The Effect of Homogeneity on the Complexity of k-Anonymity

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. 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. 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. 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 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 results also extend to some interesting generalizations of k-Anonymity.

1 Introduction

Assume that data about individuals are represented by equal-length vectors consisting of attribute values. If all vectors are identical, then we have full homogeneity and thus full anonymity of all individuals. Relaxing full anonymity to k-anonymity, in this work we investigate how the degree of (in)homogeneity influences the computational complexity of the NP-hard problem of making sets of individuals k-anonymous.

Sweeney [24] 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 concrete optimization goal while 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,

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

To appear in the Proceedings of the 18th International Symposium on Fundamentals of Computer Theory (FCT)*, Oslo, Norway, August 2011, Lecture Notes in Computer Science.*

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 problem, our results also extend to several more general problems.

We focus on a better understanding of the computational complexity and on tractable special cases of these problems; see Machanavajjhala et al. [18] and Sweeney [24] for discussions on the pros and cons of these models in terms of privacy vs preservation of meaningful data. In particular, note that in the data privacy community "differential privacy" (cleverly adding some random noise) is now the most popular method [9,14]. However, k-Anonymity is a very natural combinatorial problem (with potential applications beyond data privacy), and—for instance—in the case of "one-time anonymization", may still be valuable for the sake of providing a simple model that does not introduce noise.

k-Anonymity and many related problems are NP-hard [19], even when the input matrix is highly restricted. For instance, it is APX-hard when $k = 3$, even when the alphabet size is just two [4]; NP-hard when $k = 4$, even when the number of columns in the input dataset is 8 [4]; MAX SNP-hard when $k = 7$, even when the number of columns in the input dataset is just three [7]; and MAX SNP-hard when $k = 3$, even when the number of columns in the input dataset is 27 [3].

Confronted with this computational hardness, we study the parameterized complexity of k-Anonymity as initiated by Evans et al. [10]. The central question here is how naturally occurring parameters influence the complexity of k-Anonymity. For example, is k-Anonymity polynomial-time solvable for constant values of k? The general answer is "no" since already 3-Anonymity is NP-hard [19], even on binary data sets [4]. Thus, k alone does not give a promising parameterization.[1]

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 [21], various combinations of single parameters. Here the arity of $|\Sigma|$ may range from binary (such as gender) to unbounded. For instance, answering an open question of Evans et al. [10], Bonizzoni et al. [5] recently 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 $|\Sigma|$ [10]. We emphasize that Bonizzoni et al. [5] 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. Clearly, $|\Sigma|^m$ denotes the maximum possible number of different input rows. 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. [10] and Bonizzoni et al. [5], summarizes known and new results.

[1] However, it has been shown that 2-Anonymity is polynomial-time solvable [3].

Table 1. The parameterized complexity of k-Anonymity. 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).

	–	k	s	k, s
–	NP-hard [19]	NP-hard [19]	W[1]-hard [5]	W[1]-hard [5]
$\|\Sigma\|$	NP-hard [2]	NP-hard [2]	?	?
m	NP-hard [4]	NP-hard [4]	FPT [10]	FPT [10]
n	FPT [10]	FPT [10]	FPT [10]	FPT [10]
$\|\Sigma\|, m$	FPT [5]	FPT [10]	FPT [10]	FPT [10]
$\|\Sigma\|, n$	FPT [10]	FPT [10]	FPT [10]	FPT [10]
t_{in}	**FPT**	**FPT**	**FPT**	**FPT**
t_{out}	**NP-hard**	?	**FPT**	**FPT**

Our contributions. We introduce the "homogeneity parameters" t_{in}, the number of different input rows, and t_{out}, the number of different output rows, for studying the computational complexity of k-Anonymity and related problems. Typically, we expect $t_{\text{in}} \ll n$ and $t_{\text{in}} \ll |\Sigma|^m$. Indeed, 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 $|\Sigma|^m$ is a trivial upper bound for homogeneity.

First, we show that there is always an optimal solution (minimizing the number of suppressions) with $t_{\text{out}} \leq t_{\text{in}}$. Then, we derive an algorithm that solves k-Anonymity in $O(nm + 2^{t_{\text{in}} t_{\text{out}}} t_{\text{in}}(t_{\text{out}} m + t_{\text{in}}^2 \cdot \log(t_{\text{in}})))$ time, which compares favorably with Bonizzoni et al.'s [5] algorithm running in $O(2^{(|\Sigma|+1)^m} kmn^2)$ time. Since $t_{\text{out}} \leq t_{\text{in}}$, this shows that k-Anonymity is fixed-parameter tractable when parameterized by t_{in}. In particular, when t_{in} is a constant, our algorithm solves k-Anonymity in time *linear* in the size of the input. In contrast, when only t_{out} is fixed, then we show that the problem remains NP-hard. More precisely, opposing the trivial case $t_{\text{out}} = 1$, we show that k-Anonymity is already NP-hard when $t_{\text{out}} = 2$, even when $|\Sigma| = 4$. We remark that t_{out} is an interesting parameter since it is "stronger" than t_{in} and since, interpreting k-Anonymity as a (meta-)clustering problem (of Aggarwal et al. [1]), t_{out} may also be interpreted as the number of output "clusters".

Finally, we mention that all our results extend to more general problems, including ℓ-Diversity [18] and "k-Anonymity with domain generalization hierarchies" [23]; we defer the corresponding details to a full version of the paper.

Preliminaries and basic observations. 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 $\Sigma \uplus \{\star\}$. 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" this row. We say that a matrix is *k-anonymous* if every row type contains none or at least k rows in the matrix. 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 central problem of this work reads as follows.

k-Anonymity
Input: An $n \times m$-matrix M and nonnegative integers k, s.
Question: Can at most s elements of M be suppressed to obtain a k-anonymous matrix M'?

Our algorithmic results mostly rely on concepts of parameterized algorithmics [8,12,20]. 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 only lies in the parameterized complexity class XP.

We study two new parameters t_{in} and t_{out}, where t_{in} denotes the number of input row types in the given matrix and t_{out} denotes the number of row types in the output k-anonymous matrix. Note that using sorting t_{in} can be efficiently determined for a given matrix. To keep the treatment simple, we assume that t_{out} is a user-specified number which bounds the maximum number of output types. Clearly, one could have variations on this: for instance, one could ask to minimize t_{out} with at most a given number s of suppressions. We refrain from further exploration here and treat t_{out} as part of the input specifying an upper bound on the number of output types.[2] The following lemma says that without loss of generality one may assume $t_{\text{out}} \leq t_{\text{in}}$.

Lemma 1. *Let (M, k, s) be a* YES*-instance of k-*Anonymity*. If M has t_{in} row types, then there exists a k-anonymous matrix M' with at most t_{in} row types which can be obtained from M by suppressing at most s elements.*

2 Parameter t_{in}

In this section 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

[2] Indeed, interpreting k-Anonymity as a clustering problem with a guarantee k for the minimum cluster size (as in the work by Aggarwal et al. [1]), t_{out} can also be seen as the number of "clusters" (that is, output row types) that are built.

Algorithm 1 Pseudo-code for solving k-ANONYMITY. The function `solveRowAssignment` solves ROW ASSIGNMENT in polynomial time, see Lemma 2.

```
 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'
```

$t_{\text{in}} \leq n$, 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 already proved quite recently; Bonizzoni et al. [5] demonstrated fixed-parameter tractability for the parameter $(m, |\Sigma|)$, and Evans et al. [10] showed the same for the parameter n. Besides achieving a fixed-parameter tractability result for a more general and typically smaller parameter, we improve their results by giving a simpler algorithm with a (usually) better running time.

Let (M, k, s) be an instance of k-ANONYMITY, and let M' be the (unknown) k-anonymous 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 determine the row types R_i of M (line 2 in Algorithm 1), the algorithm constructs a trie [13] 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[3] 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 an output type to 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.

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

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 $\mathsf{R}_1[i] = \mathsf{R}_2[i] = ... = \mathsf{R}_\ell[i]$, 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 suppressions 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_{in}, 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.

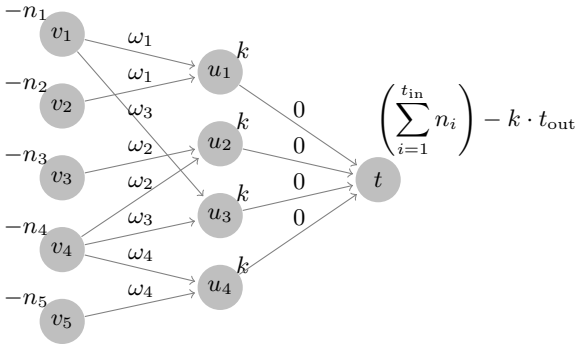

Fig. 1. Example of the constructed network with $t_{\text{in}} = 5$ and $t_{\text{out}} = 4$. The number on each arc denotes its cost. The number next to each node denotes its demand.

Lemma 2. Row Assignment *can be solved in* $O(t_{in}^3 \cdot \log(t_{in}))$ *time.*

Proof (Sketch). We reduce Row Assignment to the Uncapacitated Minimum Cost Flow problem, which is defined as follows [22]:

Uncapacitated Minimum Cost Flow

Input: A network (directed graph) $D = (V, A)$ with demands $b : V \to \mathbb{Z}$ 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:

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

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

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_i, t) with cost zero. See Figure 1 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) $D = (V, A)$ [22]. Since our constructed network has $t_{\text{in}} + t_{\text{out}}$ nodes and $t_{\text{in}} \cdot t_{\text{out}}$ 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. Since, by Lemma 1, $t_{\text{in}} \geq t_{\text{out}}$, the running time is $O(t_{\text{in}}^3 \cdot \log(t_{\text{in}}))$. □

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

Theorem 1. k-Anonymity *can be solved in* $O(nm + 2^{t_{in}t_{out}} t_{in}(t_{out}m + t_{in}^2 \cdot \log(t_{in})))$ *time, and so, in* $O(nm + 2^{t_{in}^2} t_{in}^2(m + t_{in} \log(t_{in})))$ *time.*

We remark that the described algorithm can be modified to use *domain generalization hierarchies* (DGH) [23] or to solve ℓ-Diversity [18]. We defer the details to a full version of the paper.

We end this section by comparing our algorithmic results to the closely related ones by Bonizzoni et al. [5]. They presented an algorithm for k-Anonymity with a running time of $O(2^{(|\Sigma|+1)^m} kmn^2)$ which 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 guess the output types like Bonizzoni et al. 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 using 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.

As we mentioned above, our algorithm can easily be modified to solve k-Anonymity using DGHs with no significant increase in the running time. Since in the DGH setting the alphabet size $|\Sigma|$ increases, it is not immediately clear how the algorithm due to Bonizzoni et al. can be modified to solve this problem without an exponential increase in the running time. Finally, when the parameters t_{in} and $(|\Sigma|, m)$ are constants, then Bonizzoni et al.'s algorithm runs in $O(kmn^2)$ time while our algorithm runs in linear time $O(mn)$.

3 Parameter t_{out}

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 [1,6,11]. The clustering view on k-Anonymity makes the number t_{out} of output types (corresponding to the number of clusters) of a k-anonymous matrix an interesting parameter.

There is also a more algorithmic motivation for investigating the (parameterized) complexity of k-Anonymity for the parameter t_{out}. As we saw in Theorem 1, k-Anonymity is fixed-parameter tractable for the parameter t_{in}. Due to Lemma 1, we know that t_{out} is a stronger parameter than t_{in}, in the sense that $t_{\text{out}} \leq t_{\text{in}}$. Hence, it is a natural question whether k-Anonymity is already fixed-parameter tractable with respect to the number of output types. Answering this 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". The hardness proof uses a polynomial-time many-one reduction from Balanced Complete Bipartite Subgraph:

Balanced Complete Bipartite Subgraph (BCBS)
Input: A bipartite graph $G = (V, E)$ and an integer $k \geq 1$.
Question: Is there a complete bipartite subgraph of G whose partition classes are of size at least k each?

BCBS is NP-complete [15]. 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 clearly 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. When the input graph is not balanced, that is, $|A| - |B| \neq 0$, 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 one vertex to A and make it adjacent to each vertex from B and add one vertex to B and make it adjacent to each vertex from A. If $k > |V|/4$, then add $k - |V|/4$ isolated vertices to each of A and B.

We devise a reduction from BCBS with balanced input graph and $k = |V|/4$ to show the NP-hardness of k-Anonymity.

Theorem 2. *k-Anonymity is NP-complete for two output row types and alphabet size four.*

Proof (Sketch). We only have to prove NP-hardness, since containment in NP is clear. Let $(G, n/4)$ be a BCBS-instance with $G = (V, E)$ being a balanced bipartite graph and $n := |V|$. 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.
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 2. 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] $j \leq$

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

	b_1	b_2	...	$b_{n/2-1}$	$b_{n/2}$
a_1	1	1	0_1	1	1
a_2	0_1	1	0_1	1	1
⋮	⋮	⋮	⋮	⋮	⋮
$a_{n/4}$	1	1	0_1	0_1	1
$a_{n/4+1}$	1	1	0_2	0_2	1
⋮	⋮	⋮	⋮	⋮	⋮
$a_{n/2}$	0_2	1	0_2	1	1
□	□	□	□	□	□
1	1	1	1	1	1

Fig. 2. Typical structure of the matrix D of the k-Anonymity instance obtained by the reduction from Balanced Complete Bipartite Subgraph.

$n/4$ and 0_2 if $j > n/4$. Additionally, there are two further rows, one containing only 1s and one containing only □-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; this part of the proof is deferred to a full version of the paper. □

Deconstructing intractability. In the remainder of this section we briefly discuss the NP-hardness proof for k-Anonymity in the spirit of "deconstructing intractability" [16,21]. 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) [5], 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 an XP-algorithm with respect to the combined parameter (t_{out}, m): 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 2:

Proposition 1. *k-Anonymity parameterized by (t_{out}, m) is in XP.*

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 also at most s. It follows that $t_{\text{in}} \leq t_{\text{out}} + s$. Now, fixed-parameter tractability follows from Theorem 1:

Proposition 2. k-Anonymity *is fixed-parameter tractable with respect to the combined parameter* (t_{out}, s).

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

4 Conclusion

This paper adopts a data-driven approach towards the design of (exact) algorithms for k-Anonymity and related problems. More specifically, the parameter t_{in} measures an easy-to-determine input property. Our central message here is that if t_{in} is small or even constant, then k-Anonymity and its related problems are efficiently solvable—for constant t_{in} even in linear time. On the contrary, already for two output types k-Anonymity becomes computationally intractable.

We contributed to a refined analysis of the computational complexity of k-Anonymity. The state of the art including several research challenges concerning natural parameterizations for k-Anonymity is surveyed in Table 1 in the introductory section. Besides running time improvements in general and open questions indicated in Table 1 such as the parameterized complexity of k-Anonymity for the combined parameter "number of suppressions plus alphabet size", it would also be interesting to determine whether our fixed-parameter tractability result for k-Anonymity with respect to the parameter t_{in} not only extends to the already mentioned generalizations of k-Anonymity but also to the (in terms of privacy) more restrictive t-Closeness problem [17].

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. G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proc. 10th ICDT*, volume 3363 of *LNCS*, pages 246–258. Springer, 2005.
3. 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.
4. P. Bonizzoni, G. DellaÂăVedova, and R. Dondi. Anonymizing binary and small tables is hard to approximate. *J. Comb. Optim.*, 22:97–119, 2011.
5. P. Bonizzoni, G. D. Vedova, R. Dondi, and Y. Pirola. Parameterized complexity of k-anonymity: Hardness and tractability. In *Proc. 21st IWOCA*, volume 6460 of *LNCS*, pages 242–255. Springer, 2010.
6. R. Bredereck, A. Nichterlein, R. Niedermeier, and G. Philip. Pattern-guided data anonymization and clustering. In *Proc. 36th MFCS*, LNCS. Springer, 2011. To appear.
7. V. T. Chakaravarthy, V. Pandit, and Y. Sabharwal. On the complexity of the k-anonymization problem. *CoRR*, abs/1004.4729, 2010.
8. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
9. C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54:86–95, 2011.
10. P. A. Evans, T. Wareham, and R. Chaytor. Fixed-parameter tractability of anonymizing data by suppressing entries. *J. Comb. Optim.*, 18(4):362–375, 2009.
11. A. M. Fard and K. Wang. An effective clustering approach to web query log anonymization. In *Proc. SECRYPT*, pages 109–119. SciTePress, 2010.
12. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
13. E. Fredkin. Trie memory. *Commun. ACM*, 3(9):490–499, 1960.
14. 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.
15. D. S. Johnson. The NP-completeness column: An ongoing guide. *J. Algorithms*, 8:438–448, 1987.
16. C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Deconstructing intractability– A multivariate complexity analysis of interval constrained coloring. *J. Discrete Algorithms*, 9:137–151, 2011.
17. N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. 23rd ICDE*, pages 106–115. IEEE, 2007.
18. A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1, 2007. 52 pages.
19. A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. 23rd PODS*, pages 223–228. ACM, 2004.
20. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
21. R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proc. 27th STACS*, volume 5 of *LIPIcs*, pages 17–32. IBFI Dagstuhl, 2010.
22. J. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proc. 20th STOC*, pages 377–387. ACM, 1988.
23. L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *IJUFKS*, 10(5):571–588, 2002.
24. L. Sweeney. k-anonymity: A model for protecting privacy. *IJUFKS*, 10(5):557–570, 2002.