that a solution is found for $LCSP$ at the end of the solving stage. In fact, it guarantees that T^* contains a solution to $LCSP$ at the end of each iteration. The reason for this is that LM is an inductive process. Before the first iteration ($j = 0$), T^* is initialized with a preliminary solution to $LCSP$. Now, assume that T^* contained a solution to $LCSP$ after iteration j ($j \geq 0$) and csp_{j+1} is then solved in iteration $j+1$. Clearly, csp_{j+1} has at least one solution, consisting of the current values of the cells in T^* that the variables in $X_{csp_{j+1}}$ represent. If csp_{j+1} has more than one solution, the one returned by the solver may or may not constitute a local modification of T^* from the previous iteration (differing by up to $q \cdot |A^{LCSP}|$ values). In any case, T^* necessarily contains a solution to $LCSP$ also after iteration $j+1$.

Unlike LM, IG does not guarantee that csp_{j+1} has at least one solution (even if csp_j was solved), and therefore does not guarantee that a solution

not all CSP packages support the required variable domains and/or constraints. Thus, the choice of CSP package can affect the extendability of the method.

is found for $LCSP$ (even if a solution does exist). Furthermore, there are COPs for which IG is guaranteed to fail. For example, consider an attribute A_i with domain D_i being the positive integers and a single global rule that requires that the total of all tuples for A_i equals some constant t. After the first iteration of IG, the solution of csp_1 is a set of values whose total already equals t and so no solution can ever be found for csp_2 (recall that D_i consists only of positive integers). This is a clear disadvantage of the IG heuristics compared to LM. On the other hand, IG does not require T^* to be initialized with a preliminary solution, which is sometimes hard to find.

The complete method for solving COPs is formulated in Algorithm 2.

5.5. The special case of PDOAT COPs

The PDOAT scenario has two important properties that will be utilized in the following subsections: (1) The rules in R^*_T do not depend on the values of T (see Section 3.2) (2) Since T satisfies all rules in R and $R^*_T = R \bigcup \{|T^*| = |T|\}$, it also satisfies all rules in R^*_T and thus $T \in U^*_T$.

5.5.1. A preliminary solution to be used with LM

Recall that LM requires T^* to be initialized with a preliminary solution. Since in the $PDOAT$ scenario $T \in U^*_T$, then T can be used as such a preliminary solution. However, such initialization provides the adversary more knowledge for exploitation as we explain next.

In Section 4.1 we denoted by W_{T^*} the set of all relations that could be obfuscated into T^*, given the adversarial background knowledge BK. An adversary who knows the set W_{T^*} may infer that $\hat{P}^{\mathcal{A}}_{T^*}(\overline{T}) = 0$, for all obfuscation algorithms $\mathcal{A}$ and for all relations $\overline{T} \in U \backslash W_{T^*}$. However, without knowing the specific algorithm that was used, the adversary cannot infer the probability that some relation in W_{T^*} is the original relation T, and thus all relations in W_{T^*} are equally likely to be T. An ideal obfuscation algorithm is an algorithm that does not add any information about these probabilities. In other words, such an ideal algorithm induces a uniform distribution $\hat{P}^{\mathcal{A}}_{T^*}$ on W_{T^*} (i.e., it does not provide the adversary any additional knowledge). If this property is held for all $T^* \in U^*$, we call such an algorithm a *zero-knowledge obfuscation algorithm*:

Definition 5.3. *An obfuscation algorithm $\mathcal{A}$ is called a zero-knowledge obfuscation algorithm if and only if* $\forall_{T^* \in U^*} \forall_{\overline{T} \in W_{T^*}} \hat{P}^{\mathcal{A}}_{T^*}(\overline{T}) = 1/|W_{T^*}|$.

Algorithm 2 solveCOP(T,S,R,h,q,s)

Input:
T: the original relation (together with the associated set of attributes A and the domain of tuples D).
R_T^*: the set of rules.
h: the chosen heuristic, either LM or IG.
q: the preferable number of tuples for each sub-CSP.
s: the CSP solver.

Output:
T^*: the obfuscated relation.

```
1: $RSs \leftarrow$ ruleSetDecomposition($R_T^*$); // Stage 1.1 (see Figure 4)
2: for all ($RS \in RSs$)
3:     if ($RS$ contains a global rule)
4:         $LCSP \leftarrow$ the single sub-CSP generated from $RS$; // Stage 1.2
5:         $LCSP^{dep} \leftarrow$ defineDependentCSPs($LCSP$,$|T^*|$,$q$,$h$); // Stage 2
6:         for all ($csp_j \in LCSP^{dep}$)
7:             solve($s$,$csp_j$); // stage 3
8:         end for
9:     else
10:        $CSPs \leftarrow$ independent sub-CSPs generated from $RS$; // Stage 1.2
11:        merge groups of $q$ sub-CSPs in $CSPs$ into a single CSP.
12:        for all ($csp \in CSPs$)
13:            solve($s$,$csp$); // Stage 3
14:        end for
15:    endif
16: end for
```

The next theorem claims that in some cases IG is a zero-knowledge obfuscation algorithm.

Theorem 5.1. *In the PDOAT scenario, if IG can provide at least one solution for each $\overline{T} \in U$ then IG is a zero-knowledge obfuscation algorithm.*

Proof. We assume that for each relation $\overline{T} \in U$, the set of solutions returned by IG has at least one member. Consider an arbitrary member of this set, denoted by $\overline{T}^* \in U^*_{\overline{T}}$. In the PDOAT scenario, IG completely disregards its input (recall that the rules in R^*_T do not depend on the values of T and IG does not need to be initialized with a preliminary solution). Therefore, we get that for any two relations $T_1, T_2 \in U_{\overline{T}^*}$, $\hat{P}^{IG}_{\overline{T}^*}(T_1) = \hat{P}^{IG}_{\overline{T}^*}(T_2)$. In particular we get that $\hat{P}^{IG}_{\overline{T}^*}(T_i) = 1/|W_{\overline{T}^*}|$ for any $T_i \in U_{\overline{T}^*}$. Hence, IG is a zero-knowledge obfuscation algorithm.
□

In contrast to the above, LM is not necessarily a zero-knowledge obfuscation algorithm in the PDOAT scenario. For example, consider a relation T that contains 100 tuples, one attribute A_1 with domain $D_1 = \{1, \ldots, 100\}$ and one rule R_1 which requires that the values in attribute A_1 are all different. Since T satisfies R_1 it is necessarily a permutation of the values $1, ..., 100$. Clearly, obfuscating T using LM and $q = 1$ will always produce an obfuscated relation T^* which is identical to T and thus $\hat{P}^{LM}_{T^*}(T) = 1$. On the other hand, $|W_{T^*}| = 100!$ because there exist 100! permutations of the values $1, \ldots, 100$ so $\hat{P}^{LM}_{T^*}(T) \neq 1/|W_{T^*}|$ and LM is therefore not a zero-knowledge obfuscation algorithm.

LM provides the adversary the following additional knowledge about the obfuscation process in the PDOAT scenario:

1. T^* was initialized with T as a preliminary solution.
2. Each block of variables contains maximum $q \cdot |A^{LCSP}|$ variables.
3. The order of selecting the variables is determined in advanced.

While LM heavily relies on the first two properties, the third property can easily be relaxed by adding randomization to the process of selecting the variables: for csp_1 we select q random tuples (instead of tuples $1, \ldots, q$), then for csp_2 we select q random tuples that were not selected for csp_1 (instead of tuples $q + 1, \ldots, 2q$), and so forth: for csp_j we select q random tuples that were not selected in any csp_ℓ, where $1 \leq \ell < j$. We denote henceforth this randomized version of the LM heuristics by LM^*. Clearly, LM^* provides less information to the adversary than LM does. As we shall see in Section 6, LM^* achieves significantly better privacy levels than LM.

5.5.2. Estimating ℓ

Assume that given a CSP as an input, the CSP solver outputs each one of the possible solutions with equal probabilities. Then, it is possible to derive reverse engineering algorithms for LM, LM^* and IG in the PDOAT scenario. In Section 6 we will use those reverse engineering algorithms in order to estimate the diversity level, ℓ, of the three heuristics, as explained in Section 4.2.

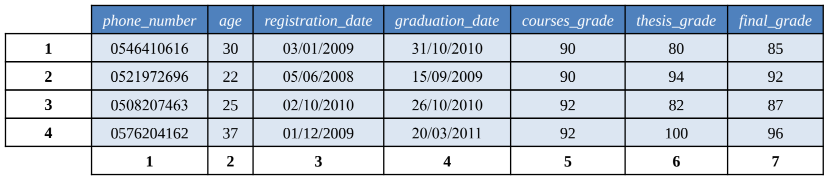
A reverse engineering algorithm for LM, denoted by LM^R, acts exactly as LM, except for the following two modifications: (1) adding the rule $\Pi_{ID,QID}(T^*) = BK$ to the set R^*_T and (2) selecting the tuples for each csp_j in a reverse order. That is, denoting the number of dependent sub-CSPs as $B = \lceil |T^*|/q \rceil$, the tuples selected for csp_j, $1 \leq j \leq B$ are between $(B - j) \times q + 1$ and $min(|T^*|, (B - j + 1) \times q)$. It is easy to verify that the combination of these two modifications ensures that executing LM^R with the input T^* outputs a relation $\overline{T} \in W_{T^*}$ with probability $\hat{P}^{LM}_{T^*}(\overline{T})$.

Similarly, a reverse engineering algorithm for LM^*, denoted by $(LM^*)^R$, acts exactly as LM^*, except for the following single modification: (1) adding the rule $\Pi_{ID,QID}(T^*) = BK$ to the set R^*_T. This modification ensures that executing $(LM^*)^R$ with the input T^* outputs a relation $\overline{T} \in W_{T^*}$ with probability $\hat{P}^{LM^*}_{T^*}(\overline{T})$.

As for IG, as explained earlier, in cases where it does find a solution, all relations in W_{T^*} are equally likely to be the original relation T, and thus $\forall_{\overline{T} \in W_{T^*}} \hat{P}^{\mathcal{A}}_{T^*}(\overline{T}) = 1/|W_{T^*}|$. Therefore, a reverse engineering algorithm for IG should output each relation $\overline{T} \in W_{T^*}$ with probability $1/|W_{T^*}|$. If the number of tuples in T is relatively small (i.e. $|T| \leq q$), such a reverse engineering algorithm can be obtained by modifying our COP solving method as follows: (1) adding the rule $\Pi_{ID,QID}(T^*) = BK$ to the set R^*_T and (2) omitting Stage 3 from the method (i.e. large sub-CSPs are not further decomposed into dependent sub-CSPs).

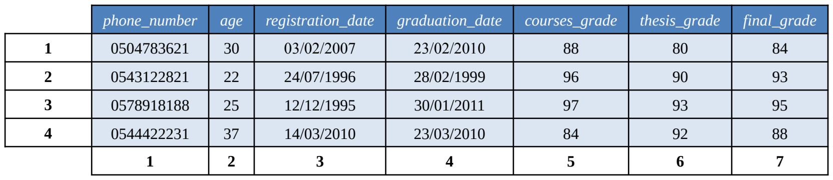
5.6. A toy example

In this section we demonstrate our obfuscation method in the PDOAT scenario using a toy example. Consider the M.Sc. students relation T from Figure 6. The set of attributes A consists of: A_1=phone_number, A_2=age,



	phone_number	age	registration_date	graduation_date	courses_grade	thesis_grade	final_grade
1	0546410616	30	03/01/2009	31/10/2010	90	80	85
2	0521972696	22	05/06/2008	15/09/2009	90	94	92
3	0508207463	25	02/10/2010	26/10/2010	92	82	87
4	0576204162	37	01/12/2009	20/03/2011	92	100	96
	1	**2**	**3**	**4**	**5**	**6**	**7**

Figure 6: A relation that stores tuples of M.Sc. students



	phone_number	age	registration_date	graduation_date	courses_grade	thesis_grade	final_grade
1	0504783621	30	03/02/2007	23/02/2010	88	80	84
2	0543122821	22	24/07/1996	28/02/1999	96	90	93
3	0578918188	25	12/12/1995	30/01/2011	97	93	95
4	0544422231	37	14/03/2010	23/03/2010	84	92	88
	1	**2**	**3**	**4**	**5**	**6**	**7**

Figure 7: An obfuscation of the relation in Figure 6

A_3=registration_date, A_4=graduation_date, A_5=courses_grade, A_6=thesis_grade and A_7=final_grade; the corresponding domains are: D_0 ={all strings}, $D_1 = \{1, \ldots, 120\}$, $D_2 = D_3$={all dates} and $D_4 = D_5 = D_6 = \{80, \ldots, 100\}$; the set of sensitive attributes $\tilde{S} = A \backslash \{\text{age}\}$ and the set of rules R consists of the following rules:

- R_1: The phone number attribute is a unique primary key.
- R_2: The phone number attribute must be a valid cellular phone number that conforms to the following regular expression pattern: "05[0247][0-9][0-9][0-9][0-9][0-9][0-9][0-9]".
- R_3: The registration date must precede the graduation date.
- R_4: The final grade is the average of the courses grade and the thesis grade.
- R_5: The average of all final grades is exactly 90 (as required by the faculty).

The three-stage obfuscation process is performed as follows:

1. The independent CSPs are generated as follows. First, we perform the rule set decomposition, as shown in Figure 8, and get the rule subsets: $RS_1 = \{R_1, R_2\}$, $RS_2 = \{R_3\}$ and $RS_3 = \{R_4, R_5\}$.

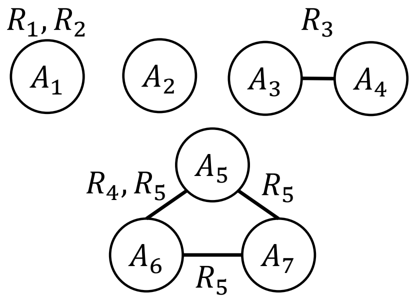


Figure 8: The rule graph of the toy problem

Second, we define the independent sub-CSPs for each rule subset that does not contain any global rule, i.e. RS_2, and get
$CSP_1 = \langle \{x_{1,3}, x_{1,4}\}, \{lt(x_{1,3}, x_{1,4})\} \rangle$
$CSP_2 = \langle \{x_{2,3}, x_{2,4}\}, \{lt(x_{2,3}, x_{2,4})\} \rangle$
$CSP_3 = \langle \{x_{3,3}, x_{3,4}\}, \{lt(x_{3,3}, x_{3,4})\} \rangle$
$CSP_4 = \langle \{x_{4,3}, x_{4,4}\}, \{lt(x_{4,3}, x_{4,4})\} \rangle$
where for each of the sub-CSPs above, the two sets represent the set of variables and the set of constraints respectively.

2. Assuming that we use the LM heuristic with $q = 2$, then we define the following two sets of dependent sub-CSPs: $\{CSP_5, CSP_6\}$ and $\{CSP_7, CSP_8\}$ where:
$CSP_5 = \langle \{x_{1,1}, x_{2,1}\}, \{allDifferent(x_{1,1}, x_{2,1}, x_{3,1}^{const}, x_{4,1}^{const}), regular(regexp, x_{1,1}), regular(regexp, x_{2,1})\} \rangle$
$CSP_6 = \langle \{x_{3,1}, x_{4,1}\}, \{allDifferent(x_{1,1}^{const}, x_{2,1}^{const}, x_{3,1}, x_{4,1}), regular(regexp, x_{3,1}), regular(regexp, x_{4,1})\} \rangle$
$CSP_7 = \langle \{x_{1,5}, x_{1,6}, x_{1,7}, x_{2,5}, x_{2,6}, x_{2,7}\}, \{\frac{x_{1,5}+x_{1,6}}{2} = x_{1,7}, \frac{x_{2,5}+x_{2,6}}{2} = x_{2,7}, \frac{x_{1,7}+x_{2,7}+x_{3,7}^{const}+x_{4,7}^{const}}{4} = 90\} \rangle$
$CSP_8 = \langle \{x_{3,5}, x_{3,6}, x_{3,7}, x_{4,5}, x_{4,6}, x_{4,7}\}, \{\frac{x_{3,5}+x_{3,6}}{2} = x_{3,7}, \frac{x_{4,5}+x_{4,6}}{2} = x_{4,7}, \frac{x_{1,7}^{const}+x_{2,7}^{const}+x_{3,7}+x_{4,7}}{4} = 90\} \rangle$
where $regexp = 05[0247][0-9][0-9][0-9][0-9][0-9][0-9]$ and $const$ represents a constant label.

3. Each one of the eight CSPs is solved using the CSP solver. The first four CSPs can be solved independently (possibly simultaneously). However, as described above, in order to reduce the number of calls to the CSP solver, they are first merged into groups of CSPs of size q, and then each group is solved by the CSP solver independently. The two dependent CSPs: CSP_5 and CSP_6 and the two dependent CSPs: CSP_7 and CSP_8 are solved iteratively.

Note that the *age* attribute is copied from T at stage 1 and is not modified later. A possible obfuscation of T, is given in Figure 7. A close inspection will show that it indeed satisfies all the rules in R.

6. Evaluation

The evaluation had the following goals: (1) demonstrating the flexibility of the COP framework by implementing and examining three of the aforementioned obfuscation scenarios, namely PDOAT, k-anonymity, and ℓ-diversity; (2) illustrating the feasibility of the new method in terms of execution time; (3) empirically comparing the influence of using different q values; and (4) empirically comparing the different heuristics in the PDOAT scenario.

All experiments were conducted on an Intel Core 2 Duo CPU 2.4 GHz personal computer with 8 GB of RAM, running Windows 8 Enterprise and MySQL 5.136 with default settings. For the solving stage, we used the CHOCO CSP solver [18], version 2.1.1. In Section 6.1 we describe our evaluation for the PDOAT scenario. In Section 6.2 we describe our evaluation for the k-anonymity scenario. We conclude in Section 6.3 which is devoted to the ℓ-diversity scenario.

6.1. PDOAT

Experiments in this section were performed on relations that have the same structure and comply with the same set of rules as described in the toy example in Section 5.6. In the first experiment we wanted to check the influence of q on the execution time. In order to achieve this goal, we created an original relation with $N = 100000$ tuples and obfuscated it using the $h \in \{IG, LM\}$ heuristic with $q \in \{10, 50, 100, 200, 333, 500, 1000\}$. Figure 9 demonstrates the influence of q and h on the execution time. Intuitively, using smaller values of q generates smaller CSPs which are easier to solve and thus take the CSP solver less time. At the same time, decreasing q increases the number of calls to the CSP solver, and that result in a higher overhead. In this experiment, using $q = 100$ was the best choice for both heuristics.

In the second experiment we wanted to check the influence of q and h on the obtained level of privacy. In order to achieve this goal, we created an original relation with $N = 1000$ tuples and obfuscated it using $h \in \{IG, LM, LM^*\}$ with $q \in \{1, 5, 10, 50, 100, 200, 333, 500, 1000\}$. Then, for each one of those obfuscated relations, we used the technique described in Section 4.2 in order to estimate the obtained level of diversity ℓ. More

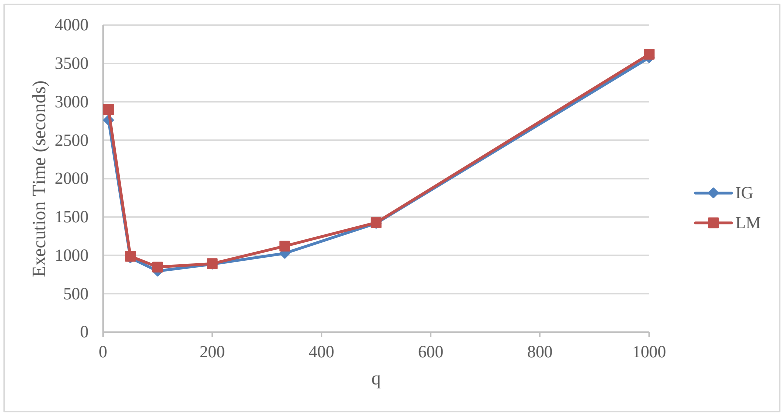


Figure 9: PDOAT: Execution time for different values of q.

specifically, we de-obfuscated each obfuscated relation, using the reverse engineering algorithms, $t = 510$ times (corresponding to $d = 0.05$ and $\alpha = 0.05$, see Table 1), and calculated the maximum probability of the different linkages (in this context we assumed that ID={phone_number} and S={final_grade}). In order to illustrate better the differences between the heuristics, we sorted the original table by the final_grade attribute before starting the obfuscation process. As a result, each group of q tuples that were considered by the LM heuristics contained final grades that were relatively close one to another. Figure 10 demonstrates the influence of q and h on the obtained level of diversity ℓ. IG achieves the best ℓ and this value does not depend on q. LM achieves the worst results since it uses a predefined grouping into blocks. LM^* achieves an intermediate level of privacy between IG and LM. Moreover, for values of q greater than or equal to 200, there is no difference between IG and LM^*. Using LM^* with an appropriate value of q seems to be the best choice, since it guarantees a solution, as opposed to IG. Note the extreme cases of $q = 1$ where LM and LM^* fail to achieve any level of privacy, and of $q = N$ where all three heuristics are identical.

In the third experiment we checked the dependence of the execution time on N and h. In order to do so, we created original relations with $N \in \{20000, 40000, 60000, 80000, 100000\}$ and obfuscated them using $h \in \{IG, LM^*\}$. In all those runs we used $q = 200$ (which, in view of the previous two experiments, seems to be the best choice for those heuristics, in terms of runtime and privacy). As can be seen by examining the curves that are denoted IG-1 and LM^*-1 in Figure 11, the execution time grows roughly lin-

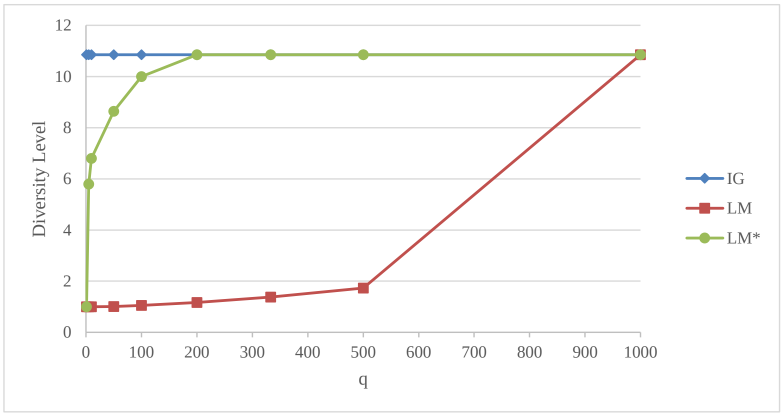


Figure 10: PDOAT: Estimated ℓ for different values of q

early with the number of tuples. Moreover, as expected, IG achieves slightly better execution times than LM^*. The explanation for this is that for global rules LM^* always solves problems with q tuples of variables and $N-q$ tuples of constants, while IG starts with a much smaller problem of only q tuples of variables and 0 tuples of constants, and only its last problem includes q tuples of variables and $N-q$ tuples of constants.

Finally, our fourth experiment examined the advantages of parallelizing the obfuscation process. Towards that end, we repeated each of the runs that were executed in the previous experiment using two processors. The resulting runtimes are shown in the curves denoted IG-2 and LM^*-2 in Figure 11. As can be seen, using two processors instead of one reduced the execution time by more than 30%.

To conclude this section, even though the original database contained several rules of different types and despite the fact that CSP is NP-hard in general, the solving process was completed in a reasonable time. Furthermore, the execution time was practically the same for all three heuristics, and using concurrency obviously improved execution time. It is important to note that when heuristics were not used, the CHOCO CSP solver was not able to complete its execution even for relatively small sized relations (e.g., $N = 15000$).

6.2. k-Anonymity

Our k-anonymity experiments were performed on two datasets: CENSUS [42] and ADULT [20]. The findings from the experiments on ADULT were

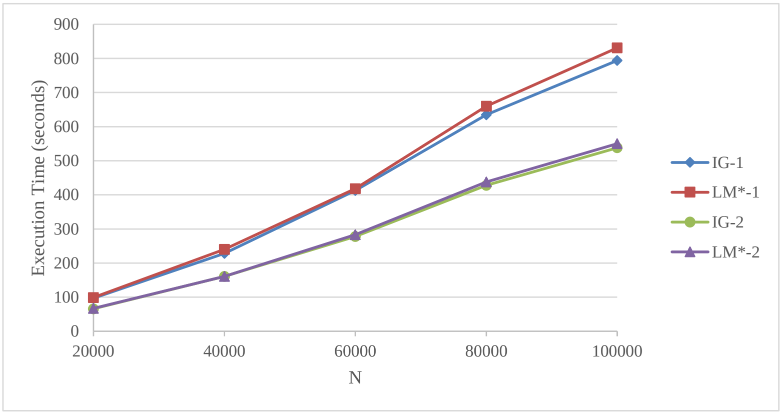


Figure 11: PDOAT: The influence of N, h and p on the execution time.

consistent with those from the experiments on CENSUS. In this section we report in detail the experiments with respect to CENSUS; the results with respect to ADULT are provided in Appendix B.

The CENSUS dataset has 500000 tuples and eight attributes: age, gender, education level, marital status, race, work class, country, and a sensitive attribute. We considered the two attributes age and gender to be quasi-identifiers, and used for them generalization taxonomies of heights 4 and 2 respectively.

We compared our new method (using the IG heuristic) with a dedicated k-anonymization algorithm, Mondrian [35], in terms of information loss and execution time. Information loss in this experiment was measured using the Loss Metric measure [32]. That measure takes the average information loss over all obfuscated cells, $T_i^*(j)$, $1 \leq i \leq N^*$, $1 \leq j \leq m^*$. The information loss in the cell $T_i^*(j)$ is defined as $(|T_i^*(j)|-1)/(|A_j|-1)$, where $|T_i^*(j)|$ is the size of the subset that replaces the original value in the jth attribute in the ith tuple, while $|A_j|$ is the number of possible values in that attribute. Note that in terms of privacy, both methods achieve the required level of privacy by definition, since the output in both methods issue k-anonymized tables.

In the first experiment we wanted to compare the two methods in terms of information loss and execution time for different relation sizes N. In order to achieve this goal, we used relations of size $N = 10000 \times i$, $1 \leq i \leq 10$, (i.e. the first N tuples from the CENSUS dataset) and we fixed $q = 2000$. The required privacy target was set to 5-anonymity. As can be seen in Figure 12, while the average information loss in the Mondrian obfuscated relations

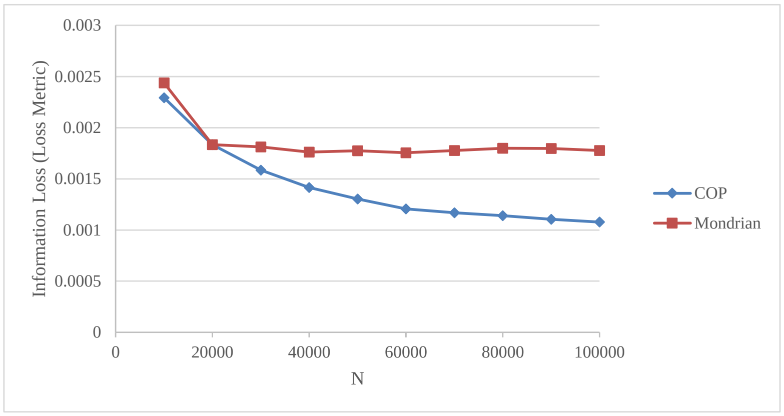


Figure 12: k-Anonymity: Information loss for different values of N.

remains almost unchanged for all $N \geq 20000$, the information loss in the output of our obfuscation method is always smaller, and it keeps decreasing with N. For $N = 100000$ the information loss in our method is roughly 60% of the information loss in the Mondrian algorithm, namely, the improvement is significant. As for the execution time, Figure 13 shows that the execution time of the new method grows roughly linearly with the number of tuples. For comparison, the Mondrian algorithm finished its execution within a few seconds even for $N = 100000$. Having said that, we note that anonymization is a process that is executed on tables that contain data that was accumulated during very long periods of time. Hence, an anonymization algorithm that runs in few seconds offer no real advantage compared to another algorithm that runs in few minutes or even hours. In our case, as our method issues relations with significantly smaller information losses and its runtime, even for very large tables, is manageable, it appears to be the better choice.

In the second experiment we wanted to test the influence of different q values on the information loss and execution time. In order to achieve this goal, we compared the two methods for a relation with a fixed size of $N = 100000$ and for $q \in \{200, 400, 600, 800, 1000, 2000, 3000, 4000, 5000, 10000\}$. Again, the privacy target was set to 5-anonymity. As can be seen in Figure 14, the information loss for the new method decreases with q and, for the largest value of q that we tested, the resulting information loss in our method was less than 0.5% of the information loss in the Mondrian algorithm (i.e., an improvement factor of more than 200). Figure 15 shows that the execution time of the new method grows roughly linearly with q. This is encouraging

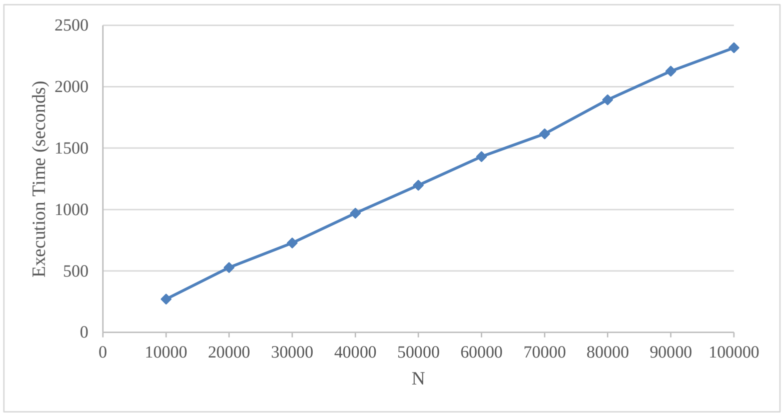


Figure 13: k-Anonymity: Execution time for different values of N.

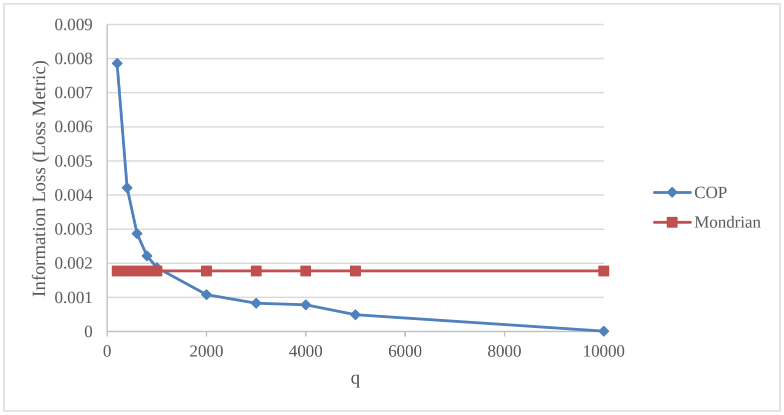


Figure 14: k-Anonymity: Information loss for different values of q.

since it allows us to choose larger values of q in order to obtain lower levels of information loss while not sacrificing scalability.

The above experiments were also conducted with the Entropy measure [25], instead of the Loss Metric measure. The findings were quite consistent with the ones reported above. Due to space limitations, we include herein only Figure 16 which shows the entropy information loss for the same settings of N, q, and k as in Figure 14.

6.3. ℓ-Diversity

The experiments in this section were performed on the CENSUS dataset [42]. In Appendix B we report the corresponding results with respect to the

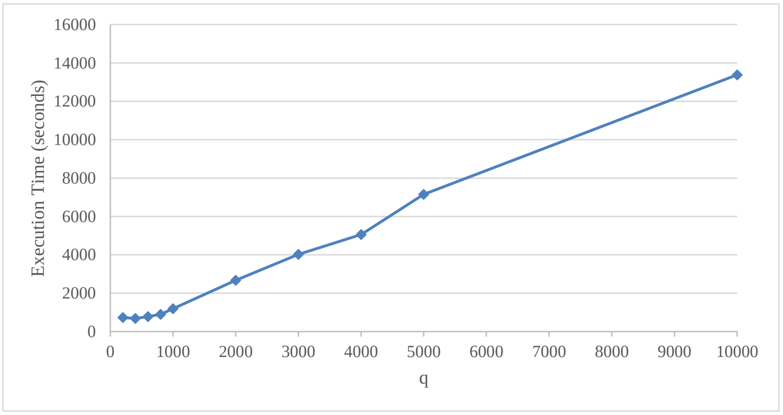


Figure 15: k-Anonymity: Execution time for different values of q.

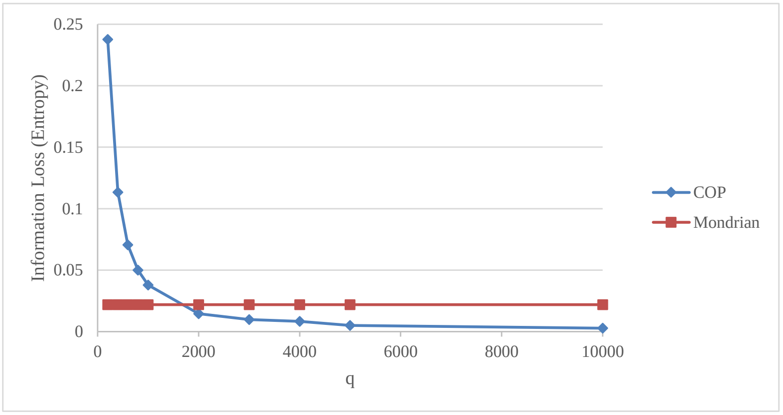


Figure 16: k-Anonymity: Information loss for different values of q (Entropy).

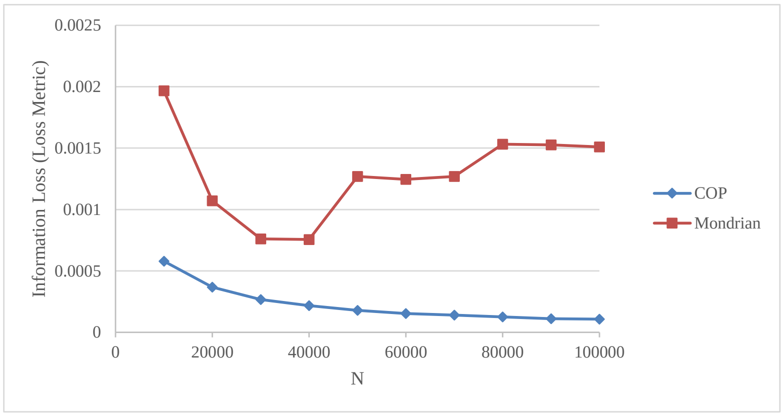


Figure 17: ℓ-Diversity: Information loss for different values of N.

ADULT dataset [20].

Here too, we compared our method (using the IG heuristic) with the Mondrian algorithm [35]. In the first experiment we compared the two methods in terms of information loss and execution time for different relation sizes N. As in Section 6.2, we used relations of size $N = 10000i$, $1 \leq i \leq 10$, and we fixed $q = 2000$. The required privacy target was set to 2-diversity. As can be seen in Figure 17, while the average information loss in the Mondrian obfuscated relations fluctuates, the information loss in the output of our obfuscation method is always smaller, and it keeps decreasing with N. For $N = 100000$ the information loss in our method is roughly 7% of the information loss in the Mondrian algorithm. As for the execution time, Figure 18 shows that the execution time of the new method grows roughly linearly with the number of tuples. The Mondrian algorithm, on the other hand, was always much faster and finished its execution within few seconds. As explained in Section 6.2, the runtime is a secondary consideration when choosing a method to anonymize tables that contain data that was accumulated over months and years.

In the second experiment we tested the influence of q on the information loss and execution time. We compared the two methods for a relation with a fixed size of $N = 100000$ and for $q \in \{200, 400, 600, 800, 1000, 2000, 3000, 4000, 5000, 10000\}$. Also here, the privacy target was set to 2-diversity. As can be seen in Figure 19, the information loss in the output of our method (as measured by the Loss Metric measure) decreases with q and, for the largest value of q that we tested, the resulting information loss in our method was about 1% of the in-

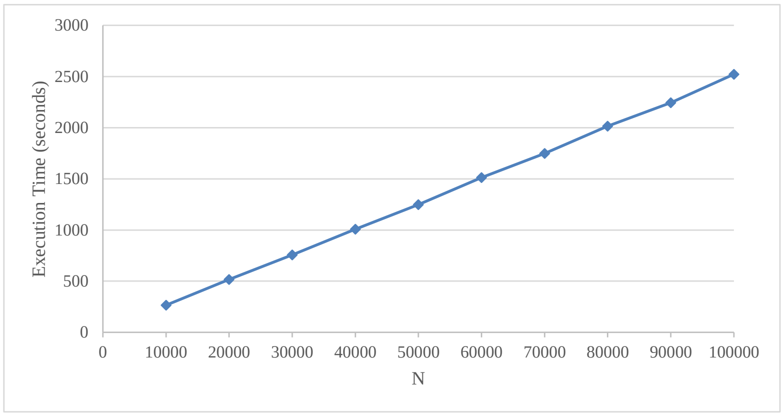


Figure 18: ℓ-Diversity: Execution time for different values of N.

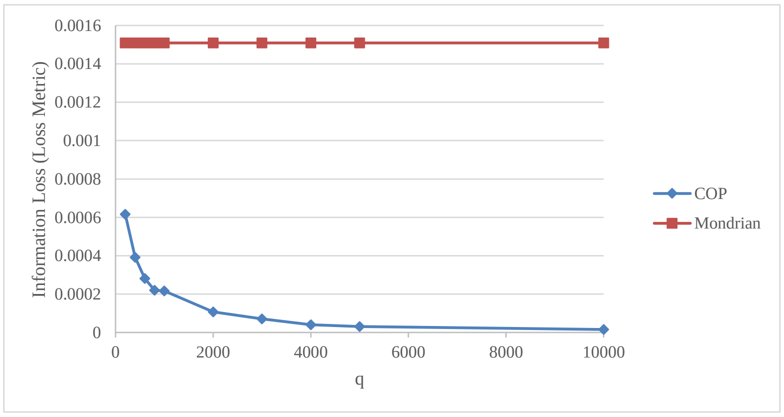


Figure 19: ℓ-Diversity: Information loss for different values of q.

formation loss in the Mondrian algorithm. We achieved similar results when using the entropy measure instead of the Loss Metric Measure, see Figure 20. Finally, Figure 21 shows that the execution time of the new method grows roughly linearly with q.

7. Summary and Future Work

In this paper we introduced the concept of COPs as a general framework for obfuscating relational databases. The flexibility of the COP definition was demonstrated by modeling several different problems of obfuscation. We suggested a general approach for solving COPs via the CSP framework, thereby

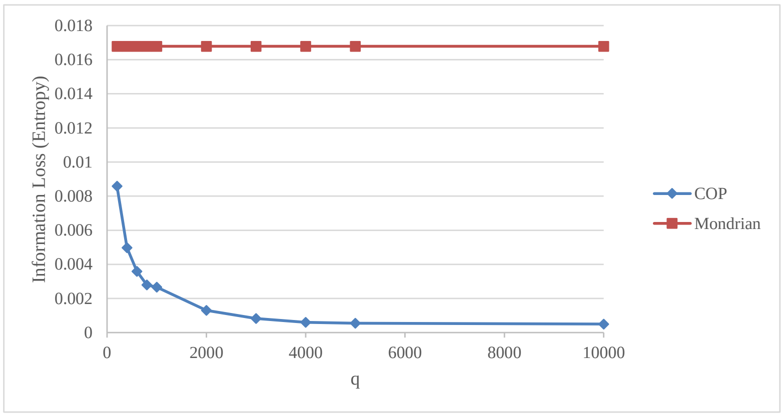


Figure 20: ℓ-Diversity: Information loss for different values of q (Entropy).

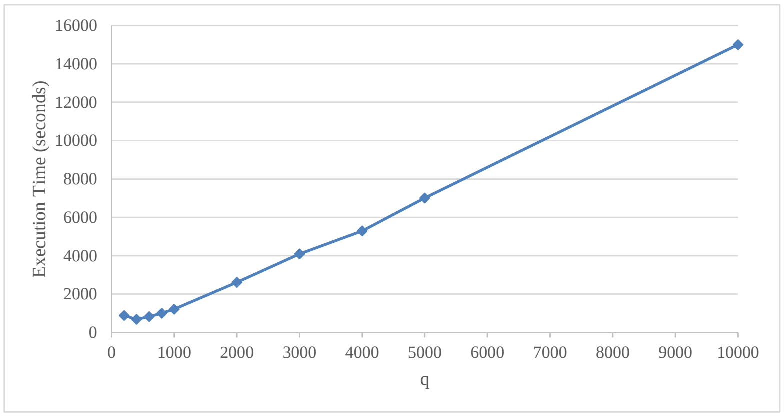


Figure 21: ℓ-Diversity: Execution time for different values of q.

supporting a wide variety of complicated COP rules. In order to cope with intractability issues that may arise from this approach, we provided two useful heuristics, namely *Incremental Generation (IG)* and *Local Modification (LM)*. We also showed how the well-known ℓ-diversity privacy measure can be incorporated into the COP framework in order to evaluate the privacy level of COP solutions.

Existing obfuscation methods are either confined to a certain application domain and their applicability is thus limited (e.g. dedicated algorithms for k-anonymization) or general enough but incapable of handling complicated rules (e.g. generation methods). Our method is general and thus can be used in cases where a dedicated algorithm cannot be used (e.g. if no dedicated algorithm exists). Our method can also be used in cases where a dedicated algorithm exists and, as we demonstrated, it can even achieve better results than those of the dedicated algorithm. Moreover, although our method is general, it is capable of handling complicated rules, since it harnesses the power of the well-studied CSP framework.

The COP framework is designed mainly for obfuscating relational databases. As we showed in Section 3.4, it is general enough to model even obfuscation problems of other types of data, such as graph data (and, more generally, social networks). This is achieved by representing the graph data as a relational database. However, from experimentation that we conducted, it appears that representing the graph data as a relational database and then translating the graph obfuscation problem into a COP, we get problems that are very hard to solve for large problem parameters. In the future, we intend to explore other ways of extending the COP framework to deal with graph data in a practical and efficient manner. In another line of research, we intend to compare our generic method with dedicated algorithms in other scenarios. As for the PDOAT scenario, we intend to develop methodologies for automatically extracting the COP rules (like the one described in [56]), and evaluate our method on a real or a benchmark database.

A. Summary of Notations

Notation	Meaning
U	The source universe
T	The original pre-obfuscated relation from U
$A_1, \ldots, A_m$	The attributes of T
$D_1, \ldots, D_m$	The domains of attributes of T
N	The number of tuples in T
U^*	The target universe
T^*	An obfuscation of T from U^*
$A_1^*, \ldots, A_m^*$	The attributes of T^*
$D_1^*, \ldots, D_m^*$	The domains of attributes of T^*
N^*	The number of tuples in T^*
ID	The set of identifier attributes of T
QID	The set of quasi-identifier attributes of T
S	The set of sensitive attributes of T
O	The set of other attributes of T
BK	The adversarial background knowledge
$R = \{R_1, \ldots, R_k\}$	Rules that apply on relations in U
$R_T^* = \{R_1^*, \ldots, R_q^*\}$	Rules that apply on relations in U^*
U_T^*	The set of all relations in U^* that satisfy R_T^*
$\mathcal{A}$	An obfuscation algorithm
$P_T^{\mathcal{A}}(T^*)$	The probability that algorithm $\mathcal{A}$ outputs T^* when given T as input
W_{T^*}	The set of all relations $T \in U$ that could possibly be the pre-image of T^*
$\hat{P}_{T^*}^{\mathcal{A}}(\overline{T})$	The posterior probability that $\overline{T} \in W_{T^*}$ is the pre-image of T under $\mathcal{A}$
$W_{T^*}^{id,s}$	The set of relations from W_{T^*} that contain the pair (id, s)
$p_{id,s}$	The probability that the original relation T contains the pair (id, s)
$p_{\max}$	The maximal probability over the different $p_{id,s}$
$\mathcal{A}^R$	A reverse engineering algorithm for $\mathcal{A}$
$\tilde{p}_{id,s}$	An estimator for $p_{id,s}$
t	The number of times that $\mathcal{A}^R$ should be executed
$x_{ij}^{T^*}$	The CSP variable corresponding to the cell in tuple i of attribute A_j
c	A CSP constraint
c^*	A derived global constraint
A^{Rule} or A^{CSP}	The attributes of a given rule or CSP
$X_{Constraint}$ or X_{CSP}	The variables of a given constraint or CSP
C_{CSP}	The constraint set of a given CSP
RS	A rule subset
$LCSP$	A large CSP that originates from a rule subset with a global rule
C_{NG}	The set of non-global constraints in $LCSP$
q_s	The maximum number of variables that a CSP solver can handle
q	The preferable number of tuples in a sub-CSP ($q \cdot \lvert A^{LCSP} \rvert \leq q_s$)
IG	The incremental generation heuristic
LM	The local modification heuristic
LM^*	The randomized version of the LM heuristic
h	A heuristic - either IG, LM or LM^*
$LCSP^{dep}$	The set of dependent sub-CSPs into which $LCSP$ was decomposed

Table 2: Notation table

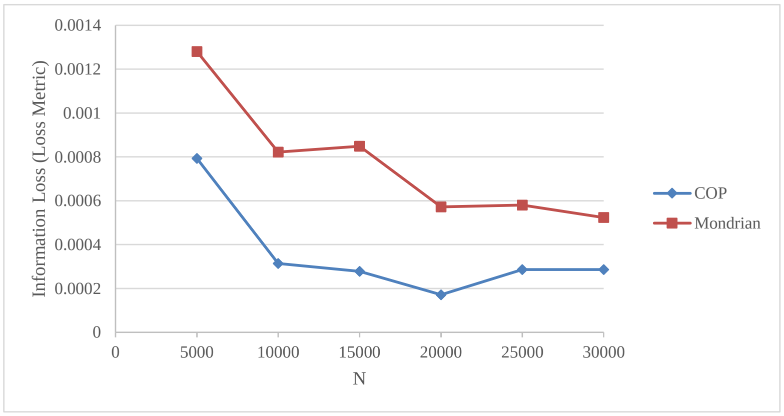


Figure 22: k-Anonymity: Information loss for different values of N.

B. Additional Experiments

B.1. k-Anonymity

In Section 6.2 we reported a suite of k-anonymity experiments that we conducted on the CENSUS datasets. We repeated the same suite of experiments for the ADULT dataset from the UCI Machine Learning Repository [20]. That dataset was extracted from the US Census Bureau Data Extraction System. It holds demographic information of a small sample of the US population with 14 quasi-identifiers such as age, gender, education level, marital status, and native country, and it contains 32,561 tuples. Here too we considered the two attributes age and gender to be quasi-identifiers, and used for them generalization taxonomies of heights 4 and 2 respectively.

Figures 22 and 23 show the information loss and execution time as obtained on subsets of ADULT of various sizes. (The corresponding figures in Section 6.2 are Figures 12 and 13). Figures 24 and 25 show the information loss and execution time for different values of q over the complete dataset (compare to Figures 14 and 15). Finally, Figure 26 shows the information loss, by the entropy measure, for different values of q (compare to Figure 16). As can be seen, the findings here are similar to those with respect to CENSUS. For example, concentrating on Figure 24, we see that the information loss for the new method decreases with q and, for the largest value of q that we tested, our method achieved an information loss that was more than 7 times smaller than that in the Mondrian algorithm.

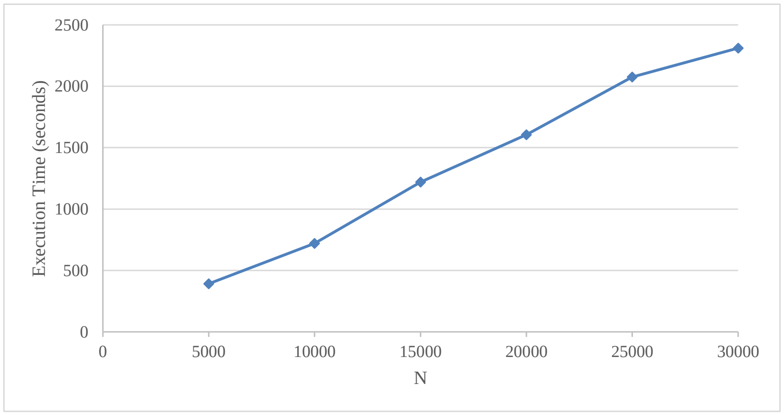


Figure 23: k-Anonymity: Execution time for different values of N.

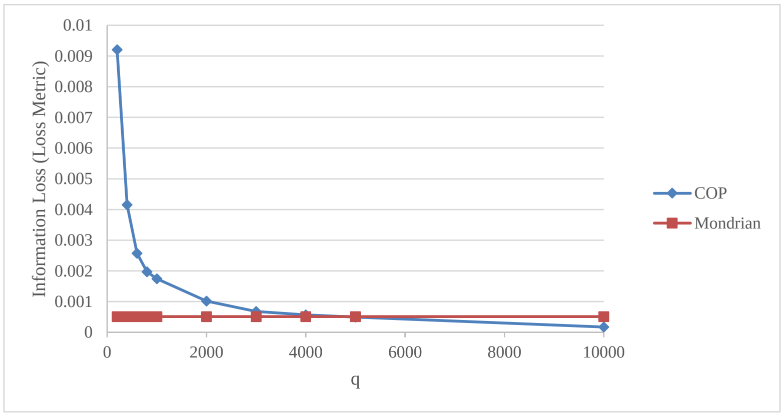


Figure 24: k-Anonymity: Information loss for different values of q.

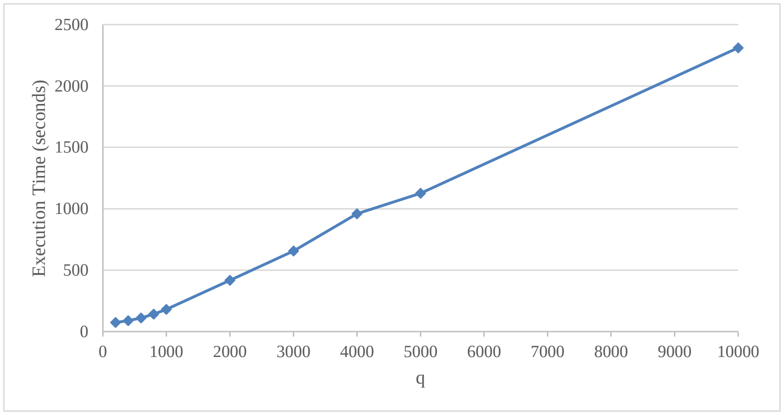


Figure 25: k-Anonymity: Execution time for different values of q.

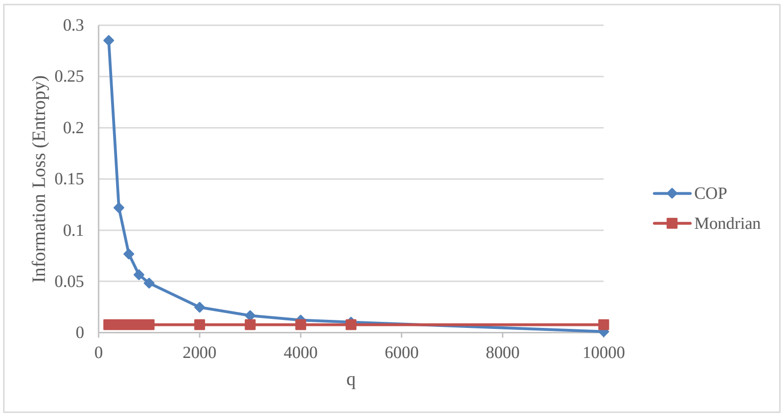


Figure 26: k-Anonymity: Information loss for different values of q (Entropy).

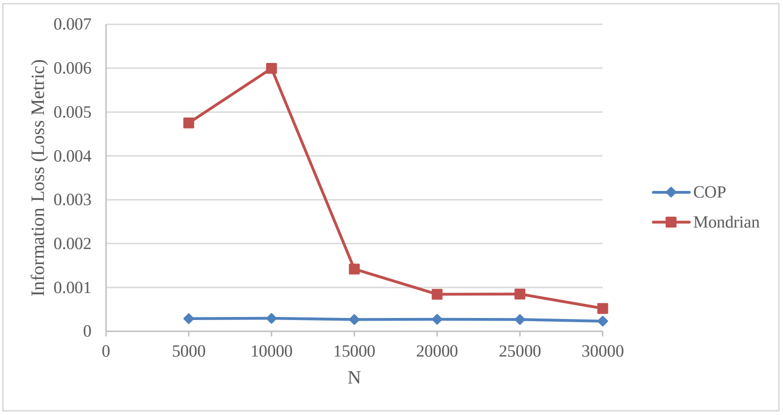


Figure 27: ℓ-Diversity: Information loss for different values of N.

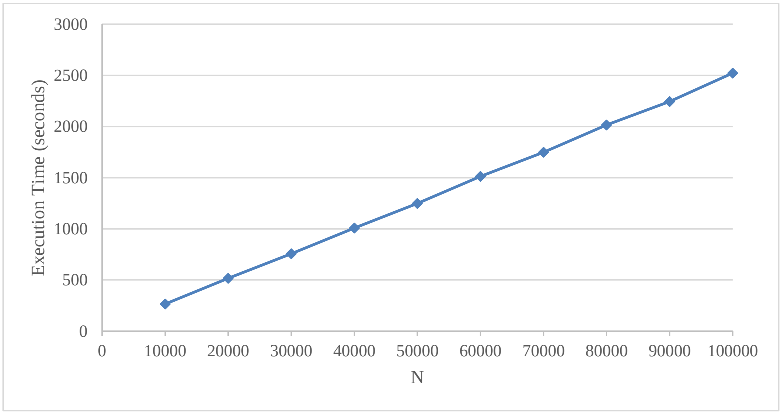


Figure 28: ℓ-Diversity: Execution time for different values of N.

B.2. ℓ-Diversity

In Section 6.3 we reported a suite of ℓ-diversity experiments that we conducted on the CENSUS datasets. We repeated the same suite of experiments for the ADULT dataset [20]. Figures 27-30 are parallel to Figures 17-20 in Section 6.3.

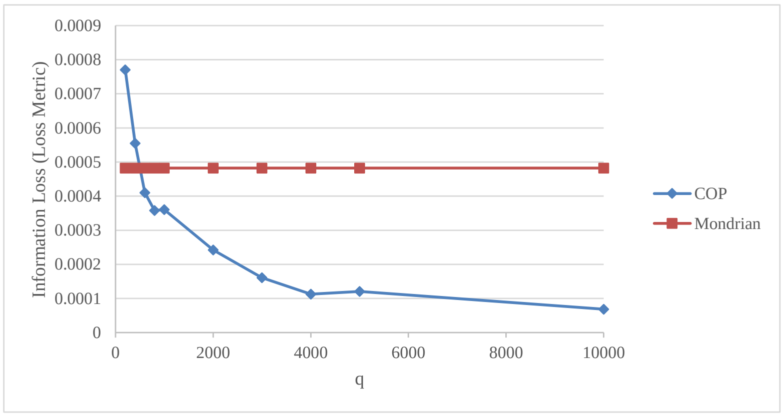


Figure 29: ℓ-Diversity: Information loss for different values of q.

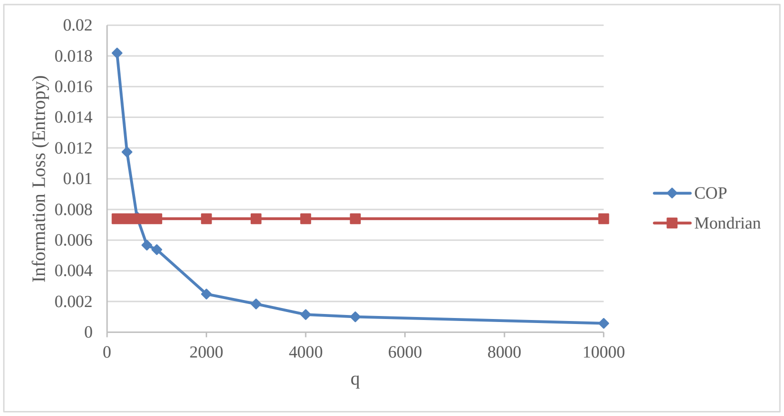


Figure 30: ℓ-Diversity: Information loss for different values of q (Entropy).

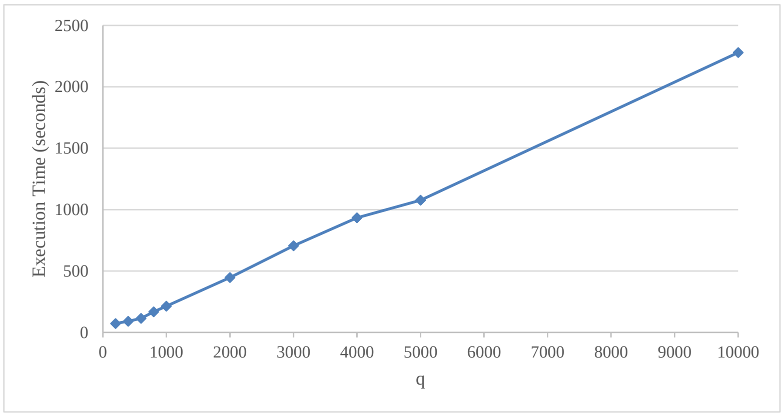


Figure 31: ℓ-Diversity: Execution time for different values of q.

References

[1] Aggarwal, C. and Yu, P. 2008. *Privacy-preserving data mining: models and algorithms*. Springer-Verlag New York Inc.

[2] Agrawal, R. and Srikant, R. 2000. Privacy-preserving data mining. *Sigmod Record 29*, 439–450.

[3] Bacchus, F., Grove, A. J., Koller, D., and Halpern, J. Y. 1992. From statistics to beliefs. In *Proceedings of the The National Conference on Artificial Intelligence*. 602–608.

[4] Backstrom, L., Dwork, C., and Kleinberg, J. M. 2007. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the International World Wide Web Conference*. 181–190.

[5] Bakken, D., Rarameswaran, R., Blough, D., Franz, A., and Palmer, T. 2004. Data obfuscation: anonymity and desensitization of usable data sets. *Security & Privacy, IEEE 2*, 34–41.

[6] Bayardo, R. and Agrawal, R. 2005. Data privacy through optimal k-anonymization. In *Proceedings of the International Conference on Data Engineering*. 217–228.

[7] Beldiceanu, N., Carlsson, M., and Rampon, J. 2005. Global constraint catalog.

[8] Binnig, C., Kossmann, D., Lo, E., and Özsu, M. T. 2007. Qagen: generating query-aware test databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 341–352.

[9] Boldi, P., Bonchi, F., Gionis, A., and Tassa, T. 2012. Injecting uncertainty in graphs for identity obfuscation. In *Proceedings of the International Conference on Very Large Data Bases*. Vol. 5. 1376–1387.

[10] Bonchi, F., Gionis, A., and Tassa, T. 2011. Identity obfuscation in graphs through the information theoretic lens. In *Proceedings of the International Conference on Data Engineering*. 924–935.

[11] Bruno, N. and Chaudhuri, S. 2005. Flexible database generators. In *Proceedings of the International Conference on Very Large Data Bases.* 1097–1107.

[12] Burnett, L., Barlow-Stewart, K., Proos, A., and Aizenberg, H. 2003. The" GeneTrustee": a universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine 10*, 506.

[13] Camouflage Software Inc. 2009a. *Camouflage Transformers.* Camouflage Software Inc. http://www.datamasking.com/usr/pdfs/CamouflageTransformers.pdf.

[14] Camouflage Software Inc. 2009b. *Enterprise-Wide Data Masking with the Camoufl age Translation Matrix.* Camouflage Software Inc. http://www.datamasking.com/usr/pdfs/CamouflageTransformers.pdf.

[15] Camouflage Software Inc. 2009c. *Secure Analytics - Maximizing Data Quality & Minimizing Risk for Banking and Insurance Firms.* Camouflage Software Inc. http://online.datamasking.com/forms/maximizingdatautility.

[16] Campan, A. and Truta, T. M. 2008. Data and structural k-anonymity in social networks. In *Proceedings of the International Workshop on Privacy, Security, and Trust in KDD*. 33–54.

[17] Castellanos, M., Zhang, B., Jimenez, I., Ruiz, P., Durazo, M., Dayal, U., and Jow, L. 2010. Data desensitization of customer

data for use in optimizer performance experiments. In *Proceedings of the International Conference on Data Engineering*. 1081–1092.

[18] CHOCO Team 2010. *CHOCO: an open source java constraint programming library*. CHOCO Team. http://www.emn.fr/z-info/choco-solver/pdf/choco-presentation.pdf.

[19] Duncan, K. and Wells, D. 1999. A Rule-Based Data Cleansing. *Journal of Data Warehousing 4*, 146–159.

[20] Frank, A. and Asuncion, A. 2010. UCI machine learning repository [http://archive.ics.uci.edu/ml]. *University of California, Irvine, School of Information and Computer Sciences*.

[21] Fung, B., Wang, K., Chen, R., and Yu, P. 2010. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys 42*, 1–53.

[22] Fung, B., Wang, K., and Yu, P. 2005. Top-down specialization for information and privacy preservation. In *Proceedings of the International Conference on Data Engineering*. 205–216.

[23] Ghinita, G., Karras, P., Kalnis, P., and Mamoulis, N. 2007. Fast data anonymization with low information loss. In *Proceedings of the International Conference on Very Large Data Bases*. 758–769.

[24] Gionis, A., Mazza, A., and Tassa, T. 2008. k-anonymization revisited. In *Proceedings of the International Conference on Data Engineering*. 744–753.

[25] Gionis, A. and Tassa, T. 2009. k-Anonymization with minimal loss of information. *Transactions on Knowledge and Data Engineering 21*, 206–219.

[26] Goldberger, J. and Tassa, T. 2010. Efficient anonymizations with enhanced utility. *Transactions on Data Privacy 3*, 149–175.

[27] Gray, J., Sundaresan, P., Englert, S., Baclawski, K., and Weinberger, P. 1994. Quickly generating billion-record synthetic databases. *SIGMOD Record 23*, 252.

[28] GridTools Ltd. 2009. *Simple Data Masking.* GridTools Ltd. http://www.grid-tools.com/download/Simple_Data_Masking.pdf.

[29] Hanhijärvi, S., Garriga, G., and Puolamaki, K. 2009. Randomization techniques for graphs. In *Proceedings of the SIAM International Conference on Data Mining.* 780–791.

[30] Hoag, J. E. and Thompson, C. W. 2007. A parallel general-purpose synthetic data generator. *SIGMOD Record 36*, 19–24.

[31] Houkjar, K., Torp, K., and Wind, R. 2006. Simple and realistic data generation. In *Proceedings of the International Conference on Very Large Data Bases.* 1246.

[32] Iyengar, V. 2002. Transforming data to satisfy privacy constraints. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining.* 279–288.

[33] Kenig, B. and Tassa, T. 2012. A practical approximation algorithm for optimal k-anonymity. *Data Mining and Knowledge Discovery 25*, 134–168.

[34] Last, M., Tassa, T., Zhmudyak, A., and Shmueli, E. 2014. Improving accuracy of classification models induced from anonymized datasets. *Information Sciences 256*, 138–161.

[35] LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. 2006. Mondrian multidimensional k-anonymity. In *Proceedings of the International Conference on Data Engineering.* 25.

[36] Li, N., Li, T., and Venkatasubramanian, S. 2010. Closeness: A new privacy measure for data publishing. *Transactions on Knowledge and Data Engineering 22*, 943–956.

[37] Liu, K. and Terzi, E. 2008. Towards identity anonymization on graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* 93–106.

[38] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. 2007. l-diversity: Privacy beyond k-anonymity. *Transactions on Knowledge Discovery from Data 1*, 3.

[39] Meyerson, A. and Williams, R. 2004. On the complexity of optimal k-anonymity. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 223–228.

[40] Parameswaran, R. and Blough, D. 1999. A Robust Data Obfuscation Approach for Privacy Preservation of Clustered Data. In *Proceedings of the International Conference on Data Mining*. 18–25.

[41] Rebollo-Monedero, D., Forne, J., and Domingo-Ferrer, J. 2010. From t-closeness-like privacy to postrandomization via information theory. *Transactions on Knowledge and Data Engineering 22*, 1623–1636.

[42] Ruggles, S., Alexander, T., Genadek, K., Goeken, R., Schroeder, M., and Sobek, M. 2010. Integrated public use microdata series: Version 5.0 [machine-readable database]. *Minneapolis: University of Minnesota*.

[43] Russell, S., Norvig, P., Canny, J., Malik, J., and Edwards, D. 1995. *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, NJ.

[44] Samarati, P. and Sweeney, L. 1998. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 188.

[45] Sweeney, L. 2002. k-Anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 557–570.

[46] Tassa, T. and Cohen, D. J. 2013. Anonymization of centralized and distributed social networks by sequential clustering. *Transactions on Knowledge and Data Engineering 25*, 311–324.

[47] Tassa, T., Mazza, A., and Gionis, A. 2012. k-concealment: An alternative model of k-type anonymity. *Transactions on Data Privacy 5*, 189–222.

[48] Thompson, S. 1987. Sample size for estimating multinomial proportions. *The American Statistician 41*, 42–46.

[49] Trabelsi, S., Salzgeber, V., Bezzi, M., and Montagnon, G. 2010. Data disclosure risk evaluation. In *Proceedings of the International Conference on Risks and Security of Internet and Systems*. 35–72.

[50] Wang, J., Luo, Y., Zhao, Y., and Le, J. 2009. A survey on privacy preserving data mining. In *Proceedings of the International Workshop on Database Technology and Applications*. 111–114.

[51] Wang, K., Fung, B., and Yu, P. 2005. Template-based privacy preservation in classification problems. In *Proceedings of the International Conference on Data Mining*. 8.

[52] Wang, K., Fung, B., and Yu, P. 2007. Handicapping attacker's confidence: an alternative to k-anonymization. *Knowledge and Information Systems 11*, 345–368.

[53] Wong, W. K., Mamoulis, N., and Cheung, D. W.-L. 2010. Nonhomogeneous generalization in privacy preserving data publishing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 747–758.

[54] Wu, W., Xiao, Y., Wang, W., He, Z., and Wang, Z. 2010. *k*-Symmetry model for identity anonymization in social networks. In *Proceedings of the International Conference on Extending Database Technology*. 111–122.

[55] Wu, X., Wang, Y., Guo, S., and Zheng, Y. 2007. Privacy preserving database generation for database application testing. *Fundamenta Informaticae 78*, 595–612.

[56] Wu, X., Wang, Y., and Zheng, Y. 2003. Privacy preserving database application testing. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*. 128.

[57] Xiao, X. and Tao, Y. 2006. Anatomy: simple and effective privacy preservation. In *Proceedings of the International Conference on Very Large Data Bases*. 139–150.

[58] Xiao, X. and Tao, Y. 2007. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 700.

[59] Yahalom, R., Shmueli, E., and Zrihen, T. 2010. Constrained Anonymization of Production Data: A Constraint Satisfaction Problem Approach. In *Proceedings of the VLDB Workshop on Secure Data Management*. Springer, 41–53.

[60] Zheleva, E. and Getoor, L. 2007. Preserving the privacy of sensitive relationship in graph data. In *Proceedings of the International Workshop on Privacy, Security, and Trust in KDD*. 153–171.

[61] Zhou, B. and Pei, J. 2008. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the International Conference on Data Engineering*. 506–515.