# Pattern-Guided Data Anonymization and Clustering

Robert Bredereck[1,⋆], André Nichterlein[1], Rolf Niedermeier[1], Geevarghese Philip[2]

[1]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
[2]The Institute of Mathematical Sciences, Chennai, India
{robert.bredereck,andre.nichterlein,rolf.niedermeier}@tu-berlin.de
gphilip@imsc.res.in

**Abstract.** A matrix $M$ over a fixed alphabet is $k$-anonymous if every row in $M$ has at least $k-1$ identical copies in $M$. Making a matrix $k$-anonymous by replacing a minimum number of entries with an additional $\star$-symbol (called "suppressing entries") is known to be NP-hard. This task arises in the context of privacy-preserving publishing. We propose and analyze the computational complexity of an enhanced anonymization model where the user of the $k$-anonymized data may additionally "guide" the selection of the candidate matrix entries to be suppressed. The basic idea is to express this by means of "pattern vectors" which are part of the input. This can also be interpreted as a sort of clustering process. It is motivated by the observation that the "value" of matrix entries may significantly differ, and losing one (by suppression) may be more harmful than losing the other, which again may very much depend on the intended use of the anonymized data. We show that already very basic special cases of our new model lead to NP-hard problems while others allow for (fixed-parameter) tractability results.

## 1 Introduction

The notion of $k$-anonymity is a basic concept in privacy-preserving data publishing [9]. An $n \times m$-matrix $M$—called "table" in database theory—over a fixed alphabet is called $k$-anonymous if for every row $r$ in $M$ there are at least $k-1$ further rows in $M$ that are identical with $r$. The intuitive idea motivating this notion is that if each row in $M$ contains data about a distinct person, and if $M$ is $k$-anonymous, then it is hard to identify the data row corresponding to some specific individual [14]. Clearly matrices are, in general, not $k$-anonymous for any $k \geq 2$. It is NP-hard to make a given matrix $k$-anonymous by suppressing a minimum number of entries [2,11], that is, by replacing a minimum number of matrix entries with the $\star$-symbol. However, in the classical scenario it remains unspecified whether certain entries are less harmful to suppress than

⋆ Supported by the DFG, research project PAWS, NI 369/10.

*To appear in the* Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, Warsaw, Poland, August 2011, Lecture Notes in Computer Science.* 

others.[1] Here, we present a simple combinatorial model that allows the user of the anonymized data to specify, as part of the input, which row entries (respectively, which combinations of row entries) may be suppressed in order to achieve $k$-anonymity. Studying the computational complexity, we identify both tractable and intractable cases of the underlying combinatorial problem which allows for user-specified "anonymization patterns".

Sweeney [13], who pioneered the notion of $k$-anonymity, pointed out that in the context of $k$-anonymization it is desirable to guide the process of entry suppression. We convert this idea into a formal model where the end-user of anonymized data specifies a number of pattern vectors from $\{\square, \star\}^m$, where $m$ is the number of columns of the underlying matrix. A pattern vector $v \in \{\square, \star\}^m$ is associated with a set of matrix rows fulfilling the following condition: If the $i^{\text{th}}$ pattern vector entry is a $\square$-symbol, then all rows associated with this pattern must have identical symbols at this position; they may differ in other positions. The corresponding minimization problem, which we refer to as Pattern Clustering, is to find a mapping of matrix rows to pattern vectors such that sanitizing[2] the rows according to their mapped pattern vectors makes the matrix $k$-anonymous with a minimum number of suppressions. Refer to Section 2 for the formal model and a simple example.

*Related Work.* Data anonymization is an active area of research with a considerable amount of published work. See, for example, the recent survey by Fung et al. [9]. Note that there are some weaknesses of the $k$-anonymity concept and it is well-known that it does not always assure privacy [7,9]. Typically, $k$-anonymity is most useful where there is a single release of data from a single publisher. However, $k$-anonymity provides a basic, clear, and easy to handle mathematical concept of privacy and related topics. Our research perhaps is most closely related to the recent work of Aggarwal et al. [1] who proposed a new model of data anonymization based on clustering. While developing several polynomial-time approximation algorithms, their modeling idea roughly is to cluster the matrix rows and then to publish the "cluster centers"; importantly, it is required that each cluster contains at least $k$ rows, which corresponds to the $k$-anonymity concept. The fundamental difference to our model is that we allow to prespecify cluster centers by the user of anonymized data whereas in Aggarwal et al.'s model the end-user of the anonymized data has no influence on selecting which entries to suppress. Indeed, our Pattern Clustering model may be interpreted as a form of clustering via anonymization whereas Aggarwal et al. perform anonymization via clustering.

[1] For instance, suppose that $M$ contains data about patients used in medical research, where each row corresponds to a patient and each column is an attribute of the patient. Then an attribute like blood pressure is—typically, but not always—more useful to preserve than, say, hair color.

[2] That is, suppressing all row positions where the corresponding pattern vector has a $\star$-symbol.

**Table 1.** The computational complexity of Pattern Clustering with respect to various parameters.

| $k$ | $m$ | $n$ | $\|\Sigma\|$ | $s$ | $t$ | $p$ |
|---|---|---|---|---|---|---|
| NP-hard for $k = 1$ ($\|\Sigma\| = 2$, $s = \infty$) | NP-hard for $m = 4$ | FPT | NP-hard for $\|\Sigma\| = 2$ ($k = 1$, $s = \infty$) | XP | FPT | XP |

*Our Results.* We formally define a simple model of user-specified data anonymization based on the concepts of $k$-anonymity and pattern vectors. The central combinatorial problem is called Pattern Clustering and it is shown to be NP-hard for every $k \geq 1$ and matrix alphabet size $|\Sigma| = 2$. It is also shown to be NP-hard for matrices containing only four columns. In contrast, Pattern Clustering is fixed-parameter tractable (FPT) for the parameters $n$ (the number of matrix rows) and $t$ (the number of different matrix rows). Moreover, it can be solved in polynomial time for a constant number $p$ of given pattern vectors (in other words, Pattern Clustering is in the parameterized complexity class XP for the parameter $p$). Membership in XP also holds for the parameter number $s$ of suppressions. See Table 1 for a list of our results with respect to single parameterizations. Clearly, several of our findings in Table 1 suggest investigations in the spirit of multivariate algorithmics [8,12], that is, the study of combined parameters. Here, the following results are known: Pattern Clustering is fixed-parameter tractable for the combined parameters $(m, |\Sigma|)$, and $(s, p)$ (due to upper bound arguments using $t$) whereas the parameterized complexity status is open for the combined parameters $(m, p)$ and $(m, k)$.

Due to the lack of space some proofs and some further details are deferred to a full version of this paper.

## 2 Preliminaries and Basic Model

As mentioned in the introductory section, the main motivation for our new model is—in contrast to standard $k$-anonymization models—to let the end-user influence the data sanitization process by selecting—to some extent—how matrix entries may be suppressed. We now formally define a model that captures this intuitive notion. To this end, it is helpful to interpret a matrix simply as a multiset of rows, as we do in the next definition.

**Definition 1.** *Let $M \in \Sigma^{n \times m}$ be a matrix over the finite alphabet $\Sigma$. Then $\mathrm{R}(M)$ is the multiset of all the rows in $M$.*

The heart of our pattern-guided anonymization model lies in a function that "consistently" maps input matrix rows to some given pattern vectors. This is described in the following definition, where we use $v[i]$ and $x[i]$ to refer to the $i^{\text{th}}$ vector and row entry, respectively.

**Definition 2.** *Let $\Sigma$ be a finite alphabet and let $M \in \Sigma^{n \times m}$ and $P \in \{\square, \star\}^{p \times m}$ be two matrices. A function $\varphi : \mathrm{R}(M) \to \mathrm{R}(P)$ is* consistent *if for all $x, y \in \mathrm{R}(M)$ with $v := \varphi(x) = \varphi(y)$, and for all $1 \leq i \leq m$: $v[i] = \square \Rightarrow x[i] = y[i]$.*

Our cost measure that shall be minimized is the number of suppressed matrix entries. First, we define the cost of a pattern vector in the natural way.

**Definition 3.** *The* cost $c(v)$ *of vector* $v \in \{\square, \star\}^m$ *is the number of its* $\star$*-symbols.*

We now define the cost of a mapping.

**Definition 4.** *Let* $M \in \Sigma^{n \times m}$ *and* $P \in \{\square, \star\}^{p \times m}$ *be two matrices and let* $\varphi : \mathrm{R}(M) \to \mathrm{R}(P)$ *be a mapping from the rows of* $M$ *to the pattern vectors of* $P$*. Let* $\#(v) := |\{x : x \in \mathrm{R}(M) \wedge \varphi(x) = v\}|$*. Then, the* cost *of* $\varphi$ *is* $\sum_{v \in P} c(v) \cdot \#(v)$*.*

Next we define the concept of $k$-anonymity for functions.

**Definition 5.** *A function* $f : A \to B$ *is* $k$-anonymous *if for each* $a \in A$ *with* $f(a) = b$ *it holds that* $|\{x : x \in A \wedge f(x) = b\}| \geq k$*.*

Finally we are ready to define the central computational problem (formulated in its decision version) of this work.

Pattern Clustering
*Input:* A matrix $M \in \Sigma^{n \times m}$, a "pattern mask" $P \in \{\square, \star\}^{p \times m}$, and two positive integers $s$ and $k$.
*Question:* Is there a consistent and $k$-anonymous function $\varphi$ mapping the rows of $M$ to the pattern vectors of $P$, with cost at most $s$?

Figure 1 provides a simple example to illustrate and further motivate our above definition of Pattern Clustering.

We use the following notation in the rest of the paper. The mapping $\varphi$ (see Definition 4) plays a central role in the definition of Pattern Clustering. We often talk about it implicitly when saying that a row is mapped to a pattern vector. Moreover, we speak about assigning a $\square$-symbol of a pattern vector $v$ to a symbol $x$ which means that every row mapped to $v$ has an $x$ at the position of the $\square$-symbol.

## 3 Intractability Results

In this section, we show that Pattern Clustering is NP-complete even in very restricted cases. The membership in NP is easy to see: Guessing a mapping $\varphi$ of the rows from $M$ to pattern vectors from $P$, it is easy to verify in polynomial time that $\varphi$ is consistent, $k$-anonymous, and has cost at most $s$. In the following, we provide two polynomial-time many-one reductions: One from CNF-Satisfiability to show NP-hardness for the unweighted variant (that is, $s = \infty$) with $k = 1$ and $|\Sigma| = 2$; and a second reduction from Set Cover to show NP-hardness for $m = 4$.

Before doing the reductions we show how to get rid of big alphabets. The structural properties of Pattern Clustering allow us to replace any alphabet with a binary alphabet, by encoding the alphabet in binary.[3]

[3] As consequence of the binarization, the question whether Pattern Clustering is fixed-parameter tractable with respect to the combined parameter $(p, |\Sigma|)$ is equivalent to the question whether Pattern Clustering is fixed-parameter tractable with respect to $p$ alone.

| ~~name~~ | hair color | disease | age |
|---|---|---|---|
| ~~Alice~~ | blond | asthma | 40-60 |
| ~~Bob~~ | blond | asthma | 40-60 |
| ~~Clara~~ | blond | laziness | 20-30 |
| ~~Doris~~ | brown | laziness | 20-30 |
| ~~Emil~~ | blond | asthma | 20-30 |
| ~~Frank~~ | brown | laziness | 20-30 |
| ~~George~~ | blond | asthma | 40-60 |

standard $k$-anonymity, $k = 2$, minimize $s$:

| | | | |
|---|---|---|---|
| blond | asthma | 40-60 | Alice, Bob, George |
| blond | $\star$ | 20-30 | Emil, Clara |
| brown | laziness | 20-30 | Doris, Frank |

Pattern Clustering, $k = 2$, minimize $s$:

| | | | | |
|---|---|---|---|---|
| $(\star, \square, \square)$ | $\star$ | asthma | 40-60 | Alice, Bob |
| $(\star, \square, \square)$ | $\star$ | laziness | 20-30 | Clara, Doris, Frank |
| $(\star, \square, \star)$ | $\star$ | asthma | $\star$ | George, Emil |

**Fig. 1.** Comparing Pattern Clustering with standard $k$-anonymization. Consider an extract from a medical database (table on the left). For privacy reasons, one only wants to publish $k$-anonymous data for $k \geq 2$. Clearly, the first step is to remove the identifier column "name". Using standard $k$-anonymity, one may end up with the database on the top right. Indeed, this is the 2-anonymous database with the fewest suppressions, requiring only two suppressions. Unfortunately, researchers cannot use it to deduce a correlation between age and disease, because the disease of Emil and Clara is suppressed. In our new model, researchers may specify their particular interests by providing the three shown pattern vectors—in this example they specify that they do not care about hair color. Thus, using Pattern Clustering one may get the database down right. It is one of the databases with fewest suppressions which, at the same time, "respects the interests of the scientists". In contrast to the result for classical $k$-anonymity, the scientists can see that only young people have the disease "laziness".

**Lemma 1.** *Let $I = (M, P, s, k)$ be an instance of* Pattern Clustering *with $M$ being a matrix over the alphabet $\Sigma$. Then there is an equivalent instance $I' = (M', P', s', k)$ such that $M'$ is a matrix over a binary alphabet $\Sigma'$ and the number of columns $m'$ of $M'$ is $\lceil \log(|\Sigma|) \rceil$ times the number of columns $m$ of $M$.*

*Proof.* Given the instance $I = (M, P, s, k)$, construct $I' = (M', P', s', k)$ as follows. Assign to each symbol in $\Sigma$ a unique integer from $\{0, 1, \ldots, |\Sigma| - 1\}$. Each column will be replaced with $\lceil \log(|\Sigma|) \rceil$ columns. The corresponding columns are used to binary encode (filling up with zeros on the left) the identifier of the original symbol. The pattern vectors are extended analogously: Each $\star$- (respectively $\square$-) symbol is replaced by $\lceil \log(|\Sigma|) \rceil$ many consecutive $\star$- (respectively $\square$-) symbols. The new cost-bound $s'$ is $\lceil \log(|\Sigma|) \rceil$ times the old cost-bound $s$. It is not hard to see that the new instance is equivalent to the original one. $\square$

In both reductions to follow we need unique entries in the input matrix $M$. For ease of notation we introduce the $\triangle$-symbol with an unusual semantics. Each occurrence of a $\triangle$-symbol stands for a *different* unique symbol in the alphabet $\Sigma$. One could informally state this as "$\triangle \neq \triangle$".

Now we present our first NP-hardness result.

**Theorem 1.** Pattern Clustering *is NP-complete, even if $k = 1$, $|\Sigma| = 2$, and $s = \infty$.*

*Proof (Sketch).* We provide a polynomial-time many-to-one reduction from the NP-complete CNF-Satisfiability. In the following, let $(X, C)$ be a CNF-Satisfiability instance with $X := \{x_1, \ldots, x_n\}$ being the set of variables and

matrix:

| | | | |
|---|---|---|---|
| 1 | 1 | △ | $(x_1 \vee x_2)$ |
| 0 | 1 | △ | $(\neg x_1 \vee x_2)$ |
| △ | 0 | 0 | $(\neg x_2 \vee \neg x_3)$ |
| 1 | 0 | 1 | $(x_1 \vee \neg x_2 \vee x_3)$ |
| 1 | 0 | 0 | $(x_1 \vee \neg x_2 \vee \neg x_3)$ |
| 0 | 0 | 0 | $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ |

pattern vectors:
$(\square, \star, \star)$
$(\star, \square, \star)$
$(\star, \star, \square)$

**Fig. 2.** Example for the reduction from CNF-Satisfiability to Pattern Clustering. Given is the following formula: $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Applying the reduction, we get a Pattern Clustering instance as illustrated. Note that each row in the table represents $n+1$ rows differing only in the $\triangle$-positions. It is quite easy to see that every solution, e.g. mapping $\boxed{1}\boxed{0}\boxed{1}$ and $\boxed{1}\boxed{0}\boxed{0}$ to $(1, \star, \star)$, $\boxed{1}\boxed{1}\boxed{\triangle}$ and $\boxed{0}\boxed{1}\boxed{\triangle}$ to $(\star, 1, \star)$, and $\boxed{\triangle}\boxed{0}\boxed{0}$ and $\boxed{0}\boxed{0}\boxed{0}$ to $(\star, \star, 0)$, corresponds to the satisfying assignment $x_1 = 1, x_2 = 1, x_3 = 0$.

$C := \{c_1, \ldots, c_m\}$ being the set of clauses. We construct an equivalent Pattern Clustering instance $(M, P, s, k)$ with $k = 1$ and $s = \infty$ as described in the following (see Figure 2 for an example).

The columns of the input matrix $M$ correspond to the variables. Blocks of rows of $M$ correspond to the clauses: For each clause we have $n+1$ rows. The alphabet $\Sigma$ contains 1, 0, and the unique symbols $\triangle$. Summarizing, $M$ is an $m(n+1) \times n$ matrix over an alphabet with at most $m \cdot (n+1) \cdot n$ symbols.

In the following we describe the rows of $M$. Recall that there are $n+1$ rows for each clause. Let $e_{i,j}[z]$ denote the entry in the $z^{\text{th}}$ column of the $j^{\text{th}}$ row for the $i^{\text{th}}$ clause.

- If $c_i$ contains variable $x_z$ as a positive literal, then $\forall 1 \leq j \leq n+1 : e_{i,j}[z] = 1$.
- If $c_i$ contains variable $x_z$ as a negative literal, then $\forall 1 \leq j \leq n+1 : e_{i,j}[z] = 0$.
- Otherwise, $\forall 1 \leq j \leq n+1 : e_{i,j}[z] = \triangle$.

For each variable $x_i \in X$ we have a pattern vector $v_i$ where all entries are $\star$-symbols aside from the entry in the $i^{\text{th}}$ position. The idea is that the assignment of the $\square$-symbol from the pattern vector $v_i$ corresponds to the assignment of the variable $x_i$.

The theorem now follows from Lemma 1. □

After proving NP-completeness for constant values of $k$ and $|\Sigma|$, we show that Pattern Clustering is intractable even for a constant number of columns.

**Theorem 2.** Pattern Clustering *is NP-complete, even for* $m = 4$.

*Proof (Sketch).* We show the hardness by giving a polynomial-time many-one reduction from the NP-hard Set Cover problem.

Set Cover
*Input:* A set family $\mathcal{F} = \{F_1, \ldots, F_{|\mathcal{F}|}\}$ over a universe $U = \{u_1, \ldots, u_{|U|}\}$, and a positive integer $h$.
*Question:* Is there a set cover $\mathcal{F}' \subseteq \mathcal{F}$ of size at most $h$ such that $\bigcup_{F \in \mathcal{F}'} F = U$?

We first describe the main idea of the reduction. Each element $u \in U$ is represented by $|\mathcal{F}| + 1$ rows in $M$. All these $|\mathcal{F}| + 1$ rows corresponding to one element $u$ can be mapped to one pattern vector $v \in P$. By setting $k$ to $|\mathcal{F}|$, we allow at most one of these rows to be mapped to another, cheaper pattern vector. The construction guarantees that if one or more rows are mapped to any of these cheaper pattern vectors, then the elements represented by these rows are contained together in at least one $F \in \mathcal{F}$. If $|U|$ rows (one row for each element $u \in U$) can be mapped to $h$ cheaper pattern vectors, then this assignment denotes a set cover of size $h$ in the given Set Cover-instance and vice versa.

Next, we describe the construction in detail. Given an instance $(\mathcal{F}, U, h)$ of Set Cover, we construct a Pattern Clustering-instance $(M, P, s, k)$ as follows.

For each element $u_i \in U$ add a set $R_i^U = \{r_{i,0}^U, \ldots, r_{i,|\mathcal{F}|}^U\}$ of $|\mathcal{F}| + 1$ rows to the input matrix $M$ and one pattern vector $v_i^U = (\star, \star, \square, \star)$ to the pattern mask $P$. We call $v_i^U$ an *expensive* pattern vector since its cost $c(v_i^U)$ is three. Set $r_{i,0}^U := \boxed{\triangle}\boxed{\triangle}\boxed{i}\boxed{\triangle}$. If $u_i \in F_j$, $1 \leq j \leq |\mathcal{F}|$, then set $r_{i,j}^U := \boxed{j}\boxed{j}\boxed{i}\boxed{\triangle}$, else set $r_{i,j}^U := \boxed{\triangle}\boxed{\triangle}\boxed{i}\boxed{\triangle}$. Further, let $R_i^F := \{r_{i,j}^U \mid u_i \in F_j\}$. All rows of $R_i^U$ coincide in the third column, and so they can all be mapped together to the pattern vector $v_i^U$. Intuitively, the first row $r_{i,0}^U$ in $R_i^U$ is a dummy row that has to get mapped to the expensive pattern vector $v_i^U$. By setting $k := |\mathcal{F}|$, we ensure that at least $|\mathcal{F}| - 1$ other rows of $R_i^U$ get mapped to the same expensive pattern vector. Hence, at most one row of $R_i^U$ can be mapped to a cheaper pattern vector. The construction ensures that this one row is an element of $R_i^F$.

Now we specify the cheaper pattern vectors: Add $h$ pattern vectors $v_1^{\mathcal{F}}, \ldots, v_h^{\mathcal{F}}$ of the form $(\square, \square, \star, \star)$. We call these pattern vectors *cheap* as they need one suppression less than the expensive pattern vectors. The idea is that each of these pattern vectors $v_i^{\mathcal{F}}$ corresponds to one set in a set cover. Note that there are $|U|$ rows of $R_1^U, \ldots, R_{|U|}^U$ that can be mapped to the $h$ cheap pattern vectors (one row of each $R_i^U$). Since all the rows mapped to a pattern vector $v_i^{\mathcal{F}}$ have to coincide in the first two columns, the only possible candidates are the rows belonging to the sets $R_i^F$. If there are $|U|$ rows fulfilling these requirements, then the $h$ different assigned numbers in the pattern vectors $v_1^{\mathcal{F}}, \ldots, v_h^{\mathcal{F}}$ denote the set cover $\mathcal{F}'$ in the Set Cover-instance.

To ensure that at least $k$ rows are mapped to each pattern vector from $\{v_1^{\mathcal{F}}, \ldots, v_h^{\mathcal{F}}\}$, add for each $F_i \in \mathcal{F}$ a set $R_i^{\mathcal{F}} = \{r_{i,1}^{\mathcal{F}}, \ldots, r_{i,|\mathcal{F}|+1}^{\mathcal{F}}\}$ of $|\mathcal{F}|+1$ rows with $r_{i,j}^{\mathcal{F}} := \boxed{i}\boxed{i}\boxed{X}\boxed{X}$ [4], $1 \leq j \leq |\mathcal{F}| + 1$, to the input matrix $M$. Since not all these rows can be mapped to some pattern vector in $\{v_1^{\mathcal{F}}, \ldots, v_h^{\mathcal{F}}\}$, add a pattern vector $v^X$ of the form $(\star, \star, \square, \square)$ to the pattern mask $P$. The rows $r_{i,j}^{\mathcal{F}}$ can be mapped to $v^X$ or to one of $v_1^{\mathcal{F}}, \ldots, v_h^{\mathcal{F}}$. Since all these pattern vectors are cheap, the only constraint on the mapping of these rows is that each pattern vector $v^X, v_1^{\mathcal{F}}, \ldots, v_h^{\mathcal{F}}$ contains at least $k$ rows. We finish the construction by setting $s := 2 \cdot |U| + 3 \cdot |U| \cdot |\mathcal{F}| + 2 \cdot (|\mathcal{F}| + 1) \cdot |\mathcal{F}|$. □

[4] Here, we use $X$ as a fixed symbol which is not used in $M$ elsewhere.

Dom et al. [6] showed that Set Cover parameterized by $(h, |U|)$ does not admit a problem kernel of size $(|U| + h)^{O(1)}$ unless an unexpected complexity-theoretic collapse occurs, namely the polynomial-time hierarchy collapses to the third level. Given a problem instance $I$ with parameter $x$, a *problem kernel* is an equivalent polynomial-time computable instance $I'$ with parameter $x' \leq x$ such that the size of $I'$ is upper-bounded by some function $g$ only depending on $x$ [3,10]; $g(x)$ is called the size of the problem kernel.

Bodlaender et al. [4] introduced a refined concept of parameterized reduction (called polynomial time and parameter transformation) that allows to transfer such hardness results to new problems. Indeed, the reduction above is such a parameterized reduction. The parameters $h$ and $|U|$ are transformed to $m$ and $p$ as follows: $m = 4$ and $p = h + |U| + 1$. Hence Pattern Clustering does not admit a problem kernel of size $(m + p)^{O(1)}$.

**Corollary 1.** Pattern Clustering *parameterized by* $(m, p)$ *has no problem kernel of polynomial size unless coNP* $\subseteq$ *NP/poly.*

## 4 Tractable Cases

In the previous section, we showed computational intractability for various special cases of Pattern Clustering. Here, we complement these hardness results by presenting some tractable cases. To this end, we consider several parameterizations of Pattern Clustering with respect to natural parameters and reasonable combinations thereof. Since Pattern Clustering allows the user to specify pattern vectors to influence the solution structure, the number of pattern vectors $p$ seems to be one of the most natural problem-specific parameters. It is quite reasonable to assume that there are instances with a small amount of pattern vectors, for instance, when the user wants a clustering with few but huge clusters. We start with a general observation on the solution structure of Pattern Clustering instances. To this end, we introduce the concept of row types. A *row type* is a string from $\Sigma^m$. We say that a set of rows in the matrix has a certain row type if they are identical. In this sense, the number $t$ of different matrix rows is the number of row types. The following lemma says that without loss of generality one may assume that at most $t$ many pattern vectors are used in the solution.

**Lemma 2.** *Let* $(M, P, k, s)$ *be a* YES*-instance of* Pattern Clustering*. If* $M$ *has* $t$ *row types, then there exists a mapping* $\varphi$*, which is solution for* $(M, P, k, s)$*, whose codomain contains at most* $t$ *elements.*

We now take up the question of whether the problem is still intractable (in form of NP-hardness) when $p$ is a constant.

*Parameters $p$ and $t$.* We describe a fixed-parameter algorithm for Pattern Clustering with respect to the combined parameter $(p, t)$. This algorithm can be interpreted as an XP-algorithm for Pattern Clustering parameterized by $p$, that is, it has polynomial running time for constant values of $p$.

**Theorem 3.** Pattern Clustering *can be solved in* $O(t^{\min(t,p)} \cdot 2^p \cdot (\min(t,p) \cdot t \cdot m + t^3 \log(t)) + n \cdot m)$ *time.*

*Proof.* As preprocessing we have to compute the input row types in $O(n \cdot m)$ time (by constructing a trie on the rows [5]). Our algorithm works in three phases:

1. For each pattern vector $v$, determine whether it is *used* in the solution, that is, whether $v$ occurs in the codomain of the mapping.
2. For each pattern vector $v$ that is used in the solution determine which input row types may contain rows that are mapped to $v$ in the solution by guessing a representative element. In the following we call these input row types *preimage types* of the pattern vector $v$.
3. For each pattern vector $v$ that is used in the solution determine how many rows from each preimage type are mapped to $v$ in the solution.

Due to Lemma 2, Phase 1 can be realized by branching on all $\sum_{i=1}^{\min(t,p)} \binom{p}{i} \leq 2^p$ possibilities.

Phase 2 is realized by guessing for each pattern vector a prototype, that is, a row that is mapped to this vector in the solution. Clearly, knowing one preimage of the mapping for each pattern vector is sufficient to compute which input row types may contain rows that are mapped to the vector. Guessing the prototypes and computing the preimage types can be done in $O(t^p \cdot m)$ time.

In Phase 3, we have the following situation. We have $t$ input row types and $p' \leq t$ pattern vectors that are used in the solution. In the following the set of input row types is represented by $T_{\text{in}} := \{1, \ldots, t\}$ whereas the set of pattern vectors is represented by $T_{\text{out}} := \{1, \ldots, p'\}$. For each pair consisting of an input row type $R$ and a pattern vector $v$ we already know whether rows from $R$ may be mapped to $v$ in the solution. Let $a : T_{\text{in}} \times T_{\text{out}} \to \{0, 1\}$ be a function expressing this information. Furthermore, let $\omega_i$ with $i \in T_{\text{out}}$ denote the cost of the $i^{\text{th}}$ pattern vector and let $n_j$ with $j \in T_{\text{in}}$ denote the number of rows in the $j^{\text{th}}$ input row type. A consistent and $k$-anonymous mapping that has cost at most $s$ and respects the preimage types (determined in Phase 2) corresponds to a solution of the Row Assignment [5] problem which is defined as follows.

Row Assignment

*Input:* Nonnegative integers $k$, $s$, $\omega_1, \ldots, \omega_{p'}$ and $n_1, \ldots, n_t$ with $\sum_{i=1}^{t} n_i = n$, and a function $a : T_{\text{in}} \times T_{\text{out}} \to \{0, 1\}$.

*Question:* Is there a function $g : T_{\text{in}} \times T_{\text{out}} \to \{0, \ldots, n\}$ such that

$$a(i,j) \cdot n \geq g(i,j) \qquad \forall i \in T_{\text{in}}, \forall j \in T_{\text{out}} \tag{1}$$

$$\sum_{i=1}^{t} g(i,j) \geq k \qquad \forall j \in T_{\text{out}} \tag{2}$$

$$\sum_{j=1}^{p'} g(i,j) = n_i \qquad \forall i \in T_{\text{in}} \tag{3}$$

$$\sum_{i=1}^{t} \sum_{j=1}^{p'} g(i,j) \cdot \omega_j \leq s \tag{4}$$

The mapping is represented by the function $g$. Inequality (1) ensures that for each pattern vector $v$ only rows from its preimage types are mapped to $v$. Inequality (2) ensures that the mapping is $k$-anonymous. Equation (3) ensures that each row is mapped to one pattern vector. Inequality (4) ensures that the costs of the mapping are at most $s$. Row Assignment can be solved in $O(t^3 \cdot \log(t))$ time [5, Lemma 2] and computing the function $a$ takes $O(\min(t, p) \cdot t \cdot m)$ time. □

We showed fixed-parameter tractability for Pattern Clustering with respect to the combined parameter $(t, p)$ by an algorithm with polynomial running time for constant values $p$. We leave it open whether there also exists an algorithm where the degree of the polynomial is independent of $p$, that is, whether Pattern Clustering is fixed-parameter tractable for parameter $p$. Next, we develop a fixed-parameter algorithm for the parameter $t$. This is mainly a classification result because its running time is impractical.

**Corollary 2.** Pattern Clustering *is fixed-parameter tractable with respect to the parameter $t$ as well as with respect to the parameter $n$.*

*Proof (Sketch).* When $p \leq t$, we use the algorithm from Theorem 3 without any modification. The corresponding running time is $O(t^t \cdot 2^t \cdot (p \cdot t \cdot m + t^3 \log(t)) + n \cdot m)$. For the other case we show a refined realization of Phase 1 of the algorithm from Theorem 3.

In Phase 1 we determine a set $P'$ of pattern vectors that are used in the solution, meaning that the codomain of the mapping function is $P'$. Due to Lemma 2 we know that w.l.o.g. $|P'| \leq t$. In Theorem 3 we simply tried all size-at-most-$t$ subsets of $P$. Here, we show that for guessing $P'$ we only have to take into account a relatively small subset $P^* \subseteq P$ with $|P^*| \leq g(t)$ with $g$ being a function which depends only on $t$.

Consider a pattern vector $v$ of the unknown $P'$. In Phase 2 of the algorithm, we determine the preimage types, that is, the set of input row types that may contain rows that are mapped to $v$ in the solution. Assume that the preimage types for all pattern vectors from $P'$ are fixed. To determine which concrete pattern vector corresponds to a set of preimage types, we only have to take into account the $t$ cheapest "compatible" pattern vectors, where compatible means that all rows of these preimage types coincide at the $\Box$-symbol positions. By definition, there exist at most $2^t$ many different sets of preimage types. Thus, keeping for each set of preimage types the $t$ cheapest pattern vectors and removing the rest results in a set $P^*$ of size $2^t \cdot t$.

Hence, when $p > t$, we realize Phase 1 by computing $P^*$ as described above and branch on all subsets $P' \subseteq P^*$ of size at most $t$. This can be done in $O\binom{2^t \cdot t}{t} \leq O(2^{t^2} t^t)$ time. As preprocessing we have to compute the input row types in $O(n \cdot m)$ time. Altogether, we can solve Pattern Clustering in $O(t^t \cdot (2^{t^2} t^t) \cdot (t^2 \cdot m + t^3 \log(t)) + n \cdot m)$ time. Clearly, since $t \leq n$, our result also holds for the parameter $n$. □

*Combined parameters.* As a corollary of Theorem 3 we show fixed-parameter tractability for some interesting combined parameters. All results rely on the fact that one can bound the number $t$ of input row types from above by a function only depending on the respective combined parameter.

**Corollary 3.** Pattern Clustering *is fixed-parameter tractable with respect to the combined parameters* $(|\Sigma|, m)$ *and* $(p, s)$.

*Proof.* As for the parameter $(|\Sigma|, m)$, $\Sigma^m$ is an upper bound for the number $t$ of input row types.

As for the parameter $(p, s)$: in the following, we call rows that are mapped to pattern vectors with at least one $\star$-symbol *costly rows* and their corresponding row types *costly row types*. Analogously, rows that are mapped to pattern vectors without $\star$-symbols are called *costless rows* and their row types *costless row types*. Clearly, every input row type is costly or costless (or both). There are at most $s$ costly rows and, hence, at most $s$ costly row types. Furthermore, the number of pattern vectors without $\star$-symbols is at most $p$. Since no two costless rows from different input row types can be mapped to the same pattern vector, the number of costless row types is also at most $p$. Hence, in a YES-instance the number $t$ of input row types is at most $s + p$. □

**Corollary 4.** Pattern Clustering *is in XP with respect to the parameter* $s$.

*Proof.* Using the definitions of costly and costless from Corollary 3 we give a simple algorithm that shows membership in XP. The first step is to guess, from $\sum_{i=0}^{s} \binom{n}{i}$ possibilities, the rows which are costly. The second step is to guess, from $\sum_{i=0}^{s} \binom{p}{i}$ possibilities, the pattern vectors (that contain $\star$-symbols) which are used in the solution. Then, guess the mapping between at most $s$ rows and at most $s$ pattern vectors and check whether it is consistent and $k$-anonymous. In the last step, the costless rows are greedily mapped to pattern vectors without $\star$-symbols. □

## 5 Conclusion

We initiated the investigation of a new variant of $k$-anonymity for ensuring "user-guided data privacy and clustering". The corresponding NP-hard combinatorial problem Pattern Clustering has a number of tractable and intractable special cases; our results are listed in Table 1 in the introduction. Several open questions remain. For example, the parameterized complexity of the problem for the parameters "number $s$ of suppressions" and the combined parameters $(m, p)$ and $(m, k)$, where $m$ is the number of columns and $p$ is the number of pattern vectors, are all open. A particularly interesting question is whether Pattern Clustering for parameter $p$ is fixed-parameter tractable or W[1]-hard. An equally interesting open question is whether Pattern Clustering becomes tractable for $m = 2$—note that we have shown it NP-hard for $m = 4$ and, based on some preliminary results, conjecture it to be NP-hard for $m = 3$. Finally,

it seems worth investigating Pattern Clustering also from the viewpoint of polynomial-time approximation.

Summarizing, we believe that Pattern Clustering is a well-motivated combinatorial problem relevant for data anonymization and data clustering; it deserves further investigation.

*Acknowledgements.* We are grateful to the anonymous referees of the *MFCS-2011* for helping to improve this work by spotting some flaws and providing the idea behind Corollary 4.

## References

1. G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. *ACM Trans. Algorithms*, 6(3):1–19, 2010.
2. J. Blocki and R. Williams. Resolving the complexity of some data privacy problems. In *Proc. 37th ICALP*, volume 6199 of *LNCS*, pages 393–404. Springer, 2010.
3. H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 17–37. Springer, 2009.
4. H. L. Bodlaender, S. Thomassé, and A. Yeo. Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels. Technical Report UU-CS-2008-030, Department of Information and Computing Sciences, Utrecht University, 2008.
5. R. Bredereck, A. Nichterlein, R. Niedermeier, and G. Philip. The effect of homogeneity on the complexity of $k$-anonymity. In *Proc. 18th FCT*, LNCS. Springer, 2011. To appear.
6. M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *Proc. 36th ICALP*, volume 5555 of *LNCS*, pages 378–389. Springer, 2009.
7. J. Domingo-Ferrer and V. Torra. A critique of $k$-anonymity and some of its enhancements. In *Proc. 3rd ARES*, pages 990–993. IEEE Computer Society, 2008.
8. M. R. Fellows. Towards fully multivariate algorithmics: Some new results and directions in parameter ecology. In *Proc. 20th IWOCA*, volume 5874 of *LNCS*, pages 2–10. Springer, 2009.
9. B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, 2010.
10. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.
11. A. Meyerson and R. Williams. On the complexity of optimal $k$-anonymity. In *Proc. 23rd PODS*, pages 223–228. ACM, 2004.
12. R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proc. 27th STACS*, volume 5 of *LIPIcs*, pages 17–32. IBFI Dagstuhl, 2010.
13. L. Sweeney. Achieving $k$-anonymity privacy protection using generalization and suppression. *IJUFKS*, 10(5):571–588, 2002.
14. L. Sweeney. $k$-anonymity: A model for protecting privacy. *IJUFKS*, 10(5):557–570, 2002.