Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 11, no. 1, pp. 239–257 (2007)

Finding a Nonempty Algebraic Subset of an Edge Set in Linear Time

Mauro Mezzini

Department of Computer Science
University of Rome, "La Sapienza"
http://www.di.uniroma1.it/
mezzini@di.uniroma1.it

Abstract

A set of edges of a hypergraph H is an *algebraic* set if its characteristic vector can be expressed as a linear combination of rows of the (node-edge) incidence matrix of H. Recently it was proven that deciding whether or not a given edge-set of H contains a non-empty algebraic set is an NP-complete problem. In this paper we give a linear time algorithm to decide if a given edge-set contains a non-empty algebraic set when the hypergraph is a graph.

Article Type	Communicated by	Submitted	Revised
Regular paper	L. Arge	November 2005	January 2007

1 Introduction

Let G be a graph with edge set $E(G)$ and node set $V(G)$. Let $\mathbf{M}$ be the node-edge incidence matrix of G, that is the binary $|V(G)| \times |E(G)|$ matrix where each element $m_{v,e}$ of $\mathbf{M}$ is given by

$$m_{v,e} = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise} \end{cases} \qquad v \in V(G) \text{ and } e \in E(G)$$

Let F be a non-empty subset of $E(G)$ and let $\mathbf{f}$ be the characteristic vector of F, that is

$$f(e) = \begin{cases} 1 & \text{if } e \in F \\ 0 & \text{if } e \notin F \end{cases}$$

We call F an *algebraic* [16, 21, 24] set if its characteristic vector can be expressed as a linear combination of rows of $\mathbf{M}$. In other words F is an algebraic set if there exist real coefficients $(c_v)_{v \in V(G)}$ such that

$$\mathbf{f} = \sum_{v \in V(G)} c_v \mathbf{m}_v \tag{1}$$

where $\mathbf{m}_v$ is a row of $\mathbf{M}$, for $v \in V(G)$. In this paper we address the problem to decide if a subset F of edges of $E(G)$ contains a proper non-empty algebraic set. We refer to this problem as the *NAS problem.* Recently it was proven that the NAS problem is *NP*-complete when G is a hypergraph [19], that is, when $\mathbf{M}$ is an arbitrary binary matrix. In this paper we will show that the NAS problem can be solved in polynomial time when G is a graph by giving a linear time algorithm to find (if any) a non-empty algebraic subset of a given edge set of G.

The NAS problem arises in the context of the inference problem of summary data (SD). An SD [3, 4, 5, 6, 10, 19] is a triple $[sum(S), C, t]$ where S is a numerical attributes (such as SALARY or COST), C is a Boolean condition made up using qualitative attributes, t is a numerical value given by the sum of the values of the numerical attribute S over the category of individuals qualified by C. The term $sum(S)$ is called the summary attribute of an SD. For example an SD can be [*sum*(COST), "PRODUCT=*computers*", t] where t is taken to be the sum of the cost of all products in the database that fall in the category "computers".

The inference problem is to decide whether or not an SD of interest (whose numerical value is unknown) can be evaluated (i.e., computed) from a given set of SD with the same summary attribute, whose domain, say Φ, is either the set of reals (R) or the set of integers (Z) or the set of non negative reals ($R+$) or the set of non negative integers ($Z+$).

The information content of the input SD set is modelled by a linear equation system whose variables are constrained to take their values from Φ and where the coefficient matrix of the system is a binary matrix $\mathbf{M}$ [5, 15, 19, 20, 22]. In this context we associate with an SD δ of interest a binary vector $\mathbf{f}$. When $\mathbf{f}$

can be expressed as a linear combination of rows of $\mathbf{M}$, i.e. $\mathbf{f}$ is algebraic, we have that δ is evaluable for any choice of Φ.

If the SD of interest is not evaluable or is computationally hard to evaluate (as in the case where $\Phi = Z+$ [15, 19]), we search for a nontrivial evaluable SD whose category is maximally contained in the category of individuals of interest. It was shown in [19] that in order to solve this problem we need to solve the NAS problem for an arbitrary binary matrix.

The inference problem arises also in the field of privacy protection in statistical database systems [1, 12, 13, 14, 15, 20, 23]. In this case one wants to determine if (exact or approximate) confidential information can be implicitly inferred from a set of released SDs. Since in this case the information content of the set of SD, can be represented by a system of linear equation where the coefficient matrix is the node-edge incidence matrix of a graph [17, 18, 20, 24], the solution of the NAS problem for graphs could find here a natural application.

The outline of this paper is as follows. From section 2 through section 5 we will go into the mathematical details needed to solve the NAS problem for graphs. In section 2 we give the basic definitions and some properties of algebraic sets. In section 3 we introduce the key concept of the *kernel* of a set F of edges. The kernel of a set F is a key concept because it contains all the algebraic subsets of F. In section 4 we give the algorithm to solve the NAS problem for graphs. In section 5 we discuss the computational aspects of the algorithm. Finally, in section 6, we give some closing remarks.

2 Algebraic sets

Let $G = (V(G), E(G))$ be a graph without parallel edges where loops may exist. An edge is a *link* if it is not a loop. If U and W are two non-empty subsets of $V(G)$, we denote by $[U, W]$ all the edges of $E(G)$ with one end point in U and the other in W. A *subgraph* of graph G is a graph H with $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. An edge in $E(G) - E(H)$ having exactly one endpoint in $V(H)$ is said to be *attached* to H.

A graph G is *bipartite* if it contains no odd cycle. If G is bipartite and $V(G)$ is not a singleton, then there exists a bipartition (U, V) of $V(G)$ such that $[U, U] = [V, V] = \emptyset$. We call U and V *sides* of G.

A *star* of a node v of G, denoted by $star(v)$, is the set of the edges incident to v. If W is a set of nodes then the union of the stars of the nodes of W is called a *starset*, denoted by $S(W)$; furthermore, if W is a *stable* set (i.e. the set of nodes in W are pairwise non-adjacent), then $S(W)$ is called an *open starset*. Now we introduce two fundamental classes of algebraic sets (see also [12, 13, 14, 16, 21]). An *open-flower* set is an open starset or the proper difference of two open starsets $S(W_1)$ and $S(W_2)$ where $S(W_2) \subseteq S(W_1)$ and $W_1 \cap W_2 = \emptyset$ (see Fig. 1). An open-flower set is an algebraic set since its characteristic vector can be written as

$$\sum_{v \in W_1} \mathbf{m}_v - \sum_{v \in W_2} \mathbf{m}_v$$

Let G be a loopless, non-bipartite graph. A *closed-flower* set is the proper difference of two starsets $S(W_1)$ and $S(W_2)$ where (W_1, W_2) is a partition of $V(G)$ and $S(W_2)$ is an open starset or is empty. Note that if A is a closed-flower set then $G - A$ is a bipartite graph. Also note that a closed-flower set is an algebraic set since its characteristic vector can be written as (see Fig. 2)

$$\frac{1}{2}\Big(\sum_{v\in W_1}\mathbf{m}_v - \sum_{v\in W_2}\mathbf{m}_v\Big)$$

Without loss of generality, henceforth G is assumed to be connected, since it is easily proven that the intersection of an algebraic set with the edge set of a connected component of G is an algebraic set, too.

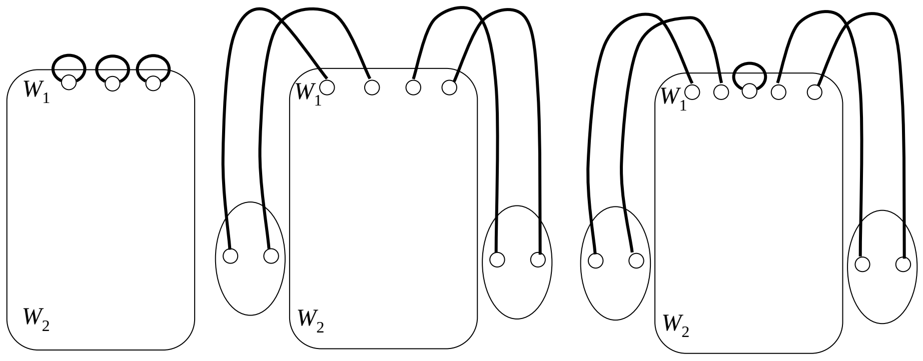

Figure 1: The bold edges form open-flower sets.

Given a real-valued vector $(c_v)_{v\in V(G)}$, the *signed support* [2] of $\mathbf{c}$ is the couple (P, N) where $P = \{v : v \in V(G), c_v > 0\}$ and $N = \{v : v \in V(G), c_v < 0\}$ and the *support* of $\mathbf{c}$ is the set $P \cup N$.

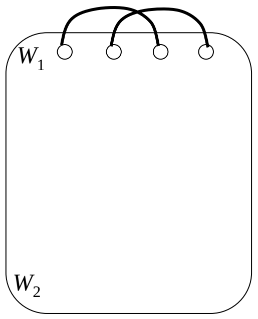

Figure 2: The bold edges form a closed-flower set.

Let A be an algebraic set, $\mathbf{a}$ its characteristic vector and $\mathbf{c}$ a solution of (1). Let (P, N) be the signed support of $\mathbf{c}$. Since each equation of (1) is in the form

$$\begin{array}{ll} a((v,u)) = c_v + c_u & \text{if } (u,v) \in E(G) \\ a((v,v)) = c_v & \text{if } (v,v) \in E(G) \end{array}$$

then it's easily seen that

$$[N, V(G) - P] = \emptyset$$

and

$$[P, V(G) - N] \subseteq A \subseteq [P, V(G)]$$

Moreover if $(v_1, \ldots, v_k)$ is a simple path in G, then

$$c_{v_{i+1}} = a((v_i, v_{i+1})) - c_{v_i} \qquad i = 1, \ldots, k-1$$

so that

$$c_{v_k} = (-1)^{k-1} c_{v_1} + \pi \tag{2}$$

where π is an integer [22]. Therefore since G is connected given two nodes u and v of G, either both c_u and c_v are integers or neither is an integer. In fact we can state the following (the proof of which can be found in Lemma (4) of [21]):

Lemma 1 *[21] Let G be a connected graph and A be a non-empty algebraic set. Let (P, N) be the signed support of a solution of (1). If $P \cup N \neq V(G)$ and $[P, P]$ is not empty then $[P, P]$ is a set of loops. If $P \cup N = V(G)$ and $[P, P]$ is not empty then either contains all loops of G or it contains only links.*

The following theorem shows the importance of closed-flower and open-flower sets.

Theorem 1 *Every non-empty algebraic set of a connected graph contains either an open-flower or a closed-flower set.*

Proof: By definition, both open-flower and closed-flower sets are algebraic sets. If G is a bipartite graph, then it is known [12, 16] that a non-empty subset of $E(G)$ is an algebraic set if and only if it is a disjoint union of open-flower sets. Assume that G is non-bipartite. Let (P, N) be the signed support of a solution of (1). We can distinguish two cases depending on whether or not $P \cup N = V(G)$.

Case 1: $P \cup N = V(G)$. First note that $[P, P]$ cannot be empty, for otherwise G would be bipartite. By Lemma (1), $[P, P]$ either is a set of loops or is a set of links. In the first case $[P, P]$ is an open-flower set and in the second case is a closed-flower set.
Case 2: $P \cup N \neq V(G)$. First note that at least $[P, V(G) - P \cup N]$ is not empty otherwise G would be disconnected. By Lemma (1), if $[P, P]$ is not empty then it is a set of loops. Therefore the edge set $[P, V(G) - N]$ is an open-flower set. □

3 The kernel of an edge set

In this section we introduce a particular subset of a given edge set F that we call the *kernel* of F, which has the property of containing all the algebraic subsets of F. This will be proved in the Theorem (2). Lemma (2), Lemma (3) and Lemma (4) are technical lemmas about the kernel needed for the rest of the paper. Consider the following linear system

$$\mathbf{M}\,\mathbf{x} = \mathbf{b} \tag{3}$$

where $\mathbf{b} = (b_v)_{v \in V(G)}$ is obtained as follows

$$b_v = |star(v) - F| \qquad v \in V(G) \tag{4}$$

A solution of system (3) is given by the vector $\mathbf{x}^*$ with

$$x^*(e) = \begin{cases} 0 & \text{if } e \in F \\ 1 & \text{otherwise} \end{cases}$$

The general solution of system (3) is given by

$$\mathbf{x} = \mathbf{x}^* + \mathbf{y}$$

where $\mathbf{y}$ is a solution of the homogeneous system

$$\mathbf{M}\,\mathbf{y} = \mathbf{0} \tag{5}$$

The set of solutions of (5) is called the *null space* of $\mathbf{M}$. According to the terminology introduced in [19] if $\mathbf{X}$ is the (non-empty) set of non negative solutions of (3), the set

$$K = \{e : x(e) = 0, \forall \mathbf{x} \in \mathbf{X}\}$$

will be referred to as the *kernel* of F. Clearly since $F = \{e : x^*(e) = 0\}$, then K is definitely a subset of F. The next theorem clarifies the importance of the kernel of a set F.

Theorem 2 *Let F be a non-empty edge set of a graph and K the kernel of F. An algebraic set is a subset of F if and only if it is a subset of K.*

Proof: (*if*) Trivially every algebraic subset of K is an algebraic subset of F since $K \subseteq F$.
(*only if*) Let A be an algebraic subset of F. We first show that $\sum_{e \in A} x(e)$ takes on the same value for every non negative solution $\mathbf{x}$ of (3). In fact let $\mathbf{a}$ be the characteristic vector of A. By definition $\mathbf{a}$ is a vector of the row space of $\mathbf{M}$. Therefore $\mathbf{a}$ is orthogonal to the null space of $\mathbf{M}$. Now if $\mathbf{x}_1$ and $\mathbf{x}_2$ are any two non negative solutions of (3), then $\mathbf{x}_1 - \mathbf{x}_2$ is a solution of (5). Therefore, $\sum_{e \in E(G)} a(e)[x_1(e) - x_2(e)] = 0$ and then $\sum_{e \in A} x_1(e) = \sum_{e \in A} x_2(e)$.
Since $\mathbf{x}^*$ is a non negative solution of (3) then $\sum_{e \in A} x(e) = \sum_{e \in A} x^*(e) = 0$ because $A \subseteq F$. By the non negativity of $\mathbf{x}$ we have that $x(e) = 0$, $\forall e \in A$, and for any non negative solution $\mathbf{x}$ of (3). It follows that $A \subseteq K$. $\square$

By definition of kernel, for every edge e of $F-K$, there exists a non negative solution $\mathbf{x}^{\#}$ of (3) such that $x^{\#}(e) > 0$. It follows that there exists a solution $\mathbf{y} = \mathbf{x}^{\#} - \mathbf{x}^{*}$ of system (5) such that $y(e) > 0$. More generally we have the following

Lemma 2 *Let K be the kernel of an edge set. Then there exists a non negative solution $\mathbf{x}'$ of (3) such that $x'(e) > 0$ for every $e \in E(G) - K$.*

Proof: Let F be an edge set and K its kernel. If $F - K$ is empty we have done since we take $\mathbf{x}' = \mathbf{x}^*$. Let $F - K = \{e_1, \ldots, e_p\} \neq \emptyset$. By definition of kernel, there exists a solution $\mathbf{y}_i$ of (5) such that $y_i(e_i) > 0$ and $\mathbf{x}^* + \mathbf{y}_i \geq \mathbf{0}$, i=1,…,p. Let $\mathbf{y} = \sum_{i=1,\ldots,p} \mathbf{y}_i$ and let

$$0 < \varphi < \mathbf{min}\{x^*(e)/|y(e)| : y(e) < 0 \text{ and } e \in E(G) - F\}$$

We have that

$$\mathbf{x}' = \mathbf{x}^* + \varphi\mathbf{y} \geq 0$$

and

$$x'(e) > 0 \qquad \forall e \in E(G) - K$$

in fact, by definition, for all the edges $e \in K$ we have $y_i(e) = 0$ and hence $y(e) = 0$. It follows that $x'(e) = 0$. Consider now any edge e in $F - K$. First we see that $y(e) \geq 0$. In fact if $e \in F - K$ then

$$0 \leq x^*(e) + y_i(e) = y_i(e) \qquad \text{for } i = 1, \ldots, p$$

and then

$$y(e) = \sum_{i=1,\ldots,p} y_i(e) \geq 0$$

Moreover since $y_i(e_i) > 0$ we have that $y(e_i) > 0$ for all $e_i \in F - K$. In this case $x'(e_i) > 0$. Finally consider any edge e in $E(G) - F$. If $e \in E(G) - F$ and $y(e) \geq 0$, then clearly, $x'(e) > 0$, otherwise if $y(e) < 0$ since

$$x^*(e)/|y(e)| > \varphi$$

we have that $x^*(e) - \varphi|y(e)| > 0$, that is $x^*(e) + \varphi y(e) > 0$. □

We now state a useful property of the edges of the kernel of F (see also [9, 17]).

Lemma 3 *If C is an even cycle of G then either $C \cap K = \emptyset$ or $|C \cap K| > 1$ and at least two edges of $C \cap K$ are at odd distance each other in C.*

Proof: Suppose for contradiction that there exists an even cycle $C = \{e_0, \ldots, e_p\}$ such that either $|C \cap K| = 1$ or $|C \cap K| > 1$ and all the edges of $C \cap K$ are at even distance each other. Suppose without loss of generality, that $e_0 \in K$. By Lemma (2), there exists a non negative solution $\mathbf{x}'$ of (3) such that $x'(e) > 0$

for every $e \in E(G) - K$. Now let $0 < \epsilon < \mathbf{min}\{x'(e) : x'(e) > 0\}$ and let $\mathbf{y} = (y(e))_{e \in E(G)}$ be defined as follows

$$y(e) = \begin{cases} 0 & \text{if } e \notin C \\ +\epsilon & \text{if } e \text{ has an even position in } C \\ -\epsilon & \text{if } e \text{ has an odd position in } C \end{cases}$$

Clearly $\mathbf{y}$ is a solution of system (5). But then we have that $\mathbf{x'} + \mathbf{y}$ is a non negative solution of (3) and $x'(e) + y(e) = +\epsilon > 0$ for all edges of $C \cap K$, contradicting the fact that they are in the kernel of F. □

Finally we state another useful property of the kernel which can be also obtained as a corollary of Lemma (2) of [20].

Lemma 4 *Let K be the kernel of F. Then there always exists a real valued solution $\mathbf{c}$ to the following system of linear constraints*

$$\sum_{v \in V(G)} c_v \mathbf{m}_v = \begin{cases} > 0 & \textit{if } e \in K \\ = 0 & \textit{if } e \notin K \end{cases} \tag{6}$$

The proof of this Lemma will be given in the section 5, where we will give an algorithm that given a set F, computes a solution of (6) for the kernel of F. Let (P, N) be the signed support of a solution of (6). Since each equation of (6) is in either form

$$\begin{array}{ll} c_v + c_u \geq 0 & \text{if } (u, v) \in E(G) \\ c_v \geq 0 & \text{if } (v, v) \in E(G) \end{array}$$

we have that $[N, V(G) - P] = \emptyset$ and that $[N, V(G) - P] \subseteq K \subseteq [P, V(G)]$. Also if $Z = V(G) - (P \cup N)$ then $K \cap [Z, Z] = \emptyset$

Example 1 In the graph of Fig. 3(a) the edges of a subset F of $E(G)$ are shown in bold. Fig. 3(b) shows the edges of the kernel K along with a solution of (6).

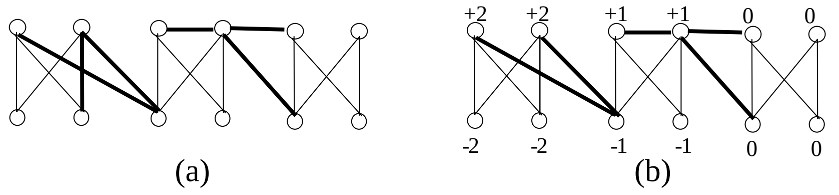

Figure 3: Example of a support for the kernel.

4 Finding a non-empty algebraic subset of the kernel

By Theorem (2) all the algebraic subsets of a set F are contained in the kernel of F. By Theorem (1), there is a non-empty algebraic subset of K if and only if K contains an open-flower or a closed-flower set. The following two lemmas show how to find a closed-flower or an open-flower set contained in K. Thus giving a solution for the NAS problem.

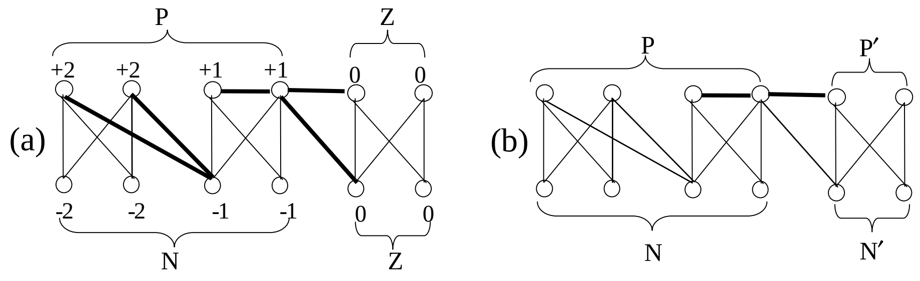

Figure 4: (a) a solution of (6). (b) a closed-flower set contained in K

Lemma 5 *Let G be a non-bipartite and loopless graph. An edge set F with kernel K contains a closed-flower set if and only if $G - K$ is bipartite.*

Proof: (*only if*) Let A be a closed-flower subset of F. By Theorem (2), A is a subset of K. By definition of closed-flower set, $G - A$ is bipartite and, then, $G - K$ must be bipartite too.
(*if*) Let G be non-bipartite and loopless. Let (P, N) be the signed support of a solution of system (6). Let $Z = V(G) - P \cup N$. If Z is empty clearly $[P, P]$ is a closed-flower set. Otherwise, since $G - K$ is bipartite, the subgraph induced by Z is bipartite too because $K \cap [Z, Z] = \emptyset$. Now if P' and N' are two sides of the subgraph induced by Z, then let $A = [P \cup P', P \cup P']$. Since $[N, V(G) - P] = \emptyset$, we have that $G - A$ is bipartite too. Note that A cannot be empty for otherwise G would be bipartite. Clearly A is a closed-flower set. □

Example 1 (contd) Fig. 4(a) highlights the signed support of a solution of (6). Fig. 4(b) $[P \cup P', P \cup P']$ is the set of bold edges. Note that $G - K$ is a bipartite graph. □

Lemma 6 *Let G be a connected graph and F a subset of $E(G)$. The kernel K of F contains an open-flower set if and only if $G - K$ has a bipartite component B^* such that*

(i) *each edge in K with both endpoints in $V(B^*)$ is a loop*
(ii) *no two edges in K are attached to opposite sides of B^**

Proof: (*if*) The subset of K formed by the edges that are attached to B^* is an open-flower set.
(*only if*) Let A be an open-flower set contained in K and (P, N) the signed support of a solution of (1). Let B be the subgraph induced by $[P, N]$. If B contains no edge of K then the statement is trivially true since we can take $B^* = B$. Otherwise let $B_1, \ldots, B_p$ be the bipartite connected components of $B - K$. Recall that (P, N) is the bipartition of B such that all the edges of A are attached to P. Also let (P_i, N_i) be the bipartition of B_i such that $P_i \subseteq P$ and $N_i \subseteq N$, $i = 1, \ldots, p$.
Suppose by contradiction that for every component B_i of $B - K$ there always exists at least one edge of K attached to P_i and at least one edge of K attached to N_i. Take the component $B_i = B_{i_1}$. If, for contradiction, an edge of K has both endpoints in the same component B_i clearly it closes an even cycle C such that $|C \cap K| = 1$, which contradicts Lemma (3). Then since $[N, V(G) - P] = \emptyset$ every edge of K attached to N_{i_1} has the other end point attached to some other component B' of $\{B_1, \ldots, B_p\} - \{B_{i_1}\}$.
Let e_{i_1} be one of such edges attached to B_{i_1} and also attached to $B' = B_{i_2}$. Repeating this argument we obtain a sequence $B_{i_1}, e_{i_1}, B_{i_2}, \ldots, e_{i_{k-1}}, B_{i_k}$ of components of $B - K$ and edges e_{i_j} of K (see Fig. 5(a)). Let B_{i_k} be the first component in the above sequence such that $B_{i_k} = B_{i_h}$ for some $1 \leq h < k$ (see Fig. 5(b)). Let $(v_{i_j}, u_{i_j}) = e_{i_j}$ such that $v_{i_j} \in N_{i_j}$ and $u_{i_j} \in P_{i_{j+1}}$. Consider now the sequence $B_{i_h}, e_{i_h}, B_{i_{h+1}}, e_{i_{h+1}}, \ldots, B_{i_{k-1}}, e_{i_{k-1}}$. Let p_{i_j} be a simple path through B_{i_j} from $u_{i_{j-1}}$ to v_{i_j} and let p_{i_h} be a simple path through B_{i_h} from $u_{i_{h-1}}$ to v_{i_h}. Clearly, we have obtained an even cycle $C = p_{i_h}, e_{i_h}, p_{i_{h+1}}, e_{i_{h+1}}, \ldots, p_{i_{k-1}}, e_{i_{k-1}}$. It is not difficult to see that all the edges of $C \cap K$ have an even distance each other in C. But, then, by Lemma (3), all the edges $e_{i_h}, e_{i_{h+1}}, \ldots, e_{i_{k-1}}$ are not in the kernel, a contradiction. □

To sum up we have the following algorithm to find an algebraic subset of F.

```
FIND_ALGEBRAIC_SUBSET
input: Graph $G$ and an edge set $F$
output: A non-empty algebraic subset of $F$ if any

begin
    find the kernel $K$ of $F$;
    if $G$ is not bipartite and loopless and $G - K$ is bipartite then begin
        compute the signed support $(P, N)$ of a solution of (6);
        let $(P', N')$ be a bipartition of the subgraph of $G$ induced by
        $V(G) - (P \cup N)$;
        output $[P \cup P', P \cup P']$ and EXIT;
    else begin
        for each bipartite component $B$ of $G - K$ do begin
            let $A^* = \{e : e \in K$ and $e$ is attached to $B\}$;
            if condition (i) and (ii) of Lemma (6) are satisfied for $A^*$ then
                output $A^*$;
```

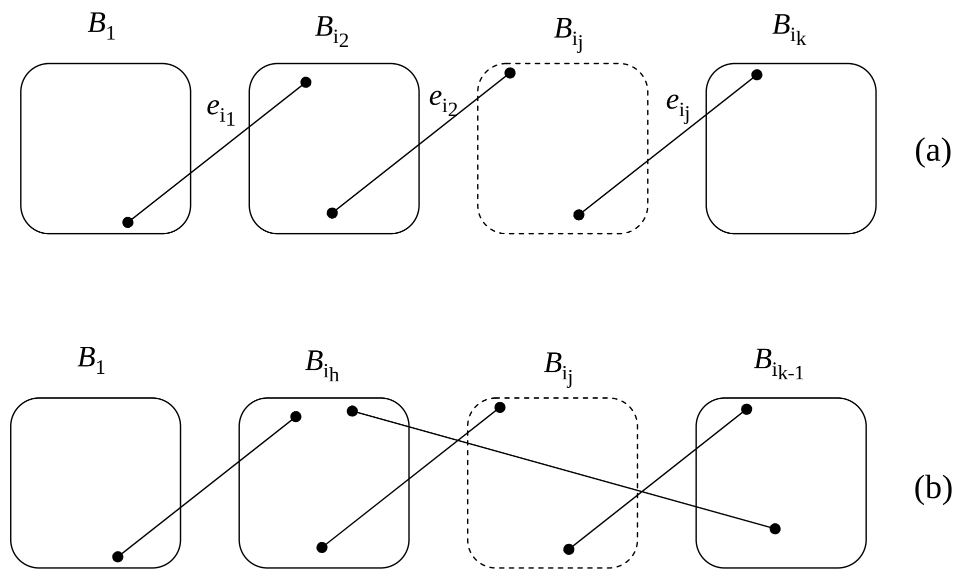

Figure 5: The connected bipartite components of $B - K$ and some edges of K connecting those bipartite components.

```
        end
    end
end
```

Theorem 3 *Algorithm FIND_ALGEBRAIC_SUBSET correctly finds a non-empty algebraic subset of a given edge set.*

Proof: The correctness of the algorithm follows from Lemma (5) and Lemma (6). □

5 Computational aspects

In this section we show how to solve efficiently the tasks of algorithm FIND_ ALGEBRAIC _SUBSET : finding the kernel of F, finding a solution of system (6) and checking the condition (i) and (ii) of Lemma (6) for an edge set.
Gusfield [9] gave an algorithm to find the kernel of a set F in the case of bipartite graphs. Let (P, N) be a bipartition of G. Direct all the edges of F from P to N thus obtaining a mixed graph G'. The algorithm is based on the following proposition

Proposition 1 *[9]. All the edges not in any strongly connected component of G' are in the kernel of F.*

By Proposition (1), in a bipartite graph the kernel of an edge set can be found using the following algorithm [9]:

```
FIND_KERNEL
input: A bipartite connected graph $G$ and a subset $F$ of $E(G)$
output: The kernel $K$ of $F$
begin

    Let $(P, N)$ be a bipartition of $G$;
    Direct all the edges $e \in F$ from $P$ to $N$. We thus obtain a mixed
    graph $G'$;
    Compute the strongly connected components of $G'$;
    Output the set of directed edges joining distinct strongly connected
    components of $G'$;

end
```

In order to prove Lemma (4) we give an algorithm that given a set F finds a solution of (6) for the kernel of F. First we see how to find a solution of (6), when the graph is bipartite. Next we extend the algorithm to non-bipartite graphs. We will use the concept of the *superstructure* [8] of a directed graph. If G is a directed graph its superstructure H is the directed graph where the node set is the set of strongly connected components of G and the edge set $E(H) = \{(u, v) :$ there exists in G at least one directed edge from component u to component $v\}$

```
COMPUTE_SUPPORT
input: A bipartite connected graph $G$ and a subset $F$ of $E(G)$
output: A solution of (6) for the kernel of $F$

begin

    Let $(P, N)$ be a bipartition of $G$;
    Direct all the edges $e \in F$ from $P$ to $N$. We thus obtain a mixed
    graph $G'$;
    Find the strongly connected components of $G'$ ;
    Let $H$ be the superstructure of $G'$;
    Let $(B_1, B_2, \ldots, B_h)$ be a topological sort of $H$ where $B_i$ is a
    strongly connected component of $G'$;
    Let $(P_i, N_i)$ be the bipartition of $B_i$ such that $P_i \subseteq P$ and $N_i \subseteq N$,
    $i = 1, \ldots, h$.;
    Set

        $c_v = \begin{cases} h - i & \text{if } v \in P_i \\ -h + i & \text{if } v \in N_i \end{cases}$

    Output $(c_v)_{v \in V(G)}$;
```

end

Lemma 7 *The algorithm COMPUTE_SUPPORT correctly finds a solution of (6).*

Proof: Let $(c_v)_{v\in V(G)}$ be the output of algorithm COMPUTE_ SUPPORT and let $\mathbf{k} = \sum_{v\in V(G)} c_v \mathbf{m}_v$. Let $(B_1, B_2, \ldots, B_h)$ be a topological sort of H where B_i is a strongly connected component of G'. First note that if $e = (u, v)$ is in a strongly connected component B_i then $c_u + c_v = 0$. Therefore $k(e) = 0$ if and only if $e = (u, v)$ is in a strongly connected component, and this is correct since, by Proposition (1), e is not in the kernel of F.
Now let (u, v) be an edge not in any strongly connected component of G'. Suppose that $u \in P$ and $v \in N$. Thus (u, v) is directed from u to v. If B_i is the component containing u and B_j the component containing v then B_i is before B_j in the topological sort of H, that is, $i < j$. Since $c_u = h - i$ and $c_v = -h + j$, then $k((u, v)) = c_u + c_v = h - i - h + j = j - i > 0$ as supposed to be. $\Box$

Algorithms FIND_KERNEL and COMPUTE _SUPPORT apply to bipartite graphs. In the case of non-bipartite graphs we can use what is called in [17] the *bipartite transform* of G which is a bipartite graph (this trasformation is also called *monotonization* in [11]). Then we can apply the above two algorithms to the bipartite transform to find both the kernel and a solution of (6). Here the details.

Let V' and V'' two copies of $V(G)$. If u is a node of $V(G)$ with u' and u'' we denote the copy of u in V' and V'' respectively. Then the bipartite transform of G is the graph H where $V(H) = V' \cup V''$ and

$$E(H) = \{(u', v'') : (u, v) \in E(G), u' \in V', v'' \in V''\} \cup \\ \cup \{(u'', v') : (u, v) \in E(G), u'' \in V'', v' \in V'\}$$

where for every edge (u, v), with $u \neq v$, of G we denote with (u', v'') and (u'', v') the two edges of the bipartite graph H in correspondence to (u, v). If (u, v) is a loop, that is $u = v$, then $(u', v'') = (u'', v') = (u', u'')$ and there is one single edge of H in correspondence to (u, v). Therefore the set

$$D = \{(u', v'') : (u, v) \in F, u' \in V', v'' \in V''\} \cup \\ \cup \{(u'', v') : (u, v) \in F, u'' \in V'', v' \in V'\}$$

is the subset of $E(H)$ in correspondence of F. Note that H is bipartite with bipartition (V', V''). Now let $\mathbf{N}$ be the incidence matrix of H and consider the following system of linear equations

$$\mathbf{N}\mathbf{z} = \mathbf{d} \tag{7}$$

where $d_{u'} := d_{u''} := b_u$ for all $u \in V(G)$ where b_u was defined in equation (4). Note that if $\mathbf{x}^*$ is a non negative solution of (3) then

$$z^*((u', v'')) := z^*((u'', v')) := x^*((u, v)) \qquad (u, v) \in E(G)$$

is a non negative solution of (7). The following proposition shows the relation between the graph G and its bipartite transform.

Proposition 2 *[17, 11] Let* $\mathbf{x}$ *be a non negative solution of (3) and* $\mathbf{z}$ *be a non negative solution of (7). Then*

$$z^{\#}((u', v'')) := z^{\#}((u'', v')) := x((u, v)) \qquad (u, v) \in E(G)$$

is a non negative solution of (7) and

$$x^{\#}((u, v)) = \frac{1}{2}[z((u', v'')) + z((u'', v'))] \qquad (u, v) \in E(G)$$

is a non negative solution of (3).

Note that H is a symmetric graph in the sense that, given a non negative solution $\mathbf{z}$ of (7), if we take $z^{\#}((u', v'')) := z((u'', v'))$ and $z^{\#}((u'', v')) := z((u', v''))$ for all $(u, v) \in E(G)$ then we obtain again a non negative solution $\mathbf{z}^{\#}$ of (7). It follows that the kernel K' of D is also symmetric, i.e. $(u', v'') \in K'$ if and only if $(u'', v') \in K'$. Therefore, by Proposition (2), we have the following

Proposition 3 *[17] The kernel of* F *is the set*

$$K = \{(u, v) : (u, v) \in E(G), (u', v'') \in K' \textit{ and } (u'', v') \in K'\}$$

where K' *is the kernel of* D*.*

The following Lemma gives a method to find a solution of (6) when G is a non-bipartite graph. It shows how to obtain from a solution of (6) for the kernel of D, a solution of (6) for the kernel of F. The solution of (6) for the kernel of D can be obtained using the algorithm COMPUTE_SUPPORT since H is a bipartite graph.

Lemma 8 *If* $(g_v)_{v \in V(H)}$ *is a solution of (6) for the kernel of* D *then*

$$(g_{v'} + g_{v''})_{v \in V(G)}$$

is a solution of (6) for the kernel of F*.*

Proof: For every node v of $V(G)$ let $c_v = g_{v'} + g_{v''}$. Let K' be the kernel of D. If $(u', v'') \in K'$ then $(u'', v') \in K'$, $g_{u'} + g_{v''} > 0$ and $g_{u''} + g_{v'} > 0$. Therefore $(u, v) \in K$ and

$$c_u + c_v = g_{v'} + g_{v''} + g_{u'} + g_{u''} > 0$$

Also if $(u', v'') \notin K'$ then $(u'', v') \notin K'$, $g_{u'} + g_{v''} = 0$ and $g_{u''} + g_{v'} = 0$. Therefore $(u, v) \notin K$ and

$$c_u + c_v = g_{v'} + g_{v''} + g_{u'} + g_{u''} = 0$$

To sum up $(c_v)_{v \in V(G)}$ is a solution of (6) for the kernel of F. □

The strongly connected components and a topological sort of a graph can be found using standard graph algorithms [7] and all take time linear in the size of the graph.

Remark 1 *The time complexity of COMPUTE_SUPPORT and FIND_KERNEL is linear in the size of the graph G.*

Example 2 Consider the graph G of Fig. 6. Let F be the set of bold edges. We have that $K = F$. Fig. 7 shows the bipartite transform H of G where the zero-weighted edges of D are directed. There are six strongly connected components of H. They are ordered from left to right. The coefficients of a solution of (6) are $c_1 = +5$, $c_2 = -5$, $c_3 = +3$, $c_4 = -3$, $c_5 = +1$ and $c_6 = -1$ as it is easily checked using algorithm COMPUTE_SUPPORT and Lemma 8.□

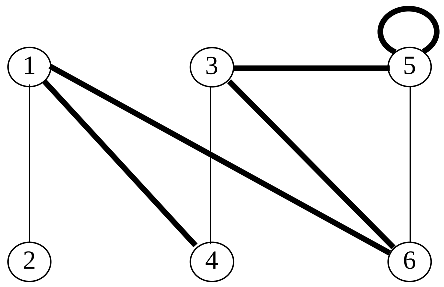

Figure 6: A graph and, in bold, a subset F of edges.

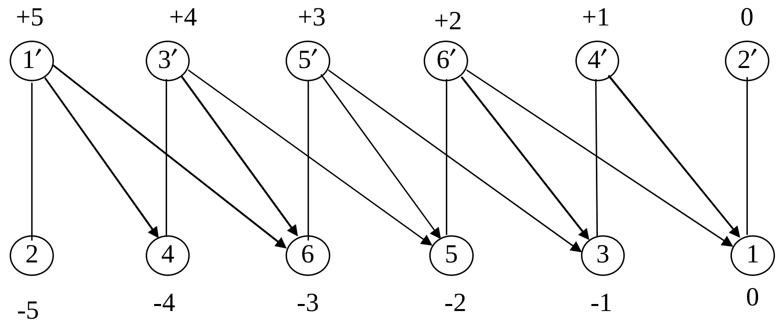

Figure 7: The bipartite transform H of the graph of Fig. 6. The edges of H in correspondence of F are directed from the upper part to the lower part.

In order to decide whether there exists a bipartite component of $G - K$ that satisfies the conditions (i) and (ii) of Lemma (6), we can proceed as follows. Let $B_1, \dots, B_h$ be the bipartite connected components of $G - K$. Suppose we have for each node v of B_i, $i = 1, \dots, h$ the following

$cn(v) \in \{1, \ldots, h\}$	the number i of the bipartite component B_i containing v
$side(v) \in \{1, 2\}$	one of the two sides of the bipartite component B_i containing v

and set

$$AT_j(B) = \begin{cases} 1 & \text{if at least one edge of } K \text{ is attached to side } j \text{ of } B_i \\ 2 & \text{if an edge of } K \text{ has both end point attached to side } j \text{ of } B_i \\ 0 & \text{otherwise;} \end{cases}$$

for $j \in \{1, 2\}$. For each link $e = (u_1, u_2)$ of K, if $side(u_1) = side(u_2)$ and $cn(u_1) = cn(u_2)$ then we set $AT_{side(u_1)}(B_i) := 2$. Otherwise if u_j, $j = 1, 2$ is attached to the side k of B_i then we set $AT_k(B_i) := 1$. Finally for each component B_i we check if $AT_1(B_i) + AT_2(B_i) = 1$. In this case conditions (i) and (ii) of Lemma (6) are satisfied. Since all those tasks take time linear in the size of the graph and by Remark 1, we have the following

Theorem 4 *Algorithm FIND_ALGEBRAIC_SUBSET has time complexity linear in the size of the graph.*

By Theorem (4), if $\mathbf{M}$ is the incidence matrix of a graph G we have a linear time algorithm to find a non-empty algebraic subset (if any) of a given edge set F. Moreover we can use the algorithm FIND_ALGEBRAIC_SUBSET to find a maximal algebraic subset of F as follows

```
MAXIMALLY
input: a graph $G$ and a subset $F$ of edges
output: a maximal algebraic subset of $F$

begin
   $M := \emptyset$;
   while $A$:= FIND_ALGEBRAIC_SUBSET ($G$, $F$) is not empty do
      begin
         $M := M \cup A$;
         $F := F - A$;
      end
   output $M$;
end
```

The time complexity of the algorithm MAXIMALLY is at most $|F|$ times the time complexity of FIND_ALGEBRAIC_SUBSET. Therefore MAXIMALLY takes at most quadratic time in the size of G.

6 Closing remarks

The NAS problem was first studied in the context of the inference problem of summary data. The NAS problem was proven to be *NP*-complete in the general case [19], i.e. when $\mathbf{M}$ is an arbitrary binary matrix. In this paper we showed that the NAS problem can be solved in linear time when $\mathbf{M}$ is the node-edge incidence matrix of a graph. If the edge set of the graph is algebraic, we can use the same algorithm to find a proper subset of $E(G)$ (if any) that is algebraic and contains a given edge set F. In fact if we find an algebraic subset A of $E(G) - F$ then it is not difficult to check that $E(G) - A$ is an algebraic superset of F when $E(G)$ is algebraic. Note that the edge set of every loopless graph is algebraic. At this point it is an open problem how to find an algebraic superset of a given edge set when the graph contains loops. Another open question is that of determining if there exists a polynomial time algorithm to solve the problem of finding a *maximum* algebraic subset of an edge set.

Acknowledgments

The author wishes to thank F.M. Malvestuto for his help and his many valuable comments and suggestions and the referees for their comments.

References

[1] N. R. Adam and J. C. Wortmann. Security-control methods for statistical databases: A comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[2] A. Bjorner, M. L. Vergnas, B. Sturmfels, N. White, and G. Ziegler. *Oriented Matroids.* Cambridge University Press, Cambrige, MA, 1993.

[3] M. C. Chen and L. McNamee. On the data model and access method of summary data management. *IEEE Trans. Knowl. Data Eng.*, 1(4):519–529, 1989.

[4] M. C. Chen, L. McNamee, and M. A. Melkanoff. A model of summary data and its applications in statistical databases. In M. Rafanelli, J. C. Klensin, and P. Svensson, editors, *SSDBM*, volume 339 of *Lecture Notes in Computer Science*, pages 356–372. Springer, 1988.

[5] F. Y. L. Chin and G. Özsoyoglu. Auditing and inference control in statistical databases. *IEEE Trans. Software Eng.*, 8(6):574–582, 1982.

[6] S. Cohen, W. Nutt, and Y. Sagiv. Rewriting queries with arbitrary aggregation functions using views. *ACM Trans. Database Syst.*, 31(2):672–715, 2006.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill, 1990.

[8] S. Even. *Graph Algorithms.* Computer Science Press, 1979.

[9] D. Gusfield. A graph theoretic approach to statistical data security. *SIAM J. Comput.*, 17(3):552–571, 1988.

[10] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[11] D. S. Hochbaum. Monotonizing linear programs with up to two nonzeroes per column. *Oper. Res. Lett.*, 32(1):49–58, 2004.

[12] M.-Y. Kao. Data security equals graph connectivity. *SIAM J. Discrete Math.*, 9(1):87–100, 1996.

[13] M.-Y. Kao. Efficient detection and protection of information in cross tabulated tables ii: Minimal linear invariants. *J. Comb. Optim.*, 1(2):187–202, 1997.

[14] M.-Y. Kao and D. Gusfield. Efficient detection and protection of information in cross tabulated tables i: Linear invariant test. *SIAM J. Discrete Math.*, 6(3):460–476, 1993.

[15] J. M. Kleinberg, C. H. Papadimitriou, and P. Raghavan. Auditing boolean attributes. In *PODS*, pages 86–91. ACM, 2000.

[16] F. M. Malvestuto. A universal-scheme approach to statistical databases containing homogeneous summary tables. *ACM Trans. Database Syst.*, 18(4):678–708, 1993.

[17] F. M. Malvestuto and M. Mezzini. A linear algorithm for finding the invariant edges of an edge-weighted graph. *SIAM J. Comput.*, 31(5):1438–1455, 2002.

[18] F. M. Malvestuto and M. Mezzini. Auditing sum queries. In D. Calvanese, M. Lenzerini, and R. Motwani, editors, *ICDT*, volume 2572 of *Lecture Notes in Computer Science*, pages 126–142. Springer, 2003.

[19] F. M. Malvestuto, M. Mezzini, and M. Moscarini. An analytical approach to the inference of summary data of additive type. *to appear in Theoretical Computer Science*.

[20] F. M. Malvestuto, M. Mezzini, and M. Moscarini. Auditing sum-queries to make a statistical database secure. *ACM Trans. Inf. Syst. Secur.*, 9(1):31–60, 2006.

[21] F. M. Malvestuto, M. Mezzini, and M. Moscarini. Minimal invariant sets in a vertex-weighted graph. *Theoretical Computer Science*, 362(1-3):140–161, 2006.

[22] F. M. Malvestuto and M. Moscarini. Query evaluability in statistical databases. *IEEE Trans. Knowl. Data Eng.*, 2(4):425–430, 1990.

[23] F. M. Malvestuto and M. Moscarini. Computational issues connected with the protection of sensitive statistics by auditing sum queries. In M. Rafanelli and M. Jarke, editors, *SSDBM*, pages 134–144. IEEE Computer Society, 1998.

[24] F. M. Malvestuto and M. Moscarini. An audit expert for large statistical databases. In *Conf. on Statistical Data Protection*, pages 29–43. EUROSTAT, 1999.