Preprint. To appear in Data Mining and Knowledge Discovery.
DOI 10.1007/s10618-012-0293-7 – Online available.

The Effect of Homogeneity on the Computational Complexity of Combinatorial Data Anonymization

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

Received: 18 April 2012 / Accepted: 24 September 2012

Abstract A matrix M is said to be k-anonymous if for each row r in M there are at least $k-1$ other rows in M which are identical to r. The NP-hard k-Anonymity problem asks, given an $n \times m$-matrix M over a fixed alphabet and an integer $s > 0$, whether M can be made k-anonymous by suppressing (blanking out) at most s entries. Complementing previous work, we introduce two new "data-driven" parameterizations for k-Anonymity—the number t_{in} of different input rows and the number t_{out} of different output rows—both modeling aspects of data homogeneity. We show that k-Anonymity is fixed-parameter tractable for the parameter t_{in}, and that it is NP-hard even for $t_{\text{out}} = 2$ and alphabet size four. Notably, our fixed-parameter tractability result implies that k-Anonymity can be solved in *linear time* when t_{in} is a constant. Our computational hardness results also extend to the related privacy problems p-Sensitivity and ℓ-Diversity, while our fixed-parameter tractability results extend to p-Sensitivity and the usage of domain generalization hierarchies, where the entries are replaced by more general data instead of being completely suppressed.

Keywords k-Anonymity · p-Sensitivity · ℓ-Diversity · Domain generalization hierarchies · Matrix modification problems · Parameterized algorithmics · Fixed-parameter tractability · NP-hardness

An extended abstract entitled "The Effect of Homogeneity on the Complexity of k-Anonymity" appeared in *Proceedings of the 18th International Symposium on Fundamentals of Computation Theory (FCT '11)*, volume 6914 of *LNCS*, pages 53-64, Springer 2011. Apart from the full proofs omitted in that version, the current article also contains new results on ℓ-Diversity, p-Sensitivity, and on the usage of domain generalization hierarchies.

⋆Supported by the DFG, research project PAWS, NI 369/10.
†Supported by the Indo-German Max Planck Centre for Computer Science (IMPECS).

[1]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
[2]Max-Planck-Institut für Informatik, Saarbrücken, Germany
{robert.bredereck,andre.nichterlein,rolf.niedermeier}@tu-berlin.de
gphilip@mpi-inf.mpg.de

1 Introduction

Assume that data about n individuals are represented by length-m vectors consisting of attribute values. In other words, assume that our input is an $n \times m$ data matrix with entries from some (potentially large) alphabet Σ. If all vectors (that is, rows) are identical, then we have full homogeneity and thus full anonymity of all individuals. Relaxing full anonymity to k-anonymity (requiring that every vector occurs at least k times), in this work we investigate how the degree of (in)homogeneity (measured by the number of different vectors) both with respect to the input data matrix as well as the corresponding anonymized output data matrix influences the computational complexity of the NP-hard problem of making sets of individuals k-anonymous. Moreover, we extend our investigations to further combinatorial data anonymization problems including the concepts of p-sensitivity and ℓ-diversity.

Samarati and Sweeney (1998), Samarati (2001), and Sweeney (2002b) devised the notion of k-anonymity to better quantify the degree of anonymity in sanitized data. This notion formalizes the intuition that entities who have identical sets of attributes cannot be distinguished from one another. For a positive integer k we say that a matrix M is k-anonymous if, for each row r in M, there are at least $k-1$ other rows in M which are identical to r. Thus k-anonymity provides a clear and simple combinatorial model for sanitizing data: choose a value of k which would satisfy the relevant privacy requirements, and then try to modify—"at minimum cost"—the matrix in such a way that it becomes k-anonymous. The corresponding decision problem k-Anonymity asks, additionally given an upper bound s for the number of suppressions allowed, whether a matrix can be made k-anonymous by suppressing (blanking out) at most s entries. While k-Anonymity is our central data anonymization problem, our results also extend to several more restrictive anonymization problems,[1] including p-Sensitivity (Truta and Vinay, 2006) and ℓ-Diversity (Machanavajjhala et al, 2007).

Motivation. The increased use of technology brings with it various non-obvious threats to personal privacy. For example, location data from tracking devices—such as Global Positioning System (GPS) devices used in public transport systems—or communication devices—such as mobile phones—can be used by an adversary in space or time-correlated inference attacks to deduce private information about individuals. Another—seemingly unlikely—source of breach of privacy lies in the search logs of web search engines, which are sometimes released as a valuable aid for research on information retrieval. Yet other, perhaps more obvious, sources of private information are medical data or the "relationship graphs" of social networks. The protection of privacy is widely recognized as a human right, and at the same time there are large economic and scientific incentives for analyzing data sets which could potentially reveal private information. To be able to do this without harming privacy, k-

[1] See Section 2 for formal definitions.

Anonymity and its variants have been proposed as a possible approach to addressing privacy concerns arising in all these scenarios (Campan and Truta, 2009; Gedik and Liu, 2008; Gkoulalas-Divanis et al, 2010; Gruteser and Grunwald, 2003; Monreale et al, 2010; Navarro-Arribas et al, 2012; Zhou and Pei, 2011).[2]

We focus on a better understanding of the computational complexity and on tractable special cases of combinatorial data privacy problems; see Machanavajjhala et al (2007) and Sweeney (2002b) for discussions on the pros and cons of these models in terms of privacy vs preservation of meaningful data. In particular, note that with "differential privacy" (cleverly adding some random noise) a "statistical model" has become very popular as well (Dwork, 2011; Fung et al, 2010); we do not study this here. However, k-Anonymity and its variants are very natural *combinatorial* problems (with potential applications beyond data privacy) that are easy to interpret with respect to performed data modifications, and—for instance—in the case of "one-time anonymization", are obviously valuable for the sake of providing simple models that do not introduce noise.

Important Variants of k-Anonymity. While the k-Anonymity problem is described in terms of the operation of suppressing—blanking out—data entries, this level of obfuscation is sometimes not required or desirable. Sweeney (2002a) introduced the notion of *domain generalization* and *domain generalization hierarchies* (DGH) as a generalization of—and an alternative to—suppression for achieving k-anonymous datasets. The operation of generalization involves replacing a data entry with a less specific element. For instance, the precise age of a person might be replaced with an age range which includes the given age. The new value, while still being "correct" for the person, is not as precise as the previous one was. This idea is extended to the notion of domain generalization *hierarchies*, which can be thought of as increasingly more general ranges of data. The operation of suppressing a data element is thus a special case of generalization. The DGH-k-Anonymity problem asks for a *minimal*—under a certain natural ordering—generalization of the input matrix which is k-anonymous. Observe that the k-Anonymity problem is a special case of DGH-k-Anonymity where the only generalization maps each data element to the "$\star$"-symbol; thus DGH-k-Anonymity is at least as hard as k-Anonymity; we extend some of our positive results for k-Anonymity to DGH-k-Anonymity.

The attributes used in the well-known linking attack method by Sweeney (2002b) for re-identification were gender, ZIP code and birth date. She called such attributes containing publicly available data *quasi-identifiers*. Published datasets, such as medical data, consist of these quasi-identifiers and private information (e.g. disease). To protect against a linking attack, it is sufficient

[2] While it is beyond the scope of this work to fully address all the potential weaknesses of k-Anonymity, we mention for the sake of completeness that k-Anonymity is known to be vulnerable against the attack models "attribute linkage", "table linkage", and "probabilistic attack" (Fung et al, 2010).

to make the quasi-identifiers k-anonymous. Hence, the input matrix M for k-Anonymity contains only quasi-identifiers. This leads to a major drawback of k-Anonymity, namely the possibility that the private information may end up being highly uniform in the output. For example, an attacker cannot determine the row corresponding to one individual but using the linking attack method of Sweeney (2000) he may identify k rows from which exactly one corresponds to the individual. If the private information stored in these k rows is equal, then the attacker does not know the row corresponding to the individual but the private information belonging to the individual is revealed. To prevent this, different concepts were introduced, e.g. p-Sensitivity (Truta and Vinay, 2006), ℓ-Diversity (Machanavajjhala et al, 2007), and t-Closeness (Li et al, 2007). We will partially extend our findings for k-Anonymity to p-Sensitivity and ℓ-Diversity.

p-Sensitivity is an enhancement to k-Anonymity in the sense that there is the additional requirement that in the output matrix each maximal set of rows that are identical in the quasi-identifiers contains at least p different private values. ℓ-Diversity is an enhancement requiring that each such set of rows contains at least ℓ "well represented" private values. The combinatorial realization of "well represented" that we use is to require that in each such set of rows the relative frequency of each private value is at most $1/\ell$ (Wong et al, 2006; Xiao and Tao, 2006).

Computational Complexity. k-Anonymity and many related problems are NP-hard (Meyerson and Williams, 2004), even when the input matrix is highly restricted. Concerning polynomial-time approximability, k-Anonymity is APX-hard when $k = 3$, even when the alphabet size is just two (Bonizzoni et al, 2011a); it is MAX SNP-hard when $k = 7$, even when the number of columns in the input dataset is just three (Chakaravarthy et al, 2010); and it is MAX SNP-hard when $k = 3$, even when the number of columns in the input dataset is 27 (Blocki and Williams, 2010). Moreover, it is APX-hard when $k = 3$, even when the input matrix has only three columns (Bonizzoni et al, 2011b). On the positive side, many polynomial-time approximation algorithms have been developed for the problem and its variants. Meyerson and Williams (2004) devised an approximation algorithm with the approximation ratio $O(k \ln k)$ which runs in $O(n^{2k})$ time, and another one which runs in time polynomial in both n and k and has the ratio $O(k \ln n)$. Gionis and Tassa (2009) came up with an algorithm which runs in $O(n^{2k})$ time and has the approximation ratio $O(\ln k)$. This was later improved upon by Park and Shim (2007) and by Kenig and Tassa (2012), who devised approximation algorithms with the same ratio and the same worst-case running time, but which run much faster on large classes of inputs.

Parameterized Complexity. Confronted with this computational hardness, we study aspects of the parameterized (or multivariate) computational complexity of k-Anonymity and its variants as initiated by Evans et al (2009). The

central question here is how naturally occurring parameters influence computational complexity. For example, consider the following parameterization. Is k-Anonymity polynomial-time solvable for constant values of k? While 2-Anonymity is polynomial-time solvable (Blocki and Williams, 2010), the general answer is no since already 3-Anonymity is NP-hard (Meyerson and Williams, 2004), even on binary data sets (Bonizzoni et al, 2011a). Thus, k alone does not yield a promising parameterization.

k-Anonymity has a number of meaningful parameterizations beyond k, including the number of rows n, the alphabet size $|\Sigma|$, the number of columns m, and, in the spirit of multivariate algorithmics (Fellows, 2009; Niedermeier, 2010), various combinations of single parameters. Here the arity of the alphabet Σ may range from binary (such as gender) to unbounded (such as net worth). For instance, answering an open question of Evans et al (2009), Bonizzoni et al (2011b) showed that k-Anonymity is fixed-parameter tractable with respect to the combined parameter $(m, |\Sigma|)$, whereas there is no hope for fixed-parameter tractability with respect to the single parameters m and $|\Sigma|$ (Evans et al, 2009). We emphasize that Bonizzoni et al (2011b) made use of the fact that the value $|\Sigma|^m$ is an upper bound on the number of different input rows, thus implicitly exploiting a very rough upper bound on input homogeneity. In this work, we refine this view by asking how the "degree of homogeneity" of the input matrix influences the complexity of k-Anonymity. In other words, is k-Anonymity fixed-parameter tractable for the parameter "number of different input rows"? In a similar vein, we also study the effect of the degree of homogeneity of the *output* matrix on the complexity of k-Anonymity. Table 1, which extends similar tables due to Evans et al (2009) and Bonizzoni et al (2011b), summarizes known and new results for k-Anonymity.

Our Contributions. We introduce the "homogeneity parameters" the number t_{in} of different input rows and the number t_{out} of different output rows, for studying the computational complexity of k-Anonymity and related problems. Typically, we expect $t_{\text{in}} \ll n$ and $t_{\text{in}} \ll |\Sigma|^m$. The latter relation is "obvious" since $|\Sigma|^m$ just refers to having all possible input rows over the alphabet Σ. The relation $t_{\text{in}} \ll n$, however, may also be justified in natural settings. First, assume that the data are already k'-anonymous and shall be made k-anonymous with $k > k'$. This may be due stronger safety requirements. Then we have $t_{\text{in}} \leq n/k'$. Second, the parameter t_{in} can be viewed as a "distance from triviality-parameterization" in the sense that if t_{in} is small then the input is almost k-anonymous and hopefully needs not many modifications to be made k-anonymous. This may be particularly true when the matrix entries represent ranges (such as age range 40-50) and are not too fine-grained (such as specifying the exact birth day). Third, we study the adult data set (Frank and Asuncion, 2010): Preparing it as described in Machanavajjhala et al (2007), it consists of $n = 30162$ rows and $t_{\text{in}} = 18755$ input row types, that is, $t_{\text{in}} \approx 0.6 \cdot n$. Furthermore, the data set has at least one column with 72 different values, its alphabet size is $|\Sigma| = 72$, and it has eight columns. Hence, for this dataset we have $|\Sigma|^m = 72^8 \approx 7.2 \times 10^{14}$, and so $t_{\text{in}} \ll |\Sigma|^m$. Indeed,

Table 1 The parameterized complexity of k-Anonymity. FPT stands for "fixed-parameter tractability" and W[1]-hardness means presumable fixed-parameter *in*tractability, see Section 2 for definitions. Results proved in this paper are in **bold**. The column and row entries represent parameters. For instance, the entry in row "–" and column "s" refers to the (parameterized) complexity for the single parameter s whereas the entry in row "m" and column "s" refers to the (parameterized) complexity for the combined parameter (s, m). An entry "NP-hard" means that k-Anonymity remains NP-hard even if the parameter adopts constant values. In addition, "?" marks a currently unknown parameterized complexity status.

	–	k	s	k, s
–	NP-hard[a]	NP-hard[a]	W[1]-hard[b]	W[1]-hard[b]
$\|\Sigma\|$	NP-hard[c]	NP-hard[c]	?	?
m	NP-hard[d]	NP-hard[d]	FPT[e]	FPT[e]
n	FPT[e]	FPT[e]	FPT[e]	FPT[e]
$\|\Sigma\|, m$	FPT[b]	FPT[e]	FPT[e]	FPT[e]
$\|\Sigma\|, n$	FPT[e]	FPT[e]	FPT[e]	FPT[e]
t_{in}	**FPT**	**FPT**	**FPT**	**FPT**
t_{out}	**NP-hard**	?	**FPT**	**FPT**
$\|\Sigma\|, t_{\text{out}}$	**NP-hard**	?	**FPT**	**FPT**

parameters	
k	degree of anonymity
s	number of suppressions
$\|\Sigma\|$	alphabet size
m	number of columns
n	number of rows
t_{in}	number of different input rows
t_{out}	number of different output rows

[a] Meyerson and Williams (2004)
[b] Bonizzoni et al (2011b)
[c] Aggarwal et al (2005)
[d] Bonizzoni et al (2011a)
[e] Evans et al (2009)

t_{in} is a "data-driven parameterization" in the sense that one can efficiently measure in advance the instance-specific value of t_{in}, whereas $|\Sigma|^m$ denotes the maximum possible degree of inhomogeneity.

As to parameterization by output homogeneity t_{out}, we basically show that this leads to computational intractability even when $t_{\text{out}} = 2$ and the alphabet size is $t_{\text{out}}+2$, irrespective of the variant of t_{out}—being part of the input or not etc.—which we consider. This holds for k-Anonymity as well as for the more restrictive privacy concepts that lead to the problems p-Sensitivity (Truta and Vinay, 2006) and ℓ-Diversity (Machanavajjhala et al, 2007).

On the positive side we show that parameterization by input homogeneity t_{in} yields several tractability results. For instance, we derive an algorithm that solves k-Anonymity in $O(nm + 2^{t_{\text{in}}^2} t_{\text{in}}^2 (m + t_{\text{in}} \log(t_{\text{in}})))$ time, which compares favorably with the Bonizzoni et al (2011b) algorithm which runs in $O(2^{(|\Sigma|+1)^m} kmn^2)$ time. In particular, when t_{in} is a constant, our algorithm

solves k-Anonymity in time *linear* in the size of the input. Moreover, we prove that k-Anonymity becomes fixed-parameter tractable for the parameter m when t_{out} is a constant, and that k-Anonymity becomes fixed-parameter tractable for the combined parameter (t_{out}, s) where s denotes the number of allowed suppressions in the matrix. We also show that our positive results for k-Anonymity parameterized by t_{in} generalize to the problems p-Sensitivity, DGH-k-Anonymity, and DGH-p-Sensitivity, with similar time bounds. In Table 1 we summarize our results on k-Anonymity and present them in the context of previous results on the parameterized complexity of k-Anonymity.

Organization of the Paper. In the next section we introduce the notation and terminology which we use in the paper. In Section 3 we consider the homogeneity of the output matrix and demonstrate that achieving a very homogeneous output matrix (as might seem desirable for privacy purposes) is computationally very hard. This applies to all problems studied in our work. In Section 4, we show that for homogeneous input matrices one can gain several fixed-parameter tractability results for most problems considered in this article, leaving the case of ℓ-Diversity as a major open question. We conclude in Section 5 with a discussion of directions for future research.

2 Preliminaries

We briefly overview some relevant concepts from parameterized complexity which we use later. We then describe the notation and terminology used in the discussion on k-Anonymity, p-Sensitivity, and ℓ-Diversity. The definitions specific to DGH-k-Anonymity can be found in Subsection 4.4.

Parameterized Complexity. Our algorithmic results mostly rely on concepts of parameterized algorithmics (Downey and Fellows, 1999; Flum and Grohe, 2006; Niedermeier, 2006). The fundamental idea herein is, given a computationally hard problem X, to identify a parameter p (typically a positive integer or a tuple of positive integers) for X and to determine whether a size-n input instance of X can be solved in $f(p) \cdot n^{O(1)}$ time, where f is an arbitrary computable function. If this is the case, then one says that X is *fixed-parameter tractable* for the parameter p. The corresponding complexity class is called FPT. If X could only be solved in polynomial running time where the degree of the polynomial depends on p (such as $n^{O(p)}$), then, for parameter p, problem X is said to lie in the—strictly larger—parameterized complexity class XP. Finally, we also consider the parameterized complexity class W[1] with FPT $\subseteq$ W[1] $\subseteq$ XP. It is widely believed that a problem which is W[1]-hard—based on the so-called parameterized reductions (Downey and Fellows, 1999)—does not have FPT algorithms.

k-Anonymity. Our inputs are datasets in the form of $n \times m$-matrices, where the n rows refer to the individuals and the m columns correspond to attributes with entries drawn from an alphabet Σ. Suppressing an entry $M[i, j]$ of an $n \times m$-matrix M over alphabet Σ with $1 \leq i \leq n$ and $1 \leq j \leq m$ means to simply replace $M[i, j] \in \Sigma$ by the new symbol "$\star$" ending up with a matrix over the alphabet $\Sigma \uplus \{\star\}$.

Definition 1 A *row type* is a string from $(\Sigma \uplus \{\star\})^m$.

We say that a row in a matrix has a certain row type if it coincides in all its entries with the row type. In what follows, synonymously, we sometimes also speak of a row "lying in a row type", and that the row type "contains" these rows.

Definition 2 (k-anonymity (Samarati, 2001; Samarati and Sweeney, 1998; Sweeney, 2002b)) A matrix is *k-anonymous* if for every row in the matrix one can find at least $k - 1$ other identical rows.

A natural objective when trying to achieve k-anonymity is to minimize the number of suppressed matrix entries. For a row y (with some entries suppressed) in the output matrix, we call a row x in the input matrix the *preimage of y* if y is obtained from x by suppressing in x the $\star$-entry positions of y. The basic problem of this work reads as follows.

k-Anonymity

Input:	An $n \times m$-matrix M over a fixed alphabet Σ and nonnegative integers k, s.
Question:	Can one suppress at most s elements of M to obtain a k-anonymous matrix M'?

We study two new parameterizations t_{in} and t_{out}, referring to the input and the output homogeneity, respectively, where t_{in} denotes the number of row types in the input matrix and t_{out} denotes the number of row types in the output k-anonymous matrix. Note that—as we show later— t_{in} can be determined efficiently. In Section 3, we discuss several possibilities for a more fine-grained and mathematically precise definition of t_{out}.

p-Sensitivity and ℓ-Diversity. For the problems p-Sensitivity and ℓ-Diversity, the input matrix M consists of a matrix where the last column is declared as *private* and the other columns as *quasi-identifiers*.[3] The row types are defined over the submatrix restricted to the quasi-identifier columns (denoted by M_{qi}). That is, when two rows are identical in their quasi-identifier entries—quasi-identifiers for short—then they belong to the same row type, irrespective of whether they differ in their private columns. This definition fits to the anonymization concepts: The more homogeneous the quasi-identifiers are, the easier p-Sensitivity and ℓ-Diversity will be to solve. A similar

[3] Following an approach due to Xiao et al (2010) we consider the case with one private information column.

argument does not hold for the private information: Homogeneous private information does *not* make these two problems more efficiently solvable. Hence, row types are defined over the quasi-identifiers only.

Definition 3 (p-sensitive (Truta and Vinay, 2006)) A matrix is p-sensitive if for each row type there are p different values in the private column.

p-Sensitivity

Input: An $n \times m$-matrix M over a fixed alphabet Σ and nonnegative integers k, s, and p.

Question: Can one suppress at most s elements in M_{qi} to obtain a p-sensitive matrix M' where M'_{qi} is k-anonymous?

Definition 4 (ℓ-diverse (Wong et al, 2006; Xiao and Tao, 2006)) A matrix is ℓ-diverse if for each row type the relative frequency of each value in the private column is at most $1/\ell$.

Note that Definition 4 implies that in an ℓ-diverse matrix each row type is of size at least ℓ.

ℓ-Diversity

Input: An $n \times m$-matrix M over a fixed alphabet Σ and nonnegative integers ℓ and s.

Question: Can one suppress at most s elements in M_{qi} to obtain an ℓ-diverse matrix M'?

3 Hardness of Achieving Homogeneous Outputs

In Section 3.1, we discuss some basic concepts of output homogeneity. In Section 3.2, we demonstrate the computational intractability of achieving homogeneous outputs for k-Anonymity, ℓ-Diversity and p-Sensitivity.

3.1 Output Homogeneity

There are two cases to consider when investigating the homogeneity of the output matrix:

- The user specifies the required output homogeneity by providing the parameter t_{out} as part of the input.
- The output homogeneity is not specified by the user.

In the following, we briefly discuss both cases.

User-Specified Output Homogeneity. In this case, the user wants to specify the number of output row types. Here, we consider three variants:

- The minimum number of output row types is specified in the input.
- The maximum number of output row types is specified in the input.
- The exact number of output row types is specified in the input.

There is a close relationship between k-Anonymity and clustering problems where one is also interested in grouping together similar objects. Such a relationship has already been observed in related work (Aggarwal et al, 2010; Bredereck et al, 2011; Fard and Wang, 2010). This clustering view on k-Anonymity makes the three described problem variants very interesting because they allow the user to specified a minimum, maximum, or exact number of clusters which is a useful feature for many applications.

Output Homogeneity not Specified by the User. If the output homogeneity is not specified, then it is a property of the set of feasible solutions. Here, the two simplest properties yield the following two parameters:

- The minimum number $t_{\text{out}}^{\min}$ of output row types among all feasible solutions.
- The maximum number $t_{\text{out}}^{\max}$ of output row types among all feasible solutions.

The parameter $t_{\text{out}}^{\max}$ is the more interesting one from the point of view of anonymization. Given a fixed number s of allowed suppressions and a required degree k of anonymity, in this case the user of the data is interested in output matrices with as many output row types as possible. Due to the higher diversity more output row types potentially provide more information to the end-user of the data.

The parameter $t_{\text{out}}^{\min}$ is more interesting than $t_{\text{out}}^{\max}$ from an algorithmic perspective. Since $t_{\text{out}}^{\min} \leq t_{\text{out}}^{\max}$, fixed-parameter algorithms with $t_{\text{out}}^{\min}$ as the parameter instead of $t_{\text{out}}^{\max}$ are potentially more effective. It is not difficult to think of real-world examples where $t_{\text{out}}^{\min}$ is much smaller than $t_{\text{out}}^{\max}$.

3.2 Hardness Results

It is a natural question whether k-Anonymity is fixed-parameter tractable with respect to the number of output types, for each of the parameters discussed in Section 3.1. It is easy to see that the problem becomes polynomial-time solvable when there is only one output row type. Answering this question for more than one output row types in the negative, we show that k-Anonymity is NP-hard even when there are only two output types and the alphabet has size four, destroying any hope for fixed-parameter tractability already for the combined parameter "number of output types and alphabet size". This intractability also transfers to all the three variants of user-specified output homogeneity discussed above. The hardness proof uses a polynomial-time many-one reduction from Balanced Complete Bipartite Subgraph:

Balanced Complete Bipartite Subgraph (BCBS)

Input: A bipartite graph $G = (A \uplus B, E)$ and an integer $k \geq 1$.

Question: Is there a complete bipartite subgraph $G' = (A' \uplus B', E')$ of G with $A' \subseteq A$ and $B' \subseteq B$ such that $|A'| = |B'| \geq k$?

BCBS is NP-complete by a reduction from Clique (Johnson, 1987). We provide a polynomial-time many-one reduction from a special case of BCBS with a *balanced* input graph, that is, a bipartite graph that can be partitioned into two independent sets of the same size, and $k = |V|/4$. This special case is also NP-complete by a simple reduction from the general BCBS problem: Let $V := A \uplus B$ with A and B being the two vertex partition classes of the input graph. If the input graph is not balanced, that is, if $|A| - |B| \neq 0$, then add $||A| - |B||$ isolated vertices to the partition class of smaller size. If $k < |V|/4$, then repeat the following until $k = |V|/4$:

- Add a new vertex a to A and a new vertex b to B.
- Make a adjacent to all vertices in B, and b adjacent to all vertices in A.
- Increment k by 1.

If $k > |V|/4$, then add $2k - |V|/2$ isolated vertices to each of A and B.

We devise a reduction from BCBS with a balanced input graph and $k = |V|/4$ to show the NP-hardness of k-Anonymity. This reduction also introduces the basic structure of the constructions used in all the hardness proofs in this work.

Proposition 1 *k-Anonymity is NP-complete for two output row types and alphabet size four.*

Proof We only have to prove NP-hardness, since containment in NP is clear. Let $(G, n/4)$ be a BCBS-instance where $G = (V, E)$ is a balanced bipartite graph, and where $n := |V|$. Let $A = \{a_1, a_2, \ldots, a_{n/2}\}$ and $B = \{b_1, b_2, \ldots, b_{n/2}\}$ be the two vertex partition classes of G. We construct an $(n/4+1)$-Anonymity-instance that is a yes-instance if and only if $(G, n/4) \in$ BCBS. To this end, the main idea is to use a matrix expressing the adjacencies between A and B as the input matrix. Partition class A corresponds to the rows and partition class B corresponds to the columns. The salient points of our reduction are as follows.

1. By making a matrix with $2x$ rows x-anonymous we ensure that there are at most two output types. One of the types, the *solution type*, corresponds to the solution set of the original instance: Solution set vertices from A are represented by rows that are preimages of rows in the solution type and solution set vertices from B are represented by columns in the solution type that are not suppressed.
2. We add one row that contains the $\square$-symbol in each entry. Since the $\square$-symbol is not used in any other row, this enforces the other output type to be *fully suppressed*, that is, each column is suppressed.

	b_1	b_2	$\dots$	$b_{n/2-1}$	$b_{n/2}$
a_1	1	1	0_1	1	1
a_2	0_1	1	0_1	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{n/4}$	1	1	0_1	0_1	1
$a_{n/4+1}$	1	1	0_2	0_2	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{n/2}$	0_2	1	0_2	1	1
$\square$	$\square$	$\square$	$\square$	$\square$	$\square$
1	1	1	1	1	1

Fig. 1 Typical structure of the matrix D of the k-Anonymity instance obtained by the reduction from Balanced Complete Bipartite Subgraph for $t_{\text{out}} = 2$.

3. Since the rows in the solution type have to agree on the columns that are not suppressed, we have to ensure that they agree on adjacencies to model BCBS. This is done by using two different types of 0-symbols representing non-adjacency. The 1-symbol represents adjacency.

The matrix D is described in the following and illustrated in Figure 1. There is one row for each vertex in A and one column for each vertex in B. The value in the i^{th} column of the j^{th} row is 1 if a_j is adjacent to b_i and, otherwise, 0_1 if[4] $j \leq n/4$ and 0_2 if $j > n/4$. Additionally, there are two further rows, one containing only 1s and one containing only $\square$-symbols. The number of allowed suppressions is $s := (n/4+1) \cdot n/2 + (n/4+1) \cdot n/4$. This completes the construction.

It remains to show that $(G, n/4)$ is a yes-instance of BCBS if and only if the constructed matrix can be transformed into an $(n/4+1)$-anonymous matrix by suppressing at most s elements. We start with the easier direction:

"$\Rightarrow$": Let $A' \subset A$ and $B' \subset B$ be the vertex subsets forming a balanced complete bipartite subgraph of G such that $|A'| = |B'| = n/4$. The $(n/4+1)$-anonymous matrix looks as follows: The first output type is fully suppressed and contains the all-$\square$ row, and each row which corresponds to a vertex from $A \setminus A'$. The second output type contains the remaining rows; in it each column corresponding to a vertex in $B \setminus B'$ is suppressed. These rows clearly agree in each unsuppressed column, because each vertex from A' is adjacent to each vertex from B'. Hence, each unsuppressed column contains a 1. The fully suppressed output type needs $(n/4+1) \cdot n/2$ suppressions and the second output type needs at most $(n/4+1) \cdot n/4$ suppressions. Thus, the total number of suppressions is at most s.

"$\Leftarrow$": The $(n/4+1)$-anonymous matrix consists of exactly two output types because having only one type would cause $(n/2+2) \cdot n/2 > s$ suppressions (due to the all-$\square$ row everything would have to be suppressed) and an $(n/4+1)$-anonymous matrix with $n/2+2$ rows cannot have more than two output types. One output type contains $n/4+1$ fully suppressed rows because of the all-$\square$

[4] We assume without loss of generality that n is divisible by four.

	b_1	b_2	...	$b_{n/2-1}$	$b_{n/2}$	d_1	d_2	...	$d_{n/2}$
a_1	1	1	0_1	1	1	**1**	**1**	...	**1**
a_2	0_1	1	0_1	1	1	**1**	**1**	...	**1**
⋮	⋮	⋮	⋮	⋮	⋮	**1**	**1**	...	**1**
$a_{n/4}$	1	1	0_1	0_1	1	**1**	**1**	...	**1**
$a_{n/4+1}$	1	1	0_2	0_2	1	**1**	**1**	...	**1**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$a_{n/2}$	0_2	1	0_2	1	1	**1**	**1**	...	**1**
□	□	□	□	□	□	**1**	**1**	...	**1**
1	1	1	1	1	1	**1**	**1**	...	**1**
n/4 + 1 ×	**3**	**3**	**3**	**3**	**3**	**3**	**3**	...	**3**
n/4 + 1 ×	**4**	**4**	**4**	**4**	**4**	**4**	**4**	...	**4**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n/4 + 1 ×	**t**	**t**	**t**	**t**	**t**	**t**	**t**	...	**t**

Fig. 2 Structure of the matrix D of the k-Anonymity instance obtained by the extended reduction from Balanced Complete Bipartite Subgraph for general t_{out}. New parts of the reduction are in boldface.

row in the input. Since the total number s of suppressions is $(n/4+1) \cdot n/2 + (n/4+1) \cdot n/4$ and for the first type one already needs at least $(n/4+1) \cdot n/2$ suppressions, the other type contains at most $n/4$ suppressed columns. Since no symbol other than 1 appears in more than $n/4$ rows (by construction), each unsuppressed column consists entirely of 1s. Now, consider the vertex subset $A' \subset A$ corresponding to the rows of the second output type and the vertex subset $B' \subseteq B$ corresponding to the unsuppressed columns. Since at most one row in the second output type does not correspond to a vertex from A (the all-1 row), $|A'| \geq n/4$. Furthermore, $|B'| \geq n/4$ because at most $n/4$ of the $n/2$ columns can be suppressed in the second output type. Since the second output type only consists of $\star$- and 1-entries, the subgraph induced by A' and B' is a balanced complete bipartite subgraph of G. □

Observe that, for the matrix D constructed in the proof of Proposition 1, $t_{\text{out}}^{\min} = t_{\text{out}}^{\max} = 2$: If the matrix D is a yes-instance, then by construction it must have exactly two output types. It follows that k-Anonymity is NP-complete for the maximization of the number of output row types, for its minimization, as well as when asking for an exact number of output row types. In the following, we show that this hardness result holds for all fixed constants greater than or equal to two.

Theorem 1 *For any fixed constant $t_{\text{out}} \geq 2$, k-Anonymity is NP-complete, where $t_{\text{out}} = t_{\text{out}}^{\min} = t_{\text{out}}^{\max}$ and the alphabet size is $t_{\text{out}} + 2$, even if the exact, minimum, or maximum number of output row types is additionally specified in the input.*

Proof We will show that the reduction from Proposition 1 can be extended such that for any given constant $t_{\text{out}} \geq 2$, a solution must have exactly t_{out} output row types. An illustration of the reduction can be found in Figure 2.

As before, the reduction is from BCBS. First, we construct a k-ANONYMITY instance as described in Proposition 1. Then, we add $n/2$ dummy columns $(d_1, \ldots, d_{n/2})$ filled with 1-entries. Finally, for each integer $2 < i \leq t_{\text{out}}$, we add $n/4+1$ dummy rows filled with the new symbol i in each entry. As in the previous proof, the number of allowed suppressions is $s := (n/4+1) \cdot n/2 + (n/4+1) \cdot n/4$, and the degree of anonymity is $k := (n/4+1)$.

"$\Rightarrow$": Given a solution for BCBS, constructing the solution for the extended k-ANONYMITY instance works exactly like before. Each dummy row occurs $n/4+1$ times in the reduced instance, and so none of their entries need be suppressed. The resulting solution has exactly t_{out} output types : two output types as in the previous proof, and $t_{\text{out}} - 2$ other output types each of which consists of all the dummy rows which have the same symbol.

"$\Leftarrow$": Consider a solution for the k-ANONYMITY instance. If a dummy row and a row from another type have to be included in the same output type, then this requires suppressing all the entries of both these rows, at a cost of $(n/4+1) \cdot n$ suppressions. Since this is more than the allowed cost, it follows that in the output the dummy rows are left as they are. So the dummy rows form $t_{\text{out}} - 2$ output row types. By the same argument as in the proof of Proposition 1, it follows that there is a solution for the corresponding BCBS instance which is encoded in two further output row types.

Observe that the $(n/4+1)$-anonymous output matrix consists of exactly t_{out} output row types. Hence the theorem holds for $t_{\text{out}}^{\min}$ and $t_{\text{out}}^{\max}$. This clearly transfers to the cases where t_{out} is specified in the input. □

By allowing more symbols in the alphabet, the intractability result of Theorem 1 also extends to p-SENSITIVITY and ℓ-DIVERSITY.

Corollary 1 *For any fixed constant $t_{\text{out}} \geq 2$,*

1. *p-SENSITIVITY is NP-complete, where $t_{\text{out}} = t_{\text{out}}^{\min} = t_{\text{out}}^{\max}$ and the alphabet size is $\max(t_{\text{out}}+2, p)$, even if the exact, minimum, or maximum number of output row types is additionally specified in the input.*
2. *ℓ-DIVERSITY is NP-complete, where $t_{\text{out}} = t_{\text{out}}^{\min} = t_{\text{out}}^{\max}$, even if the exact, minimum, or maximum number of output row types is additionally specified in the input.*

Proof To extend the reduction given for the proof of Theorem 1 to also work for ℓ-DIVERSITY and p-SENSITIVITY, we define all original columns as quasi-identifiers, and add one extra private column. To keep the argument simple, we only sketch the proof for 2-SENSITIVITY. Cases with larger values of p can be handled in a similar fashion, with an alphabet of size $\max(t_{\text{out}}+2, p)$.

1. For 2-SENSITIVITY, set the private entries for the all-1 row and the all-□ row to "1". For each dummy row type, set half of the private entries to "0" and the rest to "1". Finally, set the private entries of the remaining rows to "0" and set p to "2".

2. For ℓ-Diversity, set the private entry for each row to a new unique symbol and set $\ell := k = (n/4 + 1)$. Now, for every row type containing at least $(n/4 + 1)$ rows, the relative frequency of each private value is at most $1/(n/4 + 1)$. Furthermore, for every row type containing less than $(n/4 + 1)$ rows, the relative frequency of each private value is at least $1/(n/4)$. Thus, every ℓ-diverse matrix is also k-anonymous (restricted on the quasi-identifiers).

The rest of the proof works analogously to the proof of Theorem 1. □

4 Tractability Results

As we saw in the previous section, k-Anonymity and its variants—ℓ-Diversity and p-Sensitivity—are all NP-hard even when there are just two different row types in the output. As such, unless P=NP, these problems cannot be solved in polynomial time—where the degree of the polynomial is a constant independent of the input—even when it is specified as part of the input that the output has to contain a bounded number of row types. In this section we see that the situation changes for the better when the number of row types in the *input* matrix is small. We show, for instance, that when the number of row types in the input matrix is a constant, then we can solve k-Anonymity in time *linear* in the input size (see Subsection 4.1). We then show how to adjust the algorithm to achieve fixed-parameter tractability for the combined parameter (t_{out}, s) (where s denotes the allowed number of suppressions) and for the parameter number m of columns when t_{out} is a constant (see Subsection 4.2). We also derive results for p-Sensitivity (see Subsection 4.3) and for anonymization using the so-called *domain generalization hierarchies* (see Subsection 4.4).

The algorithms described in this section find solutions that have at most t_{out} many output row types. If t_{out} is not part of the input, then by iteratively trying the values $1, \ldots, t_{\text{out}}$ the minimum value such that a solution exists can be determined. Thus, if t_{out} is not given, we have $t_{\text{out}} = t_{\text{out}}^{\min}$.

4.1 Parameter t_{in}

In this subsection we show that k-Anonymity is fixed-parameter tractable with respect to the parameter number t_{in} of input row types. Since $t_{\text{in}} \leq |\Sigma|^m$ and $t_{\text{in}} \leq n$, this result implies that k-Anonymity is also fixed-parameter tractable with respect to the combined parameter $(m, |\Sigma|)$ and the single parameter n. Both these latter results were known previously; Bonizzoni et al (2011b) demonstrated fixed-parameter tractability for the parameter $(m, |\Sigma|)$, and Evans et al (2009) showed the same for the parameter n. Besides achieving a fixed-parameter tractability result for a typically smaller parameter, we improve their results by giving a simpler algorithm with a (usually) better running time.

Algorithm 1 Pseudo-code for solving k-Anonymity. The function `solveRowAssignment` solves Row Assignment in polynomial time, see Lemma 1.

```
1:  procedure solveKAnonymity(M, k, s, t_out)
2:      Determine the row types R_1, ..., R_{t_in}                    ▷ Phase 1, Step 1
3:      for each possible A : [0,1]^{t_in × t_out} do                 ▷ Phase 1, Step 2
4:          for j ← 1 to t_out do                                     ▷ Phase 1, Step 3
5:              if A[1,j] = A[2,j] = ... = A[t_in,j] = 0 then
6:                  delete empty output row type R'_j
7:                  decrease t_out by one
8:              else
9:                  Determine all entries of R'_j
10:         if solveRowAssignment then                                ▷ Phase 2
11:             return 'yes'
12:     return 'no'
```

Let (M, k, s) be an instance of k-Anonymity, and let M' be the (unknown) k-anonymous solution matrix which we seek to obtain from M by suppressing at most s elements. Our fixed-parameter algorithm works in two phases. In the first phase, the algorithm guesses the entries of each row type R'_j in M'. In the second phase, the algorithm computes an assignment of the rows of M to the row types R'_j in M'—see Algorithm 1 for an outline.

We now explain the two phases in detail, beginning with Phase 1. To first determine the row types R_i of M (line 2 in Algorithm 1), the algorithm constructs a trie (Fredkin, 1960) on the rows of M. The leaves of the trie correspond to the row types of M. For later use, the algorithm also keeps track of the numbers $n_1, n_2, \ldots, n_{t_{\text{in}}}$ of each type of row that is present in M; this can clearly be done by keeping a counter at each leaf of the trie and incrementing it by one whenever a new row matches the path to a leaf. All of this can be done in a single pass over M.

For implementing the guess in Step 2 of Phase 1, the algorithm goes over all binary matrices of dimension $t_{\text{in}} \times t_{\text{out}}$; such a matrix A is interpreted as follows: A row of type R_i is mapped[5] to a row of type R'_j if and only if $A[i,j] = 1$ (see line 3). Note that we allow in our guessing step that an output type may contain no row of any input row type. These "empty" output row types are deleted. Hence, with our guessing in Step 2, we guess not only output matrices M' with *exactly* t_{out} types, but also matrices M' with *at most* t_{out} types.

Now the algorithm computes the entries of each row type R'_j, $1 \leq j \leq t_{\text{out}}$, of M' (Step 3 of Phase 1). Assume for ease of notation that $\mathsf{R}_1, \ldots, \mathsf{R}_\ell$ are the row types of M which contribute (according to the guessing) at least one row to the (as yet unknown) output row type R'_j. Now, for each $1 \leq i \leq m$, if $\mathsf{R}_1[i] = \mathsf{R}_2[i] = \ldots = \mathsf{R}_\ell[i]$, then set $\mathsf{R}'_j[i] := \mathsf{R}_1[i]$; otherwise, set $\mathsf{R}'_j[i] := \star$. This yields the entries of the output type R'_j, and the number ω_j of suppres-

[5] Note that not all rows of an input type need to be mapped to the same output type.

sions required to convert any input row (if possible) to the type R'_j is the number of $\star$-entries in R'_j.

The guessing in Step 2 of Phase 1 takes time exponential in the parameter $(t_{\text{in}}, t_{\text{out}})$, but Phase 2 can be done in polynomial time. To show this, we prove that Row Assignment is polynomial-time solvable. We do this in the next lemma, after formally defining the Row Assignment problem. To this end, we use the two sets $T_{\text{in}} = \{1, \ldots, t_{\text{in}}\}$ and $T_{\text{out}} = \{1, \ldots, t_{\text{out}}\}$.

Row Assignment

Input: Nonnegative integers k, s, $\omega_1, \ldots, \omega_{t_{\text{out}}}$ and $n_1, \ldots, n_{t_{\text{in}}}$ with $\sum_{i=1}^{t_{\text{in}}} n_i = n$, 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

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

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

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

$$\sum_{i=1}^{t_{\text{in}}} \sum_{j=1}^{t_{\text{out}}} g(i,j) \cdot \omega_j \leq s \tag{4}$$

Row Assignment formally defines the remaining problem in Phase 2: At this stage of the algorithm the input row types $\mathsf{R}_1, \ldots, \mathsf{R}_{t_{\text{in}}}$ and the number of rows $n_1, \ldots, n_{t_{\text{in}}}$ in these input row types are known. The algorithm has also computed the output row types $\mathsf{R}'_1, \ldots, \mathsf{R}'_{t_{\text{out}}}$ and the number of suppressions $\omega_1, \ldots, \omega_{t_{\text{out}}}$ in these output row types. Now, the algorithm computes an assignment of the rows of the input row types to output row types such that:

- The assignment of the rows respects the guessing in Step 2 of Phase 1. This is secured by Inequality (1).
- M' is k-anonymous, that is, each output row type contains at least k rows. This is secured by Inequality (2).
- All rows of each input row type are assigned. This is secured by Equation (3).
- The total cost of the assignment is at most s. This is secured by Inequality (4).

Note that in the definition of Row Assignment no row type occurs and, hence, the problem is independent of the specific entries of the input or output row types.

Lemma 1 Row Assignment *can be solved in* $O((t_{\text{in}}+t_{\text{out}}) \cdot \log(t_{\text{in}}+t_{\text{out}})(t_{\text{in}} \cdot t_{\text{out}} + (t_{\text{in}} + t_{\text{out}}) \log(t_{\text{in}} + t_{\text{out}})))$ *time.*

Proof We reduce Row Assignment to the Uncapacitated Minimum Cost Flow problem, which is defined as follows (Orlin, 1988):

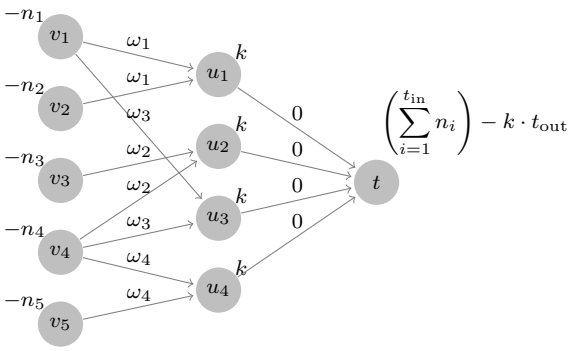

Fig. 3 Example of the constructed network with $t_{\text{in}} = 5$ and $t_{\text{out}} = 4$. The number on each arc denotes its cost. The number next to each node denotes its demand.

Uncapacitated Minimum Cost Flow

Input: A network (directed graph) $D = (V, A)$ with demands $d : V \to \mathbb{Z}$ on the nodes and costs $c : V \times V \to \mathbb{N}$.

Task: Find a function f which minimizes $\sum_{(u,v)\in A} c(u, v) \cdot f(u, v)$ and satisfies:

$$\sum_{\{v|(u,v)\in A\}} f(u, v) - \sum_{\{v|(v,u)\in A\}} f(v, u) = d(u) \qquad \forall u \in V$$

$$f(u, v) \geq 0 \qquad \forall (u, v) \in A$$

We first describe the construction of the network with demands and costs. For each n_i, $1 \leq i \leq t_{\text{in}}$, add a node v_i with demand $-n_i$ (that is, a supply of n_i) and for each ω_j add a node u_j with demand k. If $a(i, j) = 1$, then add an arc (v_i, u_j) with cost ω_j. Finally, add a sink t with demand $(\sum n_i) - k \cdot t_{\text{out}}$ and the arcs (u_j, t) with cost zero. See Figure 3 for an example of the construction. Note that, although the arc capacities are unbounded, the maximum flow over one arc is implicitly bounded by n because the sum of all supplies is $\sum_{i=1}^{t_{\text{in}}} n_i = n$.

The Uncapacitated Minimum Cost Flow problem is solvable in $O(|V| \cdot \log(|V|)(|A|+|V| \cdot \log(|V|)))$ time in a network (directed graph) $D = (V, A)$ (Orlin, 1988). Since our constructed network has $O(t_{\text{in}} + t_{\text{out}})$ nodes and $O(t_{\text{in}} \cdot t_{\text{out}})$ arcs, we can solve our Uncapacitated Minimum Cost Flow-instance in $O((t_{\text{in}} + t_{\text{out}}) \cdot \log(t_{\text{in}} + t_{\text{out}})(t_{\text{in}} \cdot t_{\text{out}} + (t_{\text{in}} + t_{\text{out}}) \log(t_{\text{in}} + t_{\text{out}})))$ time.

It remains to prove that the Row Assignment-instance is a yes-instance if and only if the constructed network has a minimum cost flow of cost at most s.

"⇒": Assume that g is a function fulfilling constraints (1) to (4). Then define a flow f as follows: For each $1 \leq i \leq t_{\text{in}}, 1 \leq j \leq t_{\text{out}}$, set $f(v_i, u_j) = g(i, j)$ and $f(u_j, t) = \sum_{i=1}^{t_{\text{in}}} g(i, j) - k$. The flow f fulfills the demands on the nodes due to Equation (3) and Inequality (2). Since g fulfills Inequality (4) and the cost of each arc (u_j, t), $1 \leq j \leq t_{\text{out}}$, is zero, flow f has cost of at most s.

"$\Leftarrow$": Assume that f is a flow with cost of at most s. All costs and demands are integer valued, and hence, due to the Integrality Property of network flow problems , an optimal flow has also integer values. Then set $g(i,j) = f(v_i, u_j)$ for each $1 \leq i \leq t_{\text{in}}, 1 \leq j \leq t_{\text{out}}$. Note that g fulfills Equation (3) and Inequality (2) due to the demands on the nodes of the network. Since $n_i \leq n$ for all $1 \leq i \leq t_{\text{in}}$, also Inequality (1) is fulfilled. Note that f has cost of at most s and, hence, g fulfills Inequality (4). □

Putting all these together, we arrive at the following theorem:

Theorem 2 *k-Anonymity can be solved in $O(nm + 2^{t_{\text{in}} t_{\text{out}}}(t_{\text{in}} t_{\text{out}} m + (t_{\text{in}} + t_{\text{out}}) \cdot \log(t_{\text{in}} + t_{\text{out}})(t_{\text{in}} \cdot t_{\text{out}} + (t_{\text{in}} + t_{\text{out}}) \log(t_{\text{in}} + t_{\text{out}}))))$ time .*

Proof The correctness of the described two-phase-algorithm (see Algorithm 1) follows from the fact that Phase 1 performs an exhaustive search, Lemma 1, and (as discussed above) that Row Assignment is indeed the remaining problem.

As for the running time: As described above, Step 1 of Phase 1 (line 2 in Algorithm 1) can be done in one pass of the input matrix, that is, in $O(nm)$ time. The loop starting at line 3 iterates $O(2^{t_{\text{in}} t_{\text{out}}})$ times. The loop starting at line 4 iterates at most t_{out} times; the condition inside this loop can be checked in $O(t_{\text{in}})$ time, and the column in A corresponding to an empty output row type can be marked as deleted in constant time. The entries of each R'_j can be computed in $O(mt_{\text{in}})$ time, as described above. As we saw in Lemma 1, the Row Assignment problem can be solved in $O((t_{\text{in}} + t_{\text{out}}) \cdot \log(t_{\text{in}} + t_{\text{out}})(t_{\text{in}} \cdot t_{\text{out}} + (t_{\text{in}} + t_{\text{out}}) \log(t_{\text{in}} + t_{\text{out}})))$ time. Putting all these together, the algorithm runs in $O(nm + 2^{t_{\text{in}} t_{\text{out}}}(t_{\text{in}} t_{\text{out}} m + (t_{\text{in}} + t_{\text{out}}) \cdot \log(t_{\text{in}} + t_{\text{out}})(t_{\text{in}} \cdot t_{\text{out}} + (t_{\text{in}} + t_{\text{out}}) \log(t_{\text{in}} + t_{\text{out}}))))$ time.

In case that t_{out} is not given, the algorithm simple tries all possible values from 0 to t_{out}. The running time is $O(nm + \sum_{i=1}^{t_{\text{out}}} 2^{t_{\text{in}} i}(t_{\text{in}} im + (t_{\text{in}} + i) \cdot \log(t_{\text{in}} + i)(t_{\text{in}} \cdot i + (t_{\text{in}} + i) \log(t_{\text{in}} + i)))) = O(nm + 2^{t_{\text{in}} t_{\text{out}}}(t_{\text{in}} t_{\text{out}} m + (t_{\text{in}} + t_{\text{out}}) \cdot \log(t_{\text{in}} + t_{\text{out}})(t_{\text{in}} \cdot t_{\text{out}} + (t_{\text{in}} + t_{\text{out}}) \log(t_{\text{in}} + t_{\text{out}}))))$. □

We now show that the exponential dependence of the running time of the algorithm of Theorem 2 on the number t_{out} of output types can be done away with. To this end, we exploit that the problem definition (see Section 2) does not impose any restriction on t_{out}. In particular, we are free to look for a solution which *minimizes* the number of output types. We take advantage of this to show that, without loss of generality, one may assume $t_{\text{out}} \leq t_{\text{in}}$ (remember that in this section we set $t_{\text{out}} := t_{\text{out}}^{\min}$).

Lemma 2 *Let (M, k, s) be a yes-instance of k-Anonymity. If M has t_{in} row types, then by suppressing at most s elements one can construct a k-anonymous matrix M'.*

Proof Let M' be a k-anonymous matrix obtained from M by suppressing at most s elements. If M' has at most t_{in} row types, then there is nothing to prove. So let M' have more than t_{in} row types. We now describe an operation

that reduces the number of row types in M' without increasing the cost (in terms of the number of suppressed elements) of obtaining M' from M.

Call a row type R in M' *redistributable* if, for each row r in R, there is a row type R_r in M' such that (i) the preimage r' of r in M can be converted to the type R_r by suppressing in r' the $\star$-symbol positions of R_r, and (ii) R_r contains at most as many $\star$-symbols as R. Observe that if a row type in M' is redistributable, then we can eliminate this row type from M' by "moving" each of its rows to a different, at most as expensive row type, while preserving the condition that there are at least k rows per (output) row type. This operation reduces the number of row types in M' without increasing the cost of obtaining M' from M. As long as there are redistributable row types left in M', we repeatedly eliminate row types in this manner. If the number of remaining row types is at most t_{in}, then we are done. So let there be more than t_{in} row types left in M', none of which is redistributable.

Consider a row type R' in M' that is not redistributable. Let r' be a row in R' such that the cost of suppressing the preimage r of r' to match any row type R'' in M' with $\mathsf{R}' \neq \mathsf{R}''$ is more than the cost of suppressing r to match R'. Since R' is not redistributable such a row r' must exist. Let R be the row type of r in M. Then the cost of suppressing *any* row in R to match *any* row type R'' in M' with $\mathsf{R}' \neq \mathsf{R}''$ is more than the cost of suppressing this row to match R'. It follows that R' is the *only* row type in M' having this property with respect to R. Let f be a mapping that takes each row type R' in M' to some row type R for which R' is the unique cheapest row type. From the above argument, such an f exists and is one-to-one. It follows that M has more than t_{in} row types, a contradiction. □

Combining Lemma 2 with Theorem 2 we get:

Theorem 3 *k-Anonymity can be solved in $O(nm + 2^{t_{\text{in}}^2} t_{\text{in}}^2 (m + t_{\text{in}} \log(t_{\text{in}})))$ time.*

In the beginning of this section, we remarked that our algorithm implies the fixed-parameter tractability of k-Anonymity with respect to the combined parameter $(|\Sigma|, m)$. This was previously shown by Bonizzoni et al (2011b). They presented an algorithm for k-Anonymity which runs in $O(2^{(|\Sigma|+1)^m} kmn^2)$ time and works—similarly to our algorithm—in two phases: First, their algorithm guesses all possible output row types together with their entries in $O(2^{(|\Sigma|+1)^m})$ time. In Phase 1 our algorithm guesses the output row types producible from M within $O(2^{t_{\text{in}} t_{\text{out}}} t_{\text{in}} t_{\text{out}} m + mn)$ time using a different approach. Note that, in general, t_{in} is much smaller than the number $|\Sigma|^m$ of all possible different input types. Hence, in general the guessing step of our algorithm is faster. For instances where $|\Sigma|^m \leq t_{\text{in}} \cdot t_{\text{out}}$, one can exchange the guessing step with the one from Bonizzoni et al. which runs in $O(2^{(|\Sigma|+1)^m})$ time.

Next, we compare Phase 2 of our algorithm to the second step of Bonizzoni et al.'s algorithm. In both algorithms, the same problem Row Assignment is solved. Bonizzoni et al. did this by finding a maximum matching on a bipartite

graph with $O(n)$ nodes, while we do it using a flow network with $O(t_{\text{in}})$ nodes. Consequently, the running time of our approach depends only on t_{in}, and its proof of correctness is—arguably—simpler.

4.2 Combining Parameters with t_{out}

In this subsection we briefly discuss the NP-hardness proof for k-Anonymity from Section 3 in the spirit of "deconstructing intractability" (Komusiewicz et al, 2011; Niedermeier, 2010). In our reduction the alphabet size $|\Sigma|$ and the number t_{out} of output row types are constants whereas the number n of rows, the number m of attributes, number s of suppressions, and the anonymity quality k are unbounded. This suggests a study of the computational complexity of those cases where at least one of these quantities is bounded. Some of the corresponding parameterizations have already been investigated, see Table 1 in Section 1. While for parameters $(|\Sigma|, m)$, $(|\Sigma|, n)$, and n, k-Anonymity is fixed-parameter tractable, it is open whether combining t_{out} with m, k, or s helps to obtain fixed-parameter tractability. In particular, the parameterized complexity for the combined parameter $(|\Sigma|, s, k)$ is still open. In contrast, k-Anonymity is W[1]-hard for (s, k) (Bonizzoni et al, 2011b), that is, it is presumably fixed-parameter *intractable* for this combined parameter.

Whereas k-Anonymity is NP-hard for constant m and unbounded t_{out}, one can easily construct a fixed-parameter algorithm with respect to the parameter m when t_{out} is a constant: In $O(2^{m \cdot t_{\text{out}}} \cdot m \cdot t_{\text{out}})$ time guess the suppressed columns for all output row types. Then, guess in $n^{O(t_{\text{out}})}$ time one prototype for each output row type, that is, one input row that is a preimage of a row from the output row type. Now, knowing the entries for each output row, one can simply apply the Row Assignment algorithm from Section 4:

Proposition 2 *k-Anonymity parameterized by the number m of columns is fixed-parameter tractable when the number t_{out} of output row types is a constant.*

Next, we prove fixed-parameter tractability for k-Anonymity with respect to the combined parameter (t_{out}, s) by showing that the number t_{in} of input types is at most $(t_{\text{out}} + s)$. To this end, consider a feasible solution for an arbitrary k-Anonymity instance. We distinguish between input row types that have rows which have at least one suppressed entry in the solution (*suppressed input row types* in the following) and input row types that do only have rows that remain unchanged in the solution (*unsuppressed input row types* in the following). Clearly, every unsuppressed input row type needs at least one unsuppressed output row type. Thus, the number of unsuppressed input row type cannot exceed t_{out}. Furthermore, the number of rows that have at least one suppressed entry is at most s. Hence the number of suppressed input row types is at most s. It follows that $t_{\text{in}} \leq t_{\text{out}} + s$. Now, fixed-parameter tractability follows from Theorem 2:

Proposition 3 *k-Anonymity is fixed-parameter tractable with respect to the combined parameter (t_{out}, s).*

To achieve a better running time one might want to develop a direct fixed-parameter algorithm for (t_{out}, s), remaining a task for future research. Finally, we conjecture that an XP-algorithm for k-Anonymity with respect to the combined parameter (t_{out}, k) can be achieved.

4.3 p-Sensitivity

In this section we describe how our algorithm from Theorem 2 can be extended for solving p-Sensitivity (Truta and Vinay, 2006). We leave Phase 1 unchanged, and add extra constraints to the Row Assignment problem to ensure that the output matrix M' is p-sensitive. We first describe this modified Row Assignment problem. We then model the extra requirements using a modified flow construction, and show that this modified version can also be solved in polynomial time. To this end, we formally define the p-Row Assignment problem which enhances Row Assignment to incorporate the p-sensitivity constraint.

p-Row Assignment

Input: Nonnegative integers p, k, s, $\omega_1, \ldots, \omega_{t_{\text{out}}}$ and $n_{1,\sigma_1}, \ldots, n_{t_{\text{in}},\sigma_{|\Sigma|}}$ with $\sum_{i=1}^{t_{\text{in}}} \sum_{\sigma \in \Sigma} n_{i,\sigma} = n$, and a function $a : T_{\text{in}} \times T_{\text{out}} \to \{0, 1\}$.

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

$$a(i,j) \cdot n \geq \sum_{\sigma \in \Sigma} g(i,\sigma,j) \qquad \forall i \in T_{\text{in}} \forall j \in T_{\text{out}} \tag{5}$$

$$\sum_{i=1}^{t_{\text{in}}} \sum_{\sigma \in \Sigma} g(i,\sigma,j) \geq k \qquad \forall j \in T_{\text{out}} \tag{6}$$

$$\sum_{j=1}^{t_{\text{out}}} g(i,\sigma,j) = n_{i,\sigma} \qquad \forall i \in T_{\text{in}} \forall \sigma \in \Sigma \tag{7}$$

$$\sum_{i=1}^{t_{\text{in}}} \sum_{\sigma \in \Sigma} \sum_{j=1}^{t_{\text{out}}} g(i,\sigma,j) \cdot \omega_j \leq s \tag{8}$$

$$\sum_{\sigma \in \Sigma} \operatorname{sgn}\left(\sum_{i=1}^{t_{\text{in}}} g(i,\sigma,j) \right) \geq p \qquad \forall j \in T_{\text{out}} \tag{9}$$

Here $\operatorname{sgn}(\cdot)$ denotes the sign (signum) function which maps positive numbers to 1, negative numbers to -1, and zero to itself. The integer $n_{i,\sigma}$ denotes the number of rows in the input row type R_i with entry σ in the private column. This refined distinction of the input row types is necessary, since two rows in one input row type may differ in the private information column. Inequalities

(5) to (8) form a refinement of the Inequalities (1) to (4) used in the definition of Row Assignment. As before, these adjusted Inequalities ensure that the guessing in Step 2 of Phase 1 is respected (Inequality (5)), the output is k-anonymous (Inequality (6)), all input rows are assigned to output row types (Equation (7)), and the cost bound is respected (Inequality (8)). Additionally, Inequality (9) ensures that the output matrix is p-sensitive.

Analogously to Lemma 1 we now show that p-Row Assignment can be solved in polynomial time.

Lemma 3 *p-Row Assignment can be solved in $O((n+|\Sigma|\cdot t_{\text{in}})^4 \log(n+|\Sigma|\cdot t_{\text{in}}))$ time.*

Proof We reduce p-Row Assignment to the Capacitated Minimum Cost Flow problem, which is defined as follows (Orlin, 1988):

Capacitated Minimum Cost Flow

Input: A network (directed graph) $D = (V, A)$ with demands $d : V \to \mathbb{Z}$ on the nodes, capacities $\mathrm{f}_{\max} : A \to \mathbb{Z}$, lower bounds on the flow $\mathrm{f}_{\min} : A \to \mathbb{Z}$, and costs $c : A \to \mathbb{N}$.

Task: Find a function f which minimizes $\sum_{(u,v)\in A} c(u,v) \cdot f(u,v)$ and satisfies:

$$\sum_{\{v|(u,v)\in A\}} f(u,v) - \sum_{\{v|(v,u)\in A\}} f(v,u) = d(u) \qquad \forall u \in V$$

$$\mathrm{f}_{\max}(u,v) \geq f(u,v) \geq \mathrm{f}_{\min}(u,v) \qquad \forall (u,v) \in A$$

The construction of our reduction is a modification of the construction in the proof of Lemma 1. For each $n_{i,\sigma}$, $1 \leq i \leq t_{\text{in}}$, $\sigma \in \Sigma$, add a node $v_{i,\sigma}$ with demand $-n_{i,\sigma}$ (that is, a supply of $n_{i,\sigma}$) and for each ω_j add a node u_j with demand k. For each node u_j add the $|\Sigma|+1$ nodes $u_{j,\sigma_1}, \ldots, u_{j,\sigma_{|\Sigma|}}$, and u_j^p. Add for each $1 \leq j \leq t_{\text{out}}$ and $\sigma \in \Sigma$ the arcs $(u_{j,\sigma}, u_j^p)$ with capacity $\mathrm{f}_{\max}(u_{j,\sigma}, u_j^p) = 1$. Also add (u_j^p, u_j) with capacity and minimum flow $\mathrm{f}_{\max}(u_j^p, u_j) = \mathrm{f}_{\min}(u_j^p, u_j) = p$ and with cost $c(u_j^p, u_j) = \omega_j$. If $a(i,j) = 1$, then add for all $1 \leq i \leq t_{\text{in}}$, $1 \leq j \leq t_{\text{out}}$, and $\sigma \in \Sigma$ the arcs $(v_{i,\sigma}, u_{j,\sigma})$ with costs 0 and add the arc $(v_{i,\sigma}, u_j)$ with cost ω_j. Finally, add a sink t with demand $n - k \cdot t_{\text{out}}$ and the arcs (u_j, t). For all arcs where so far the cost c or the minimum flow $\mathrm{f}_{\min}$ is not specified the corresponding value is zero. Arcs where the capacity $\mathrm{f}_{\max}$ is not given have capacity of n. See Figure 4 for an example of the construction.

As for the running time: The Capacitated Minimum Cost Flow problem is solvable in $O(|A| \cdot \log(|V|)(|A| + |V| \cdot \log(|V|)))$ time in a network (directed graph) $D = (V, A)$ (Orlin, 1988). Since our constructed network has $O(n + |\Sigma| \cdot t_{\text{out}})$ nodes and $O((n + |\Sigma| \cdot t_{\text{out}})^2)$ arcs, we can solve our Uncapacitated Minimum Cost Flow-instance in $O((n + |\Sigma| \cdot t_{\text{out}})^2 \cdot \log(n + |\Sigma| \cdot t_{\text{out}})((n + |\Sigma| \cdot t_{\text{out}})^2 + (n + |\Sigma| \cdot t_{\text{out}}) \log(n + |\Sigma| \cdot t_{\text{out}})))$, that is $O((n + |\Sigma| \cdot t_{\text{out}})^4 \log(n + |\Sigma| \cdot t_{\text{out}}))$ time. By Lemma 2, this can be upper-bounded by $O((n + |\Sigma| \cdot t_{\text{in}})^4 \log(n + |\Sigma| \cdot t_{\text{in}}))$

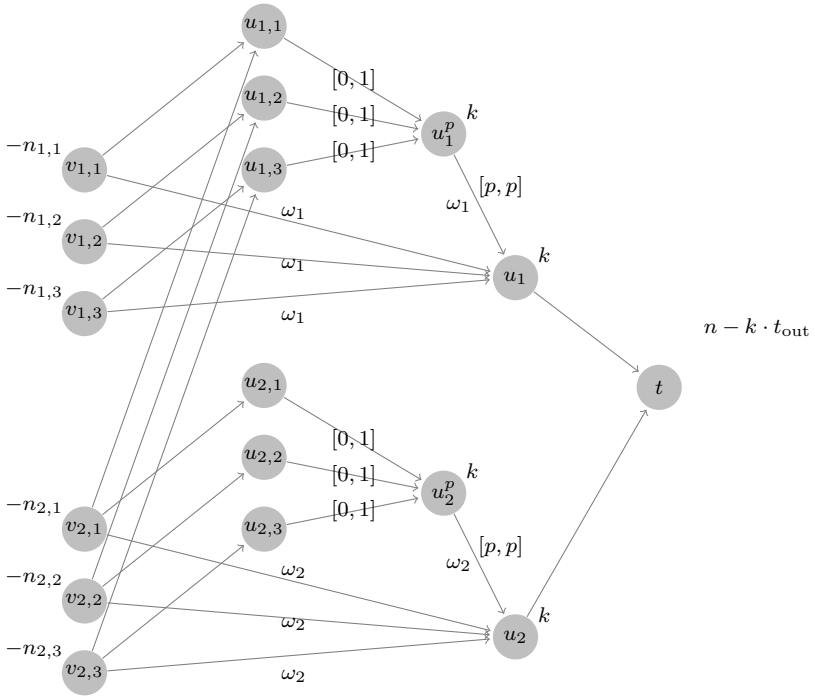

Fig. 4 A small example of the construction referring to an input matrix with two row types, each containing the three different entries "1", "2", and "3" in their respective private column. In the example two output row types are in the solution with $a(1,1) = a(2,1) = a(2,2) = 1$ and $a(1,2) = 0$, meaning that no row from the first input row type is assigned to the second output row type. The lower and upper bounds on the flow over the edges are given in brackets on the arcs: [lower bound, upper bound]. Values not being in brackets refer to the costs of the corresponding arc. If the cost is not pictured then the cost is zero. Missing bounds on the flow indicate $[0, n]$, that is lower bound zero and upper bound (capacity) n. Labels next to nodes indicate the demands of the nodes.

It remains to prove that the p-Row Assignment-instance is a yes-instance if and only if the constructed network has a minimum cost flow of cost at most s.

"⇒": Assume that g is a function fulfilling constraints (5) to (9). Then define a flow f as follows: For each $1 \leq i \leq t_{\text{in}}, 1 \leq j \leq t_{\text{out}}$, temporarily set $f(v_{i,\sigma}, u_j) = g(i, \sigma, j)$ and $f(u_j, t) = \sum_{i=1}^{t_{\text{in}}} \sum_{\sigma \in \Sigma} g(i, \sigma, j) - k$. To fulfill the minimum flow requirement on the arcs (u_j^p, u^j) adjust the flow as follows. Since g fulfills Inequality (9), there are for each u_j at least p pairwise distinct elements of the alphabet $\sigma_1, \ldots, \sigma_p \in \Sigma$ such that $g(i_1, \sigma_1, j), \ldots, g(i_p, \sigma_p, j)$ are all greater than zero for some $i_1, \ldots, i_p \in \{1, \ldots, t_{\text{in}}\}$ (not necessarily pairwise distinct). For each $1 \leq r \leq p$ adjust the flow as follows: Reduce $f(v_{i_r,\sigma_r}, u_j)$ by one and increase $f(v_{i_r,\sigma_r}, u_{j,\sigma_r})$, $f(u_{j,\sigma_r}, u_j^p)$, and $f(u_j^p, u_j)$ by one. Note that this adjustment does not violate the capacities on the arcs $(v_{i_r,\sigma_r}, u_{j,\sigma_r})$, (u_{j,σ_r}, u_j^p), and (u_j^p, u_j), since the flow on each of these rows is increased exactly once during the whole adjustment process. The flow f fulfills the demands on

the nodes due to Equation (7) and Inequality (6). Since g fulfills Inequality (8) and the cost of each arc (u_j, t), $1 \leq j \leq t_{\text{out}}$, is zero, flow f has cost of at most s. Inequality (5) is fulfilled due to the construction: if $a(i, j) = 0$ then there exists no path from $v_{i,\sigma}$ to u_j for any $\sigma \in \Sigma$. Hence, in this case $g(i, \sigma, j) = 0$ for all $\sigma \in \Sigma$.

"$\Leftarrow$": Assume that f is a flow with cost of at most s. All costs and demands are integer valued, and hence, due to the Integrality Property of network flow problems , an optimal flow has also integer values. Then set $g(i, \sigma, j) = f(v_i, \sigma, u_j) + f(v_i, \sigma, u_{j,\sigma})$ for each $1 \leq i \leq t_{\text{in}}, 1 \leq j \leq t_{\text{out}}, \sigma \in \Sigma$. Note that g fulfills Equation (7) and Inequality (6) due to the demands on the nodes of the network. Since $n_{i,\sigma} \leq n$ for all $1 \leq i \leq t_{\text{in}}$, also Inequality (5) is fulfilled. Note that f has cost of at most s and, hence, g fulfills Inequality (8). Furthermore, for each $1 \leq j \leq t_{\text{out}}$ it holds that $\mathrm{f}_{\min}(u_j^p, u_j) = p$ and $\mathrm{f}_{\max}(u_{j,\sigma}, u_j^p) = 1$ for each $\sigma \in \Sigma$. Thus, Inequality (9) is fulfilled. □

Putting all these together we arrive at the following.

Theorem 4 *p-Sensitivity can be solved in $O(nm + 2^{t_{\text{in}}^2}(t_{\text{in}}^2 m + (n + |\Sigma| \cdot t_{\text{in}})^4 \log(n + |\Sigma| \cdot t_{\text{in}})))$ time.*

The question whether also ℓ-Diversity is fixed-parameter tractable with respect to the parameter t_{in} remains open.

Note that we defined t_{in} and t_{out} as numbers of different rows in the matrix restricted to the quasi-identifiers. From a theoretical point of view the number of different rows in the matrix (including the private column), denoted by t_{in}^* and t_{out}^* for the input matrix and output matrix, respectively, is also interesting. Clearly, $t_{\text{in}} \leq t_{\text{in}}^*$.

We remark that ℓ-Diversity is fixed-parameter tractable with respect to the combined parameter $(t_{\text{in}}^*, t_{\text{out}}^*)$: Again take our Algorithm 1 and make some adjustments in Phase 2. After the guessing step in Phase 1, again compute the output row types and the costs. Then one can solve Phase 2 via a straightforward ILP formulation that is based on the inequalities in the Row Assignment definition plus one constraint ensuring that the solution is l-diverse. A similar ILP was already described by Dondi et al (2012) in order to show that ℓ-Diversity is fixed-parameter tractable with respect to the combined parameter $(|\Sigma|, m)$.

4.4 Domain Generalization Hierarchies

In this section we describe how our algorithm for k-Anonymity from Theorem 2 can be modified to find optimal domain generalization hierarchies.

The *domain* of a column in the input matrix M is the set of all values that may appear in the column. The operation of *domain generalization* consists of replacing each value v in a domain with a value v' which is at least as general as v, for a suitable definition of "general". A meaningful domain generalization takes each value to a "less specific, more general value which is faithful to the original" (Sweeney, 2002a).

A *domain generalization hierarchy* (DGH) for a column of a matrix is a sequence of generalizations. The domain of the first generalization in the sequence is the domain of the column itself. The domain of each subsequent generalization is the co-domain of its previous generalization. The range of the last generalization consists of a single value, denoted $\star$.

Observe that the notion of suppression can be thought of as a domain generalization hierarchy which consists of exactly one generalization.

We say that M' is a *minimal generalization* of M with a property Π if (i) M' is a generalization of M, (ii) M' has the property Π, and (iii) replacing any entry of M' with a more specific value will result in a matrix which does not satisfy the property Π. See Sweeney (2002a) for a formal definition.

The DGH-k-Anonymity problem is defined as follows:

DGH-k-Anonymity

Input: A matrix M, $k \in \mathbb{N}$, and a DGH for each column of M.

Question: Find a minimal generalization of M which is k-anonymous.

This problem is motivated by the fact that we may assume that any meaningful generalization loses information; it is thus of advantage to look for minimal k-anonymous generalizations.

To see how to use our framework to deal with DGH-k-Anonymity, let us take a closer look at the algorithm for k-Anonymity from Theorem 2. The algorithm computes the entries of the output row types in Step 2 in Phase 1. To modify this step to use generalization instead of suppression, we do the following: For each output row type R', set the value at position i to be the most specific value which generalizes the values at position i of all the row types which assign at least one row to R'. Since our algorithm is based on exhaustive search, it can compute a minimal generalization such that the output matrix is k-anonymous. Assuming that we can apply generalizations and do comparisons in constant time, we get

Theorem 5 DGH-k-Anonymity *can be solved in* $O(2^{t_{\text{in}}^2} t_{\text{in}}^2 (mn + t_{\text{in}} \log(t_{\text{in}})) + nm)$ *time.*

Using domain generalization hierarchies to obtain p-sensitive matrices leads to the following problem:

DGH-p-Sensitivity

Input: An $n \times m$-matrix M over a fixed alphabet Σ and nonnegative integers k, s, and p.

Question: Find a minimal generalization of M such that M_{qi} is k-anonymous and M is p-sensitive.

Similar to the modification in the algorithm of k-Anonymity to solve DGH-k-Anonymity, the algorithm for p-Sensitivity can be modified to solve DGH-p-Sensitivity. Thus we get:

Corollary 2 DGH-p-Sensitivity *can be solved in* $O(nm + 2^{t_{\text{in}}^2}(t_{\text{in}}^2 nm + (n + |\Sigma| \cdot t_{\text{in}})^4 \log(n + |\Sigma| \cdot t_{\text{in}})))$ *time.*

5 Conclusion

In this paper we looked at the effect of input and output homogeneity on the complexity of four combinatorial data anonymization problems, namely k-Anonymity, DGH-k-Anonymity, p-Sensitivity, and ℓ-Diversity. We took the number of row types as the measure of homogeneity. We showed that while the problems remain NP-hard even when the number of row types required in the output is a small constant, two of the problems become tractable—in the parameterized sense—when the number of *input* row types is small.

Our studies so far are purely theoretical. While our results contribute to a better understanding of how structural properties of the data matrices may influence the computational complexity of the—in general NP-hard—data anonymization problems studied here, it remains unclear whether some of the proposed algorithms may turn useful for practical purposes. It seems clear, however, that trying to further shrink exponential factors such as $2^{t_{\text{in}}^2}$ is of key importance and remains a task for future work.

There are several fundamental challenges concerning the investigation of the parameterized complexity of the considered combinatorial data anonymization problems. For k-Anonymity, it remains open whether it is fixed-parameter tractable for the combined parameter (k, t_{out}). This question directly extends to DGH-k-Anonymity. For ℓ-Diversity it remains a major open question to determine its parameterized complexity for the parameter t_{in}. In more general terms, exploring the parameterized and multivariate complexity landscape (Fellows, 2009; Niedermeier, 2010) of combinatorial data anonymization problems seems a natural and fruitful course of action with many natural challenges still left to be tackled.

References

Aggarwal G, Feder T, Kenthapadi K, Motwani R, Panigrahy R, Thomas D, Zhu A (2005) Anonymizing tables. In: Proceedings of the 10th International Conference on Database Theory (ICDT '05), Springer, LNCS, vol 3363, pp 246–258 → p 6.

Aggarwal G, Feder T, Kenthapadi K, Khuller S, Panigrahy R, Thomas D, Zhu A (2010) Achieving anonymity via clustering. ACM Transactions on Algorithms 6(3):1–19 → p 10.

Blocki J, Williams R (2010) Resolving the complexity of some data privacy problems. In: Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP '10), Springer, LNCS, vol 6199, pp 393–404 → pp 4 and 5.

Bonizzoni P, Della Vedova G, Dondi R (2011a) Anonymizing binary and small tables is hard to approximate. Journal of Combinatorial Optimization 22(1):97–119 → pp 4, 5, and 6.

Bonizzoni P, Della Vedova G, Dondi R, Pirola Y (2011b) Parameterized complexity of k-anonymity: hardness and tractability. Journal of Combinatorial Optimization Available online → pp 4, 5, 6, 15, 20, and 21.

Bredereck R, Nichterlein A, Niedermeier R, Philip G (2011) Pattern-guided data anonymization and clustering. In: Proceedings of the 36th International Symposium Mathematical Foundations of Computer Science (MFCS '11), Springer, LNCS, vol 6907, pp 182–193 → p 10.

Campan A, Truta TM (2009) Data and structural k-anonymity in social networks. In: Proceedings of the 2nd ACM SIGKDD International Workshop on Privacy, Security, and Trust in KDD (PinKDD '08), LNCS, vol 5456, Springer, pp 33–54 → p 3.

Chakaravarthy VT, Pandit V, Sabharwal Y (2010) On the complexity of the k-anonymization problem. CoRR abs/1004.4729 → p 4.
Dondi R, Mauri G, Zoppis I (2012) The ℓ-diversity problem: Tractability and approximability. Theoretical Computer Science Available online → p 25.
Downey RG, Fellows MR (1999) Parameterized Complexity. Springer → p 7.
Dwork C (2011) A firm foundation for private data analysis. Communications of the ACM 54(1):86–95 → p 3.
Evans PA, Wareham T, Chaytor R (2009) Fixed-parameter tractability of anonymizing data by suppressing entries. Journal of Combinatorial Optimization 18(4):362–375 → pp 4, 5, 6, and 15.
Fard AM, Wang K (2010) An effective clustering approach to web query log anonymization. In: Katsikas SK, Samarati P (eds) Proceedings of the International Conference on Security and Cryptography (SECRYPT '10), SciTePress, pp 109–119 → p 10.
Fellows MR (2009) Towards fully multivariate algorithmics: Some new results and directions in parameter ecology. In: Proceedings of the 20th International Workshop on Combinatorial Algorithms (IWOCA '09), Springer, LNCS, vol 5874, pp 2–10 → pp 5 and 27.
Flum J, Grohe M (2006) Parameterized Complexity Theory. Springer → p 7.
Frank A, Asuncion A (2010) UCI machine learning repository. URL http://archive.ics.uci.edu/ml → p 5.
Fredkin E (1960) Trie memory. Communications of the ACM 3(9):490–499 → p 16.
Fung BCM, Wang K, Chen R, Yu PS (2010) Privacy-preserving data publishing: A survey of recent developments. ACM Computing Surveys 42(4):14:1–14:53 → p 3.
Gedik B, Liu L (2008) Protecting location privacy with personalized k-anonymity: Architecture and algorithms. IEEE Transactions on Mobile Computing 7(1):1–18 → p 3.
Gionis A, Tassa T (2009) k-anonymization with minimal loss of information. IEEE Transactions on Knowledge and Data Engineering 21(2):206 –219 → p 4.
Gkoulalas-Divanis A, Kalnis P, Verykios VS (2010) Providing k-anonymity in location based services. ACM SIGKDD Explorations Newsletter 12:3–10 → p 3.
Gruteser M, Grunwald D (2003) Anonymous usage of location-based services through spatial and temporal cloaking. In: Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MOBISYS '03), ACM, MobiSys '03, pp 31–42 → p 3.
Johnson DS (1987) The NP-completeness column: An ongoing guide. Journal of Algorithms 8:438–448 → p 11.
Kenig B, Tassa T (2012) A practical approximation algorithm for optimal k-anonymity. Data Mining and Knowledge Discovery 25:134–168 → p 4.
Komusiewicz C, Niedermeier R, Uhlmann J (2011) Deconstructing intractability–A multivariate complexity analysis of interval constrained coloring. Journal of Discrete Algorithms 9:137–151 → p 21.
Li N, Li T, Venkatasubramanian S (2007) t-closeness: Privacy beyond k-anonymity and l-diversity. In: In Proceedings of the 23rd International Conference on Data Engineering (ICDE '07), IEEE, pp 106–115 → p 4.
Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) ℓ-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data 1(1) → pp 2, 3, 4, 5, and 6.
Meyerson A, Williams R (2004) On the complexity of optimal k-anonymity. In: Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '04), ACM, pp 223–228 → pp 4, 5, and 6.
Monreale A, Andrienko G, Andrienko N, Giannotti F, Pedreschi D, Rinzivillo S, Wrobel S (2010) Movement data anonymity through generalization. Transactions on Data Privacy 3:91–121 → p 3.
Navarro-Arribas G, Torra V, Erola A, Castellà-Roca J (2012) User k-anonymity for privacy preserving data mining of query logs. Information Processing & Management 48(3):476–487 → p 3.
Niedermeier R (2006) Invitation to Fixed-Parameter Algorithms. Oxford University Press → p 7.
Niedermeier R (2010) Reflections on multivariate algorithmics and problem parameterization. In: Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS '10), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik,

Leibniz International Proceedings in Informatics (LIPIcs), vol 5, pp 17–32 → pp 5, 21, and 27.

Orlin J (1988) A faster strongly polynomial minimum cost flow algorithm. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88), ACM, pp 377–387 → pp 17, 18, and 23.

Park H, Shim K (2007) Approximate algorithms for k-anonymity. In: Proceedings of the International Conference on Management of Data (SIGMOD '07), ACM, SIGMOD '07, pp 67–78 → p 4.

Samarati P (2001) Protecting respondents identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6):1010–1027 → pp 2 and 8.

Samarati P, Sweeney L (1998) Generalizing data to provide anonymity when disclosing information. In: Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '98), ACM, pp 188–188 → pp 2 and 8.

Sweeney L (2000) Uniqueness of simple demographics in the U.S. population. Tech. rep., Carnegie Mellon University, School of Computer Science, Laboratory for International Data Privacy → p 4.

Sweeney L (2002a) Achieving k-anonymity privacy protection using generalization and suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5):571–588 → pp 3, 25, and 26.

Sweeney L (2002b) k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5):557–570 → pp 2, 3, and 8.

Truta TM, Vinay B (2006) Privacy protection: p-sensitive k-anonymity property. In: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDE '06), IEEE Computer Society, p 94 → pp 2, 4, 6, 9, and 22.

Wong RCW, Li J, Fu AWC, Wang K (2006) (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining (KDD '06), ACM, pp 754–759 → pp 4 and 9.

Xiao X, Tao Y (2006) Anatomy: Simple and effective privacy preservation. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06), ACM, pp 139–150 → pp 4 and 9.

Xiao X, Yi K, Tao Y (2010) The hardness and approximation algorithms for l-diversity. In: Proceedings of the 13th International Conference on Extending Database Technology (EDBT' 10), ACM, pp 135–146 → p 8.

Zhou B, Pei J (2011) The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. Knowledge and Information Systems 28:47–77 → p 3.