Privacy Preserving and Data Mining in an On-Line Statistical Database of Additive Type

Francesco M. Malvestuto[1], Mauro Mezzini [1]

[1] Computer Science Department, "La Sapienza" University of Rome, Italy

Abstract. In an on-line statistical database, the query-answering system should prevent answers to statistical queries from leading to disclosure of confidential data. On the other hand, a statistical user is inclined to data mining, that is, to disclose pieces of information that are implicit in the (explicit) answers to his queries. A key task for both is to find data that is derivable from given summary statistics. We show that this task is easy if data is additive and the set of given summary statistics can be modelled by a graph.

1 Introduction

An on-line statistical database [1] is an ordinary database which contains information on individuals (persons, companies, organizations, et cetera) but its users are allowed to only access summary statistics over categories of individuals. For example, consider a bank database which contains a file called `DEPOSITOR` whose records have the following fields: `Name`, `Account`, `Gender`, `Age`, `Balance`. The statistical users can ask for summary statistics on `Balance` over arbitrary categories of depositors but the categories can be specified by logical formulae involving the fields `Gender` and `Age`, but not the private fields `Name` and `Account`. Typically, such summary statistics are obtained using the five aggregation functions: **sum**, **count**, **max**, **min**, **average**. If f is any of these aggregation functions, the following are three possible instances of a statistical query expressed in an SQL-like language

Q **select** f(`Balance`)
from `DEPOSITOR`
where `Gender` = `Male` and `Age` ≥ `25`

Q' **select** f(`Balance`)
from `DEPOSITOR`
where `Gender` = `Female` and `Age` ≥ `25`

Q'' **select** `Gender`, f(`Balance`)
from `DEPOSITOR`
where `Age` ≥ `25`

groupBy `Gender`

It should be noted that Q'' is equivalent to the couple $\{Q, Q'\}$; therefore, without loss of generality, we can limit our considerations to statistical queries from which the **groupBy** clause is missing.

According to the terminology introduced in [2], [3] the aggregation functions **sum** and **count** are called *additive*, the aggregation functions **min** and **max** are called *semiadditive*, and the aggregation function **average** is called *computed*. In this paper, as in [2], [3] we focus on the special class of additive aggregation functions that take on their values from a commutative group (e.g., the set of reals or the set of integers). In other words, we only consider *sum-queries*, which in our bank database are the statistical queries of the type **sum**(`Balance`). By the *value* of such a sum-query we mean exactly the total sum of the values of `Balance` reported in the records of the file `DEPOSITOR` that fall in the category specified by the logical formula contained in the **where** clause. Answering such sum-queries (and, more in general, statistical queries) raises concerns on the compromise of individual privacy and protection of confidential data should be afforded. We call *intrusive* a sum-query asking for total of a *sensitive statistic* [5], [11], [12]. Let Q be a sum-query of the type **sum**(`Balance`) on our bank database. If `Balance` is a confidential field and the value of Q is sensitive (e.g., according to the threshold criterion), then Q is intrusive. When an intrusive sum-query is asked, the query-answering system (QAS) should issue a "non-informative" response (see below). The statistical security of a database can also be attacked by a nonintrusive sum-query. In our bank database, this is the case if Q is not intrusive, but its value combined with the responses to previously answered sum-queries of the type **sum**(`Balance`) can lead to the disclosure of the total balance for some sensitive category of depositors. Then, we call Q *tricky* and the QAS should answer Q as if Q were intrusive. Finally, if a sum-query is neither intrusive nor tricky, the QAS can be safely answer it by releasing its value. The situation can be depicted as a competitive game played by the QAS, which has as its opponent a hypothetical user, henceforth referred to as the *data miner*, who at all times is well-informed of all answered sum-queries and is able to identify and compute the data that are *derivable* from their responses, that is, those data that are implicitly released. To beat the data miner, the QAS should control the amount of information released each time a new query is answered by *auditing* the whole set of answered sum-queries. More precisely, given a new sum-query Q, the auditing procedure should first find those data that are *derivable* from the responses to Q and to previously answered sum-queries; next, if no derivable data is sensitive, then the QAS can safely answer Q, otherwise in response to Q the QAS will issue a "non-informative" answer, e.g., the set of feasible values of Q consistent with the values of previously answered sum-queries. In order to find the data that are derivable from the values of a given set of sum-queries, one can exploit the additivity of the aggregation function **sum** and model the amount of information conveyed by their answers with a set of linear equations with a 0-1 coefficient matrix [10], [8]. In this paper, we address the derivability problem under the assumption that the coefficient matrix is the incidence matrix of a graph, we call the *query map*. This assumption corresponds to a query-overlap restriction, which is needed to make the auditing procedure feasible [10], [8] but is powerful enough to deal with two-

dimensional tables with arbitrary sets of suppressed cells [9]. We shall show that, for sum-queries whose values belong to a commutative group, the derivability problem can be solved in linear time thanks to a simple characterization of the so-called *invariant edges* of the query map.

Example 1. Consider again the file called DEPOSITOR, where Balance is a field of real type, and the value-set of the field Age consists of the following three intervals: Age < 25, 25 ≤ Age < 45, Age ≥ 45. We assume that Balance is a confidential field and that

Q select **sum**(Balance)
from DEPOSITOR
where Gender = Male and Age < 25

is the only intrusive sum-query. Consider the four sum-queries

Q_1 select **sum**(Balance)
from DEPOSITOR
where Gender = Male and Age < 45

Q_2 select **sum**(Balance)
from DEPOSITOR
where Age < 25 or Gender = Male and Age ≥ 45

Q_3 select **sum**(Balance)
from DEPOSITOR
where Age ≥ 45 or Gender = Male and 25 ≤ Age < 45

Q_4 select **sum**(Balance)
from DEPOSITOR
where Gender = Female and Age < 45

and assume that the values of Q_1, Q_2, Q_3 and Q_4 are 24, 29, 18 and 12, respectively. If the values of the four sum-queries are all released, the amount of information conveyed by their answers is modelled by the following equation system

$$\begin{cases} x_1 + x_2 = 24 \\ x_1 + x_3 + x_4 = 29 \\ x_2 + x_3 + x_6 = 18 \\ x_4 + x_5 = 12 \end{cases} \tag{1}$$

where the variables x_1, x_2, x_3, x_4, x_5 and x_6 stand for the total balances of the depositors belonging to the categories specified by the following six atomic formulae:

C_1 = (Gender = Male and Age < 25)
C_2 = (Gender = Male and 25 ≤ Age < 45)
C_3 = (Gender = Male and Age ≥ 45)
C_4 = (Gender = Female and Age < 25)
C_5 = (Gender = Female and 25 ≤ Age < 45)
C_6 = (Gender = Female and Age ≥ 45)

The coefficient matrix of equation system (1) is the incidence matrix of a graph and the corresponding query map is shown in Figure 1.

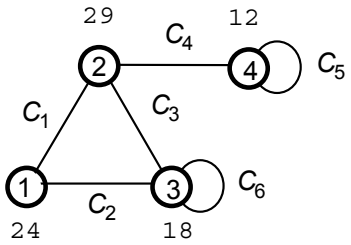


Figure 1

Note that the value of the intrusive sum-query Q is represented by x_1. Since the general solution of equation system (1) is

(x_1 = a, x_2 = 24 – a, x_3 = 29 – a – b, x_4 = b, x_5 = 12 – b, x_6 = –35 + 2a + b)

where a and b are two arbitrary real numbers, the value of x_1 is not determined and, hence, the value of Q is protected. Suppose now that a new sum-query arrives:

Q_5 select **sum**(Balance)
from DEPOSITOR
where Gender = Female and Age ≥ 25

and assume that the value of Q_5 is 7. If the QAS answers Q_5, then the amount of information conveyed by the answers to Q_1, …, Q_5 is obtained by adding the equation $x_5 + x_6 = 7$ to equation system (1) and the corresponding query map is shown in Figure 2.

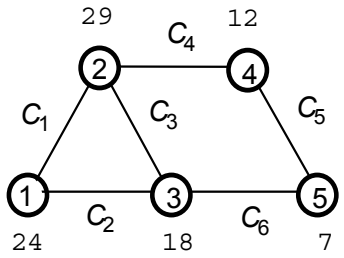


Figure 2

The general solution is now

$$(x_1 = 15, x_2 = 9, x_3 = 14 - \mathtt{b},\ x_4 = \mathtt{b}, x_5 = 12 - \mathtt{b}, x_6 = -5 + \mathtt{b})$$

so that the value of Q is disclosed. Therefore, the QAS should not answer Q_5 by releasing its value, but should issue the set of feasible values of Q_5 consistent with the answers to Q_1, …, Q_4, that is, the whole set of real numbers.

Let us now state the derivability problem in formal terms. Let A be an (additive) commutative group with zero $\mathsf{0}$, and let G be a graph (with no isolated vertices) where loops are allowed. Without loss of generality, we assume that G is connected. A *vertex labeling* (an *edge labeling*, respectively) of G is a mapping from $V(G)$ ($E(G)$, respectively) to A. Given a vertex labeling $\boldsymbol{q}$ of G, an edge labeling $\boldsymbol{x}$ of G is *compatible* with $\boldsymbol{q}$ if $\boldsymbol{x}$ is a solution of the equation system

$$\sum_{e \in E(v)} x(e) = q(v) \qquad (v \in V)$$

where $E(v)$ denotes the set of edges of G incident to v. A vertex labeling of G is *admissible* if there is an edge labeling compatible with it. For example, the *null vertex labeling* (that is, the vertex labeling being zero everywhere) is admissible. Given an admissible vertex labeling $\boldsymbol{q}$ of G, we call the vertex-weighted graph $(G, \boldsymbol{q})$ a *map*. An edge e of G is an A-*invariant edge* of the map $(G, \boldsymbol{q})$ if $x(e) = x'(e)$ for every two edge labelings $\boldsymbol{x}$ and $\boldsymbol{x}'$ compatible with $\boldsymbol{q}$. Let $\boldsymbol{X}(\boldsymbol{q})$ be the set of all edge labelings compatible with $\boldsymbol{q}$. If $\mathbf{0}$ is the null vertex labeling, then it is easily seen that $\boldsymbol{X}(\boldsymbol{q})$ is a translation of $\boldsymbol{X}(\mathbf{0})$, that is, $\boldsymbol{X}(\boldsymbol{q}) = \boldsymbol{x} + \boldsymbol{X}(\mathbf{0})$, where $\boldsymbol{x}$ is any edge labeling of G compatible with $\boldsymbol{q}$. Therefore, the set of A-invariant edges of the map $(G, \boldsymbol{q})$ is the set of edges e such that $y(e) = \mathsf{0}$ for all $\boldsymbol{y}$ in $\boldsymbol{X}(\mathbf{0})$ and, hence, it is the same for every map $(G, \boldsymbol{q})$. Accordingly, the reference to $\boldsymbol{q}$ can be omitted and such edges will be referred to as the A-invariant edges of G.

The problem is to find the set of all A-invariant edges of G and to compute the value of each of them given a map $(G, \boldsymbol{q})$. This problem was solved in linear time in the following two cases:

— $A = Z$ (the set of integers) and G is bipartite [7]: the Z-invariant edges are all and the only bridges of G;

— $A = R$ (the set of reals) and G is arbitrary [9]: the R-invariant edges are all and the only edges of G whose removal increases the number of bipartite components of G.

The above result for $A = R$ was achieved using matroid-theoretic arguments, which are useless for an arbitrary commutative group A. In this paper, using some results of the theory of magic graphs [6] we show that the set of A-invariant edges of an arbitrary graph and the value of each of them can be found in linear time.

2 Background

Let A be a commutative group with zero 0. If a is an element of A, by $2a$ we denote the sum $a+a$. An element b of A is *even* if there is an element a of A such that $b = 2a$. If b is even, by *half*(b) we denote the set $\{a \in A: 2a = b\}$. Accordingly, *half*(0) $= \{a \in A: 2a = 0\} = \{a \in A: a = -a\}$. For example, if A is the set $\{0, 1, \ldots, p-1\}$ with the integer addition mod p then, if p is even, say $p = 2k$, then *half*(0) $= \{0, k\}$; otherwise, *half*(0) $= \{0\}$. The following result is borrowed from the theory of magic graphs [6] where only loopless graphs are considered.

Proposition 1 Let G be a connected, loopless graph and let $\boldsymbol{q}$ be a vertex labeling of G.

(*i*) If G is bipartite, then $\boldsymbol{q}$ is admissible if and only if $\sum_{v \in U} q(v) = \sum_{v \in W} q(v)$, where $\{U, W\}$ is the bipartition of $V(G)$; otherwise, $\boldsymbol{q}$ is admissible if and only if the sum $\sum_{v \in V(G)} q(v)$ is even.

(*ii*) If $\boldsymbol{q}$ is admissible, then an edge labeling compatible with $\boldsymbol{q}$ can be found in linear time using the following algorithm, where by a *leaf* of a graph we mean a vertex with exactly one incident edge that is not a loop.

Algorithm 1

(1) Find a spanning tree T of G.
(2) If G is not bipartite, find an edge $e^* = (u^*, v^*)$ whose addition to T creates an odd cycle, and set $T := T + e^*$.
(3) For each edge $e \in E(G)–E(T)$, set $x(e) := 0$.

(4) Until T contains no leaves, repeat:

> Find a leaf u of T. Let e be the edge incident to u and let w be the other end-point of e. Set $x(e) := q(u)$, $q(w) := q(w)–q(u)$, and delete u and e from T.

(5) If $E(T) = \emptyset$ (that is, if G is bipartite), then Exit. Otherwise, let $\{U, W\}$ be the bipartition of the vertex set of the tree $T–e^*$ with U containing the end-points u^* and v^* of e^*. Set $\mathsf{b} := \sum_{v\in U} q(v) - \sum_{v\in W} q(v)$. Choose an element $\mathsf{a}\in \mathit{half}(\mathsf{b})$ and set $x(e^*) := \mathsf{a}$, $q(u^*) := q(u^*)–\mathsf{a}$ and $q(v^*) := q(v^*)–\mathsf{a}$. Delete e^* from T.

(6) Until T is a one-point graph repeat:

> Find a leaf u of T. Let e be the edge incident to u and let w be the other end-point of e. Set $x(e) := q(u)$, $q(w) := q(w)–q(u)$, and delete u and e from T.

(In Appendix I Algorithm 1 is applied to the map shown in Figure 2.)

Consider now a connected graph G where loops are allowed and let $\boldsymbol{q}$ be any vertex labeling of G. An edge labeling compatible with $\boldsymbol{q}$ can be always found using the algorithm (henceforth referred to as Algorithm 2) obtained from Algorithm 1 by re-placing step (2) by the step

> (2′) Find a loop e^*, and set $T := T + e^*$.

and steps (5) and (6) by the single step

> (5′) If v^* is the end-point of e^*, set $x(e^*) := q(v^*)$.

(In Appendix II Algorithm 2 is applied to the information model shown in Figure 1.) So, one has

Proposition 2 Every vertex labeling $\boldsymbol{q}$ of a graph containing loops is admissible and an edge labeling compatible with $\boldsymbol{q}$ can be found in linear time.

3 Characterization of invariant edges

Let G be a connected graph and let $\boldsymbol{Y} = \boldsymbol{X}(\mathsf{0})$. An edge labelling in $\boldsymbol{Y}$ will be called a *circulation* in G *over* A (an A-*circulation*, for short); moreover, if $\boldsymbol{y}$ is an A-circulation in G, the edge set $\{e \in E(G): y(e) \neq \mathsf{0}\}$ is called the *support* of $\boldsymbol{y}$. Bearing in mind that an edge e of G is A-invariant if and only if $y(e) = 0$ for all $\boldsymbol{y}$ in $\boldsymbol{Y}$, we have

that an edge of G is A-invariant if and only if it does not belong to the support of any A-circulation. Let us distinguish the following three cases: G is bipartite, G is not bipartite and is loopless, G contains loops.

Case 1. G is bipartite. If G is a tree then $\boldsymbol{Y} = \{\mathbf{0}\}$ (see Algorithm 1) so that each edge of G is A-invariant. Assume that G is not a tree. For every cycle C, no edge in C is A-invariant since, arbitrarily chosen a nonzero element a of A, one can construct an A-circulation (see Figure 3) whose support is C.

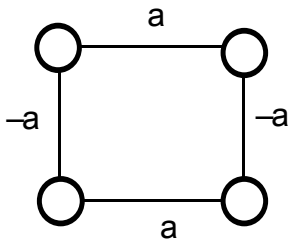


Figure 3 An A-circulation associated with an even cycle

Therefore, the A-invariant edges of G are all bridges. On the other hand, if e is a bridge of G and G' is either component of $G–e$, then

$$y(e) = [\sum_{v\in U} \sum_{e\in E(v)} y(e)] - [\sum_{v\in W} \sum_{e\in E(v)} y(e)] = \mathsf{0}$$

where $\{U, W\}$ is the bipartition of $V(G')$ and e is incident to U. To sum up, the A-invariant edges of G are all and the only bridges of G.

Case 2. G is not bipartite and is loopless. Let T be a spanning tree of G with the addition of an edge e^* (see Algorithm 1) creating an odd cycle, say C. Given an arbitrary element a of $\mathit{half}(\mathsf{0})$, with C we can associate an A-circulation (see Figure 4), whose support is empty or C depending on whether $\mathsf{a} = \mathsf{0}$ or $\mathsf{a} \neq \mathsf{0}$, respectively.

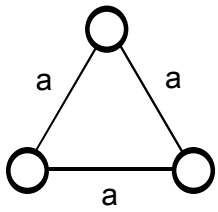


Figure 4 An A-circulation associated with an odd cycle

Let $\boldsymbol{c}_{\mathsf{a}}$ be such an A-circulation associated with C. If $G = T$, then $\boldsymbol{Y} = \{\boldsymbol{c}_{\mathsf{a}}\colon \mathsf{a} \in \mathit{half}(\mathsf{0})\}$ (see Algorithm 1) so that, if $\mathit{half}(\mathsf{0}) = \{\mathsf{0}\}$ then each edge of G is A-invariant; otherwise (that is, if $\mathit{half}(\mathsf{0}) \neq \{\mathsf{0}\}$), an edge is A-invariant if and only if it is

a bridge. Let now assume that $G \neq T$ and let $E(G)–E(T) = \{e_1, \ldots, e_k\}$. The addition of e_i to T creates a closed even walk C_i which is either an even cycle or an *L-odd set* [4], that is, a pair of edge-disjoint odd cycles joined by a (possibly one-point) path. Given an arbitrary element a of A, with C_i we can associate an A-circulation as follows. If C_i is an even cycle, then the A-circulation is of the form shown in Fig. 3; if C_i is an L-odd set, then the A-circulation is of the form shown in Fig. 5.

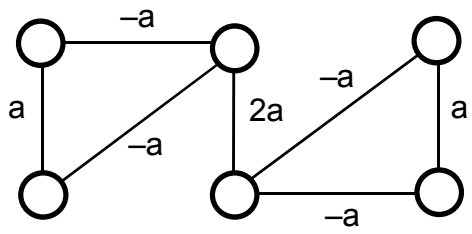


Figure 5 An A-circulation associated with an L-odd set

Let $\boldsymbol{c}_{i,\mathsf{a}_i}$ be such an A-circulation associated with C_i for some element a_i of A. As proven in [6], every A-circulation $\boldsymbol{y}$ in G can be written as

$$\boldsymbol{y} = \boldsymbol{c}_{\mathsf{a}} + \sum_{i=1,\ldots,k} \boldsymbol{c}_{i,\mathsf{a}_i}$$

for some element a of $\mathit{half}(\mathsf{0})$ and some elements $\mathsf{a}_1, \ldots, \mathsf{a}_k$ of A. Let us distinguish two subcases depending on whether $\mathit{half}(\mathsf{0}) = \{\mathsf{0}\}$ or $\mathit{half}(\mathsf{0}) \neq \{\mathsf{0}\}$.

Case 2(*i*): $\mathit{half}(\mathsf{0}) = \{\mathsf{0}\}$. Then, an edge e is A-invariant if and only if e does not belong to any even cycle and to any L-odd set, that is, if and only if either e is a bridge and $G–e$ has a bipartite component or e belongs to all odd cycles of G. Note that in both cases, e is characterized by the property that $G–e$ has one more bipartite component than G. As an example, the R-invariant edges of the graph of Figure 2 are the edges (1,2) and (1, 3).

Case 2(*ii*): $\mathit{half}(\mathsf{0}) \neq \{\mathsf{0}\}$. Then, the A-invariant edges are all bridges since they belong to no cycles. Furthermore, if $\mathit{half}(\mathsf{0}) \neq \mathsf{A}$ then an edge e is A-invariant if and only if e is a bridge and $G–e$ has a bipartite component; otherwise (that is, if $\mathit{half}(\mathsf{0}) = \mathsf{A}$) then $2\mathsf{a} = \mathsf{0}$ for all a so that an edge is A-invariant if and only if it is a bridge.

Case 3. G contains loops. Let T be a spanning tree of G with the addition of a loop e^* (see Algorithm 2). If $G = T$ then $\boldsymbol{Y} = \{\mathbf{0}\}$ so that each edge of G is A-invariant. Otherwise, let $E(G)–E(T) = \{e_1, \ldots, e_k\}$. The addition of e_i to T again creates a closed

even walk C_i which is either an even cycle or an L-odd set having e^* as one of its cycles. Given an arbitrary element a of A, with C_i we can associate an A-circulation as follows. If C_i is an even cycle, then the A-circulation is of the form shown in Fig. 3; if C_i is an L-odd set, then the A-circulation is of either form shown in Fig. 6.

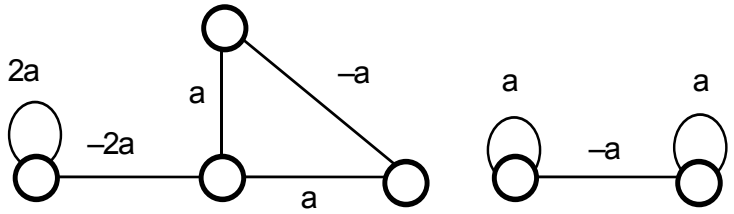


Figure 6 An A-circulation associated with an L-odd set containing a loop

Set $\boldsymbol{c}_{i,\mathsf{a}_i}$ be such an A-circulation associated with C_i for some element a_i of A. It can be proven that every A-circulation $\boldsymbol{y}$ in G can be written as

$$\boldsymbol{y} = \sum_{i=1,\ldots,k} \boldsymbol{c}_{i,\mathsf{a}_i}$$

for some elements $\mathsf{a}_1, \ldots, \mathsf{a}_k$ of A. Let us distinguish two subcases depending on whether $half(\mathsf{0}) = \{\mathsf{0}\}$ or $half(\mathsf{0}) \neq \{\mathsf{0}\}$.

Case 3(*i*): $half(\mathsf{0}) = \{\mathsf{0}\}$. Then, an edge e is A-invariant if and only if e does not belong to any even cycle and to any L-odd set, that is, if and only if e either is a bridge and $G–e$ has a bipartite component or e belongs to all odd cycles of G. As an example, the graph of Figure 1 has no R-invariant edges.

Case 3(*ii*): $half(\mathsf{0}) \neq \{\mathsf{0}\}$. If $half(\mathsf{0}) \neq \mathsf{A}$ then an edge e is A-invariant if and only if either e is a bridge and $G–e$ has a bipartite component or e is a loop and $G–e$ is loopless; otherwise (that is, if $half(\mathsf{0}) = \mathsf{A}$), an edge e is A-invariant if and only if either e is a bridge and $G–e$ has a loopless component or e is a loop and $G–e$ is loopless.

To sum up, we have the following.

Proposition 3 Let G be a connected graph and A a commutative group. If $half(\mathsf{0}) = \{\mathsf{0}\}$, then an edge e of G is A-invariant if and only if either e is a bridge and $G–e$ has a bipartite component or e belongs to all odd cycles of G. If $\{\mathsf{0}\} \subset half(\mathsf{0}) \subset \mathsf{A}$, then an edge e of G is A-invariant if and only if either e is a bridge and $G–e$ has a bipartite component or e is a loop and $G–e$ is loopless. If $half(\mathsf{0}) = \mathsf{A}$, then an edge e of G is A-

invariant if and only if either e is a bridge and $G–e$ has a loopless component or e is a loop and $G–e$ is loopless.

4 Computational aspects

A consequence of Proposition 3 is that the set of A-invariant edges of a graph can be found in time linear since:

— the set of bridges whose removal creates one more bipartite component can be found in linear time [9], and the same can be easily proven for the set of bridges whose removal creates one more loopless component;

— the presence of a loop whose removal creates a loopless graph can be checked in linear time;

— the set of edges belonging to all odd cycles can be found in linear time [9].

Once the set of A-invariant edges of a graph G has been found, in order to determine their values for a map $(G, \boldsymbol{q})$ one can use Algorithm 1 or Algorithm 2, depending on whether G is or is not loopless.

5 An open problem

We solved the problem of finding the A-invariant edges of a map and of computing their values, where A is an arbitrary commutative group. The case that A is the set of non-negative elements of an "ordered" commutative group is open. However, if A is the set of non-negative reals, then the solution is known [9] and it also applies to the case that A is the set of non-negative integers provided that the underlying graph is bipartite [7].

References

1. Adam, N.R. & Wortmann, J.C. (1989). Security control methods for statistical databases: a comparative study. *ACM Computing Surveys* **21**, 515-556.

2. Chang Chen, M., & McNamee, L., & Melkanoff, M. (1988). A model of summary data and its applications to statistical databases, Proc. IV Int. Working Conf. on "Statistical & Scientific Database Management" (M. Rafanelli et alii, eds.), *Lecture Notes in Computer Sciences* **339**, 354-372.

3. Chang Chen, M., & McNamee, L. (1989). On the data model and access method of summary data management. *IEEE Trans. on Knowledge and Data Engineering* **1**, 519-529.

4. Conforti, M. & Rao, M.R (1987). Cut set and the max cut problem. *Math. Oper. Res.*, **12** , 193-204.

5. Cox, L.H. (1980). Suppression methodology and statistical disclosure control. *J. American Statistical Association* **75**, 377-385.

6. Doob, M. (1974). Generalization of magic graphs. *J. Combinatorial Theory B* **17**, 205-217.

7. Gusfield, D. (1988). A graph-theoretic approach to statistical data security. *SIAM J. Computing* **17**, 552-571.

8. Malvestuto, F.M. (2003). A query-overlap restriction for statistical database security. UN-ECE/EUROSTAT Worksession on "Statistical Confidentiality", Luxembourg.

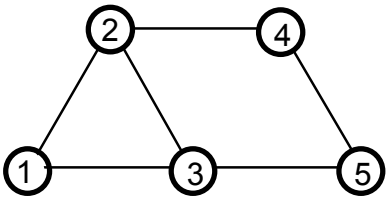
9. Malvestuto, F.M. & Mezzini, M. (2002). A linear algorithm for finding the invariant edges of an edge-weighted graph. *SIAM J. on Computing* **31**, 1438-1455.

10. Malvestuto, F.M. & Moscarini, M. (1999). An audit expert for large statistical databases. In *Statistical Data Protection*, EUROSTAT, 29-43.

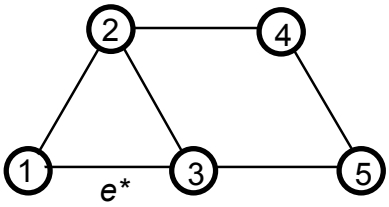
11. Willenborg, L. & de Waal, T. (1996). *Statistical Disclosure Control in Practice*. Lecture Notes in Statistics, Vol. 111, Springer-Verlag, New York.

12. Willenborg, L. & de Waal, T. (2000). *Elements of Statistical Disclosure*. Lecture Notes in Statistics, Vol. 155, Springer-Verlag, New York.

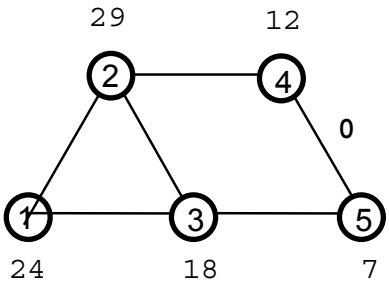
Appendix I



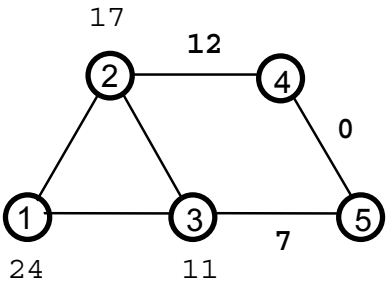
Step 1



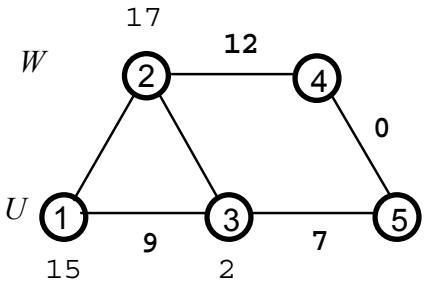
Step 2



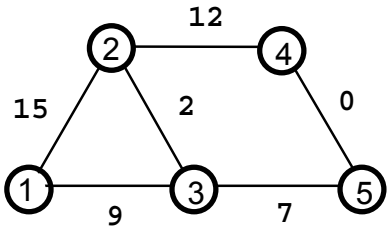
Step 3



Step 4

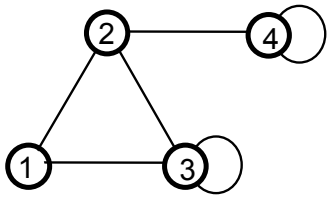


Step 5

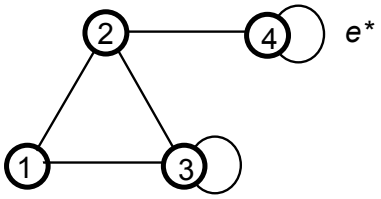


Step 6

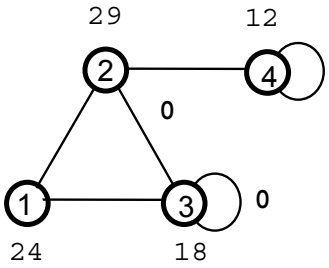
Appendix II



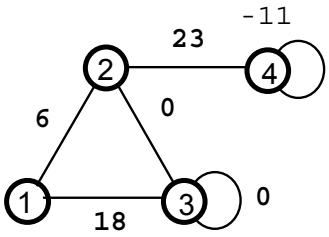
Step 1



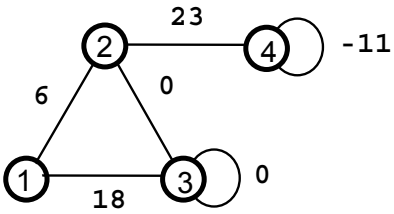
Step 2'



Step 3



Step 4



Step 5'