Diversifying Anonymized Data with Diversity Constraints

Mostafa Milani  
Western University  
London, Ontario, Canada  
mostafa.milani@uwo.ca

Yu Huang  
McMaster University  
Hamilton, Ontario, Canada  
huang223@mcmaster.ca

Fei Chiang  
McMaster University  
Hamilton, Ontario, Canada  
fchiang@mcmaster.ca

ABSTRACT
Recently introduced privacy legislation has aimed to restrict and control the amount of personal data published by companies and shared to third parties. Much of this real data is not only sensitive requiring anonymization, but also contains characteristic details from a variety of individuals. This diversity is desirable in many applications ranging from Web search to drug and product development. Unfortunately, data anonymization techniques have largely ignored diversity in its published result. This inadvertently propagates underlying bias in subsequent data analysis. We study the problem of finding a diverse anonymized data instance where diversity is measured via a set of diversity constraints. We formalize diversity constraints and study their foundations such as implication and satisfiability. We show that determining the existence of a diverse, anonymized instance can be done in PTIME, and we present a clustering-based algorithm. We conduct extensive experiments using real and synthetic data showing the effectiveness of our techniques, and improvement over existing baselines. Our work aligns with recent trends towards responsible data science by coupling diversity with privacy-preserving data publishing.

1 INTRODUCTION
Organizations often share user information with third parties to analyze collective user behaviour and for targeted marketing. For example, in the pharmaceutical industry, hospital and medical records are shared and sold to data brokers who aggregate longitudinal data from patient records, insurance claims and lab tests to derive collective insights for research and drug development. Protecting user privacy is critical to safeguard personal and sensitive data. The European Union General Data Protection Regulation (GDPR), and variants such as the California Consumer Protection Act (CCPA) aim to control how organizations manage user data. For example, a major tenet in GDPR is data minimization that states companies should collect and share only a minimal amount of personal data sufficient for their purpose. CCPA takes this one step further requiring companies to document and track onward transfer of data to third parties. Given the impossibility of knowing how a published data instance will be used in the future, determining a minimal amount of personal data to share is a challenge.

One solution is to apply differential privacy techniques to the entire data instance that provide provable guarantees. These guarantees often rely on aggregation queries over sufficiently large samples such that the output is not influenced by the presence (or absence) of any single record [12]. Unfortunately, applications often experience poor data utility and accuracy due to the necessary data randomization in differential privacy. Privacy-preserving data publishing (PPDP) provides a middle-ground to safeguard individual privacy while ensuring the published data remains practically useful for subsequent analysis. One of the benefits of PPDP is the focus on publishing actual data, rather than statistical summaries and relationships about the data. Anonymization is the most common form of PPDP, where quasi-identifiers and/or sensitive values are obfuscated via suppression or generalization [14].

As anonymized instances are shared with third parties for decision making and analysis, there is growing interest to ensure that data (and the algorithms that generate and use the data) are diverse and fair. Diversity is a rather established notion in data analytics that refers to the property of a selected set of individuals. Diversity requires the selected set to have a minimum representation from each group of individuals [11, 23] while determining the minimum bound for each group is often domain and user dependent.

To avoid biased decision making, incorporating diversity into computational models is essential to prevent and minimize discrimination against disadvantaged and minority groups. In this paper, we focus on diversity, and study how diversity requirements can be modeled and satisfied in PPDP. In PPDP, non-diverse data instances that exclude minority group give an inaccurate representation of the population in subsequent data analysis. Unfortunately, early PPDP work [14, 22, 24], and recent work on PPDP for linked data and graphs [15, 16] have not studied techniques to include diversity in published data instances. Consider the following example demonstrating the challenges of applying diversity in PPDP.

Example 1.1. Table 1 shows relation $R$ containing patients medical records describing gender (GEN), ethnicity (ETH), age (AGE), province (PRV), city (CTY), and diagnosed disease (DIAG). Third-parties such as pharmaceuticals, insurance firms are interested in an anonymized $R$ containing patients from diverse geographies, gender, and ethnicities. Let GEN, ETH, AGE, CTY, PRV, be quasi-identifier (QI) attributes, and let DIAG be a sensitive attribute. Existing PPDP methods such as $k$-anonymity prevent re-identification of an individual along the QI attributes from $k − 1$ other tuples. Table 2 shows a $k$-anonymized instance for $k = 3$ where tuples are clustered along the QI attributes via value suppression [22, 24].

The $k$-anonymization problem is to generate a $k$-anonymous relation through an anonymization process, such as generalization and suppression, while incurring minimum information loss. Suppression replaces some QI attribute values with $\ast$s to achieve $k$-anonymity. There are several measures of information loss in PPDP [14], e.g., counting the number of $\ast$s. Existing $k$-anonymization techniques do not preserve diversity in $R$ since these information loss measures do not capture diversity semantics.

Unfortunately, existing methods fail to provide any diversity guarantees in published, privatized data instances. This leads to inaccurate and biased decision making in downstream data analysis. For example, in health care, anonymized patient records that
We make the following contributions:

1. We study the foundations of diversity constraints; their validation, implication, satisfiability, and finding a minimal cover. We also give an axiomatization of diversity constraints, and present an algorithm for checking implication using this axiomatization.

2. We define the \((k, \Sigma)\)-anonymization problem that seeks a \(k\)-anonymous relation with value suppression that satisfies \(\Sigma\). We introduce DIVA, a clustering-based algorithm that solves the \((k, \Sigma)\)-anonymization problem with minimal suppression.

3. We present two selection strategies to improve the DIVA algorithm performance by selectively ordering candidate constraints and clusterings to minimize conflict and save computation.

4. We conduct an extensive evaluation using real data collections demonstrating the effectiveness and efficiency of our selection strategies over the naive version of DIVA and show the utility of diversity constraints over an existing baseline.

**Paper Organization.** In Section 2, we present necessary definitions and notation. We study foundations of diversity constraints in Section 3, and introduce the DIVA algorithm and our selection strategies in Section 4. We present our evaluation results in Section 5, related work in Section 6, and conclude in Section 7.

### 2 PRELIMINARIES

#### 2.1 Relations and Dependencies

A relation \(R\) with a schema \(\mathcal{R} = \{A_1, \ldots, A_n\}\) is a finite set of \(n\)-ary tuples \(\{t_1, \ldots, t_N\}\). We denote by small letters \(x, y, z\) as variables. Let \(A, B, C\) refer to single attributes and \(X, Y, Z\) as sets of attributes.

A cell \(c = t[A_i]\) is the \(i\)-th position in tuple \(t\) with value denoted by \(c\). We use \(c\) to refer to \(c\) value if it is clear from the context.

Table 5 summarizes our symbols and notations.

| Symbol | Description |
|--------|-------------|
| \(R, \mathcal{R}\) | relation and relational schema |
| \(A, B\) | relational attributes |
| \(X, Y, Z\) | sets of relational attributes |
| \(*, \star\) | suppression relation, symbol for a suppressed value |
| \(\phi, \sigma, \Sigma\) | single and set of diversity constraints |
| \(C, S\) | cluster and clustering (set of clusters) |

#### 2.2 Privacy-Preserving Data Publishing

\(k\)-anonymity prevents re-identification of an individual in an anonymized data set [22, 24]. Attributes in a relation are either identifiers such as SSN that uniquely identify an individual, quasi-identifier (QI) attributes such as ethnicity, address, age that together can identify an individual, or sensitive attributes that contain personal information.

**Definition 2.1 (QI-group and \(k\)-anonymity).** A relation \(R\) is \(k\)-anonymous if every QI-group has at least \(k\) tuples. A QI-group is a set of tuples with the same values in the QI attributes.

For example, Table 2 has three QI-groups, \([-1, 1, 2, 3]\), \([-2, 3, 4, 5]\), and \([-1, 2, 3, 4, 5]\), and is \(3\)-anonymous. Recent extensions of \(k\)-anonymity include \(t\)-diversity, \(t\)-closeness, and \((X, Y)\)-anonymity, which provide improved privacy confidence (cf. [14] for a survey). We apply \(k\)-anonymity for its ease of presentation, however, our definitions and techniques can be extended to include recent PPDP models.
2.3 Suppression

Suppression generates an anonymized relation \( R' \) from a relation \( R \) by replacing some QI values in \( R \) with \( * \). We denote this by \( R \subseteq R' \). Suppression clearly causes information loss which is typically measured by the number of \( * \)'s in \( R' \).

**Definition 2.2 (k-anonymization problem [24]).** Given \( R \), the problem of \( k \)-anonymization is finding \( R' \) such that (1) \( R \subseteq R' \); (2) \( R' \) is \( k \)-anonymous; and (3) \( R' \) incurs minimum information loss.

The \( k \)-anonymization problem is NP-hard for \( k \geq 3 \) even when QI attributes have only two values but it is in PTIME for \( k = 2 \). The best approximation for a general value of \( k \) is a \( O(\log k) \), and for the special case \( k = 3 \), there is a \( 2 \)-approximation algorithm [14].

2.4 Diversity Constraints

Diversity constraints are originally proposed for the set selection problem as proposed by Stoyanovich et al. [23] (Section 2.4). We introduce a formal definition of \( \varphi \) of the form

\[ \text{Definition 2.1 (Diversity Constraints).} \]

\[ R = \text{constraint} \sigma \text{ of the form } \forall a \in \{\text{QI values}\} \text{ check if this number lies in the frequency range } [a, l]. \]

Let there be \( d \) distinct values of the sensitive attribute and \( m_i \) with \( i \in [1, d] \) be the number of selected items with each distinct value such that \( m_i \in [0, M] \) and \( \sum_i(m_i) = M \). A diversity constraint \( \phi \) of the form \( floor_i \leq m_i \leq ceiling_i \) specifies upper and lower bounds on \( m_i \), i.e., the number of items with the \( i \)-th sensitive value. These constraints ensure representation from each category known as coverage-based diversity. To avoid tokenism, where there is only a single representative from each category, we can increase the lower bound, e.g., \( m_i > 1 \). Given a set of diversity constraints \( \Sigma \) of the form \( \phi \in \Sigma \), we define our initial problem statement.

**Definition 2.3 (Problem Statement ((k, \( \Sigma \))-anonymization)).** Consider a relation \( R \) of schema \( \mathcal{R} \), a constant \( k \), a set of diversity constraints \( \Sigma \). The \((k, \Sigma)\)-anonymization problem is to find a relation \( R' \) where: (1) \( R \subseteq R' \), (2) \( R' \) is \( k \)-anonymous, (3) \( R' \mid \Sigma \), and (4) \( R' \) has minimum information loss, i.e., a minimum number of \( * \)'s.

3 FOUNDATIONS

We apply the concept of diversity constraints as proposed by Stoyanovich et al. [23] (Section 2.4). We introduce a formal definition of these diversity constraints, study their validation, implication and satisfaction, define minimal cover, and present an axiomatization.

**Definition 3.1 (Diversity Constraints).** A diversity constraint over a relation schema \( \mathcal{R} \) is of the form \( \sigma = (A[a], \lambda_1, \lambda_r) \) in which \( A \in \mathcal{R} \), \( a \in \text{dom}(A) \) and \( \lambda_1, \lambda_r \) are non-negative integers. The diversity constraint \( \sigma \) is satisfied by a relation \( R \) of schema \( \mathcal{R} \) denoted \( R \models \sigma \) if and only if there are at least \( \lambda_1 \) and at most \( \lambda_r \) occurrences of the value \( a \) in attribute \( A \) of relation \( R \). We call \( [\lambda_1, \lambda_r] \) the frequency range and \( A[a] \) the target value of \( \sigma \). A set of diversity constraints \( \Sigma \) is satisfied by \( R \), denoted by \( R \models \Sigma \), iff \( R \) satisfies every \( \sigma \in \Sigma \).

3.1 Validation

The validation problem is to decide whether \( R \models \sigma \). Assuming\( \sigma = (A[a], \lambda_1, \lambda_r) \), we can run a query that counts the number of occurrences of the target value \( a \) in attribute \( A \) of \( R \) and then check if this number lies in the frequency range \( [\lambda_1, \lambda_r] \).

Diversity constraints can be extended to multiple attributes by replacing \( A[a] \) with \( X[t] \), where \( X \) is a set of attributes and \( t \) is a tuple with values from these attributes. This extended diversity constraint \( \sigma = (X[t], \lambda_1, \lambda_r) \) is satisfied by \( R \) if there are at least \( \lambda_1 \) and at most \( \lambda_r \) tuples in \( R \) with the same attribute values in \( t \). The validation problem for a multi-attribute diversity constraint is answered in a similar manner as the single attribute diversity constraint by extending the conditions to include each target attribute values, and aggregating the results via a count query. Similar to traditional functional dependencies, validation is in PTIME since we can automatically generate SQL queries from the diversity constraints [13].

3.2 Implication and Axiomatization

We present an axiomatization for diversity constraints, and formally define the logical implication problem.

**Definition 3.2 (Logical Implication).** Given a set of diversity constraints \( \Sigma \) over schema \( \mathcal{R} \), and a diversity constraint \( \sigma \notin \Sigma \), we say \( \Sigma \) implies \( \sigma \), denoted by \( \Sigma \models \sigma \), if and only if any relation \( R \models \Sigma \), then \( R \models \sigma \). Given any finite set \( \Sigma \) and a constraint \( \sigma \), the implication problem is to determine whether \( \Sigma \models \sigma \).

To test for logical implication \( \Sigma \models \sigma \), and infer a new \( \sigma \), we give a sound and complete axiomatization for diversity constraints.

**Axiom 1 (Fixed Attributes):** If \( \sigma = (X[t], \lambda_1, \lambda_r), \sigma' = (X[t], \lambda'_1, \lambda'_r), [\lambda_1, \lambda_r] \subseteq [\lambda'_1, \lambda'_r], \) then \( \sigma \models \sigma' \).

For example, let \( \sigma' = (\text{GEN}[\text{Female}], 1, 5) \), and \( \sigma = (\text{GEN}[\text{Female}], 2, 4) \), which require \( [1,5] \) and \( [2,4] \) females, respectively. The frequency range of \( \sigma' \) subsumes the range of \( \sigma \), indicating that \( \sigma \) is more restrictive. Thus, if a relation \( R \) satisfies \( \sigma \), it also satisfies \( \sigma' \).

**Axiom 2 (Attribute Extension):** Let \( \sigma = (X[t], \lambda_1, \lambda_r), \sigma' = (X'[t'], 0, \lambda_1), X'[t'] \subseteq X[t], \) then \( \sigma \models \sigma' \).

Intuitively, if we add new target attribute values to a satisfied constraint, we cannot guarantee that there exist tuples with the added values \( (\lambda'_r = 0) \). In contrast, if there exist tuples that contain the new target attribute values, their frequency would be upper bounded by \( \lambda_r \). For example, if \( \sigma = (\text{ETH}[\text{Female}], 1, 5) \) then we can infer \( \sigma'' = (\text{GEN}, \text{ETH}[\text{Female}, \text{Caucasian}], 0, 5) \).

**Axiom 3 (Attribute Reduction):** Let \( \sigma = (X[t], \lambda_1, \lambda_r), \sigma' = (X'[t'], \lambda_1, \lambda_r), X'[t'] \subseteq X[t], \) then \( \sigma \models \sigma' \).

Axiom 3 states that for a satisfied diversity constraint \( \sigma \), if we remove a set of target attribute values from \( X[t] \), we can infer at least \( \lambda_1 \) occurrences of the values \( X'[t'] \subseteq X[t] \). For example, if \( \sigma = ((\text{GEN}, \text{ETH}[\text{Female}, \text{Caucasian}], 1, 5) \) holds over \( R \), then we can conclude at least 1 individual is female, i.e., \( \sigma'' = (\text{ETH}[\text{Female}], 1, +\infty) \) holds. However, we cannot claim the number of females in \( R \) is limited to 5 since there may be individuals from other ethnicities in \( R \).

**Axiom 4 (Range Intersection):** Let \( \sigma = (X[t], \lambda_1, \lambda_r), \sigma' = (X[t], \lambda'_1, \lambda'_r), \) then for any \( \sigma'' = (X[t], \lambda'_1, \lambda'_r) \) where \( [\lambda'_1, \lambda'_r] \subseteq [\lambda_1, \lambda_r] \cap [\lambda'_1, \lambda'_r] \), it follows that \( \{\sigma, \sigma'\} \models \sigma'' \).
Algorithm 1: Implies ($\Sigma, \sigma = (X[t], \lambda_1, \lambda_2)$)

Output: $\Sigma \models \sigma$.

1. $\delta := [0, +\infty)$;
2. foreach $\sigma' = (X'[t'], \lambda'_1, \lambda'_2) \in \Sigma$
   do
   if $X'[t'] = X[t]$ then $\delta := \delta \cap [\lambda'_1, \lambda'_2]$;
3. if $X \subset X'$ and $t \subset t'$ then $\delta := \delta \cap [0, \lambda'_1]$;
4. if $X' \subset X$ and $t' \subset t$ then $\delta := \delta \cap [\lambda'_1, +\infty)$;
5. return $\delta \subseteq [\lambda_1, \lambda_r]$.

Intuitively, the set of tuples satisfying $\sigma, \sigma'$ would also satisfy a new diversity constraint $\sigma''$ that is more restrictive whose frequency range is the intersection of $[\lambda_1, \lambda_r]$ and $[\lambda'_1, \lambda'_2]$. For example, let $\sigma = (\text{GEN}[\text{Female}], 1, 5)$ and $\sigma' = (\text{GEN}[\text{Female}], 3, 7)$, we can infer $\sigma'' = (\text{GEN}[\text{Female}], 3, 5)$.

Theorem 3.3. The axiomatization (Ax. 1-4) is sound and complete.

Proof Sketch. Axioms 1-4 are sound as shown with the above examples. The axiomatization is also complete since any constraint $\sigma$ that can be inferred from $\Sigma$ can be obtained by applying Axioms 1-4 in a sequence. We can prove this by showing for any $\sigma'$ that does not follow from $\Sigma$ via these axioms is not a logical implication of $\Sigma$, i.e., by construction of a relation that satisfies $\Sigma$ but not $\sigma'$.

We present Algorithm 1 that tests for logical implication by applying Axioms 1-4, i.e., checking whether $\Sigma \models \sigma$. The algorithm starts with a diversity constraint $(X[t], 0, +\infty)$ with the most general target range $\delta = [0, +\infty)$ (Line 1), which is satisfied by any relation, hence, is inferred from $\Sigma$. The algorithm iterates over each constraint in $\Sigma$ to find constraints $\sigma'$ with target values in $\sigma$ to infer more restricted ranges $\delta$. Using Axioms 1-3 and in Lines 3-5, the algorithm subsequently finds target ranges $[\lambda'_1, \lambda'_2]$, $[0, \lambda'_2]$, $\lambda'_1$, and $\lambda'_2$, if applies Axiom 4 to restrict $\delta$. If $\delta$ is included in $[\lambda_1, \lambda_r]$ (applying Axiom 1 in Line 6 and checking if $(X[t], \delta) \models \sigma$), then $\Sigma$ implies $\sigma$. Algorithm 1 runs in linear time w.r.t. $|\Sigma|$, and proves the implication problem can be solved in linear time.

Example 3.4. Consider the following execution of Algorithm 1 to check whether $\Sigma \models \sigma$, where $\Sigma = \{\sigma', \sigma''\}$, $\sigma' = (\text{CTY}[\text{Calgary}], 2, 10)$, $\sigma'' = (\text{GEN}, \text{ETH}, \text{CTY})(\text{Female}, \text{Caucasian}, \text{Calgary}), 4, 7)$, and $\sigma = ((\text{ETH}, \text{CTY})\text{Calgary}, 5, 8)$. The range $\delta$ first reduces to $[0, 10]$, and then to $[4, 10]$ after considering $\sigma'$ and $\sigma''$. Line 6 returns true since $[5, 8] \subseteq [4, 10]$, and thus, $\Sigma$ implies $\sigma$.

3.3 Satisfiability

The satisfiability problem is to determine whether a set of constraints $\Sigma$ is satisfiable, i.e., does there exist a relation $R$ such that $R \models \Sigma$. We can apply Axioms 1-4, and test whether $\Sigma$ implies the false diversity constraint $\phi$, i.e., $\phi = (X[t], \lambda_1, \lambda_r)$ with empty range $[\lambda_1, \lambda_r] = \emptyset$. Since there is no relation $R$ that satisfies $\phi$, if we infer that $\Sigma \models \phi$, then there is no $R$ that satisfies $\Sigma$, and $\Sigma$ is not satisfiable.

Example 3.5. Let $\Sigma = \{\sigma', \sigma''\}$, where $\sigma' = ((\text{ETH}, \text{CTY})\text{Calgary}, 6, 8)$ and $\sigma'' = (\text{CTY}[\text{Calgary}], 1, 5)$. Clearly, $\Sigma$ is unsatisfiable since the target ranges are not compatible for persons from Calgary. From Algorithm 1, we can check that $\Sigma$ implies the false constraint $\phi = (\text{CTY}[\text{Calgary}], \emptyset)$, where $\emptyset = \emptyset$. Given this, we conclude that $\Sigma$ implies $\phi$, and $\Sigma$ is not satisfiable.

3.4 Minimal Cover

To avoid redundancy, it is preferable to have a minimal set of constraints that are equivalent to $\Sigma$, i.e., a minimal cover of $\Sigma$.

Definition 3.6. (Minimal Cover). Given two sets of diversity constraints, $\Sigma$ and $\Sigma'$, we say $\Sigma'$ covers $\Sigma$, if for every constraint $\sigma \in \Sigma$, $\Sigma' \models \sigma$. A minimal cover $\Sigma'$ of $\Sigma$ is a set of diversity constraints such that $\Sigma'$ covers $\Sigma$, and there is no subset of $\Sigma'$ that covers $\Sigma$.

Intuitively, a set of constraints $\Sigma'$ is minimal if every constraint $\sigma' \in \Sigma'$ is necessary. That is, there is no constraint in $\Sigma'$ such that $\Sigma' \setminus \{\sigma'\} \models \sigma'$. In Example 3.4, the set of constraints $\Sigma = \{\sigma', \sigma''\}$ is minimal since neither $\{\sigma''\} \not\models \sigma'$ nor $\{\sigma''\} \not\models \sigma'$. However, $\Sigma \cup \{\sigma\}$ is not minimal since $\{\sigma', \sigma''\} \models \sigma$ and $\sigma$ is redundant. We can check the minimality of a set of constraints $\Sigma$ using Algorithm 1, by testing the logical implication of every constraint in $\Sigma$.

In the remainder of the paper, we assume $\Sigma$ is satisfiable and minimal. We verify and reject unsatisfiable sets of constraints, and verify minimality by removing redundant constraints.

3.5 ($k, \Sigma$)-Anonymization: Decision Problem

We now turn to the decision problem of ($k, \Sigma$)-anonymization, and show that the decision problem is tractable but unfortunately, the problem in Defn 2.3 is not. First, given our updated Defn. 3.1 of diversity constraints, we update $\Sigma$ in our ($k, \Sigma$)-anonymization problem statement in Defn 2.3 to reflect these constraints.

The Decision Problem. Given relation $R$, value $k$, diversity constraints $\Sigma$, the ($k, \Sigma$)-anonymization decision problem is to decide whether there exists an $R'$ such that: (1) $R \subseteq R'$; (2) $R'$ is $k$-anonymous; and (3) $R' \models \Sigma$.

We assume for any constraint $\sigma = (X[t], \lambda_1, \lambda_r)$ in $\Sigma$, $\lambda_1 \geq k$. Constraint $\sigma$ can only be satisfied when the frequency of value $a$ is greater than or equal to $k$ due to the $k$-anonymity condition in $R'$.

Theorem 3.7. Consider relation $R$, value $k$, and a constraints $\Sigma$. The ($k, \Sigma$)-anonymization decision problem is in PTIME w.r.t. $|\Sigma|$.

Proof Sketch. The proof of Theorem 3.7 is based on a naive algorithm that exhaustively checks every possible clustering of tuples in $R$ to generate $X$-groups that satisfy $\Sigma$. Since the number of possible clusterings is polynomial in the size of $R$, and exponential in the size of $\Sigma$, the decision problem is tractable. In Section 4, we propose an algorithm for solving the ($k, \Sigma$)-anonymization decision problem by optimizing the naive algorithm, i.e., our new algorithm generates a $k$-anonymized instance $R'$ that satisfies $\Sigma$.

Proposition 3.8. Consider relation $R$, value $k$, and constraints $\Sigma$. The ($k, \Sigma$)-anonymization problem is NP-hard w.r.t. $|R|$.

Proof Sketch. The ($k, \Sigma$)-anonymization problem extends the $k$-anonymization problem, which is proved to be NP-hard [14].

4 THE DIVA ALGORITHM

We present the DIVersity and Anonymization algorithm (DIVA) that solves the ($k, \Sigma$)-anonymization problem. DIVA takes as input a relation $R$, a minimal and satisfiable set of diversity constraints $\Sigma$, constant $k$, and returns a $k$-anonymous and diverse relation $R'$ that satisfies $\Sigma$. DIVA is a clustering-based anonymization algorithm that works in two phases: (i) clustering, by partitioning $R$ into disjoint
Algorithm 2: DIVA ($R, \Sigma, k$)

Output: $k$-anonymous and diverse relation.

1. $S_\Sigma := \text{DiverseClustering}(R, \Sigma, k)$;
2. if $S_\Sigma = \emptyset$ then return unsatisfiable;
3. $R_k := \text{Suppress}(S_\Sigma)$;
4. foreach $C_i \in S_\Sigma$ do $R := R \setminus C_i$;
5. $R_k := \text{Anonimize}(R, k)$;
6. return $\text{Integrate}(R, R_k)$;

clusters of size $\geq k$; and (ii) suppression, by suppressing a minimal number of QI values in each cluster such that they have the same QI values, and form a QI-group of size $\geq k$. The result is a $k$-anonymous relation, as every QI-group is of size $\geq k$.

Algorithm 2 presents the DIVA algorithm details. In the clustering phase in Line 1, DIVA uses the DiverseClustering procedure to generate a set of diverse clusters $S_\Sigma$. These clusters guarantee that the diversity constraints in $\Sigma$ will be satisfied by $R_k$ after the suppression phase in Line 3. If there is no $k$-anonymous relation $R'$ that satisfies $\Sigma$, there is no such clustering, and DIVA returns $S_\Sigma = \emptyset$. We provide details of DiverseClustering in Section 4.1.

In the suppression phase, DIVA suppresses values according to the clusters in $S_\Sigma$. The Suppress procedure iterates over tuples in each cluster of $S$, and suppresses $A_i$ attribute values if there is more than one value for $A_i$ in the same cluster. Assuming each cluster in $S$ contains at least $k$ tuples, the result of Suppress in $R$ is a $k$-anonymous relation.

Returning to Algorithm 2, DIVA anonymizes the remaining tuples of $R$ that are not in $S_\Sigma$ (Line 4) by applying an existing $k$-anonymization algorithm (Line 5). DIVA is amenable to any $k$-anonymization algorithm. In Line 6, Integrate returns $R' = R_k \cup R_k'$ if $R' \equiv \Sigma$. Otherwise, $R'$ falsifies the upper bounds of some of the constraints in $\Sigma$ because of $R_k$, and Integrate resolves this by suppressing minimal values in $R'$ to satisfy $\Sigma$.

Example 4.1. Consider relation $R$ in Table 1, $k = 2$, and $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$, where $\sigma_1 = \{\text{ETH} = \text{Asian}, 2, 5\}$, $\sigma_2 = \{\text{ETH} = \text{African}, 1, 3\}$ and $\sigma_3 = \{\text{CTY} = \text{Vancouver}, 2, 4\}$. DiverseClustering returns a clustering $S_\Sigma = \{C_1, C_2, C_3\}$ where $C_1 = \{t_5, t_{10}\}$, $C_2 = \{t_5, t_6\}$, and $C_3 = \{t_6, t_10\}$. Tuples $t_5, t_{10}$ contain the same value ETH = Asian, and together with $C_1$ guarantee that the lower bound in $\sigma_1$ will be satisfied. $C_2$ and $C_3$ satisfy the lower bounds of $\sigma_2$ and $\sigma_3$ for ETH = African and CTY = Vancouver, respectively. Note that other clusterings, which satisfy $\Sigma$, are possible, such as $\{C_2, \{t_5, t_{10}\}\}$. In Section 4.1, we describe how we select one of these clusterings.

DiverseClustering returns an empty set if there is no clustering that satisfies $\Sigma$. For example, if $k = 3$ there is no possible anonymization that satisfies $\sigma_1, \sigma_3$. In particular, there are no clusters of size 3 that preserve both Vancouver and Asian. For $k = 2$, DIVA continues with the Suppress procedure that transforms the tuples in $S_\Sigma$ to $R_k = \{g_5, g_6, g_9\}$ as shown in Table 3. DIVA anonymizes the remaining tuples $R \setminus S_\Sigma = \{t_1, t_2, t_3, t_4\}$ using an existing $k$-anonymization algorithm that minimizes the number of $\ast$s. In this case, the optimal result is $R_k = \{g_1, g_2, g_3, g_4\}$ in Table 3. Integrate procedure returns $R_k \cup R_k'$, which satisfies $\Sigma$.

Integrate resolves any inconsistency caused by adding $R_k$. For example, if $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ in which $\sigma_4 = \{\text{GEN} = \text{Male}, 1, 3\}$, $R_k \cup R_k'$ because there are 4 males in $R_k \cup R_k$. Integrate suppresses GEN in $g_5, g_6, g_9, g_4$ to satisfy $\sigma_4$.

4.1 Diverse Clustering

We now describe the DiverseClustering routine in the DIVA algorithm, and define a clustering that satisfies a diversity constraint.

Definition 4.2. Given a diversity constraint $\sigma$ over a relation $R$ and a clustering $S$ with clusters of tuples in $R$, $S$ satisfies $\sigma$, denoted as $S \models \sigma$ if Suppress($S$) $\models \sigma$. The clustering $S$ satisfies a set of constraints $\Sigma$, if $S \models \sigma$ for every $\sigma \in \Sigma$.

In Example 4.1, $S = \{C_1\}$ satisfies $\sigma_1$ since Suppress($S$) $\models \{g_5, g_6\}$ (cf. Table 3) satisfies $\sigma_1$. The objective of DiverseClustering is to find $S_\Sigma$ that satisfies $\Sigma$. This works by computing clustering $S_\Sigma$ that satisfy diversity constraints $\sigma_i \in \Sigma$, and then computing $S_{\Sigma_i}$ by merging the clusterings $S_{\Sigma_i}$. The main challenge is to ensure the clustering for each $\sigma_i$ is consistent with clusterings for the other constraints in $\Sigma$. If so, this allows us to merge the $S_{\Sigma_i}$ to obtain $S_{\Sigma_i}$.

Definition 4.3 (Consistent clusterings). Consider diversity constraints $\sigma_i$ and $\sigma_j$ over relation $R$. Two clusterings $S_{\Sigma_i}$ and $S_{\Sigma_j}$ are consistent if and only if $S_{\Sigma_i} \models \sigma_i$ and $S_{\Sigma_j} \models \sigma_j$ implies Merge($S_{\Sigma_i}, S_{\Sigma_j}$) $\models \{\sigma_i, \sigma_j\}$.

Merge in Defn. 4.3 merges clusters if they overlap, otherwise their union is computed, e.g., Merge($\{t_5, t_6\}, \{t_6, t_7\}$) = $\{t_5, t_6, t_7\}$, and Merge($\{t_5, t_6\}, \{t_7\}$) = $\{t_5, t_6, t_7\}$. We can check the consistency of two clusterings using Merge and Suppress.

Example 4.4. In Example 4.1, $S_2 = \{\{t_5, t_6\}\}$ and $S_3 = \{\{t_6, t_7\}\}$ are not consistent w.r.t $\sigma_2$ and $\sigma_3$, because $S_2 \equiv \sigma_2$ and $S_3 \equiv \sigma_3$, but Merge($S_2, S_3$) $\models \{\{t_6, t_7\}\} \not\models \{\sigma_2, \sigma_3\}$. This occurs since $t_6$ appears in two different clusters $\{t_5, t_6\}$ and $\{t_6, t_7\}$ in $S_2$ and $S_3$, respectively. Consequently, the value Vancouver will be suppressed in the clustering Merge($S_2, S_3$) because $t_6[\text{CTY}] \neq t_6[\text{CTY}]$, and hence, $\sigma_3$ will not be satisfied.

It is straightforward to show that if $S_{\Sigma_i} \models \sigma_i$ for every $\sigma_i \in \Sigma$, and every pair of $S_{\Sigma_i}, S_{\Sigma_j}$ are consistent, we can generate $S_{\Sigma_i} \equiv \Sigma$ by merging all clusterings $S_{\Sigma_j}$. Note that it is not necessary to check consistency of every pair of clusterings $S_{\Sigma_i}, S_{\Sigma_j}$, as we only need to check if $\sigma_i, \sigma_j$ apply to some tuples that are common to both constraints. We use this intuition to transform our problem of computing all $S_{\Sigma_i}$ to the problem of graph coloring.

4.1.1 Modeling as Graph Coloring.

We model the problem of finding the clusterings $S_{\Sigma_i}$ as a graph coloring problem. Given an undirected graph $G = (\Gamma, E)$, where $\Gamma$ and $E$ denote the set of vertices and edges, respectively, and $\delta$ distinct colors, the graph coloring problem is to color all vertices subject to certain constraints. In its simplest form, no two adjacent vertices can have the same color.

For relation $R$ and diversity constraints $\Sigma$, we model each diversity constraint $\sigma_i \in \Sigma$ as a vertex $v_i \in \Gamma$. We use $v_i[\text{constraint}]$ to refer to $\sigma_i$. We define the relevant tuples of $\sigma_i$, denoted $I_{\sigma_i} \subseteq R$, as tuples containing the target values of $\sigma_i$. We record the relevant tuples of $\sigma_i$ in vertex $v_i$. An edge $e_{ij} \in E, e_{ij} = (v_i, v_j)$, exists between vertices $v_i$ and $v_j$ if there is at least one tuple in the intersection of their relevant tuple sets, i.e., $(I_{\sigma_i} \cap I_{\sigma_j}) \neq \emptyset$. In Example 4.1, $G$ contains three vertices corresponding to $\sigma_1$, $\sigma_2$, $\sigma_3$ (cf. Figure 1), and...
Section 4.2, we present two strategies for selecting candidate ver-
coloring. We then try to color vertices \( v_2 \) and \( v_3 \) by recursively calling \( \text{Coloring} \) in Line 11. If vertex \( v_2 \) is selected, the only clustering is \( S_{\sigma_3} = \{ t_5, t_6 \} \) that is consistent with \( S_{\sigma_1} \). Considering the last vertex \( v_3 \), we iterate over the clusterings for \( \sigma_3 \), and determine that the only consistent clustering (w.r.t. \( S_{\sigma_1} \) and \( S_{\sigma_2} \)) is \( \{ t_5, t_6 \} \), which we assign to \( S_\sigma \). Since we have found a clustering satisfying all constraints (i.e., a coloring of all vertices), the \( \text{Coloring} \) routine returns true with \( V \) containing the vertices and their colors (i.e., clusterings). The calling routine \( \text{DiverseClustering} \) uses \( V \) to compute the final clustering as \( S_\Sigma = \{ t_5, t_6 \} \).

**Algorithm 3: DiverseClustering(\( R, \Sigma, k \))**

Output: Clustering \( S_\Sigma \).

1. \( G := \text{BuildGraph}(R, \Sigma) \);
2. \( V := \emptyset; S_\Sigma := \emptyset \);
3. if \( \text{Coloring}(G, V, R) \) then
   4.     foreach \( (v_i, c_i) \in V \) do \( S_\Sigma := \text{Merge}(S_\Sigma, c_i, \text{clustering}) \);
5. return \( S_\Sigma \);

**Algorithm 4: Coloring(G, V, R)**

Output: true if there exists a coloring of \( G \), otherwise false.

1. if \( V \) contains all vertices of \( G \) then return true;
2. \( v := \text{NextVertex}(G, V) \);
3. foreach \( S \in \text{Clusterings}(v, \text{constraint}, R) \) do
   4.     \( \text{consistent} := \text{true} \);
   5.     foreach \( (v', c') \in V \) s.t. \( v' \) is adjacent to \( v \) do
      6.         if \( S \) and \( c' \) are inconsistent then
          7.             \( \text{consistent} := \text{false}; \text{break} \);
   8.     if \( \text{consistent} \) then
      9.         \( c := \text{new color with clustering} S; \)
   10.        \( V := V \cup \{ (v, c) \} \);
   11.        if \( \text{Coloring}(G, V, R) \) then return true;
   12.    \( V := V \backslash \{ (v, c) \} \);
13. return false.

two edges \( E = \{(v_1, v_2), (v_2, v_3)\} \). The relevant sets \( I_{\sigma_1} = \{ t_8, t_9, t_{10} \}, I_{\sigma_2} = \{ t_6, t_7, t_{10} \}, \) have a non-empty intersection of \( \{ t_8, t_{10} \} \). Similarly, for \( I_{\sigma_3} = \{ t_5, t_6 \}, I_{\sigma_1} \cap I_{\sigma_2} = \{ t_6 \} \). We note that \( I_{\sigma_2} \cap I_{\sigma_3} = \emptyset \). Choosing a color \( c_i \) for vertex \( v_i \) is analogous to finding a clustering \( S_{\sigma_i} \) for \( \sigma_i \). In our setting, to color two adjacent \( v_i, v_j \), we must check that their clusterings \( S_{\sigma_i} \) and \( S_{\sigma_j} \) are consistent. We define \( c_i, c_j \) to refer to clustering corresponding to \( c_i, c_j \).

Algorithm 3 presents the details of \( \text{DiverseClustering} \). We build the graph \( G \) for \( \Sigma \) and \( R \) (Line 1). We then initialize the clustering \( S_\Sigma \) and a mapping \( V \) that stores the color (assigned clustering) for each vertex (Line 2), and checks if a coloring exists via \( \text{Coloring} \).

Algorithm 4 presents the recursive function, \( \text{Coloring} \), that takes a graph \( G \), the mapping \( V \) (specifying the colored vertices), relation \( R \), and returns true if the remaining vertices of \( G \) can be colored; otherwise it returns false. In the naive version, \( \text{Coloring} \) randomly selects an uncolored vertex (Line 2) to color using \( \text{NextVertex} \). In Section 4.2, we present two strategies for selecting candidate vertices. Given a vertex \( v \), we try to color \( v \) by checking whether the candidate clustering of \( v \) and its adjacent vertices are inconsistent (Lines 3-12). The routine \( \text{Clusterings} \) returns candidate clusterings \( S \) that satisfy \( v, \text{constraint} \) (\( \text{Suppress}(S) = v, \text{constraint} \)). For example, in Example 4.1, \( \text{Clusterings}(\sigma_1, R) \) contains four different clusterings \( \{ t_5, t_6 \}, \{ t_8, t_{10} \}, \{ t_8, t_9, t_{10} \} \), while \( \text{Clusterings}(\sigma_2, R) \) contains only one clustering \( \{ t_5, t_6 \} \). In the naive algorithm, we assume \( \text{Clusterings} \) returns clusterings in random order. We present strategies in Section 4.2 to order the clusterings to minimize inconsistencies. In Lines 4-12, we check whether \( S \) has inconsistency with the clustering of any constraint modeled by an adjacent vertex \( v' \). If they are consistent, we generate a new color \( c \) assigned to the clustering \( S \), and temporarily color \( v \) with \( c \) by adding \( (v, c) \) to \( V \). We then recursively call \( \text{Coloring} \) to check whether the remaining vertices in \( G \) can be colored. If the color \( c \) does not work, i.e., \( \text{Coloring} \) returns false in Line 11, we remove \( (v, c) \) from \( V \), and try another color. If all clusterings are inconsistent, i.e., there is no successful coloring of \( v \), we return false in Line 13, to backtrack and evaluate a different vertex.

**Example 4.5.** Consider an execution of Alg. 4 \( \text{Coloring} \) on the graph \( G \) in Figure 1, with vertices \( \{ v_1, v_2, v_3 \} \) representing constraints \( \{ \sigma_1, \sigma_2, \sigma_3 \} \), respectively. The candidate clusterings that satisfy each constraint (i.e., the output of the routine \( \text{Clusterings} \)) are shown beside each vertex. Consider vertex \( v_1 \) first (Line 2), and we select \( S_{\sigma_1} = \{ t_5, t_6 \} \), which is consistent with any other clustering. We then try to color vertices \( v_2 \) and \( v_3 \) by recursively calling \( \text{Coloring} \) in Line 11. If vertex \( v_2 \) is selected, the only clustering is \( S_{\sigma_2} = \{ t_5, t_6 \} \) that is consistent with \( S_{\sigma_1} \). Considering the last vertex \( v_3 \), we iterate over the clusterings for \( \sigma_3 \), and determine that the only consistent clustering (w.r.t. \( S_{\sigma_1} \) and \( S_{\sigma_2} \)) is \( \{ t_5, t_6 \} \), which we assign to \( S_\sigma \). Since we have found a clustering satisfying all constraints (i.e., a coloring of all vertices), the \( \text{Coloring} \) routine returns true with \( V \) containing the vertices and their colors (i.e., clusterings). The calling routine \( \text{DiverseClustering} \) uses \( V \) to compute the final clustering as \( S_\Sigma = \{ t_5, t_6 \} \).

**Runtime Analysis.** \( \text{DIVA} \) runs in polynomial time w.r.t. \(|R|\) since \( \text{DiverseClustering} \), \( \text{Anonymize} \), and \( \text{Suppress} \) run in polynomial time. \( \text{DiverseClustering} \) and its recursive procedure \( \text{Coloring} \) run in polynomial time w.r.t. \(|R|\) since the number of candidate clusterings for each constraint is polynomial w.r.t. \(|R|\). In particular, the size of these clusters is in \([k, 2k - 1]\) and there are polynomially many clusters of each size. Note that there is no cluster of size \( \geq 2k \) because we can split them into clusters of size \( \geq k \).

**4.2 Selection Strategies**

In the naive version of \( \text{DIVA} \), we randomly select a constraint and a clustering to evaluate. These choices impact algorithm performance as poor initial selections can lead to increased backtracking operations downstream. We selectively order the constraints (vertices) and clusterings (colors) that most likely lead to a graph coloring while minimizing the need to backtrack. We start evaluating constraints (vertices) that are the most difficult to satisfy. By postponing these candidates, we may encounter fewer or no possible consistent clusterings as we assign clusterings to less restrictive constraints. We apply this intuition to propose the following two strategies.

**DIVA-MinChoice:** Our preference is to select constraints with the fewest candidate clusterings, as we start with the most restrictive
Table 6: Data characteristics.

| | Pantheon | Census | Credit | Population (Syn) |
|---|----------|--------|--------|------------------|
| $| R | 11,341 | 299,285 | 1000 | 100,000 |
| n | 17 | 40 | 20 | 7 |
| $| \Pi_Q(R) | 5,636 | 12,405 | 60 | 24,630 |
| $| \Sigma | 24 | 21 | 18 | 10 |

Constraints first, i.e., those with the fewest chances, ensuring that these constraints are first satisfied. In the routine NextVertex, we initially select a vertex $v$ with a minimum value $\max_{\sigma}(\text{Clusterings}(v, \text{constraint}, R))$. As we visit vertices and assign (colors) clusterings, we update the candidate clusterings for their neighbors.

**DIVA-MaxFanOut:** In this strategy, we target constraints that overlap with the highest number of other constraints. This is modeled in the graph $G$ as vertices with the maximum number of unvisited edges. We preferentially select these constraints due to their high number of interactions with other constraints, which lead to an increased number of target attributes, and bounds that the relevant tuples must satisfy. This heuristic strategy aims to satisfy “maximum overlap” constraints first, and perform early pruning of unsatisfiable clusterings to reduce the number of clustering evaluations downstream. The vertex selection in this strategy is similar to incidence degree ordering in graph coloring [9].

In both strategies, clusterings returns a list of clusterings in ascending order of the number of overlapping tuples. For instance, for a clustering $|S|$ and a neighboring vertex $v$ (constraint $\sigma$), overlapping tuples are in the target $I_t$ and in a cluster in $S$. In Section 5.4, we show these strategies improve runtime by an average 24%.

**Example 4.6.** In Fig. 1, the DIVA-MinChoice strategy first selects vertex $v_2 (\sigma_2)$, since $|\text{Clusterings}(\sigma_1, R)| = 4, |\text{Clusterings}(\sigma_2, R)| = 1, |\text{Clusterings}(\sigma_3, R)| = 12$. After assigning cluster $\{t_8, t_9\}$ to $v_2$, we update the clusterings, and vertices $v_1, v_3$ will each have 4 clusterings; we break ties randomly. In DIVA-MaxFanOut, we first select vertex $v_3 (\sigma_3)$ containing two unvisited edges. Clusterings then computes cluster $\{t_7, t_8\}$ has 2 overlapping tuples $t_8, t_{10}$ in $I_{t_7}$. Similarly, cluster $\{t_7, t_8\}$ has 1 overlapping tuple $t_8$ in $I_{t_7}$. Hence, clustering $\{t_7, t_8\}$ is ranked first assuming it wins the tie against clustering $\{t_7, t_{10}\}$. We randomly select between $v_1$ and $v_2$ given their equal number of unvisited edges.

5 EXPERIMENTS

Our evaluation has the following objectives: (1) We evaluate DIVA’s accuracy using three types of diversity constraints as we vary $k$, and the conflict rate among tuples. (2) We evaluate the accuracy and performance of all DIVA variants as we vary $k, |\Sigma|$, the conflict rate, and the target attribute(s) data distribution. (3) We compare against an existing $k$-anonymization baseline algorithm to evaluate the cost of introducing diversity constraints into data anonymization.

5.1 Experimental Setup

We implement DIVA using Python 3.6 on a server with 32 Core Intel Xeon 2.2 GHz processor with 32GB RAM. We describe the datasets, diversity constraints, and baseline comparative algorithm.

**Datasets.** We use three real data collections and one synthetic dataset. Table 6 gives the data characteristics, showing a range of data sizes w.r.t. the number of tuples ($|R|$), number of attributes $(n)$, number of unique values in the QI attributes ($|\Pi_Q(R)|$), and the total number of defined diversity constraints ($|\Sigma|$).

Pantheon [1]. This dataset describes individuals based on the popularity of their biographical page in Wikipedia. Attributes include name, sex, city, country, continent. We select sex, city, country and continent as QI attributes, and define diversity constraints on sex and continent, where the attribute domain is two and six, respectively. We use this dataset to evaluate algorithm accuracy.

Census [3]. The U.S. Census Bureau describes population data for 1970, 1980 and 1990. We select sex, workplace, marital status, family relationship, race, and native country as QI attributes. We define (single and multi-attribute) diversity constraints on the sex and race attribute domains with size two and five, respectively. We evaluate accuracy, runtime, and comparative performance with this dataset.

German Credit [3]. This dataset classifies persons as good or bad credit risk according to attributes such as credit history, credit amount, sex, job, housing, marital status, and stratified savings account balances. We select sex, job, housing, saving account as QI attributes, and define diversity constraints on sex and job containing two and four values, respectively. We comparatively evaluate against an existing $k$-anonymization baseline with this dataset.

**Synthetic Population Data (Pop-Syn).** We use the Synnerio tool to generate realistic synthetic data by declaratively specifying the desirable distribution properties in the target attributes [18]. We generate a synthetic dataset describing population characteristics (age, education, race, gender, income, marital status, occupation). We select a subset of these attributes as target attributes, and vary their statistical distributions (uniform, Gaussian, Zipfian) to study the impact on DIVA’s accuracy.

**Diversity Constraints.** We implement different notions of diversity such as minimum frequency, average and proportional representation from the attribute domain. We use the diversity definitions presented by Stoyanovich et al. that define three classes of diversity constraints as described below [23]. We generate a set of satisfiable diversity constraints $\Sigma_i$ for each class, $i = \{1, 2, 3\}$, for each dataset.

In the original definition, Stoyanovich et al. define these diversity constraint classes w.r.t. the number of selected elements from a set [23]. In our setting, we consider an equivalent notion as the number $U$ of published (non-suppressed) tuples in $R'$. To estimate $U$, recall the tuples in the QI attributes are suppressed to achieve the indistinguishability of a tuple among $(k - 1)$ other tuples in a cluster group. We can estimate $U$ by computing the cardinality of the QI attribute(s) domain, and subtracting this value from the size of $R$. Let $\Pi_Q(R)$ represent the projection of relation $R$ on the QI attributes, i.e., the set of unique tuples w.r.t. the QI attributes. These unique values will need to be suppressed among an average of $|\Pi_Q(R)| / \lambda$ groups to achieve $k$-anonymity. Hence, we estimate $U = |R| - |\Pi_Q(R)|$ as the number of tuples that are published (unsuppressed) tuples in $R'$. We now describe each class of diversity constraints. Let $d = |\text{dom}(A)|$, i.e., the number of unique values in the target attribute(s) $A$ domain. The full set of diversity constraints, datasets and our code are available at [2].

- **Minimum:** Cover as many values in the attribute(s) domain as possible. If $U > d$, set $\lambda = \lambda_r = 1$ for all $d$ (value) constraints.
Then, compute \( w = U - d \). If \( w > 0 \), then assign these values to a random constraint \( \sigma_j^\prime \) by setting its \( \lambda_r^j = \lambda_r^j + w \). Select \( \sigma_j^\prime \) randomly where \( \text{freq}(a) \geq \lambda_r^j + w \).

If \( U < d \), set \( \lambda_l = \lambda_r = 1 \) to a random set of \( U \) out of \( d \) constraints, and set \( \lambda_l = \lambda_r = 0 \) to the remaining \( d - U \) constraints.

- **Average:** Select equal numbers for each value in the attribute domain. If \( U \geq d \), set \( \lambda_l = \min([U/d], \text{freq}(a)) \), \( \lambda_r = \min([U/d], \text{freq}(a)) \), where \( \text{freq}(a) \) represents the frequency of value(s) \( a \) in attribute(s) \( A \) in \( R \). Next, compute \( w = \sum_{i=1}^{d} \lambda_r^j \). If \( w < U \), then assign these values to a random constraint \( \sigma_j^\prime \) by setting its \( \lambda_r^j = \lambda_r^j + w \). Select \( \sigma_j^\prime \) randomly where \( \text{freq}(a) \geq \lambda_r^j + w \). If \( U < d \), define in **minimum** class.

- **Proportion:** Select equal proportions for each value in \( \text{dom}(A) \). If \( U \geq d \), set \( \lambda_l = \left[ \frac{U * \text{freq}(a) \mod |R|}{} \right], \lambda_r = \left[ \frac{U * \text{freq}(a) \mod |R|}{} \right] \). If \( U < d \), set constraints as in **minimum** class above.

**Comparative Baseline.** As far as we know, DIVA is the first work to couple diversity and privacy-preserving anonymization. The closest comparative baseline is the \( k \)-member anonymization algorithm takes a greedy, clustering-based approach to group similar records by minimizing the distance between values and between records [7]. \( k \)-member aims to minimize distortion among the values, and minimize the information loss in the anonymized relation. Although \( k \)-member considers both generalization and suppression, we only apply suppression in our comparative evaluation.

### 5.2 Metrics and Parameters

**Metrics.** We compute the average runtime over five executions. To quantify accuracy, we use an intuitive measure to model desirable anonymizations that minimize a cost function. Existing anonymization algorithms use cost functions that minimize information loss from suppression [14]. The resulting anonymized relation \( R' \) can be considered as imposing a penalty on each tuple that reflects its information loss due to suppression. The **discernibility metric**, \( \text{disc}(R', k) \), quantifies the differentiation between tuples for a given \( k \) value, by assigning a penalty to each tuple based on the number of tuples that are indistinguishable from it in \( R' \) [5]. If an unsuppressed tuple lies in a cluster of size \( j \), it is assigned a penalty of \( j \). If a tuple is suppressed, it is assigned a penalty of \( 1 / j \) since the tuple cannot be differentiated from other tuples in \( R' \) [5]. We define the normalized discernibility score as \( \frac{\text{disc}(R', k)}{|R'|} \). To quantify accuracy, we compare \( \text{disc}(R', k) \) for \( R' \) computed by DIVA against \( \text{disc}(R'', k) \) for the best \( R'' \) computed by sampling among all the possible clusters and selecting the best clustering. We compute accuracy as the ratio of the normalized discernibility scores \( \frac{\text{disc}(R', k)}{|R'|} \) to quantify the penalty to enforce diversity in \( R' \).

**Parameters.** Unless otherwise stated, Table 7 shows the range of parameter values we use, with default values in bold. We measure the **conflict rate** between the diversity constraints by measuring the number of overlapping relevant tuples between a pair of diversity constraints. We use Jaccard similarity to quantify the similarity between two sets, computed as the size of the intersection divided by the size of the union of the sets. Similarly, we define the conflict rate \( c_f(\sigma_i, \sigma_j) = \frac{|I_{\sigma_i \cap I_{\sigma_j}}|}{|I_{\sigma_i} \cup I_{\sigma_j}|} \) between constraints \( \sigma_i, \sigma_j \), and \( I_{\sigma_i} \) refers to the relevant tuples of \( \sigma_i \). For all \( \sigma_i \in \Sigma \), we compute \( c_f(\Sigma) = \frac{1}{|\Sigma|} \sum_{\sigma_i, \sigma_j} c_f(\sigma_i, \sigma_j) \), i.e., the average of all conflict scores for every pair of diversity constraints. Values of \( c_f(\Sigma) \) range from \([0, 1]\), where 0 indicates no overlapping relevant tuples, and 1 indicates full overlap (exact similarity) of the relevant tuples among the constraints.

### 5.3 Accuracy

We evaluate accuracy using three classes of constraints, and then vary \( |\Sigma|, c_f \), and the data distribution in the target attribute values.

**Exp-1: Vary \( |\Sigma| \) and \( k \).** Figure 2a gives the DIVA accuracy for the three variations of DIVA as we vary \( k \) using the Census dataset, across the three diversity constraint classes. Accuracy increases for larger \( k \) values as more values are suppressed to achieve anonymization.

**Exp-2: Vary \( |\Sigma| \) and Conflict Rate.** Figure 2b shows the DIVA accuracy as we vary the conflict rate \( c_f \) across the three constraint classes. Accuracy declines for increasing \( c_f \) as it is more difficult to find a clustering. Again, DIVA-Naive achieves the lowest accuracy, whereas DIVA-MaxFanOut performs best by first selecting clusterings that satisfy a maximal number of dependent constraints. In contrast, DIVA-MinChoice does not consider this constraint interaction. The proportion class of constraints achieves the best tradeoff between accuracy and adapting to the relative frequency of values in the data. Although the minimum class of constraints achieves higher accuracy in some cases (given the minimal lower bound values), this can lead to tokenization in \( R' \).

**Exp-3: Vary \( |\Sigma| \).** Figure 2c and Figure 2d show the DIVA accuracy as we vary the number of (proportion) constraints \( |\Sigma| \) using the Pantheon and Census dataset, respectively. DIVA-MaxFanOut outperforms DIVA-Naive and DIVA-MinChoice by +27\% and +9\%, respectively, (Pantheon), and +30\% and +7\% (Census). As \( |\Sigma| \) increases, we see accuracy decline but at a relatively slow linear rate. As a new constraint \( \sigma \notin \Sigma \) is added, we observe new relevant tuples w.r.t. \( \sigma \) join existing clusters of relevant tuples from \( \Sigma \) leading to a smaller decline in accuracy. This occurs with multi-attribute constraints that share target attributes with single attribute constraints.

**Exp-4: Vary Conflict Rate.** Figure 2e shows the DIVA accuracy as we vary the conflict rate \( c_f \). As expected, accuracy declines for increasing \( c_f \), with DIVA-MaxFanOut and DIVA-MinChoice
Diversifying Anonymized Data with Diversity Constraints

outperforming DIVA-Naive by +17% and +9%, respectively. DIVA-MaxFanOut shows improved accuracy over DIVA-MinChoice since targeting constraints with a high number of interactions (with other constraints) first allows it to eliminate unsatisfying clusterings sooner, while also satisfying dependent diversity constraints.

**Exp-5: Vary Data Distribution.** We generate target attribute values according to the Zipfian, uniform, and Gaussian distributions in the Pop-Syn dataset with $|\Sigma| = 100k$ and $|\cal{R}| = 8$. Figure 2f shows that DIVA-MaxFanOut performs best across all distributions by 8% and 17% over DIVA-MinChoice and DIVA-Naive, respectively. The target uniform distribution performs best as domain values are spread evenly across the tuples, avoiding contention among a small set of tuples. This conflict occurs more often in the Zipfian case than the Gaussian, leading to lower accuracy.

**5.4 Performance**

**Exp-6: Scale $|\Sigma|$.** Figure 2g shows the DIVA runtime as we vary the number of constraints over the Census dataset. As expected, DIVA-Naive shows exponential growth for increasing $|\Sigma|$ since we can assign $O(|\cal{R}|)$ different clusterings to each constraint. Our selection strategies to restrict clusterings and perform early pruning in DIVA-MinChoice and DIVA-MaxFanOut show linear scale-up with a 29% and 18%, respectively, reduction in runtime over the naive version.

**Exp-7: Vary Conflict Rate.** Figure 2h shows runtimes as we vary the conflict rate. DIVA-MinChoice outperforms DIVA-MaxFanOut and DIVA-Naive by 16% and 23%, respectively. We observe that when conflicts occurs among a set of tuples, leaving residual tuples that are unique and the only ones that can satisfy a constraint, e.g., vertex $v_2$ ($\sigma_2$) in Figure 1, DIVA-MinChoice performs well. By selecting these special constraints first (with fewer clustering choices), we reduce the number of clusterings to evaluate.

**5.5 Overhead of Diversity Constraints**

**Exp-8: Vary $k$.** Figure 3a and Figure 3b show the comparative discernibility scores and runtimes between DIVA and $k$-member [7]. DIVA-MinChoice and DIVA-MaxFanOut incur an average 32% and 44% higher runtime, respectively, than $k$-member, reflecting the cost of computing a diverse data instance. As $k$ increases, we expect more tuples to be suppressed leading to higher penalty costs, and higher $\hat{d}_{\text{disc}}(R', k)$ scores. For DIVA-MaxFanOut and DIVA-MinChoice, a 10% reduction in $\hat{d}_{\text{disc}}(R', k)$ costs 13m and 9m, respectively, whereas for $k$-member, the cost is 4m. We believe that the overhead and trade-off are still acceptable in practice since constraint validation and anonymization is often done offline. As next steps, we are exploring techniques to reduce the overhead via parallel processing of the Coloring routine on subgraphs of $G$.

**Exp-9: Vary $|\cal{R}|$.** Figure 3c shows that as $|\cal{R}|$ increases, $\hat{d}_{\text{disc}}(R', k)$ scores slightly improve as QI and target attribute values from the new tuples align with existing tuples, and do not incur additional suppression (penalty). In contrast, when new attribute values are suppressed to satisfy diversity constraints (at $|\cal{R}| = 240k$), we incur increased penalty costs. Figure 3d shows that DIVA runtimes

![Figure 2](image1.png)  
**Figure 2:** DIVA effectiveness and efficiency.

![Figure 3](image2.png)  
**Figure 3:** Comparative evaluation.
increase linearly w.r.t $|R|$ with an average overhead of 36% over the baseline, as new tuples and clusterings need to be evaluated.

6 RELATED WORK

Privacy Preserving Data Publishing. Extensions of $k$-anonymity include $l$-diversity, $t$-closeness, $(X,Y)$-privacy, and $(X,Y)$-anonymity with tighter privacy guarantees [14]. DIVA is extensible to re-define the clustering criteria according to these privacy semantics. Differential privacy (DP) provides a higher level of protection for individuals where the existence (or not) of a single record should not impact the outcome of any statistical analysis [12]. As next steps, we intend to study similar decision problems, and quantify the randomization to satisfy both DP and a set of diversity constraints. Cuenca et. al, study PPDP in linked data by formalizing the anonymization problem and its complexity for RDF graphs [15]. Hay et. al., present a data publishing algorithm that guarantee anonymity over social network data [16]. In generalization, data values are replaced with less specific, but semantically consistent values according to a generalization hierarchy [14]. While DIVA currently considers suppression (a special case of generalization), we are exploring distance metrics to include generalization in DIVA.

Fairness and Diversity. Achieving fair and equal treatment of groups and individuals is difficult in data-driven decision making [4]. Despite a strong need for algorithmic fairness and data diversity, such principles are rarely applied in practice [25]. Data sharing of private data has been studied along two primary lines. First, causality reasoning aims to recognize discrimination to achieve algorithmic transparency and fairness. Recent techniques have proposed influence measures to identify correlated attributes [10], statistical reasoning about discrimination [19], and reasoning between causality and fairness to generate bias-free, differentially private synthetic data [28]. Secondly, recent work has studied variants of DP to release synthetic data with similar statistical properties to the input data [6], publishing differentially private histograms [26], and studying the impact of differentially private algorithms on equitable resource allocation, especially for strict privacy-loss budgets [21]. Our work is complementary to these efforts, with a different goal; to publish diverse and anonymized versions of the original data with minimal information loss for applications where statistical summaries, synthetic data, and aggregate queries are inadequate. Recent work by Stoyanovich et al., study diversity in the set selection problem and introduce diversity constraints to guarantee representation for each category in the selected set [23, 27]. We build upon this work, and are the first to formalize diversity constraints and study their foundations. We propose algorithms to couple diversity with data anonymization, a problem not considered in existing work.

Diverse Clustering. Incorporating diversity into clustering has been limited to producing more diverse results. Nguyen et. al., start with an initial clustering and then generate additional clusterings that minimize error from the initial set [8]. Phillips et. al., argue that there is limited success by being too reliant on the initial clustering, and propose a sampling approach to select a diverse, large sample of non-redundant clusters while maximizing a quality metric [29]. The only work we are aware of that combines clustering with anonymization is by Li et. al., that study a 2-approximation algorithm for $l$-diversity, an extension of $k$-anonymity, where each cluster is of size at least $l$, and each point is a different color (i.e., sensitive value) [17]. However, while our work shares a similar spirit, Li et. al., show that a solution may not be possible depending on the color distribution, and record deletion may be necessary. DIVA does not consider tuple deletion, and we use graph coloring to model tuple overlap between constraints, focusing instead on a declarative specification of diversity that is realizable in practice.

7 CONCLUSION

We introduce DIVA, a DIVersity-driven Anonymization algorithm that computes a privatized data instance guaranteed to satisfy a set of diversity constraints. We studied the foundations of diversity constraints, and presented a sound and complete axiomatization. We showed that the $(k, \Sigma)$-anonymization decision problem is in \textsc{PTime}, presented a clustering-based algorithm, and proposed optimizations to improve performance. Our evaluation showed the performance benefits of the optimizations, and the overhead of enforcing diversity constraints over the baseline. As future work, we intend to study more expressive statistical-based diversity constraints, and privacy extensions beyond $k$-anonymity. We are also investigating a distributed version of the coloring algorithm in DiverseClustering for improved scalability.

REFERENCES

[1] 2014. Pantheon Dataset. (2014). https://pantheon.world/
[2] 2020. DIVA: Extended Evaluation Details. (2020). https://diva1234567.github.io/
[3] 2020. UCI Machine Learning Repository. (2020). https://archive.ics.uci.edu/ml/datasets/
[4] S. Barocas and A. Selbst. 2016. Big Data’s Disparate Impact. California Law Review 104, 671 (2016), 671–732.
[5] R. Bayardo and R. Agrawal. 2005. Data Privacy through Optimal K-Anonymization. In ICDE. 217âĂŞ228.
[6] V. Bindeschneider, R. Shokri, and C. Gunter. 2017. Plausible Deniability for Privacy-Preserving Data Synthesis. Proc. VLDB Endow. 10, 5 (2017), 481IAÂŠ492.
[7] J. Byun, A. Kamra, E. Bertino, and N. Li. 2007. Efficient k-anonymization using clustering techniques. In DASFAA. 188–200.
[8] R. Caruana, Elhawary, N. Nguyen, and C. Smith. 2006. Meta Clustering. Proc. ICDM, 107–118.
[9] T. Coleman and J. MorÁľ. 1983. Estimation of Sparse Jacobian Matrices and Graph Coloring Problems. SIAM J. Numer. Anal. 20, 1 (1983), 187–209.
[10] A. Datta, S. Sen, and Y. Zick. 2016. Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems. In 2016 IEEE Symposium on Security and Privacy. 598–617.
[11] M. Drosou, H. Jagadish, E. Pitsoura, and J. Stoyanovich. 2017. Diversity in Big Data: A Review. Big Data 5, 2 (2017), 73–84.
[12] C. Dwork. 2006. Differential Privacy. In ICALP. 1–12.
[13] W. Fan, F. Geerts, X. Jia, and T. KementiTsiudis. 2008. Conditional Functional Dependencies for Capturing Data Inconsistencies. TODS 33, 2 (2008).
[14] B. Fung, K. Wang, R. Chen, and F. Yu. 2010. Privacy-Preserving Data Publishing: A Survey of Recent Developments. ACM Comput. Surv. 42, 4, Article 14 (2010).
[15] B. Grau and E. Kostylev. 2016. Logical Foundations of Privacy-preserving Publishing of Linked Data. In AAAI. 943–949.
[16] M. Hay, G. Miklau, D. Jensen, D. Towsley, and C. Li. 2010. Resisting structural reidentification in anonymized social networks. VLDB Journal 19, 6 (2010), 797–823.
[17] J. Li, K. Yi, and Q. Zhang. 2010. Clustering with Diversity. In Automata, Languages and Programming. 188–200.
[18] M. Mannino and A. Abouzeid. 2019. Is This Real? Generating Synthetic Data That Looks Real. In UST. 549AÂŠ561.
[19] R. Nabi and I. Shpitser. 2018. Fair Inference on Outcomes. In AAAI. 1931–1940.
[20] J. Phillips, P. Raman, and S. Venkatasubramanian. 2011. Generating a Diverse Set of High-Quality Clusterings. CEUR Workshop Proceedings 772 (2011).
[21] D. Pujol, R. McKenna, S. Kuppam, M. Hay, A. Machanavajjhala, and G. Miklau. 2020. Fair Decision Making Using Privacy-Protected Data. In FAccT. 189AÂŠ199.
[22] P. Samarati. 2001. Protecting respondents identities in microdata release. TKDE 13, 6 (2001), 1010–1027.

Diversifying Anonymized Data with Diversity Constraints

[23] J. Stoyanovich, K. Yang, and H. Jagadish. 2018. Online set selection with fairness and diversity constraints. In *EDBT*. 241–252.

[24] L. Sweeney. 2002. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 10, 05 (2002), 537–570.

[25] L. Sweeney. 2013. Discrimination in Online Ad Delivery. *Commun. ACM* 56 (2013).

[26] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett. 2013. Differentially Private Histogram Publication. *The VLDB Journal* 22, 6 (2013), 797–822.

[27] K. Yang and J. Stoyanovich. 2017. Measuring Fairness in Ranked Outputs. In *SSDBM*. 22:1–22:6.

[28] M. Young, L. Rodriguez, E. Keller, F. Sun, B. Sa, J. Whittington, and B. Howe. 2019. Beyond Open vs. Closed: Balancing Individual Privacy and Public Accountability in Data Sharing. In *FAccT*. 191–200.