Approximability of a $\{0,1\}$-matrix problem

Ljiljana Branković [1*]
Henning Fernau [1,2,3]

[1] The University of Newcastle
School of Electrical Engineering and Computer Science
University Drive, NSW 2308 Callaghan, Australia
{lbrankov,fernau}@cs.newcastle.edu.au

[2] University of Hertfordshire, Computer Science,
College Lane, Hatfield, Herts AL10 9AB, UK
h.fernau@herts.ac.uk

[3] Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Sand 13, D-72076 Tübingen, Germany
fernau@informatik.uni-tuebingen.de

Abstract

We consider the following combinatorial problem: given an $n \times m$ $\{0,1\}$-matrix M, find a minimum cardinality set S of mergings between neighboring rows or columns that yields an all-zeros matrix. Here, merging means performing a component-wise AND operation. We prove that this NP-hard minimization problem is factor-2-approximable by relating it to the VERTEX COVER problem on bipartite graphs.

1 Introduction

The combinatorial problem that we study in this paper comes from the area of statistical database security. For the benefit of a reader, we first present the problem from the database security point of view, and we then provide a formal mathematical definition of it.

Statistical database security is concerned with protecting confidentiality of individual data values that are used for statistical purposes. With the recent expansion of computer and in particular networking technologies, huge volumes of personal data are regularly collected by companies, hospitals, government departments and various organizations. These include medical, criminal, shopping and taxation records, to mention just a few. At the same time, public is becoming increasingly concerned about confidentiality of such personal data. Statistical database security focuses on protecting the confidentiality, while at the same time making data available for statistical research.

For the purpose of this study, a database can be seen as a multidimensional matrix where each dimension corresponds to a particular attribute (property) of the records stored in the database. For example, a two dimensional database M might have attributes "Age" and "Years

*This research was supported by ARC Discovery Project grant DP0452182.

of service", in addition to a confidential attribute "Salary". The element m_{ij} of M in row i and column j contains the number of all employees who are i years old and who have j years of service in the company. We refer to elements of M which are equal to 0 as *holes*, those that are equal to 1 as *single elements* and those that are greater than 1 as *multielements*. A range query is defined as a contiguous submatrix of M. Each range query provides not only the total number of records contained in the elements of the corresponding contiguous submatrix, but also the sum of "Salary" (confidential attribute), in all such elements. The database M is said to be *compromised* if there exists a linear combination of range queries that contains precisely one record, in which case the value of the confidential attribute in that record can be learned by an unauthorized party. Our goal is to prevent a database compromise, while at the same time allow as many range queries as possible. The problem is further complicated by the so-called *supplementary knowledge* that assists intruders in compromising the database [10].

If all the elements in an m-dimensional database contain at least one record, that is, if the database contains no holes, then the fraction of queries that can be answered without exact disclosure of any individual value is bounded from below by $(2^m - 1)/2^m$ [9]. The minimum is achieved in the case where there is precisely one record for each combination of attribute values, that is, when a database matrix contains only ones (or single elements). A precise formula for the usability of one-dimensional databases with arbitrary elements in the database matrix is given in [3], and databases with holes were considered in [11]. In this paper we explore merging neighboring rows or columns in order to eliminate all elements that contain holes. Once we have arrived at a database without holes, we know from [9] that we can keep the data secure while achieving very high usability by allowing only "even" range queries, that is, those queries that are represented by contiguous sub-matrices with even number of single elements. Clearly, our goal in the preprocessing is to minimize the number of row or column mergings necessary for eliminating all holes in the matrix.

How does a matrix in our example look like after this preprocessing ? Rows will now carry interval labels $[i_1, i_2]$. This means that the corresponding matrix entries report all employees of age between i_1 and i_2 years. Similarly, columns will carry interval labels $[j_1, j_2]$. Range queries of the form: "How many people are employed that are aged between i and i' ?" can only be exactly answered if there are numbers $i_1 < i_2 < \ldots < i_\ell$ such that $i = i_1$, $i' = i_\ell$, and $[i_1, i_2]$, $[i_2 + 1, i_3]$, $\ldots$, $[i_{\ell-1} + 1, i_\ell]$ are interval labels of consecutive rows in the preprocessed matrix. (Similar comments apply to the columns.) Other range queries can only be answered approximately.

In this paper, we restrict the problem to two dimensions and we ask the following question: *What is the minimum number of neighboring rows and columns that we have to merge in order to eliminate all elements that contain holes?* We now present the mathematical formulation of the problem. Note that in our formulation, zeros in the database matrix are represented by 1 while positive integers are represented by 0, to be consistent with [7].

Minimum Matrix Row/Column Merging MRCM:

Instance: $n \times m$ $\{0,1\}$-matrix M
Solution: A set S of mergings between neighboring rows or columns such that the resulting matrix contains all zeros. Here, *merging* means performing a component-wise logical AND.
Measure: Cardinality $|S|$ of the set S.

This problem was shown to be NP-hard and fixed-parameter tractable [7].[1] In this paper we show that MRCM is factor-2-approximable, this way improving on the 3-approximability of MRCM as shown in [7] by using a different approach.

[1]Refer to [2, 6, 8] for the basic notions of approximability, parameterized and classical complexity, respectively.

This means that, given an instance M of MRCM, we can find in polynomial time a solution S (i.e., a set of merging operations) such that $|S|/|S_{opt}| \leq 2$, where S_{opt} is a minimum solution of the instance M (i.e., minimal in terms of cardinality of the solution). Notice that, since MRCM is NP-hard, it is not expected that we can find a (deterministic) polynomial-time algorithm that optimally solves any given instance.

2 Approximation algorithm

A given database matrix $M \in \{0,1\}^{n \times m}$ can be interpreted as the "adjacency" matrix of a bipartite undirected graph B_M. In B_M, we have row vertices $r_1, \ldots, r_n$ and column vertices $c_1, \ldots, c_m$, where r_i and c_j are connected by an edge if and only if $M[i,j] = 1$. Let us associate $\{r_i, r_{i+1}\}$ to the operation $\mathrm{RM}(i, i+1)$ of merging rows i and $i+1$, and similarly $\{c_i, c_{i+1}\}$ to the operation $\mathrm{CM}(i, i+1)$ of merging columns i and $i+1$. Let $A(S)$ denote the set of vertices of B_M associated to a set S of merging operations. Then, we can show:

Lemma 1: Let M be an instance of MRCM and let S be a solution to this instance. Then, $A(S)$ is a vertex cover set of B_M with $|A(S)| \leq 2|S|$.

Proof: To show that $A(S)$ is a vertex cover of B_M we only need to show that all ones in M are contained in the set $A(S)$ of rows and columns. Indeed, if there were a one outside of these rows and columns, then that one would remain one even after all merging operations from S are performed and thus S would not be a solution of M. Thus every edge in B_M is adjacent to a vertex in $A(S)$ and $A(S)$ is a vertex cover of B_M. To see that $|A(S)| \leq 2|S|$, recall that each of the merging operations from S is associated with a set of two vertices from B_M. As $A(S)$ is the union of all $|S|$ such sets, it clearly has the cardinality at most $2|S|$. □

Interestingly, the converse of Lemma 1 is not true in general: A vertex cover of B_M does not necessarily give a solution for MRCM. For example, if B_M is a complete bipartite graph, the set of all the row vertices is a vertex cover of B_M; on the other hand, M is the all-ones matrix and as such does not have a solution at all. More generally, if B_M contains a vertex in, say, the row-partition which is adjacent to all the vertices in the column-partition, then the column-partition is a vertex cover of B_M while the intuitively "equivalent" merging of all columns is not a solution to the corresponding MRCM-instance M. However, in all the other cases a vertex cover of B_M can be converted to a solution of MRCM as we shall see in what follows.

The algorithmic significance of Lemma 1 is that VERTEX COVER can be solved in polynomial time on bipartite graphs by maximum matching technique and we shall use this fact to construct a factor-2-approximation algorithm for MRCM.

Before we can present an algorithm, we need to introduce some auxiliary notation. Let M be an $(n \times m)$ $\{0,1\}$-matrix and let V^* be a minimal vertex cover of the associated bipartite graph B_M. We select a row vertex and a column vertex in B_M that are not in V^* and we denote them as r_R and c_C, respectively. If V^* contains all the row vertices or all the column vertices, we set $R = n$ and $C = m$.

We are now ready to propose an algorithm based on Lemma 1. The algorithm has 3 parts. In Part 0 we check whether the matrix M contains only ones, in which case the instance is not solvable and we are done. In Part 1 we process all the lines (rows and columns) of the matrix M which contain only ones. Note that we are building the set S of mergings and at the same time we are performing the actual mergings in the matrix M. Part 2 deals with a matrix that has no lines completely filled with ones. In this part we only build the set S but we do not perform any actual mergings in the matrix M.

For the sake of simplicity, throughout the algorithm we use integers (i.e., i) to denote the indices of the rows and columns in M. We shall later be much more precise and we shall denote a row obtained by merging rows r_i and r_{i+1} by $r_{i,i+1}$. In general, $r_{j\ldots k}$ would denote a row obtained by merging rows $r_j, r_{j+1}, \ldots, r_k$. Then merging rows $r_{j\ldots k}$ and $r_{l\ldots s}$, where $l = k+1$

gives the row $r_{j...s}$. When merging, we add $MR(k,l)$ to the set S. Also, when comparing indices $j...k$ and $l...s$ in Part 2 of the algorithm we say that $j...k < l...s$ if and only if $k < l$.

```
Given: {0,1}-matrix M.

//Part 0.
If M contains only ones, the instance is not solvable ⇝ abort.

//Part 1.
Initialize solution S := ∅.
While
   there is an all-ones line i
   whose neighboring line j contains a zero
do
   S := S ∪ {merge-lines(i, j)};
   merge lines i and j in M;
//now there is no line completely filled with ones
If M contains only zeros, then return S.

//Part 2.
Solve VERTEX COVER on B_M, yielding a minimum solution V*.
Determine integers R, C from B_M   (as described in the main text).
For each r_i ∈ V* do
  if i < R then S := S ∪ {merge-rows(i, i+1)}
           else S := S ∪ {merge-rows(i-1, i)};
For each c_i ∈ V* do
  if i < C then S := S ∪ {merge-columns(i, i+1)}
           else S := S ∪ {merge-columns(i-1, i)};
return S
```

To show the correctness of Part 1, observe that whenever a line completely filled with ones has two neighboring lines which contain a zero, then it does not matter which of the two lines we choose for merging, as the resulting matrices are the same in both cases. Thus that this kind of nondeterminism in the algorithm can be arbitrarily fixed.

We illustrate the above algorithm with the following example.

$$\mathbf{M}_k = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

As M does not contain only ones the algorithm proceeds to Part 1. Row 2 contains only ones and as such it is merged with a neighboring row, say row 3. Thus we have a new matrix

$$\begin{array}{c} \\ 1 \\ 23 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

and a set $S = \{\mathrm{RM}(2,3)\}$. After processing column 2 we get, for example, a matrix

$$\begin{array}{c} \\ 1 \\ 23 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 12 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

and $S = \{\mathrm{RM}(2,3), \mathrm{CM}(1,2)\}$.

Now there are no more lines that contain only ones and we proceed to Part 2 of the algorithm. We have a bipartite graph B_M where $V_{B_M} = \{r_1, r_{23}, r_4, c_{12}, c_3, c_4\}$ and $E_{V_M} = \{(r_1, c_3), (r_{23}, c_4), (r_4, c_{12})\}$, and a minimum vertex cover $V^* = \{r_1, r_4, c_3\}$. Then $r_R = r_{23}$ and there are two possibilities for c_C; we select, for example, $c_C = c_4$. We now process all the vertices in V^*.

r_1: As $1 < 23$ we have $S := S \cup \{\mathrm{RM}(1,2)\}$.

r_4: As $4 > 23$ we have $S := S \cup \{\mathrm{RM}(3,4)\}$.

c_3: As $3 < 4$ we have $S := S \cup \{\mathrm{CM}(3,4)\}$.

The algorithm returns

$$S = \{\mathrm{RM}(1,2), \mathrm{RM}(2,3), \mathrm{RM}(3,4), \mathrm{CM}(1,2), \mathrm{CM}(3,4)\}.$$

Alternatively, as solution to the VERTEX COVER instance B_M, we could have obtained $V^* = \{r_1, r_{23}, r_4\}$. Then, we select $r_R = r_4$ and $c_C = c_4$ (in fact, arbitrary in the column case). As the reader can verify,

$$S = \{\mathrm{RM}(1,2), \mathrm{RM}(2,3), \mathrm{RM}(3,4), \mathrm{CM}(1,2)\}$$

would then have been found, which is indeed optimal.

3 Main result

In this section we first prove that the above algorithm yields a feasible solution to MRCM and then that it is a factor-2-approximation algorithm.

Lemma 2: Let M be an instance of MRCM. Let S be obtained by the algorithm above. Then, S is a feasible solution to this instance.

Proof: We may assume that M contains at least one zero as otherwise MRCM has no solutions. Part 1 of the algorithm deals with all-ones lines. After Part 1 of the algorithm, M will either contain only zeros, in which case S is indeed a solution and our proof is complete, or M will contain some ones but no all-ones lines. We thus need to show that, for such a matrix M, Part 2 of the algorithm yields a solution S.

In order to prove this we introduce a partial ordering on the set of $(n \times m)$ $\{0,1\}$-matrices as follows. We say that $M' \leq M''$ if and only if $M'[i,j] = 1$ implies $M''[i,j] = 1$ for all $i \in [1,n]$ and all $j \in [1,m]$. Note that if $M' \leq M''$ and if S is a solution of the MRCM-instance M'', then S is also a solution to M'.

Consider now a matrix M and two neighboring lines i and j in M. We introduce some further notations. Let M_i be a matrix obtained from M by deleting line i. Let M_{ij} be a matrix obtained from M by merging lines i and j. Then $M_{ij} \leq M_i$ and $M_{ij} \leq M_j$.

We are now ready to complete our proof. Recall that V is a set of rows and columns of M that contain all the ones in M. Thus deleting all the rows and columns in V yields an all-zeros matrix. From the above discussion it follows that is we replace each deletion with a unique merging then the resulting matrix would also contain only zeros. If V does not contain all the rows or all the columns of B_M then the row r_R and the column c_C guarantee that no two deletions are associated with the same merging. In the case where V contains all the rows (or all the columns, respectively) of M, we have only $n-1$ ($m-1$, respectively) mergings and

n (m, respectively) deletions, but it is easy to see that the resulting matrix still contains only zeros, as the original matrix has no all-ones lines. Note that $n \geq 2$, $m \geq 2$ and thus $n - 1 \geq 1$ and $m - 1 \geq 1$, because a matrix with a single row or a single column would be completely processed in Part 0 or Part 1 of the algorithm as it either contains a line completely filled with ones or it is an all-zeros matrix. This completes the proof. □

Theorem: The sketched algorithm is a factor-2-approximation algorithm.

Proof: The first part of the algorithm deals with all-ones lines and optimally treats them, since a line completely filled with ones can never be converted into an all-zeros line only by using merges of "orthogonal lines," e.g., an all-ones row cannot be converted into an all-zeros row by using column-mergings only.

Moreover, merging with an all-ones line has exactly the same effect as deleting that line; thus, it does not matter which neighbor we merge it with. So we now assume that an instance M has no lines completely filled with ones.

Let S^* be a minimal solution of this MRCM-instance M. Then, $A(S^*)$ is a solution of the VERTEX COVER instance B_M. An optimal solution V^* to B_M surely satisfies

$$|V^*| \leq |A(S^*)| \leq 2|S^*|$$

according to Lemma 1. Conversely, Lemma 2 shows that the set of merging operations S obtained from V^* by the algorithm is indeed a feasible solution to the instance M. It is easily observed that

$$|S| \leq |V^*|.$$

Altogether, we can see that $|S| \leq 2|S^*|$. □

4 Conclusion

In this paper we have considered an NP-hard problem called MRCM that arises in the area of statistical database security. We presented an algorithm for solving this problem and we showed that it is a factor-2-approximation algorithm.

This approach is different from the typical *local ratio* approach for covering problems, see [4]. Rather, we use a related problem, namely VERTEX COVER, to solve MRCM. It would be interesting to see if the local ratio approach (or the likewise successful integer linear programming approach) can be used to obtain similar or maybe better approximation ratios for MRCM. Alternatively, it might be worthwhile looking into other kinds of reductions between approximation problems not only from the point of view of proving hardness results but also from an algorithmic perspective.

References

[1] N.Adam and J. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, M. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer, 1999.

[3] L. Branković, P. Horak, and M. Miller An optimization problem in statistical databases. *SIAM Journal on Discrete Mathematics*, 13(3):346–353, 2000.

[4] R. Bar-Yehuda. One for the price of two: a unified approach for approximating covering problems. *Algorithmica*, 27:131–144, 2000.

[5] D. Denning. *Cryptography and Data Security.* Addison-Wesley, 1982.

[6] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[7] H. Fernau. Complexity of a $\{0,1\}$-matrix problem. *The Australasian Journal of Combinatorics*, 29:273–300, 2004.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability.* New York: Freeman, 1979.

[9] P. Horak, L. Branković, and M. Miller. A combinatorial problem in database security. *Discrete Applied Mathematics*, 91(1-3):119–126, 1999.

[10] M. Miller. A model of statistical database compromise incorporating su pplementary knowledge. *Databases in the 1990's* (B.Srinivasan and J.Zeleznik ow (editors)), World Scientific, 97-113, 1991.

[11] L. Wang, Y. Li, D. Wijesekera, and S. Jajodia. Precisely answering multi-dimensional range queries without privacy breaches. Computer Security — European Symposium on Research in Computer Security ESORICS 2003, Proceedings, LNCS 2808, pp. 100–115, 2003.

[12] V. Estivill-Castro and L. Branković. Data swapping: balancing privacy against precision in mining for logic rules. Data Warehousing and Knowledge Discovery DaWaK'99, Proceedings, LNCS 1676, pp. 389–398, 1999.