A Fixed-Parameter Approach for Privacy-Protection with Global Recoding*

Katrin Casel

FB 4-Abteilung Informatikwissenschaften,
Universität Trier, 54286 Trier, Germany
casel@informatik.uni-trier.de

Abstract. This paper discusses a problem arising in the field of privacy-protection in statistical databases: Given a $n \times m$ $\{0,1\}$-matrix M, is there a set of mergings which transforms M into a zero matrix and only affects a bounded number of rows/columns. "Merging" here refers to combining adjacent lines with a component-wise logical AND. This kind transformation models a generalization on OLAP-cubes also called *global recoding*. Counting the number of affected lines presents a new measure of information-loss for this method. Parameterized by the number of affected lines k we introduce reduction rules and an $\mathcal{O}^*(2.618^k)$-algorithm for the new abstract combinatorial problem LMAL.

1 Introduction

With the steadily increasing amount of personal data collected for statistical research, privacy has become a matter of great federal and public interest. Statistical databases allow investigating personal records for empirical studies. This access however has to be restricted carefully to avoid disclosure of individual information. In the following, we approach this task considering parameterized complexity which was recently used for similar problems to obtain security in data-tables by entry-suppression [4,6].

In the following we discuss a method to secure access to confidential data via *OLAP-cubes*: A collection of individual records $R \subset I \times Q \times S$ characterized by x unique identifiers I, p numerical (or otherwise logically ordered) non-confidential attributes Q (*quasi-identifiers*) and numerical confidential attributes S is represented by a p-dimensional table T. Each dimension corresponds to one attribute and ranges over all of its possible values in their logical order. A cell of T with the label $(w_1, \ldots, w_p)$ contains the number of records in R with these characteristics, i.e.: $|\{r \in R \colon r_{x+i} = w_i \; \forall \; i = 1, \ldots, p\}|$. An example for this kind of representation is given below. Access to the confidential attribute(s) S is granted via queries on the non-confidential attributes, identifiers I are suppressed completely.

With the restriction to SUM- and COUNT- range-queries, the set of *even* range-queries (queries addressing an even number of cells) is the largest safe query-set [2]. The corresponding security-level is *l-compromise* [3], which is defined

* Partially funded by a PhD-scholarship of the DAAD.

J. Chen, J.E. Hopcroft, and J. Wang (Eds.): FAW 2014, LNCS 8497, pp. 25–35, 2014.
© Springer International Publishing Switzerland 2014

similarly to *k-anonymity* [10] for data-tables but does not directly suffer from the diversity-problem [9]. Unfortunately, this method is only safe for databases in which each cell of the OLAP-cube contains at least one record. To avoid this problem without major information-loss, the database can be altered by combining attribute-ranges as proposed in [1].

Table 1. Collection of records and its OLAP-cube representation

name	*age*	*education*	*salary*
Adam	48	Master	2500
Dave	44	College	1500
Keith	32	Bachelor	2000
Norah	34	College	1000
Tracy	48	Master	3000

→

$_{education}\backslash^{age}$	32	34	39	44	48
College	0	1	0	1	0
Bachelor	1	0	0	0	0
Master	0	0	0	0	2

Throughout this paper, we only consider the two-dimensional version of the resulting abstract problem of transforming a Boolean matrix into a zero matrix by merging adjacent rows/columns. In this abstraction, a two-dimensional OLAP-cube is represented by a matrix $M \in \{0,1\}^{n \times m}$ where empty cells in the cube correspond to one- and non-empty cells to zero-entries. The n rows and m columns of M will always be denoted by $r_1, \ldots, r_n$ and $c_1, \ldots, c_m$, respectively. Further, since there are many statements that apply symmetrically to a row or a column, the term *line* is used to refer to both. For $M[i,j] = 1$, a one-entry in $M[i-1,j], M[i+1,j], M[i,j-1]$ or $M[i,j+1]$ is called a *neighbour*. A one-entry is called *isolated*, if it has no neighbours.

The term *merging* will be used to express the transformation of replacing two adjacent lines l_1, l_2 of M by one line l, computed by $l[i] = l_1[i] \cdot l_2[i]$. This operation, in the following also described by the term (l_1, l_2), can be seen as performing a component-wise logical AND on two adjacent lines which models combining ranges in the OLAP-cube where the resulting combined cell is empty if and only if both participating original cells are empty. This operation is commutative which allows writing the shortened term $(l_1, \ldots, l_r)$ instead of the merging-set $\{(l_1, l_2), (l_2, l_3), \ldots, (l_{r-1}, l_r)\}$.

As long as M contains at least one zero-entry, the set of all possible mergings always translates M into a zero matrix. With the original objective of minimizing information-loss, this transformation is not very reasonable. Minimizing the number of merging-operations was already discussed in [1] and [7]. A different way to measure information-loss is considering each altered original line as "lost". The idea behind this new measurement is illustrated in the example below. This objective yields the following abstract problem:

LINE-MERGING MINIMIZING AFFECTED LINES (LMAL)
Input: $M \in \{0,1\}^{n \times m}$, $k \in \mathbb{N}$.
Question: Is there a set S of operations to transform M into a zero matrix with $|\{l_i \colon ((l_{i-1}, l_i) \in S) \vee ((l_i, l_{i+1}) \in S)\}| \leq k$?

A solution for LMAL can also be described by the set of affected lines. A set of lines L will be called *feasible*, if $(l_{i-1} \in L) \vee (l_{i+1} \in L)\ \forall\ l_i \in L$. A feasible set of lines L is a solution for a LMAL instance (M, k), if the merging-set $S = \{(l_i, l_{i+1}) : l_i, l_{i+1} \in L\}$ transforms M into a zero matrix.

Example 1. Consider the following OLAP-cube with the non-confidential attributes *age* and *education*:

$education \backslash age$	30	31	32	33	34	35	36	37	38	39
None	8	7	9	4	2	0	1	0	0	0
High-School	5	6	4	2	2	1	0	1	1	1
College	4	5	7	10	3	0	2	1	0	0
Bachelor	2	2	7	6	2	1	0	0	1	1
Master	3	3	5	4	6	0	1	0	2	1
PhD	1	2	6	8	7	2	0	1	0	0

A solution with the smallest number of mergings is $\{(r_1, r_2), (r_3, r_4), (r_5, r_6)\}$ which alters 6 original ranges. This solution creates a table with only three ranges for the attribute *age*:

$education \backslash age$	30	31	32	33	34	35	36	37	38	39
None or High-School	12	13	13	6	4	1	1	1	1	1
College or Bachelor	6	7	14	16	5	1	2	1	1	1
Master or PhD	4	5	11	12	14	2	1	1	2	1

Minimizing affected lines, an optimal solution would be merging $(c_6, \ldots, c_{10})$ which requires 4 operations but affects only 5 lines. This solution combines the dense last columns and creates the more balanced table:

$education \backslash age$	30	31	32	33	34	35–59
None	8	7	9	4	2	1
High-School	5	6	4	2	2	4
College	4	5	7	10	3	3
Bachelor	2	2	7	6	2	3
Master	3	3	5	4	6	4
PhD	1	2	6	8	7	3

The new measure for information-loss prefers neighbouring operations which seems more suitable for distributions in which empty cells tend to accumulate.

In the following we study the new problem LMAL. Section 2 discusses its complexity and rules for kernelization. Section 3 derives the parameterized algorithm with time-complexity in $\mathcal{O}^*(2.618^k)$.

2 Complexity and Reduction Rules

The previous problem-variation with the objective to minimize the number of mergings was already identified as NP-complete [7]. The new measure for information-loss does not seem to simplify this problem:

Theorem 1. *The decision-problem variation of LMAL is NP-complete.*

Proof. Membership in NP is easily seen by nondeterministically guessing k operations and checking the resulting matrix for one-entries. Reduction from vertex cover, as one of Karp's famous 21 NP-complete problems [8], proves hardness. Let (G, k) be a vertex cover instance and let $v_1, \dots, v_n$ be the nodes and $e_1, \dots, e_m$ the edges of G. If $k \geq n$ ((G, k) is a trivial "yes"-instance) return a zero matrix as a trivial "yes"-instance for LMAL. If $k < n$, the following construction yields a matrix $M \in \{0,1\}^{(6n+2m)\times(3n+5m)}$ for which $(M, 2n+2m+k)$ is a "yes"-instance for LMAL if and only if (G, k) is a "yes"-instance:

Each node v_i of G is represented by three rows $r_i, \hat{r}_i, \tilde{r}_i$. Neighbouring in this order for all nodes, these build the first $3n$ rows of M. The rows $\hat{r}_i, \tilde{r}_i$ will always be altered by a minimal solution, merging r_i corresponds to v_i being in the vertex cover. Each edge e_j is represented by three columns c_j^1, c_j, c_j^2. Neighbouring in this order for all edges, these build the first $3m$ columns of M. The columns c_j^1, c_j^2 model the connection to the incident nodes. Since just one of these nodes has to be in a cover, c_j will be forced to merge with one of its neighbours c_j^1, c_j^2 leaving just one of them to invoke row-mergings. The remaining $3n+2m$ rows will be denoted by $h_1^r, \dots, h_{3n+2m}^r$ and used to force the columns c_j to be altered in every minimal optimal solution. The remaining $3n+2m$ columns $h_1^c, \dots, h_{3n+2m}^c$ similarly trigger the choice of the rows $\hat{r}_i, \tilde{r}_i$.

Each edge $e_j = (v_i, v_l)$ (undirected but incident nodes considered given in an arbitrary, fixed order) induces a one-entry in the corresponding column c_j^1 in row r_i representing v_i, and another in column c_j^2 and row r_l. These entries will be called edge-induced. To force the merging of c_j with one of its neighbours, this column has one-entries in all of the $3n + 2m$ additional rows. If c_j is not merged, all of these entries have to be eliminated by row-merging which requires $3n + 2m > 2n + 2m + k$ operations, hence exceeding the optimal solution. The rows $\hat{r}_i$ have one-entries in h_{2j-1}^c, $\tilde{r}_i$ in h_{2j}^c for $i = 1, \dots, n,\ j = 1, \dots, \lceil 3n/2 \rceil + m$. A solution that does not affect all of the rows $\hat{r}_i$ and $\tilde{r}_i$ has to merge all of the additional columns instead which again exceeds $2n + 2m + k$.

A vertex cover C for G can be translated to $L = \{\hat{r}_1, \dots, \hat{r}_n\} \cup \{\tilde{r}_1, \dots, \tilde{r}_n\} \cup \{c_1, \dots, c_m\} \cup \{r_i \colon v_i \in C\} \cup \{c_j^1 \colon (e_j = (v_{i_1}, v_{i_2})) \wedge (v_{i_1} \in C)\} \cup \{c_j^2 \colon (e_j = (v_{i_1}, v_{i_2})) \wedge (v_{i_1} \notin C)\}$. A feasible LMAL solution L for M on the other hand can be altered to a solution of the same size that deletes at least one of the edge-induced one-entries for every edge by row-merging implying that the set $\{v_i \colon r_i \in L\}$ is a vertex cover for C (observe that additionally merging c_j^1 with (c_j, c_j^2) only deletes a single one-entry and can hence be replaced by covering it by the corresponding row which, by construction, always has the merging-partner $(\hat{r}, \tilde{r})$). Since covering all the additional lines of M induces a cost of $2n + 2m$, a solution of size at least $2n + 2m + k$ contains at most k of the rows r_i. □

In the following we use the terms of parameterized complexity presented in [5]. A decision-problem can be considered *parameterized*, if its instances can be described as elements of $\Sigma^* \times \mathbb{N}$. A parameterized problem P is called *fixed parameter tractable*, if it can be solved in time $f(k)p(n)$ for every instance (I, k) of P, with $n = size(I)$ where p is an arbitrary polynomial and f an arbitrary function. An equivalent way to describe fixed parameter tractability is reduction to a *problem-kernel*, a procedure that reduces an instance (I, k) in time $p(size(I))$ to an equivalent instance (I', k') with $k', size(I') \leq f(k)$. LMAL can be interpreted as parameterized by the number of affected lines k assuming that the information-loss should be reasonably bounded.

A simple false assumption is the idea of reducing the given instance by deleting all lines without one-entries; even for two neighbouring "empty" lines, this reduction may alter the solution-size. Consider, for example, deleting the second row and column from the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The original matrix requires four lines to eliminate all one-entries, the reduced version can be solved with three. Counting affected lines, merging-operations with a line of zeros are not independent. Considering three neighbouring lines of zeros, the one in the middle however is never included in a minimal solution, which yields:

Reduction-Rule 1. Let (M, k) be a parameterized LMAL instance and r_{i-1}, r_i, r_{i+1} three neighbouring rows of zeros in M. Create the reduced matrix M' from M by deleting the row r_i. (M, k) is a "yes"-instance, if and only if (M', k) is.

Proof. For any set of affected lines L for a LMAL solution for M, consider the set $L' = \{r_j : r_j \in L, j < i\} \cup \{r_{j-1} : r_j \in L, j > i\}$. If L' contains the rows r_{i-1} or r_i without merging partners, delete them from L'. This creates a feasible set for M', since r_i is the only line possibly omitted from L. The operations among L' perform the same transformation on M' as L for M except for possible mergings with r_i. Since merging (r_{i-1}, r_i) or (r_i, r_{i+1}) in M does not delete any entries, the operations among L' build a solution for M' with $|L'| \leq |L|$.

Similarly, the set $L' = \{r_j : r_j \in L, j < i\} \cup \{r_{j+1} : r_j \in L, j > i\}$ (omitting r_{i-1} or r_{i+1} if they have no neighbours in L') translates affected lines of a solution for M' into affected lines of a solution for M, not increasing the size. □

Rows that contain more than k entries have to be affected by the solution, since deleting its entries would otherwise affect more than k columns. This observation is similar to *Buss'* rule for vertex cover; "deleting" these rows to reduce a given instance however is not as simple as deleting nodes of large degree from a graph since the feasibility-condition for the solution requires at least one neighbouring row. The following reduction-rule introduces one possible way to reduce the number of entries by marking a reduced row with an entry in an additional

column. Exchanging "row" for "column" gives the equivalent rule for columns with more than k entries, as the whole argumentation is obviously symmetrical.

Reduction-Rule 2. Let r_i be a row in M with more than k entries in a LMAL instance (M, k). Construct the reduced matrix M' from M by:

1. Delete all entries $M[i, j] = 1$ with $M[i-1, j] = M[i+1, j] = 0$.
2. Add two columns of zeros and the i-th unit-vector to the right border of M $(M \to [M|0|0|e_i])$.

With these transformations, (M, k) is a "yes"-instance, if and only if (M', k) is.

Proof. Let S be a solution for M that affects at most k lines. Since r_i has more than k entries, S has to at least merge either (r_{i-1}, r_i) or (r_i, r_{i+1}). Each of these operations delete the new one-entry in the last column of M'. With all other one-entries copied from M, S is a solution for M'.

A solution S for M' that merges the new columns, on the other hand, can be altered to a solution S' that affects the same number of lines and merges either (r_{i-1}, r_i) or (r_i, r_{i+1}) instead (observe that they both delete the only one-entry possibly deleted by column-merging the last column of M'). Each of these row-operations also deletes all of the entries of M which were omitted in the second step of creating M' which implies that S' is a solution for M. □

These rules however can only yield a quadratic kernel, since even if no line contains more than k entries, the matrix-size can still be non-linear in k. Consider, for example, for a fixed $k = 4h \in \mathbb{N}$ a matrix $A \in \{0,1\}^{(k-2)\times 6kh}$ with $A[4i+1, 3ki+3j-2] = A[4i+2, 3k(i+h)+3j-2] = 1 \;\forall\; i = 0, \ldots, h-1 \;\; j = 1, \ldots, k$.

For $k = 4$ (zero-entries denoted by $\cdot$): $\begin{pmatrix} 1\cdot\cdot1\cdot\cdot1\cdot\cdot1\cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot \\ \cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot1\cdot\cdot1\cdot\cdot1\cdot\cdot1\cdot\cdot \end{pmatrix}$

Consider $M := \begin{pmatrix} A & \mathcal{O} \\ \mathcal{O} & A^T \end{pmatrix}$ with appropriately sized zero matrices $\mathcal{O}$. An optimal solution merges $(r_{4i+1}, r_{4i+2}), (c_{6kh+4i+1}, c_{6kh+4i+2}) \;\forall\; i = 0, \ldots, h-1$, which alters k lines. No line contains more than k entries and M does not contain three neighbouring empty lines which means that none of the above reductions is applicable. With $n = m = \frac{3}{2}k^2 + k - 2$ the matrix-size is quadratic in k.

3 Parameterized Algorithm

Parameterized by the number of merged lines, branching on a single one-entry by choosing either its row or its column already provides an approach in $\mathcal{O}^*(2^k)$. Branching on both possibilities to pick a partner-line yields feasibility and preserves this running-time. Branches that consider adding one of the non-existing lines $r_0, r_{n+1}, c_0, c_{m+1}$ are to be omitted throughout this section.

Theorem 2. *Parameterized LMAL restricted to instances without neighbouring one-entries can be solved in $\mathcal{O}^*(2^k)$.*

Proof. Starting with $L = \emptyset$, compute the set of affected lines by:

while$((|L| < k) \wedge (\exists\ M[i,j] = 1 \colon (r_i \notin L) \wedge (c_j \notin L)))$ branch into:

1. if$((r_{i+1} \in L) \vee (r_{i-1} \in L))$ $L = L \cup r_i$
   else branch:
   (a) $L = L \cup \{r_{i-1}, r_i\}$
   (b) $L = L \cup \{r_i, r_{i+1}\}$
2. if$((c_{j+1} \in L) \vee (c_{j-1} \in L))$ $L = L \cup c_j$
   else branch:
   (a) $L = L \cup \{c_{j-1}, c_j\}$
   (b) $L = L \cup \{c_j, c_{j+1}\}$

These choices of lines arise from the obvious conditions for any feasible solution: **1.** For each one-entry, either its row or column has to be included in the solution. **2.** Lines in L always have a merging-partner $(r_i \in L \Rightarrow (r_{i+1} \in L) \vee (r_{i-1} \in L))$. Branching into all sets of lines that cover a one-entry satisfying these conditions yields all possibilities to compute a solution for M.

Since an isolated one-entry is eliminated by any adjacent operation, a feasible set of lines covering all one-entries transforms M into a zero matrix. With this property, any set that satisfies the conditions above gives a solution for a matrix without neighbouring one-entries. All branching-options produce a recursion solved by $T(k) = 2^k$. □

The construction above benefits from the fact that the selected lines always merge to a vector of zeros. Unfortunately, neighbouring one-entries create exceptions which do not allow for this argumentation. The following example illustrates the problems created by general instances:

Example 2. Consider the following matrix (zero-entries omitted):

	c_1	$\cdots$	c_s	$\cdots$	c_j		$\cdots$	c_l		$\cdots$	c_t	$\cdots$	c_m
r_1								1					
r_p					1								
r_{i-2}			1		1	1							
r_i					1	1					1		
r_{i+2}								1	1				1
r_q	1							1	1				
r_n						1							

With $p, s > 2$, $|p-i|, |q-i| > 3$ and $|n-q|, |m-t|, |t-l|, |l-j|, |j-s| > 2$, *the* optimal solution is the set $\{r_{i-2}, r_{i-1}, r_i, r_{i+1}, r_{i+2}, c_j, c_{j+1}, c_l, c_{l+1}\}$. Consider running the algorithm from the proof to theorem 2 for $k=9$. Choosing the one-entry $M[i-2, s]$ in the first step, the only branch that arrives at the correct "yes"-response is the one that adds the set $\{r_{i-2}, r_{i-1}\}$ (all other branches contain lines that are not included in the optimal solution).

Branching on the one-entries $M[i+2,1]$, $M[p,j]$ and $M[1,l]$ in the following three steps always leaves just one possible successful branch as well and yields the partial solution $\{r_{i-2}, r_{i-1}, r_{i+1}, r_{i+2}, c_j, c_{j+1}, c_l, c_{l+1}\}$. This set of lines contains at least the row or the column of each one-entry and the algorithm terminates. For the remaining one-entries, both row and column are already contained in the partial solution but only the choice of the row r_i gives the optimal solution. Adding a row of zeros between r_i and r_{i+1} and deleting the columns $c_{j+2}, \ldots, c_{l-2}$ creates a matrix in which the algorithm arrives at a similar partial solution and the only optimal solution is created by adding c_{j+2}.

Larger matrices built from these structures can create instances with an arbitrarily difficult optimal set to complete an unfortunate partial solution. Efficiently computing an optimal additional set turns out to be a non-trivial task. For a parameterized approach for general instances we introduce a different branching-rule and further use the following sub-problem:

ROW-MERGING MINIMIZING AFFECTED LINES (RMAL)
Input: $M \in \{0,1\}^{n \times m}$, $k \in \mathbb{N}$.
Question: Is there a set of row-mergings that transforms M into a zero matrix and affects at most k rows?

Theorem 3. *RMAL can be solved in linear time.*

Proof. The following construction yields a solution represented by the set of affected lines:

1. Start with $L = \{r_i\colon \exists\ 1 \leq p \leq m\colon M[i,p] = 1\}$.
2. Collect the rows with one-entries that are not eliminated by the operations $S := \{(r_i, r_{i+1})\colon r_i, r_{i+1} \in L\}$ in a new set R. Rows $r_i \in R$ are characterized by:

   $$(r_i \in L) \wedge (r_{i+1} \notin L) \wedge (\exists p\colon\ \forall \min\{j\colon r_t \in L\ \forall\ j \leq t \leq i\} \leq h \leq i\ M[h,p] = 1).$$

   These rows require another merging partner outside the current set L.
3. While $R \neq \emptyset$, set $i = \min\{1 \leq j \leq n\colon r_j \in R\}$, $R = R \setminus r_i$ and:
   (a) If $i = n$, add r_{n-1} to L.
   (b) If $i < n$, add r_{i+1} to L and delete other rows possibly covered by r_{i+1}: If $r_{i+2} \in L$ and $r_h \in R$ with $h = \max\{1 \leq j \leq n\colon\ \forall\ i+2 \leq t \leq j\ r_t \in L\}$, set $R = R \setminus r_h$.

Correctness of the (possibly not feasible) starting-set follows from the simple fact that every non-empty row has to be altered. Each row collected in R is either non-empty and without merging-partner in L ($\min\{j\colon r_t \in L\ \forall\ j \leq t \leq i\} = i$) or the row of largest index in a group that does not merge to an empty row. Transforming L into a feasible solution requires at least one row outside L as merging-partner for each row in R. Since the rows outside L are empty, any partner outside L suffices to delete all one-entries and produces an additional cost of one. With this observation, each set that contains L and at least one partner outside L for each row in R yields a feasible set that deletes all one-entries in M. Step 3 creates a set with this property.

Minimality of the constructed solution is seen as follows: Each row of zeros picked for the solution can be used to cover at most two rows in R. Always fixing the row of lowest index exploits the fact that choosing its upper neighbour can not cover another row in R. □

The following final algorithm first computes a partial solution which contains the row of each not-deleted one-entry. This property allows a more efficient way of expanding it to a solution: Branching for one-entries not deleted by the partial solution, the options for row-merging can be postponed. Using the *easy* subproblem RMAL as a final polynomial-time step, the "book-keeping" introduced in [7] allows pre-counting the cost of the postponed row-mergings to improve this algorithm.

Theorem 4. *Parameterized LMAL can be solved in* $\mathcal{O}^*(2.618^k)$.

Proof. The following three steps compute a solution L (set of affected lines):

Step 1: Starting with $L = \emptyset$, compute a feasible partial solution:

while$(\exists M[t,j] = 1 \colon (r_t \notin L) \wedge (c_j \notin L) \wedge (|L| < k))$

Select a row $r_i \notin L$ with $i = argmax\{C(i,j) \colon 1 \leq i \leq n, r_i \notin L\}$ where $C(i,j)$ counts the neighbouring one-entries around c_j:
$C(i,j) = \max\{u \colon M[i,h] = 1 \ \forall \ j \leq h \leq u\} - \min\{o \colon M[i,h] = 1 \ \forall \ o \leq h \leq j\} + 1$

Let $c_o, \ldots, c_u$ be the $C(i,j)$ columns that contain the one-entries in r_i around $M[i,j]$. Branch into:

1. if$((r_{i+1} \in L) \vee (r_{i-1} \in L))$ $L = L \cup r_i$
   else, branch:
   (a) $L = L \cup \{r_{i-1}, r_i\}$
   (b) $L = L \cup \{r_i, r_{i+1}\}$

2. if$((c_{u+1} \in L) \vee (c_{o-1} \in L))$ $L = L \cup \{c_o, \ldots, c_u\}$
   else, branch:
   (a) $L = L \cup \{c_{o-1}, \ldots, c_u\}$
   (b) $L = L \cup \{c_o, \ldots, c_{u+1}\}$

If the corresponding operations already transform M into a zero matrix, L is a feasible solution. Else expand the partial solution with steps 2 and 3.

Step 2: Choose additional columns for L and save row-merging for step 3:

With $B_l(j) := \min\{s \colon c_s, \ldots, c_j \in L\}$, $B_r(j) := \max\{t \colon c_j, \ldots, c_t \in L\}$ for $c_j \in L$ and $B_l(j) := B_r(j) := j$ for $c_j \notin L$, the set I of one-entries which are not deleted by the operations corresponding to L can be characterized by:

$I = \{(o,u,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\} \times \{1,\ldots,m\} \colon r_o, \ldots, r_u \in L,\ r_{o-1}, r_{u+1} \notin L$ and $M[h,s] = 1 \ \forall \ o \leq h \leq u,\ B_l(j) \leq s \leq B_r(j) = j\}$

Branch into either row- or column-merging for these one-entries and save row-merging for a final polynomial-step by collecting rows in a set R (initially $R=\emptyset$):

while$((I \neq \emptyset) \wedge (|L| + \frac{1}{2}|R| < k))$

Choose the column c_j with $j = \min\{t\colon \exists o, u\colon (o,u,t) \in I\}$ and branch into:

1. Column-merging:
   if$(c_j \in L)$
   if$(j < n)$ $L = L \cup c_{j+1}$
   if$(1 < B_l(j) < j = n)$ $L = L \cup c_{B_l(j)-1}$
   if$(c_j \notin L)$
   if$(c_{j-1}, c_{j+1} \notin L)$
   if$(j < n)$ $L = L \cup \{c_j, c_{j+1}\}$
   if$(1 < j = n)$ $L = L \cup \{c_{j-1}, c_j\}$
2. Row-merging: $R = R \cup \{(o,u)\colon (o,u,j) \in I\}$

Recalculate I for the new set L omitting one-entries (o,u,j) with $(o,u) \in R$.

Step 3: Compute additional rows with RMAL: Choose merging-partner for the set $\{r_u\colon \exists\, o,j\colon (o,u,j) \in R\}$ with step 3 of the procedure used for theorem 3.

Since the branching-step considers all minimal, feasible possibilities to delete the one-entries $M[i,o], \ldots, M[i,u]$, any solution for M contains at least one of the sets constructed by the first step. All branches of the first step give a recursion with a running-time in $\mathcal{O}^*(2^k)$. The particular choice of the branching-row $i = argmax\{C(i,j)\colon 1 \leq i \leq n, r_i \notin L\}$ in step 1 produces a partial solution S that contains the row of every one-entry that is not eliminated: Suppose a one-entry $M[t,j]$ is not deleted by S with $r_t \notin S$. Since S contains at least row or column for each one-entry, c_j is contained in S. Let r_i be the branching-row that added c_j to S among, w.l.o.g, $c_{o-1}, \ldots, c_u$. Since S applies the operation $(c_{o-1}, \ldots, c_u)$ and does not delete the one-entry $M[t,j]$, the row r_t contains the one-entries $M[t,o-1] = \ldots = M[t,u] = 1$ which would give $C(t,j) \geq u-o+2 = C(i,j)+1$, a contradiction to the choice of i.

One-entries not deleted by S are relics of neighbouring one-entries in rows that are contained in L which justifies the characterization of I used in step 2. This property further implies that rows outside L are transformed into rows of zeros by the column-mergings in L. This property is the crucial difference to the partial solution from the procedure for theorem 2. Since any additional row that merges with $r_o, \ldots, r_u$ deletes all remaining one-entries in these rows omitting one-entries (o,u,j) with (o,u) already chosen to be merged by the final polynomial step is justified and pre-counting the cost for each row in R is valid. Since each row added in the final step can cover at least two groups of rows, step 3 induces a cost of at least $\frac{1}{2}|R|$. Since the second branch increases $|R|$ by at least one, this book-keeping yields the recursion $T(k) \leq T(k-1/2) + T(k-1)$ for each branching-step which gives the stated running-time.

Always choosing the one-entry of lowest column-index allows to reduce the options in the branch that treats this one-entry by column-operations to merging

it with the *right* neighbouring column (unless c_j is the last column of M). Like in the procedure for RMAL, choosing the left neighbour does not delete other one-entries and can be replaced by choosing the right neighbour instead. With the properties of L from step 1, the set I always contains *all* one-entries that are not deleted by adding at least one row for each row in R to the current partial solution. This implies that the columns c_h with $h < j$ and especially the possible left merging-partner are already empty. The cases in branch 1 consider all possibilities to add the right neighbour preserving feasibility. □

4 Conclusion

This paper studied an abstract problem from the field of privacy-protection in statistical databases with global recoding. We studied the new problem LMAL considering parameterized complexity and presented reduction rules and a branching-algorithm based on the *easy* sub-problem RMAL.

Considering the motivating background, a generalization for matrices of higher dimension would be very interesting. While the ideas for solving the easy sub-problem remain applicable, a translation for the general parameterized algorithm seems to be more difficult. The model introduced here so far only suffices *1-compromise*. Another interesting generalization would be a generalization for *k-compromise* for larger values of k.

References

1. Branković, L., Fernau, H.: Approximability of a $\{0,1\}$-matrix problem. In: Proceedings of the 16th Australasian Workshop on Combinatorial Algorithms, pp. 39–45 (2005)
2. Branković, L., Horak, P., Miller, M.: A Combinatorial Problem in Database Security. Discrete Applied Mathematics 91(1-3), 119–126 (1999)
3. Branković, L., Giggins, H.: Security of Statistical Databases. In: Petkovic, M., Jonker, W. (eds.) Security, Privacy and Trust in Modern Data Management, pp. 167–182. Springer (2007)
4. Bredereck, R., Nichterlein, A., Niedermeier, R.: Pattern-Guided k-Anonymity. In: Fellows, M., Tan, X., Zhu, B. (eds.) FAW-AAIM 2013. LNCS, vol. 7924, pp. 350–361. Springer, Heidelberg (2013)
5. Downey, R., Fellows, M.: Parameterized Complexity. Springer (1999)
6. Evans, P., Wareham, T., Chaytor, R.: Fixed-parameter tractability of anonymizing data by suppressing entries. Journal of Combinatorial Optimization 18(4), 362–375 (2009)
7. Fernau, H.: Complexity of a $\{0,1\}$-matrix problem. The Australasian Journal of Combinatorics 29, 273–300 (2004)
8. Karp, R.: Reducibility among Combinatorial Problems. Complexity of Computer Computations, 85–103 (1972)
9. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.:l-Diversity: Privacy Beyond k-Anonymity. ACM Transactions on Knowledge Discovery from Data 1(1) (2007)
10. Samarati, P.: Protecting Respondents' Identities in Microdata Release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)