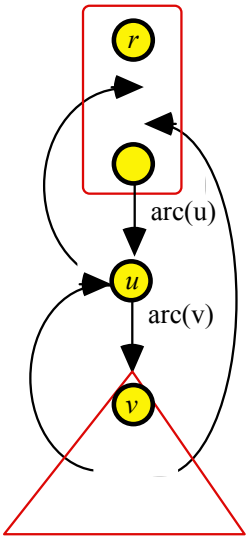
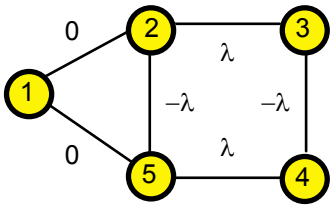
University of Rome
La Sapienza
Department of Computer Science

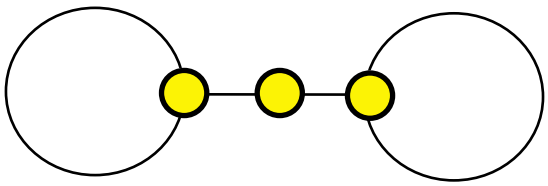
Mauro Mezzini

ANSWERING SUM-QUERIES : A SECURE AND EFFICIENT APPROACH

Advisor: Prof. F. M. Malvestuto







to my sons **Oskar** and **Lorenzo**
and to my wife **Arenika**

Content

Introduction

The use of computer systems for collecting data has exploded during the last 40 years allowing the storage and the query of an immense quantity of information. Governmental organizations, census institutes, industries and corporations usually have large information systems that collect and store information about individuals. Such databases can be very large and are often used to produce statistics over certain population. Statistics can ask for a sum, average, maximum or minimum of the value of a numerical attribute for certain category of individual (i.e. a set of records). These databases are referred in this paper to as *statistical databases* (SDB).
A statistical database is an ordinary database whose users are allowed to ask for statistical information. 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 and the categories can be specified by logical formulae involving the fields of the file. 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 [9, 10], 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 thesis, as in [9, 10] 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 *response* 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.

We address two issues concerning the processing of sum-queries in large statistical database system. The first one is concerning the privacy of the data stored in the database. Collecting and storing information about individuals raises concerns on the compromise of individual privacy. Insurance or employer could collect information to discriminate between who to insure or employ. For example suppose a medical database has a boolean attribute storing, for each individual, whether she or he has got or has not got a certain disease. Medical researcher could ask the sum of the boolean attribute over certain categories to assess the spreading of this disease. The risk is, if the category is small or, worse, is a singleton, that such statistic could lead to the disclosure of

confidential information. The problem here is to allow the query-system to answer statistical queries as long as no confidential information is, directly or indirectly, given to the users. We refer in this paper to this problem as the *security problem*.
The second issue we address in this thesis is related to the performance of processing of sum-queries in a large statistical database. As in the example above in a medical database a user could ask for the number of male people affected by a certain disease. To produce the answer require scanning exhaustively all the records falling in that category. Clearly this process can takes a lot of time if the database is very large. The idea to speed up this process is to pre-compute a set of statistics when the database is created and use these pre-computed statistics to find whole or part of the answer of a query **q** thus saving time. We refer to this problem as the *performance problem*.
We will show that this two problems are related each other. In fact technique used to solve the first problem can be used to solve the second.
The idea is that after answering a set **V** of queries or equivalently given a set **V** of pre-computed queries, one can combine the information conveyed by **V** to obtain further information, not explicitly given by the database. We also see that this information can be obtained in a very efficient manner.

The security problem
Answering 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* [17, 61, 62]. Let **q** be a sum-query of the type **sum**(`Balance`) on our bank database. If `Balance` is a confidential field and the response 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 response 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 *snooper*, 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 snooper, 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.

*Example 0.*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

$\mathbf{q}_1$ select **sum**(Balance)
from DEPOSITOR
where Gender = Male and Age < 45

$\mathbf{q}_2$ select **sum**(Balance)
from DEPOSITOR
where Age < 25 or Gender = Male and Age ≥ 45

$\mathbf{q}_3$ select **sum**(Balance)
from DEPOSITOR
where Age ≥ 45 or Gender = Male and 25 ≤ Age < 45

$\mathbf{q}_4$ select **sum**(Balance)
from DEPOSITOR
where Gender = Female and Age < 45

and assume that the values of $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$ and $\mathbf{q}_4$ 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

$$\begin{cases} x_1 + x_2 = 24 \\ x_1 + x_3 + x_4 = 29 \\ x_2 + x_3 + x_6 = 18 \\ x_4 + x_5 = 12 \end{cases} \qquad (0.1)$$

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:

V_1 = (Gender = Male and Age < 25)
V_2 = (Gender = Male and 25 ≤ Age < 45)
V_3 = (Gender = Male and Age ≥ 45)
V_4 = (Gender = Female and Age < 25)
V_5 = (Gender = Female and 25 ≤ Age < 45)
V_6 = (Gender = Female and Age ≥ 45)

Note that the value of the intrusive sum-query $\mathbf{q}$ is represented by x_1. Since the general solution of the equation system (0.1) is

$$(x_1 = \mathsf{a}, x_2 = 24 - \mathsf{a}, x_3 = 29 - \mathsf{a} - \mathsf{b}, \; x_4 = \mathsf{b}, x_5 = 12 - \mathsf{b}, x_6 = -35 + 2\mathsf{a} + \mathsf{b})$$

where a and b are two arbitrary real numbers, the value of x_1 is not determined and, hence, the response of $\mathbf{q}$ is protected. Suppose now that a new sum-query arrives:

$\mathbf{q}_5$ select **sum**(Balance)
from DEPOSITOR
where Gender = Female and Age ≥ 25

and assume that the response of $\mathbf{q}_5$ is 7. If the QAS answers $\mathbf{q}_5$, then the amount of information conveyed by the answers to $\mathbf{q}_1$, …, $\mathbf{q}_5$ is obtained by adding the equation $x_5 + x_6 = 7$ to equation system (0.1). The general solution is now

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

so that the response of $\mathbf{q}$ is disclosed. Therefore, the QAS should not answer $\mathbf{q}_5$ by releasing its response, but should issue the set of feasible values of $\mathbf{q}_5$ consistent with the answers to $\mathbf{q}_1$, …, $\mathbf{q}_4$, that is, the whole set of real numbers.

Auditing sum-queries raises some computational problems (recognizing evaluable sum-queries, updating the information model), whose solutions depend on the data type of the response variable. If it is of real type, then standard algebraic methods can be used to solve all of them efficiently;
If it is of non negative real type then a natural approach for solving the security problem consists in resorting to standard linear-programming algorithms (e.g., the simplex method [13, 19]). Unfortunately, none of them is polynomial even if they are polynomial on the average and have good performances in practice [13, 19]. Therefore, in order to solve the computational problems raised by the security of the SDB, it is convenient to make a parsimonious use of standard linear-programming algorithms and "there is considerable interest in finding alternative techniques" [22].
Accordingly we devise procedures and algorithms that minimise the use of linear programming and in a special case we see how to solve the security problem making use only of polynomial graphs and networks algorithms.

The performance problem
When a sum-query about a numeric attribute selects a large number of records, the query-answering system should scan through a large portion of the database exhaustively, and thus inefficiently. A better approach consists in trying to answer the sum-query using materialised sum-views that correspond to more frequently asked sum-queries.
Previous approaches to planning the execution of sum-queries [9, 10, 14, 26, 28, 31, 56] aim at rewriting the query set of query in terms of the query sets of views using as "rewriting language" the very query language. For example, in [9, 10] a sum-query is called derivable from a list of views if the query set can be expressed by disjoint union and proper difference of the query sets of views. However, in general the situation is more complex and the choice of the very query language as rewriting language is poor as we can see in the following example

Example 0.1 (continued). Consider the file `DEPOSITOR` and suppose the query system has a set $\mathbf{V}=\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_5\}$ of materialised sum-views. Then the amount of information conveyed by $\mathbf{V}$ is modelled by the following equation system

$$\begin{cases} x_1 + x_2 = 24 \\ x_1 + x_3 + x_4 = 29 \\ x_2 + x_3 + x_6 = 18 \\ x_4 + x_5 = 12 \\ x_5 + x_6 = 7 \end{cases} \qquad (0.2)$$

As stated before the general solution of system 0.2 is

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

clearly the query $\mathbf{q}$ can be answered without accessing the database. Also the query asking for the value of the sum for category V_2 can be answered without accessing the database since the value of x_2 is uniquely determined under system 0.2. Note that not only x_1 and x_2 are uniquely determined but also the value of $x_3 + x_6$ and the value of $x_3 + x_4$ are uniquely determined. It should be noted too

that $x_1, x_2, x_3 + x_6$, and $x_3 + x_4$ are uniquely determined no matter what was the response of the five query.
We see in the Example 0.1 that the contents of a given materialised sum-views can be modelled by a linear equation system, which naturally leads to the notions of "evaluability" of a sum-query (the answer is uniquely determined by the values of the materialised views) and of "derivability" (the answer is uniquely determined no matter which values the materialised views may assume). Accordingly, two query-execution plans for answering a sum-query are proposed and the connected computational aspects are discussed. One (called *plan D*), which is not sensitive to the values the materialised views, first checks if the sum-query is derivable and, if this is not the case, finds a "contained" sum-query that is derivable and accesses the database only to evaluate the residual part of the original sum-query. The other (called *plan E*), which is sensitive to the values the materialised views, first checks if the sum-query is evaluable from materialised views and, if this is not the case, finds a "contained" sum-query that is evaluable and accesses the database only to evaluate the residual part of the original sum-query. As to plan *D*, it is proven that (1) the problem for recognising derivable sum-queries is independent of the domain of the response variable and can be solved in polynomial time, and (2) the problem of finding a "maximally contained" sum-query is *NP*-hard. As to plan *E*, it is proven that (1) the problem for recognising evaluable sum-queries is dependent on the domain of the response variable, can be solved in polynomial time if the domain of the response variable is the set of reals, or of integers, or of nonnegative reals, but is co*NP*-hard if the domain of the response variable is the set of nonnegative integers, and (2) the problem of finding a "maximally contained" sum-query remains *NP*-hard. Finally we will show a special cases where the problem of finding a "maximally contained" sum-query can be solved in polynomial time.

Organization of the work
The work is organised in the following manner: Chapter 1 discusses the information model called the *inference model* used to represent the amount of information conveyed by a set $\mathbf{V}$ of answered sum-queries. This mathematical model is used to solve both the performance and the security problems. Part of the work of this Chapter was used in the work made in collaboration with F. M. Malvestuto and Moscarini M. "*Answering Statistical Sum-Queries Using Materialised Sum-Views: An Analytic Approach*" submitted for publication in 2005 to TODS [51].
Chapter 2 discusses the performance problem. It is described how to use the set of $\mathbf{V}$ of answered queries for efficiently find the answer to new queries. In Chapter 3, the problem of answering sum-queries without disclosing, directly or indirectly, confidential information is addressed. Some original results described in this Chapter were used in the work "*Auditing Sum-Queries to Make a Statistical Database Secure*" [50] made in collaboration with Malvestuto F. M. and Moscarini M. and accepted for publication to TISSEC (2005). In Chapter 4, we start to deal with a *graphical* information model. The information model is said to be graphical when the coefficient matrix of the system of linear equations is the node-edge incidence matrix of a graph. So Chapter 4 is devoted to solve the important problem of finding the invariant edges of an edge-weighed graph. Almost the entire Chapter 4 comes from the work made in collaboration with Malvestuto F. M, "*A Linear time algorithm for finding the invariant edges of edge-weighted graph"* [41]. In section 4.7 of Chapter 4 we discuss the invariant edges problem for an edge-weighted graph where the weights of the edges take their values from a commutative (abelian) group. This section was derived from the work made in collaboration with Malvestuto F. M. "*Privacy preserving and data mining in an on-line statistical database of additive type*" [43]. In Chapter 5, we solve the problem of finding the feasible range of the sum of the weights of a set of edges of an edge-weighted graph. This Chapter is derived from the work made in collaboration with Malvestuto F. M. "*Auditing sum-queries*"[42]. Chapter 6 solve the problem of finding a "maximally contained" sum-query when the information model is graphical. This problem is proved (see Section 2.1.1) to be an *NP*-hard problem in the general case. Finally Chapther 7 a carachterization of algebraic edge sest on graphs is given.

Acknowledgements
First of all I wish to thank my advisor, Prof. **F. M. Malvestuto**, for having given me the opportunity to start this scientifical work which eventually has made possible make this thesis and has enabled me to attend the Ph.D. course. I wish also to thank him for his constant help, his many valuable comments and suggestions. Next I wish to thank Prof**. Moscarini M.**. I become graduate under her supervision and many of the results of Chapter 4 come from the work done with her. Special thanks goes to Prof. **Pavel Serafimov** for his help in correcting the English language's errors of the original manuscript.

Chapter 1

The inference model

1.1 Introduction

Consider an ordinary database containing information on individuals (persons, households, companies, organizations *et cetera*), which either is static (e.g., a census database) or is updated periodically (e.g., daily, weekly, monthly, …). Statistical queries ask for summary statistics over categories of individuals, possibly for "analytic processing" purposes. For example, consider a database which is monthly updated and contains a relation name `Personnel` with scheme {NAME, SSN, GENDER, AGE, DEPT, SALARY}. A statistical query may ask for the summary statistic on the attribute SALARY (using aggregate functions such as `sum`, `count`, …) for some group of employees selected using the attributes GENDER, AGE and DEPT but not NAME and SSN which are private attributes. We focus on *statistical sum-queries* such as

$\mathbf{q}_1$:
```
select sum(SALARY)
from Personnel
where AGE = young
```

$\mathbf{q}_2$:
```
select sum(SALARY)
from Personnel
where GENDER = male and AGE = young
```

(Sum-queries containing the `group-by` clause will not be taken into account explicitly since they are equivalent to sets of sum-queries that the `group-by` clause is missing from). In a sum-query $\mathbf{q}$ such as $\mathbf{q}_1$ or $\mathbf{q}_2$, the attribute SALARY is called the *response variable* and the attributes involved in the `where` clause are called the *categorical variables* of $\mathbf{q}$. The `where` clause specifies a relation over the set of categorical variables of $\mathbf{q}$, we call the *target* of $\mathbf{q}$ and denoted by Q, which is computed when $\mathbf{q}$ is compiled. Accordingly, the targets of $\mathbf{q}_1$ and $\mathbf{q}_2$ are the one-tuple relation {(young)} over {AGE} and the one-tuple relation {(male, young)} over {GENDER, AGE}, respectively. If I is the current database relation of name `Personnel`, then the answer to a sum-query such as $\mathbf{q}_1$ and $\mathbf{q}_2$ is given by

$$\sum_t t(\text{SALARY})$$

the summation being extended over the tuples t in I that "fall" in the target of $\mathbf{q}$, that is, that are qualified by the condition involved in the `where` clause.
Suppose that at the beginning of a new life period of the database, in order to speed up the process of evaluating incoming sum-queries, the database administrator decides to create certain sum-views over SALARY, that is, named sum-queries with response variable SALARY such as

```
create view v1 as
select sum(SALARY)
from Personnel
where AGE in {middle, old}

create view v2 as
select sum(SALARY)
from Personnel
where AGE in {young, old}
```

```
create view v3 as
select sum(SALARY)
from Personnel
where AGE in {young, middle}
```

Also the database system can store all the answered queries for auditing purposes. The set of all answered queries can be seen as a set of sum-views.
Let $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ and let $\mathbf{v} = (v_1, v_2, v_3)$, where v_1, v_2 and v_3 are the answers to $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ computed on the current relation I of name `Personnel`. When a user will submit a sum-query $\mathbf{q}$ with response variable SALARY, a question that naturally arises is whether the query-answering system can answer $\mathbf{q}$ from $\mathbf{v}$ without accessing the database.
Here we describe the inferential model (also called the information model) that allow us to evaluate query $\mathbf{q}$ from a set of sum-views or equivalently, from a set of previously answered sum-queries. Informally, we say that $\mathbf{q}$ is *evaluable* from $\mathbf{v}$ if the answer to $\mathbf{q}$ is uniquely determined by $\mathbf{v}$, and is *derivable* from $\mathbf{V}$ if it is *evaluable* from any $\mathbf{v}$. Accordingly, the sum-query $\mathbf{q}_1$ is derivable from $\mathbf{V}$ since, for every instance of the database relation of name `Personnel`, the answer to $\mathbf{q}_1$ can be obtained as

$$\frac{1}{2}\,(-v_1 + v_2 + v_3)$$

On the other hand, it is easily seen that the sum-query $\mathbf{q}_2$ is not derivable from $\mathbf{V}$ and that, under the realistic assumption that the attribute SALARY is of nonnegative type, $\mathbf{q}_2$ is evaluable from $\mathbf{v}$ only if $v_1 = v_2 + v_3$ for, then, the answer to $\mathbf{q}_1$ is `0` and, hence, also the answer to $\mathbf{q}_2$ must be `0`.

1.2 View bases

Let us assume that we are given a relation name `Rel` whose scheme contains a numeric attribute σ with domain d. Typically, d is the set `R` of reals, or the set `Z` of integers, or the set $\mathtt{R}^+$ of nonnegative reals or the set $\mathtt{Z}^+$ of nonnegative integers. Let R be the set of attributes in the scheme of `Rel` that can be used to ask statistical queries on σ. The values of each attribute in R are assumed to be mutually exclusive and globally exhaustive from a semantic viewpoint. Let S be a nonempty subset of R. An element of the domain of S, written $dom(S)$, is an *S-cell*, and a set of S-cells is an *S-category*; accordingly, every S-category is a relation over S. For any subset S' of R that contains S, the *extension* of an S-category Q to S' is the S'-category $dom(S'–S) \times Q$ if $S' \neq S$, and is the S-category Q otherwise.
We will consider sum-queries with response variable σ in "positive normal form" [44] such as

```
q:  select sum(σ)
    from Rel
    where (A1, …, Ar) in Q
```

where Q, the target of $\mathbf{q}$, is an S-category with $S = \{A_1, \ldots, A_r\} \subseteq R$. Thus, the condition involved in the `where` clause stands for the logical formula

$$\bigvee_{c \in Q}\ A_1 = c(A_1) \wedge \ldots \wedge A_r = c(A_r).$$

We also consider the sum-query

```
u:  select sum(σ)
    from Rel
```

which has no categorical variables. By convention, we say that the target of **u** is the *universal category* and the extension of the universal category to any nonempty subset S of R is $dom(S)$.

Such sum-queries with response variable σ will be referred to as σ-*queries*.

Remark 1.1 If the target Q of a σ-query **q** is an S-category which is the extension to S of some S'-category Q', $S' \subset S$, then **q** is equivalent to the σ-query **q′** with target Q' in that the answers to **q** and **q′** computed on every instance of the database relation of name `Rel` do coincide.

Note that, owing to the assumptions of mutual exclusiveness and global exhaustiveness, every sum-query **q** with response variable σ has associated with it exactly one σ-query which is equivalent to **q**.

An *sum-view* **v** over σ (a σ-*view*, for short) is a named σ-query (over the scheme of `Rel`) whose answer is computed by the query-answering system as soon as the database is updated. A *view base* over σ (a σ-*view base*, for short) is a nonempty list of σ-views. We can also see a view base as a set of answered σ-queries. Let $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ be a σ-view base. Let S_i be the set of categorical variables of $\mathbf{v}_i$, $1 \le i \le n$, and let $S = \cup_{i=1,\ldots,n} S_i$ be the set of categorical variables of **V**. Let V_i be the extension to S of the target of $\mathbf{v}_i$. Then, it is uniquely determined (e.g., see [38], pages 132-133]) the coarsest $\boldsymbol{X}$ of the partitions of $\Omega = \cup_{i=1,\ldots,n} V_i$. such that each V_i can be recovered by taking the union of one or more classes of $\boldsymbol{X}$. The partition $\boldsymbol{X}$ is composed by the non empty elements, each one obtained as

$$\cap_{i \in \mu} V_i - \cup_{i \notin \mu} V_i \qquad\qquad \mu \in 2^{[n]}$$

where by $[n]$ we denote the set $\{1, \ldots, n\}$ and $2^{[n]}$ is the set of all subset of $[n]$. We call $\boldsymbol{X}$ the *classification system* of **V**.

Remark 1.2 A category V_i is itself a class of $\boldsymbol{X}$ if and only if, for each i', either $V_i \subseteq V_{i'}$ or $V_i \cap V_{i'} = \emptyset$.

Let $\boldsymbol{X} = \{X_1, \ldots, X_m\}$. Then, each V_i can be expressed as

$$V_i = \cup_{j \in J_i} X_j \qquad (i \in [n])$$

where $J_i = \{j \in [m]: X_j \subseteq V_i\}$. The set of these equalities defines the *dictionary* of **V**. In what follows, we arbitrarily choose an ordering of the views in **V** and of the classes of $\boldsymbol{X}$. Thus, the dictionary of **V** is fully specified by the $n \times m$ dimensional 0-1 matrix **H** with entries

$$h_{ij} = \begin{cases} 1 & \text{if } j \in J_i \\ 0 & \text{else} \end{cases}$$

which will be referred to as the *dictionary matrix* of **V**.

Example 1.1. Consider a database which is monthly updated and contains a relation name `Personnel` with scheme {`NAME`, `SSN`, `GENDER`, `AGE`, `DEPT`, `SALARY`}. We assume that domains of `GENDER`, `AGE` and `DEPT` are {`female`, `male`}, {`young`, `middle`, `old`} and {`A`, `B`, `C`, `D`, `E`, `F`, `G`, `H`, `I`}, respectively. Suppose that, at the beginning of a certain month, the database administrator decides to utilize the following eight views:

```
create view v1 as
select sum(SALARY)
from Personnel
```

```
create view v2 as
select sum(SALARY)
from Personnel
where DEPT in {A,B}
```

```
create view v3 as
select sum(SALARY)
from Personnel
where DEPT in {A,C,D,E}
```

```
create view v4 as
select sum(SALARY)
from Personnel
where DEPT in {F,G}
```

```
create view v5 as
select sum(SALARY)
from Personnel
where DEPT in {H,I}
```

```
create view v6 as
select sum(SALARY)
from Personnel
where DEPT in {B,C,F}
```

```
create view v7 as
select sum(SALARY)
from Personnel
where DEPT in {D,H}
```

```
create view v8 as
select sum(SALARY)
from Personnel
where DEPT in {E,G,I}
```

The set of categorical variables of the view base $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_8)$ is $S = \{\texttt{DEPT}\}$ and the extensions $V_1, \ldots, V_8$ to S of the targets of $\mathbf{v}_1, \ldots, \mathbf{v}_8$ are the following relations over S:

$V_1 = \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}, \texttt{E}, \texttt{F}, \texttt{G}, \texttt{H}, \texttt{I}\}$
$V_2 = \{\texttt{A}, \texttt{B}\}$
$V_3 = \{\texttt{A}, \texttt{C}, \texttt{D}, \texttt{E}\}$
$V_4 = \{\texttt{F}, \texttt{G}\}$
$V_5 = \{\texttt{H}, \texttt{I}\}$
$V_6 = \{\texttt{B}, \texttt{C}, \texttt{F}\}$
$V_7 = \{\texttt{D}, \texttt{H}\}$
$V_8 = \{\texttt{E}, \texttt{G}, \texttt{I}\}$

The classes of the classification system of $\mathbf{V}$ are:

$X_1 = \{\texttt{A}\}$ $X_2 = \{\texttt{B}\}$ $X_3 = \{\texttt{C}\}$
$X_4 = \{\texttt{D}\}$ $X_5 = \{\texttt{E}\}$ $X_6 = \{\texttt{F}\}$
$X_7 = \{\texttt{G}\}$ $X_8 = \{\texttt{H}\}$ $X_9 = \{\texttt{I}\}$

The dictionary of $\mathbf{V}$ is

$$V_1 = X_1 \cup X_2 \cup X_3 \cup X_4 \cup X_5 \cup X_6 \cup X_7 \cup X_8 \cup X_9$$
$$V_2 = X_1 \cup X_2$$
$$V_3 = X_1 \cup X_3 \cup X_4 \cup X_5$$
$$V_4 = X_6 \cup X_7$$
$$V_5 = X_8 \cup X_9$$
$$V_6 = X_2 \cup X_3 \cup X_6$$
$$V_7 = X_4 \cup X_8$$
$$V_8 = X_5 \cup X_7 \cup X_9$$

and the dictionary matrix of $\mathbf{V}$ is

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$ ■

Let $\mathbf{v} = (v_1, \ldots, v_n)$ be a vector of d^n, where v_i stands for a possible value of $\mathbf{v}_i$. For each class X_j of $\boldsymbol{X}$, $j \in [m]$, let us introduce a variable x_j which stands for the (*a priori* unknown) answer to the σ-query with target X_j. Then, the amount of information conveyed by $\mathbf{v}$ can be described by the following linear equation system

$$\sum_{j \in J_i} x_j = v_i \qquad (i \in [n])$$

which we denote by $\Sigma(\boldsymbol{X}, \mathbf{v})$ and re-write in compact form as

$$\mathbf{H}\,\mathbf{x} = \mathbf{v} \qquad (1.1)$$

where $\mathbf{x} = (x_1, \ldots, x_m)$. A solution $\mathbf{x}$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$ is *d-feasible* if $x_j \in d$ for each $j \in [m]$, and the equation $\Sigma(\boldsymbol{X}, \mathbf{v})$ is *d-consistent* if it admits at least one *d*-feasible solution. If this is the case, then we call $\mathbf{v}$ an *instance* the σ-view base $\mathbf{V}$. Note that, if is I be the current database relation of name `Rel` and each v_i is the value of $\mathbf{v}_i$ computed on I, then $\mathbf{v}$ is definitely an instance of $\mathbf{V}$ and I determines a *d*-feasible solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$, we call the *true solution* for I.

Example 1.1 (continued). Consider the instance $\mathbf{v} = (v_1, \ldots, v_8)$ of $\mathbf{V}$ where

$v_1 = 22$ $\quad v_2 = 4$ $\quad v_3 = 6$ $\quad v_4 = 8$
$v_5 = 4$ $\quad v_6 = 10$ $\quad v_7 = 10$ $\quad v_8 = 4$

Then, $\Sigma(\boldsymbol{X}, \mathbf{v})$ reads

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 = 22 \\ x_1 + x_2 = 4 \\ x_1 + x_3 + x_4 + x_5 = 6 \\ x_6 + x_7 = 8 \\ x_8 + x_9 = 4 \\ x_2 + x_3 + x_6 = 10 \\ x_4 + x_8 = 10 \\ x_5 + x_7 + x_9 = 4 \end{cases}$$

If the domain of SALARY is R (or Z), then the general feasible solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$ has

$x_1 = 0$ $\quad x_2 = 4$ $\quad x_3 = \lambda$

$x_4 = 6+\mu$ $\quad x_5 = -\lambda-\mu$ $\quad x_6 = 4-\lambda$
$x_7 = 4+\lambda$ $\quad x_8 = 4-\mu$ $\quad x_9 = \mu$

where (λ, μ) is any couple of reals (integers, respectively).
If the domain of SALARY is R^+ or Z^+, then $\Sigma(\boldsymbol{X}, \mathbf{v})$ has exactly one feasible solution:

$x_1 = 0$ $\quad x_2 = 4$ $\quad x_3 = 0$
$x_4 = 6$ $\quad x_5 = 0$ $\quad x_6 = 4$
$x_7 = 4$ $\quad x_8 = 4$ $\quad x_9 = 0$ ■

Let J be a subset of $[m]$, and let $\mathbf{b}$ be the *characteristic vector* of J, that is, the m-dimensional 0-1 vector defined as

$$b_j = \begin{cases} 1 & \text{if } j \in J \\ 0 & \text{else} \end{cases}$$

We say that the sum-expression

$$\sum_{j \in J} x_j$$

is a *d-invariant* of $\Sigma(\boldsymbol{X}, \mathbf{v})$ if it assumes the same value for every d-feasible solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$ or, equivalently, if

$$\textbf{inf } \{(\mathbf{b}, \mathbf{x}): \mathbf{H}\,\mathbf{x} = \mathbf{v}, \mathbf{x} \in d^m\} = \textbf{sup } \{(\mathbf{b}, \mathbf{x}): \mathbf{H}\,\mathbf{x} = \mathbf{v}, \mathbf{x} \in d^m\} \qquad (1.2)$$

where $(\mathbf{b}, \mathbf{x})$ denotes the inner product. We now introduce a special class of d-invariants. We say that $\mathbf{b}$ belongs to the *row space* of $\mathbf{H}$ if $\mathbf{b}$ can be expressed as a linear combination of rows of $\mathbf{H}$, that is, if the following equation system

$$\mathbf{H}^{\mathsf{T}}\,\mathbf{a} = \mathbf{b}, \qquad (1.3)$$

where $\mathbf{H}^{\mathsf{T}}$ denotes the transpose of $\mathbf{H}$, admits a real-valued solution $\mathbf{a} = (a_1, \ldots, a_n)$. Equivalently, $\mathbf{b}$ belongs to the row space of $\mathbf{H}$ if and only if $\mathbf{b}$ is orthogonal to the null space of $\mathbf{H}$, that is, if and only if the inner product $(\mathbf{b}, \mathbf{z}) = 0$ for every solution $\mathbf{z}$ of the homogeneous equation system $\mathbf{H}\,\mathbf{z} = \mathbf{0}$.

Lemma 1.1 Let $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ a σ-view base, d the domain of σ, $\mathbf{H}$ the dictionary matrix of $\mathbf{V}$ and J a subset of $[m]$. If $d \subseteq \mathsf{R}$ and the characteristic vector of J belongs to the row space of $\mathbf{H}$ then, for every instance $\mathbf{v}$ of $\mathbf{V}$, the sum-expression $\sum_{j \in J} x_j$ is a d-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$.
Proof. Assume that the characteristic vector of J can be written as a linear combination of rows of the dictionary matrix $\mathbf{H}$ with coefficients $a_1, \ldots, a_n$. Then, for every instance $\mathbf{v}$ of $\mathbf{V}$ and for every real-valued solution $\mathbf{x}$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$, one has

$$(\mathbf{b}, \mathbf{x}) = (\mathbf{H}^{\mathsf{T}}\,\mathbf{a}, \mathbf{x}) = (\mathbf{a}, \mathbf{H}\,\mathbf{x}) = (\mathbf{a}, \mathbf{v}) = \text{const.}$$

The statement then follows from the fact that, for every $d \subseteq \mathsf{R}$, every d-feasible solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$ is also a real-valued solution. □

Example 1.1 (continued). Consider the two (elementary) sum-expressions x_1 and x_2 corresponding to $J = \{1\}$ and $J = \{2\}$. If $\mathbf{b}_1$ and $\mathbf{b}_2$ denote the characteristic vectors of these two singletons, then $\mathbf{b}_1$ and $\mathbf{b}_2$ belong to the row space of the dictionary matrix **H** since

$$\mathbf{b}_1 = \mathbf{h}_1 - \mathbf{h}_6 - \mathbf{h}_7 - \mathbf{h}_8$$
$$\mathbf{b}_2 = -\mathbf{h}_1 + \mathbf{h}_2 + \mathbf{h}_6 + \mathbf{h}_7 + \mathbf{h}_8$$

where $\mathbf{h}_i$ denotes the i-the row of **H**. Moreover, it is easy to see that, for none of the remaining seven variables, the characteristic vector of the corresponding singleton belongs to the row space of **H**. By Lemma 1.1, both variables x_1 and x_2 are d-invariants of $\Sigma(\boldsymbol{X}, \mathbf{v})$ for every instance **v** and for every subset d of R. Thus, for the instance **v** = (22, 4, 6, 8, 4, 10, 10, 4) of **V**, one has $x_1 = 0$ and $x_2 = 4$. ■

1.3 The updating procedure for a view base

Given a σ-view base **V**, let **v** be a σ-view and let v the value of **v**. In this section, we see how to update the classification system, the dictionary matrix and its associated equation system when we add to **V** the new σ-view **v**. In fact computing all the non empty set in the form of

$$\cap_{i\in\mu} V_i - \cup_{i\notin\mu} V_i \qquad\qquad \mu\in 2^{[n]}$$

can be time consuming even when the value of n is small. A better approach is to construct the classification system incrementally. Also we give some basic definition needed in the next Section.
Let **V** = ($\mathbf{v}_1$, …, $\mathbf{v}_n$), let S be the set of categorical variables of **V**, let $\boldsymbol{X} = \{X_1, \ldots, X_m\}$ be the classification system of **V**, and let S' be the union of S with the set of categorical variables of **v**. Let X'_j be the extension of X_j to S', and let Q be the extension to S' of the target of **v**. Let $\Omega = \cup_{j=1,\ldots,m} X'_j$ (see Figure 1.1) and let:

$$J_o = \{j \in [m]: Q \cap X'_j = \emptyset\} \qquad J = \{j \in [m]: X'_j \subseteq Q\} \qquad J' = [m] - (J_o \cup J)$$

The sets J and J' will be referred to as the Ω-*support* and the Ω-*cosupport* of **v**.

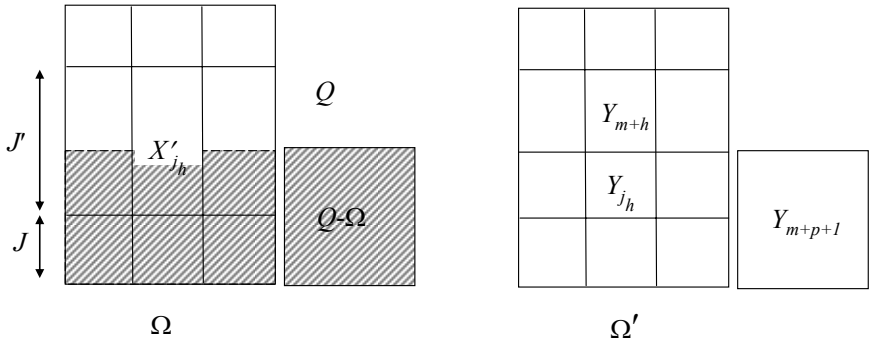


Figure 1.1

Let $p = |J'|$ and, if $p > 0$, let $J' = \{j_1, \ldots, j_p\}$. Let $\Omega' = \Omega \cup Q$ and let $\boldsymbol{Y}$ be the partition of Ω' defined as follows (see Figure 1.1):

If $p > 0$, then set $Y_{m+h} := X'_{j_h} - Q$ and $Y_{j_h} := Q \cap X'_{j_h}$ for $h = 1, \ldots, p$.

If $\Omega' \neq \Omega$, then set $Y_{m+p+1} := Q - \Omega$

Then $\boldsymbol{Y}$ is the classification system of $\mathbf{V}' = \mathbf{V} \cup \{\mathbf{v}\}$.
Without loss of generality, henceforth we assume that $J' \neq \emptyset$ and $\Omega' \neq \Omega$. Let $\mathbf{y}$ be an $(m+p+1)$-dimensional variable, where $p = |J'|$, and let $\Sigma(\boldsymbol{Y}, \mathbf{v}')$ be the equation system

$$\mathbf{K}\,\mathbf{y} = \mathbf{v}'$$

obtained from $\Sigma(\boldsymbol{X}, \mathbf{v})$ as follows

— the variable x_j is replaced by y_j for $j \in J_o \cup J$, and

— if $J' = \{j_1, \ldots, j_p\}$, then the variable x_{j_h} is replaced by $y_{j_h} + y_{m+h}$ for $h \in [p]$.

— the equation $\sum_{j \in J \cup J'} y_j + y_{m+p+1} = v$ is added to $\Sigma(\boldsymbol{Y}, \mathbf{v}')$ along with the variable y_{m+p+1}

where $\mathbf{v}' = (v_1, \ldots, v_n, v)$. Accordingly, for $j \in J_o \cup J$, the variable y_j stands for the (*a priori* unknown) answer to the σ-query with target the category Y_j and, for $h \in [p]$, the variables y_{j_h} and y_{m+h} stand for the (*a priori* unknown) answers to the σ-queries with targets Y_{j_h} and Y_{m+h}, respectively. Finally the variable y_{m+p+1} stands for the (*a priori* unknown) answer to the σ-query with target the category Y_{m+p+1}.

Example 1.1 (continued). Consider the σ-query

```
create view v as
select sum(SALARY)
from Personnel
where (GENDER=male and DEPT in {A,B,C}) or (DEPT in {D,E,I})
```

The set of categorical variables of **v** is {GENDER, DEPT} and, hence, $S' = \{\texttt{GENDER}, \texttt{DEPT}\}$. The extension to S' of X_j is $X'_j = \{\texttt{male}, \texttt{female}\} \times X_j$, $(j = 1, \ldots, 9)$, and the extension to S' of the target of **v** is

GENDER	DEPT
male	A
male	B
male	C
male	D
female	D
male	E
female	E
male	I
female	I

Then, $J_o = \{6, 7, 8\}$, $J = \{4, 5, 9\}$ and $J' = \{1, 2, 3\}$. The partition $\boldsymbol{Y}$ of the domain of S' consists of the following twelve categories:

$Y_1 = \{\texttt{male}\} \times X_1 = \{(\texttt{male}, \texttt{A})\}$
$Y_2 = \{\texttt{male}\} \times X_2 = \{(\texttt{male}, \texttt{B})\}$
$Y_3 = \{\texttt{male}\} \times X_3 = \{(\texttt{male}, \texttt{C})\}$

$Y_4 = \{\texttt{male}, \texttt{female}\} \times X_3 = \{(\texttt{male}, \texttt{C}), (\texttt{female}, \texttt{C})\}$
…
$Y_9 = \{\texttt{male}, \texttt{female}\} \times X_9 = \{(\texttt{male}, \texttt{I}), (\texttt{female}, \texttt{I})\}$
$Y_{10} = \{\texttt{female}\} \times X_1 = \{(\texttt{female}, \texttt{A})\}$
$Y_{11} = \{\texttt{female}\} \times X_2 = \{(\texttt{female}, \texttt{B})\}$
$Y_{12} = \{\texttt{female}\} \times X_3 = \{(\texttt{female}, \texttt{C})\}$

After substituting x_1 by $y_1 + y_{10}$, x_2 by $y_2 + y_{11}$, x_3 by $y_3 + y_{12}$ and x_j by y_j ($4 \le j \le 9$) in $\Sigma(\boldsymbol{X}, \mathbf{v})$, we obtain the equation system $\Sigma(\boldsymbol{Y}, \mathbf{v}')$

$$\begin{cases} y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_4 + y_{10} + y_{11} + y_{12} & = 22 \\ y_1 + y_2 + y_{10} + y_{11} & = 4 \\ y_1 + y_3 + y_4 + y_5 + y_{10} + y_{12} & = 6 \\ y_6 + y_7 & = 8 \\ y_8 + y_9 & = 4 \\ y_2 + y_3 + y_6 + y_{11} + y_{12} & = 10 \\ y_4 + y_8 & = 10 \\ y_5 + y_7 + y_9 & = 4 \\ y_1 + y_2 + y_3 + y_4 + y_5 + y_9 & = 6 \end{cases}$$

■

1.4 Evaluability and Derivability

In this section we want to establish the condition when a new query **q** can be evaluable or derivable from a set **V** of materialised views. Let S be the set of categorical variables of **V**, let $\boldsymbol{X} = \{X_1, \ldots, X_m\}$ be the classification system of **V**, and let S' be the union of S with the set of categorical variables of **q**. Let X'_j be the extension of X_j to S', and let Q be the extension to S' of the target of **q**. Let $\Omega = \cup_{j=1,\ldots,m} X'_j$. As said in the previous Section let

$$J_0 = \{j \in [m]: Q \cap X'_j = \emptyset\} \qquad J = \{j \in [m]: X'_j \subseteq Q\} \qquad J' = [m] - (J_0 \cup J)$$

We are interested in the (a priori unknown) total of σ for the S-category Q. The tightest lower bound, the total of σ for the S-category Q denoted by *lower*(Q), it is taken to be

$$lower(Q) = \mathbf{inf}\ \{\textstyle\sum_{j \in J} x_j : \mathbf{H}\,\mathbf{x} = \mathbf{v}, \mathbf{x} \in d^m\}\ .$$

The tightest upper bound, denoted by *upper*(Q), is set to $+\infty$ if Q is not contained in Ω; otherwise, it is taken to be

$$upper(Q) = \mathbf{sup}\ \{\textstyle\sum_{j \in J \cup J'} x_j : \mathbf{H}\,\mathbf{x} = \mathbf{v}, \mathbf{x} \in d^m\}\ .$$

The interval [*lower*(Q), *upper*(Q)] is called the *feasibility range* of the total of σ for Q. If *lower*(Q) = *upper*(Q), then the total for Q is said to be *evaluable*.

Remark 1.3 If $Q - \Omega \neq \emptyset$ then **q** is neither evaluable nor derivable.

Recall from previous section that $\boldsymbol{Y}$ is the classification system obtained from $\boldsymbol{X}$ by adding $\mathbf{q}$ to $\mathbf{V}$ and $\Sigma(\boldsymbol{Y}, \mathbf{v}')$ the associated equation system.
If Q is included in Ω we also will consider the equation system denoted by $\Sigma(\boldsymbol{Y}, \mathbf{v})$ obtained from $\Sigma(\boldsymbol{Y}, \mathbf{v}')$ by deleting the equation corresponding to the σ-view $\mathbf{q}$. In fact, observe that the sets of d-feasible solutions of $\Sigma(\boldsymbol{X}, \mathbf{v})$ and $\Sigma(\boldsymbol{Y}, \mathbf{v})$ are closely related to each other by the following obvious property: given a d-feasible solution $\mathbf{x} = (x_1, \ldots, x_m)$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$, we can obtain a d-feasible solution $\mathbf{y} = (y_1, \ldots, y_{m+p})$ of $\Sigma(\boldsymbol{Y}, \mathbf{v})$ by taking

$$y_j = x_j \qquad \text{for } j \in J_0 \cup J$$
$$y_{j_h} = \alpha_h \qquad y_{m+h} = x_{j_h} - \alpha_h \qquad \text{for } h \in [p]$$

where α_h is any element of d such that $x_{j_h} - \alpha_h$ also belongs to d. On the other hand, given a d-feasible solution $\mathbf{y} = (y_1, \ldots, y_{m+p})$ of $\Sigma(\boldsymbol{Y}, \mathbf{v})$, we can obtain a d-feasible solution $\mathbf{x} = (x_1, \ldots, x_m)$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$ by taking

$$x_j = y_j \qquad \text{for } j \in J_0 \cup J$$
$$x_{j_h} = y_{j_h} + y_{m+h} \qquad \text{for } h \in [p]$$

Therefore $\mathbf{q}$ is evaluable from $\mathbf{V}$ if and only if

$$\sum_{j \in J \cup J'} x_j = \mathit{lower}(Q) = \mathit{upper}(Q) = \sum_{j \in J \cup J'} y_j$$

that is if and only if $\sum_{j \in J \cup J'} y_j$ is a d-invariant of $\Sigma(\boldsymbol{Y}, \mathbf{v})$. Therefore we call

$$\sum_{j \in J \cup J'} y_j$$

the *sum expression associated to* $\mathbf{q}$.
If $J' = \emptyset$, then $\Sigma(\boldsymbol{Y}, \mathbf{v})$ is essentially the same as $\Sigma(\boldsymbol{X}, \mathbf{v})$ and $\mathbf{q}$ is evaluable from the instance $\mathbf{v}$ of $\mathbf{V}$ if and only if the sum-expression $\sum_{j \in J} x_j$ is a d-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$. For $J' = \{j_1, \ldots, j_p\}$, $p > 0$, we now state a characterisation of evaluability.

Lemma 1.2 Let $\mathbf{V}$ be a σ-view base, and let $d \in \{\mathsf{R}, \mathsf{Z}, \mathsf{R}^+, \mathsf{Z}^+\}$ be the domain of σ. Let $\mathbf{q}$ be a σ-query and let J and $J' = \{j_1, \ldots, j_p\}$, $p \geq 1$, be the Ω-support and the Ω-cosupport of $\mathbf{q}$, respectively. Then, if the target Q of $\mathbf{q}$ is contained in Ω

(*i*) if d is R or Z, then $\mathbf{q}$ is evaluable from no instance $\mathbf{v}$ of $\mathbf{V}$;

(*ii*) if d is R^+ or Z^+, then $\mathbf{q}$ is evaluable from an instance $\mathbf{v}$ of $\mathbf{V}$ if and only if
- (*a*) the sum-expression $\sum_{j \in J} x_j$ is a d-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$, and
- (*b*) for each $h \in [p]$, the variable x_{j_h} is a d-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$ with value zero.

Proof. (*i*) Let h^* be from $[p]$. Let $\boldsymbol{Y}$ be the classification system obtained adding $\mathbf{q}$ to $\mathbf{V}$. Let $\Sigma(\boldsymbol{Y}, \mathbf{v})$ be the system associated to $\boldsymbol{Y}$ obtained from $\Sigma(\boldsymbol{Y}, \mathbf{v}')$ by deleting the equation associated to $\mathbf{q}$.
Then, given an instance $\mathbf{v}$ of $\mathbf{V}$ and a d-feasible solution $\mathbf{x} = (x_1, \ldots, x_m)$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$, consider the following two d-feasible solutions $\mathbf{y} = (y_1, \ldots, y_{m+p})$ and $\mathbf{y}' = (y'_1, \ldots, y'_{m+p})$ of $\Sigma(\boldsymbol{Y}, \mathbf{v})$:

$$y_j = x_j \qquad \text{for } j \in [m]$$

$$y_{m+h} = 0 \qquad \text{for } h \in [p]$$

and

$$y'_j = x_j \qquad \text{for } j \in [m] - \{j_{h^*}\}$$
$$y'_{j_{h^*}} = x_{j_{h^*}} - 1 \quad \text{and} \quad y'_{m+h^*} = +1$$
$$y'_{m+h} = 0 \qquad \text{for } h \in [p] - \{h^*\}$$

Then, the sum-expression associated with **q** assumes two distinct values

$$\sum_{j \in J \cup J'} x_j \qquad \text{and} \qquad \sum_{j \in J \cup J'} x_j - 1$$

which proves that **q** is not evaluable from **v**.

Part (if) of (ii). By condition (*a*) the sum-expression $\sum_{j \in J} x_j$ is a *d*-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$. By condition (*b*), for each *h*, $x_{j_h} = 0$ for every *d*-feasible solution **x** of $\Sigma(\boldsymbol{X}, \mathbf{v})$ so that, owing to the nonnegativity constraint, one has

$$y_{j_h} = y_{m+h} = 0$$

for every *d*-feasible solution **y** of $\Sigma(\boldsymbol{Y}, \mathbf{v})$. Therefore, for every *d*-feasible solution **y** of $\Sigma(\boldsymbol{Y}, \mathbf{v})$, the sum-expression associated with **q** assumes the unique value of the sum-expression $\sum_{j \in J} x_j$, which proves that **q** is evaluable from **v**.

Part (only if) of (ii). Suppose by contradiction that condition (*a*) does not hold. Then, there exist two *d*-feasible solutions $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{x}' = (x'_1, \ldots, x'_m)$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$ such that

$$\sum_{j \in J} x_j < \sum_{j \in J} x'_j.$$

Let us consider the following two *d*-feasible solutions $\mathbf{y} = (y_1, \ldots, y_{m+p})$ and $\mathbf{y}' = (y'_1, \ldots, y'_{m+p})$ of $\Sigma(\boldsymbol{Y}, \mathbf{v})$ defined as follows:

$$y_j = x_j \qquad \text{for } j \in J_o \cup J$$
$$y_{j_h} = 0 \text{ and } y_{m+h} = x_{j_h} \qquad h \in [p]$$

and

$$y'_j = x'_j \qquad \text{for } j \in J_o \cup J$$
$$y'_{j_h} = 0 \text{ and } y'_{m+h} = x'_{j_h} \qquad \text{for } h \in [p]$$

Then

$$\sum_{j \in J \cup J'} y_j = \sum_{j \in J} y_j = \sum_{j \in J} x_j < \sum_{j \in J} x'_j = \sum_{j \in J} y'_j = \sum_{j \in J \cup J'} y'_j$$

which contradicts the evaluability of **q**. Therefore, condition (*a*) must hold. Suppose by contradiction that condition (*b*) does not hold. Then, there exist $h^* \in [p]$ such that either $x_{j_{h^*}}$ is not a

d-invariant or $x_{j_{h^*}}$ is a d-invariant but its value is greater than zero so that there is a d-feasible solution $\mathbf{x} = (x_1, \ldots, x_m)$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$ with $x_{j_{h^*}} > \mathtt{0}$. Consider the following two d-feasible solutions $\mathbf{y} = (y_1, \ldots, y_{m+p})$ and $\mathbf{y}' = (y'_1, \ldots, y'_{m+p})$ of $\Sigma(\boldsymbol{Y}, \mathbf{v})$:

$$y_j = x_j \qquad \text{for } j \in [m]$$
$$y_{m+h} = \mathtt{0} \qquad \text{for } h \in [p]$$

and

$$y'_j = x_j \qquad \text{for } j \in [m] - \{j_{h^*}\}$$
$$y'_{j_{h^*}} = \mathtt{0} \text{ and } y'_{m+h^*} = x_{j_{h^*}}$$
$$y'_{m+h} = \mathtt{0} \qquad \text{for } h \in [p] - \{h^*\}$$

Then, since $x_{j_{h^*}} > \mathtt{0}$, the sum-expression associated with $\mathbf{q}$ assumes two distinct values

$$\sum_{j \in J \cup J'} x_j \qquad \text{and} \qquad \sum_{j \in J \cup J'} x_j - x_{j_{h^*}}$$

which proves that $\mathbf{q}$ is not evaluable from $\mathbf{v}$. □

Example 1.1 (continued). We saw that the σ-query $\mathbf{q}$ with Ω–support $J = \{4, 5, 9\}$ and Ω–cosupport $J' = \{1, 2, 3\}$ is not evaluable from $\mathbf{v}$. If d is $\mathtt{R}$ or $\mathtt{Z}$, then Lemma 2(*i*) confirms that $\mathbf{q}$ is not evaluable from $\mathbf{v}$. If d is $\mathtt{R}^+$ or $\mathtt{Z}^+$, then we saw that there is exactly one d-feasible solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$

$$x_1 = \mathtt{0} \qquad x_2 = \mathtt{4} \qquad x_3 = \mathtt{0}$$
$$x_4 = \mathtt{6}, \qquad x_5 = \mathtt{0} \qquad x_6 = \mathtt{4}$$
$$x_7 = \mathtt{4} \qquad x_8 = \mathtt{4} \qquad x_9 = \mathtt{0}$$

Therefore, condition (*a*) of Lemma 1.2(*ii*) holds but, since $x_2 = \mathtt{4}$, condition (*b*) of Lemma 1.2(*ii*) does not hold, which confirms that $\mathbf{q}$ is not evaluable from $\mathbf{v}$. ■

We can summarise the above results by the following evaluability criterion.

Theorem 1.1 (*Evaluability Criterion*) Let $\mathbf{V}$ be a σ-view base and let $d \in \{\mathtt{R}, \mathtt{Z}, \mathtt{R}^+, \mathtt{Z}^+\}$ be the domain of σ. Let $\mathbf{H}$ be the dictionary matrix of $\mathbf{V}$ and $\mathbf{q}$ a σ-query with Ω-support J and Ω-cosupport J'.

(*i*) The target Q of $\mathbf{q}$ is contained in Ω.

(*ii*) For $d = \mathtt{R}$ or $d = \mathtt{Z}$, $\mathbf{q}$ is evaluable from an instance of $\mathbf{V}$ if and only if the characteristic vector of J belongs to the row space of $\mathbf{H}$, and $J' = \emptyset$.

(*iii*) For $d = \mathtt{R}^+$ or $d = \mathtt{Z}^+$, $\mathbf{q}$ is evaluable from an instance $\mathbf{v}$ of $\mathbf{V}$ if and only if the sum-expression $\sum_{j \in J} x_j$ is a d-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$, and either $J' = \emptyset$ or, for each $j \in J'$, x_j is a d-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$ with value $\mathtt{0}$.

Proof. Statement (i) is a necessary condition for Remark 1.3. For $d = \mathtt{R}$, statement (*ii*) is a well-known result of linear algebra. Consider now the case that $d = \mathtt{Z}$. By Lemma 1.1, it is sufficient to

prove the "only-if" part. By Lemma 1.2(*i*), $J' = \emptyset$. Suppose by contradiction that the characteristic vector of J, say **b**, does not belong to the row space of **H**. Then, **b** is not orthogonal to the null space of **H**. It follows that $J \neq \emptyset$ (for, otherwise, **b** would belong to the row space of **H**), and the dimension of the null space of **H** is not zero (for, otherwise, the null space of **H** would be equal to {**0**} and **b** would be orthogonal to the null space of **H**). Since **b** is not orthogonal to the null space of **H** and **H** is a 0-1 matrix, there is a rational-valued solution **z** of the homogeneous equation system **H z** = **0** such that (**b**, **z**) ≠ 0. Let $z_j = \frac{p_j}{q_j}$ with $q_j \geq 1$ (j = 1, …, m). Let q be the least common multiple of q_1, …, q_m. Let **x** be the true solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$ for the current relation I so that **x** is Z-valued. Consider the Z-valued solution **x**′ of $\Sigma(\boldsymbol{X}, \mathbf{v})$ defined as **x**′ = **x** + q **z**. Indeed, **x**′ is a Z-valued solution since each x'_j is definitely an integer. Finally, the value of the sum-expression $\sum_{j \in J} x_j$ corresponding to **x**′ is

$$(\mathbf{b}, \mathbf{x}') = (\mathbf{b}, \mathbf{x}) + q\,(\mathbf{b}, \mathbf{z}) \neq (\mathbf{b}, \mathbf{x})$$

since neither q nor (**b**, **z**) is zero (contradiction). Therefore, **b** must belong to the row space of **H**. Statement (*ii*) follows from Lemma 1.2(*ii*). □

Remark 1.4 For $d = \mathsf{R}^+$ or $d = \mathsf{Z}^+$, x_j is a d-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$ with value $\mathtt{0}$ if and only if

$$\mathbf{sup}\ \{x_j\colon\ \mathbf{H}\,\mathbf{x} = \mathbf{v},\ \mathbf{x} \in d^m\} = \mathtt{0}.$$

Recall that a σ-query is derivable from a σ-view base **V** if it is evaluable from every instance of **V**. The following derivability criterion characterises the σ-queries that are derivable from **V**.

Theorem 1.2 (*Derivability Criterion*) Let **V** be a σ-view base. A σ-query **q** with Ω–support J and Ω–cosupport J' is derivable from **V** if and only if

(*a*) the target Q of **q** is contained in Ω.

(*b*) the characteristic vector of J belongs to the row space of the dictionary matrix of **V**, and

(*c*) J' is empty.

Proof. (*if*) By Theorem 1.1 and Lemma 1.1.
(*only if*). Condition (*a*) is a necessary condition from Remark 1.3.
Proof of (*c*). Theorem 1.1 must hold for every instance **v** of **V**. Let I be a base relation such that the true solution **x** of $\Sigma(\boldsymbol{X}, \mathbf{v})$ has $x_j = 1$ for each j. Then, by Theorem 1.1, J' must be empty.
The proof of (*b*) is similar to the proof of part (*i*) of Theorem 1.1 for $d = \mathsf{Z}$, with the following change. Let $p = \max\{|p_1|, \ldots, |p_m|\}$ and let **x** with $x_j = pq$ for each j be the true solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$ for some base relation. Then, **x**′ = **x** + q **z** is a Z^+-feasible solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$ and, hence, a d-feasible solution of $\Sigma(\boldsymbol{X}, \mathbf{v})$ for each $d \in \{\mathsf{R}, \mathsf{Z}, \mathsf{R}^+, \mathsf{Z}^+\}$. □

Let λ(**V**) be the set of S-categories Q such that the σ-query with target Q is derivable from **V**. By Theorem 1.2, a σ-query **q** is derivable from **V** if and only if **q** is equivalent to some σ-query whose target belongs to λ(**V**), that is, if and only if the target of **q** is an extension of some S-category from λ(**V**). We now state some useful properties of λ(**V**). Of course, each Q in λ(**V**) can be obtained as union of zero or more classes of the classification system $\boldsymbol{X}$ of **V**; so, λ(**V**) is a subfamily of the set field generated by $\boldsymbol{X}$. Moreover, if $Q = \cup_{j \in J} X_j$ and **b** is the characteristic vector of J, then Q

belongs to $\lambda(\mathbf{V})$ if and only if equation system (1.1) is consistent. Therefore, $\lambda(\mathbf{V})$ contains the empty S-category, the extension to S of the target of each view $\mathbf{v}_i$ in $\mathbf{V}$ and, if present, the universal S-category $dom(S)$. It is also closed under disjoint union and proper difference [9, 10, 52]; that is,

— if Q and Q' are in $\lambda(\mathbf{V})$ and $Q \cap Q' = \emptyset$, then the disjoint union $Q \cup Q'$ is in $\lambda(\mathbf{V})$;

— if Q and Q' are in $\lambda(\mathbf{V})$ and $Q \subseteq Q'$, then the proper difference $Q' - Q$ is in $\lambda(\mathbf{V})$.

Note that, if $\lambda(\mathbf{V})$ contains the universal S-category then it is closed under proper difference and it is also closed under complementation, which makes $\lambda(\mathbf{V})$ a "λ-system" (see p. 41 in [4]). The minimal (with respect to the set-theoretic inclusion) nonempty S-categories in $\lambda(\mathbf{V})$ are called the *atoms* of $\lambda(\mathbf{V})$ so that, by the closure under proper difference, every nonempty category in $\lambda(\mathbf{V})$ is the disjoint union of atoms. Note that the number of atoms of $\lambda(\mathbf{V})$ may be exponential to n [47].

Example 1.1 (continued). The atoms of $\lambda(\mathbf{V})$ are ten:

$X_1 = \{\mathtt{A}\}$ $\quad$ $X_2 = \{\mathtt{B}\}$
$X_3 \cup X_4 \cup X_5 = \{\mathtt{C}, \mathtt{D}, \mathtt{E}\}$ $\quad$ $X_3 \cup X_5 \cup X_9 = \{\mathtt{C}, \mathtt{E}, \mathtt{I}\}$
$X_3 \cup X_6 = \{\mathtt{C}, \mathtt{F}\}$ $\quad$ $X_4 \cup X_5 \cup X_7 = \{\mathtt{D}, \mathtt{E}, \mathtt{G}\}$
$X_4 \cup X_8 = \{\mathtt{D}, \mathtt{H}\}$ $\quad$ $X_5 \cup X_7 \cup X_9 = \{\mathtt{E}, \mathtt{G}, \mathtt{I}\}$
$X_6 \cup X_7 = \{\mathtt{F}, \mathtt{G}\}$ $\quad$ $X_8 \cup X_9 = \{\mathtt{H}, \mathtt{I}\}$ ■

1.4.1 Computational aspects

We now discuss the computational complexities of the problems of deciding whether a σ-query $\mathbf{q}$ is or is not evaluable from an instance $\mathbf{v}$ of $\mathbf{V}$ and, in the affirmative case, of computing the answer q to $\mathbf{q}$ from $\mathbf{v}$. If d is $\mathtt{R}$ or $\mathtt{Z}$ then, by Theorem 1.1(a), $\mathbf{q}$ is evaluable from $\mathbf{v}$ if and only if equation system (1.3) is consistent and $J' = \emptyset$, which can be checked in $O(n^3)$ time using standard linear-algebra methods. Moreover, if $\mathbf{a}$ is a solution of equation system (1.3), then q can be computed as

$$\sum_{i=1,\ldots,n} a_i \, v_i$$

Therefore, deciding if $\mathbf{q}$ is evaluable from $\mathbf{v}$ and, if this is the case, computing q from $\mathbf{v}$ requires $O(n^3)$ time.

If d is $\mathtt{R}^+$, then testing the evaluability of $\mathbf{q}$ from $\mathbf{v}$ and computing q can be done using any polynomial-time algorithm for linear programming. However, we will see in Chapter 2 that it can be solved using the same cubic algorithm as for $d = \mathtt{R}$.

If $d = \mathtt{Z}^+$, then testing the evaluability of $\mathbf{q}$ from $\mathbf{v}$ may be hard since integer-linear-programming methods [55] are required. Indeed, as proven in Section 2.2.1, the problem of deciding if there exists some variable of $\Sigma(\boldsymbol{X}, \mathbf{v})$ that is $\mathtt{Z}^+$-invariant is co*NP*-complete. In practice, in the case $d = \mathtt{Z}^+$ it is convenient to relax the integrality constraints and apply the evaluability test developed for the case $d = \mathtt{R}^+$. In other words, we will use a sound (and possibly incomplete) evaluability test for $d = \mathtt{Z}^+$, but this is the best we can get since its possible incompleteness will be well compensated by its efficiency. However, it is well-known [55] that, if the dictionary matrix $\mathbf{H}$ is *totally unimodular* (that is, all squared submatrices of $\mathbf{H}$ have determinants +1, –1 or 0), then the evaluability test for $d = \mathtt{R}^+$ is also complete for $d = \mathtt{Z}^+$.

By Theorem 1.2, testing derivability requires $O(n^3)$ time; however, if $\mathbf{H}$ is the incidence matrix of a graph G, then a derivability test that runs in time linear in the size of G can be found in [46].

1.5 Reduction procedure for a view base and for a view base instance

Let **v** be an instance of a σ-view base **V**. Henceforth, we only consider the cases $d = \mathsf{R}$, $d = \mathsf{Z}$ and $d = \mathsf{R}^+$ since, as we said in Section 1.4.1, if $d = \mathsf{Z}^+$ then we will relax the integrality constraints and apply the results that hold for $d = \mathsf{R}^+$.

It is convenient to work with a σ-view base equivalent to **V** and with the minimum storage requirement for its dictionary. The natural storage representation of Σ(***X***, **v**) consists of an edge-labelled and node-weighted hypergraph (*H*, ***X***, **v**), we call the *map* of Σ(***X***, **v**), which is obtained from *H*, ***X*** by weighting each node *i* by the *i*-th component of **v** and. Thus, *H* is a hypergraph with node set [*n*] and *m* edges

$$e_j = \{i \in [n]: h_{ij} = 1\} \qquad (j \in [m])$$

and the quantity $\sum_{j=1,\ldots,m} |e_j|$ its *size.* Moreover, each edge e_j of *H* is labelled by the class X_j of the classification system ***X*** of **V**. We leave open the problem of finding a σ-view base that is equivalent to **V** and whose dictionary has a minimum-size map. However, we will give a procedure to find a σ-view base that is equivalent to **V** and is such that the map of its dictionary has a size not greater than the size of the map of the dictionary of **V**.

An edge e_j of *H* is *d-invariant* if x_j is a *d*-invariant of Σ(***X***, **v**). If this is the case and α is the value of x_j, then the instance **v**′ of the σ-view base $\mathbf{V}' = \{\mathbf{v}'_0, \mathbf{v}'_1, \ldots, \mathbf{v}'_n\}$, where

$\mathbf{v}'_0$ is the σ-query with target X_j and value $v'_0 = \alpha$, and

$\mathbf{v}'_i$, $i \in [n]$, is the σ-query with target $V_i - X_j$ and value

$$v'_i = \begin{cases} v_i - \alpha & \text{if } X_j \subseteq V_i \\ v_i & \text{else} \end{cases}$$

is definitely equivalent to the instance **v** of **V**; furthermore, the size of the map of Σ(***X***, **v**′) is equal to the size of the map of Σ(***X***, **v**) minus $|e_j| - 1$.

If an edge is *d*-invariant edges regardless of the instance of **v** of **V** then it is said to be *algebraic*. By Theorem 1.2 an edge is algebraic if and only if its characteristic vector can be expressed as a linear combination of row of H.

Given two σ-view bases **V** and **V**′ that have the same set of categorical variables and the same classification system, we say that an instance **v** of **V** and an instance **v**′ of **V**′ are *equivalent* with respect to evaluability, if each view in **V**′ is evaluable from **v** and each view in **V** is evaluable from **v**′. If this is the case, then a sum-query is evaluable from **v** if and only if it is evaluable from **v**′. Analogously we say that two σ-view bases **V** and **V**′ that have the same set of categorical variables and the same classification system are *equivalent* with respect to derivability, if each view in **V**′ is derivable from **V** and each view in **V** is derivable from **V**′. If this is the case, then a sum-query is derivable from **V** if and only if it is derivable from **V**′.

A node *i* of the map (*H*, ***X***, **v**) of Σ(***X***, **v**) is *redundant*, with respect to evaluability if the view $\mathbf{v}_i$ is evaluable from the instance **v**′ of $\mathbf{V}' = \mathbf{V} - \{\mathbf{v}_i\}$, where **v**′ is obtained from **v** by ignoring v_i. If this is the case, then **v**′ is definitely equivalent to the instance **v** of **V** and the size of the map of Σ(***X***, **v**′) is equal to the size of the map of Σ(***X***, **v**) minus the number of edges containing the node *i*.

Analogously a node *i* of the map (*H*, ***X***, **v**) of Σ(***X***, **v**) is redundant, with respect to derivability, if the view $\mathbf{v}_i$ is derivable from $\mathbf{V} - \{\mathbf{v}_i\}$ or equivalently, by Theorem 1.2, if the *i*-th row of **H** can be

written as a linear combination of the other rows of **H**. For example, if H is a connected graph then, since the rank of **H** is n–1 or n depending on whether H is or is not bipartite [15, 57], H has exactly one redundant node if H is bipartite, and no redundant nodes otherwise. If i is a redundant node of the map of the dictionary of $\mathbf{V}$, then the σ-view base $\mathbf{V}' = \mathbf{V} - \{\mathbf{v}_i\}$ is definitely equivalent to $\mathbf{V}$ and the size of the map of the dictionary of $\mathbf{V}'$ is equal to the size of the map of the dictionary of $\mathbf{V}$ minus the number of edges containing the node i.

We say that the map of $\Sigma(\boldsymbol{X}, \mathbf{v})$ is *reduced* if it contains no d-invariant edges and no redundant nodes wit respect to derivability if the derivability criterion is used or with respect to evaluability otherwise. If the map of $\Sigma(\boldsymbol{X}, \mathbf{v}')$ is reduced, then $\Sigma(\boldsymbol{X}, \mathbf{v}')$ is called a *reduced form* of $\Sigma(\boldsymbol{X}, \mathbf{v})$.
The algorithm below, takes as input the map $(H, \boldsymbol{X}, \mathbf{v})$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$, and yields a reduced map $(H^*, \boldsymbol{X}, \mathbf{v}^*)$ of $\Sigma(\boldsymbol{X}, \mathbf{v}^*)$ where $\mathbf{v}^*$ is a view-base instance equivalent to $\mathbf{v}$.

reduction procedure for a view base instance

Input: $(H, \boldsymbol{X}, \mathbf{v})$.

Output: $(H^*, \boldsymbol{X}, \mathbf{v}^*)$.

Step 1. $H^* = \emptyset$, $l := 0$.

Step 2. Find the set of d-invariant edges of H. If it is empty, then go to Step 3. Otherwise, set l to its cardinality. Let $e_{j_1}, \ldots, e_{j_l}$ be the d-invariant edges of H and let $\alpha_1, \ldots, \alpha_l$ be their values. For $k = 1, \ldots, l$, do*:*

add to H^* the node k and set $v^*_k := \alpha_k$;

add to H^* the loop $\{k\}$ and label the loop by the label (X_{j_k}) of e_{j_k};

for each node i of H contained in e_{j_k}, set $v_i := v_i - \alpha_k$;

delete e_{j_k} from H.

If the edge set of H is empty, then Exit.

Step 3. For each node i of H, delete i from H if it is redundant. If the node set of H is empty, then Exit.

Step 4. Let $\{i_1, \ldots, i_r\}$ be the node set of H. After renaming its nodes by $l+1, \ldots, l+r$, add $(H, \boldsymbol{X}, \mathbf{v})$ to $(H^*, \boldsymbol{X}, \mathbf{v}^*)$.

Example 1.1 (continued). If the domain of SALARY is R or Z, then the output of the reduction procedure is the map shown in Figure 1.2.

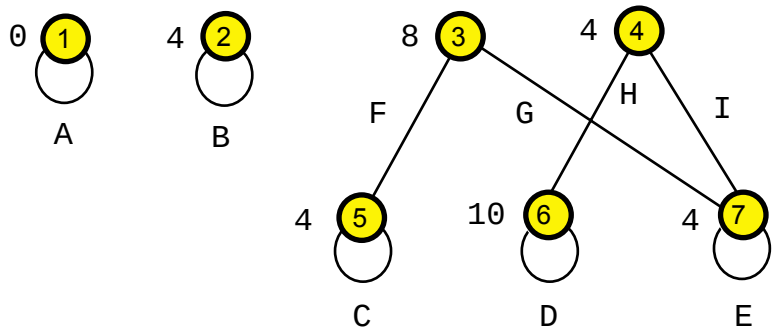


Figure 1.2

If the domain of SALARY is R^+, then we saw that each edge is invariant so that the output of the reduction procedure is the map shown in Figure 1.3.

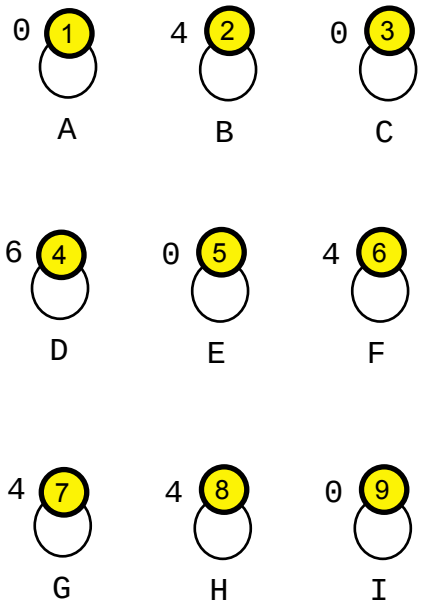


Figure 1.3

■

When the criterion utilised is derivability the following algorithm, we call the *reduction procedure for a view base*, takes as input the map $(H, \boldsymbol{X})$ of the dictionary of $\mathbf{V}$ and yields the map $(H^*, \boldsymbol{X})$ of the dictionary of a reduced σ-view base $\mathbf{V}^*$ equivalent to $\mathbf{V}$, we call a *reduced form* of $\mathbf{V}$. In what follows, an edge e is said to be *incident* to node i if $i \in e$, and is called a *loop* if $|e| = 1$.

reduction procedure for a view base

Input: $(H, \boldsymbol{X})$

Output: $(H^*, \boldsymbol{X})$.

Step 1. $H^* = \emptyset$, $l := 0$.

Step 2. Find the set of algebraic edges of H. If it is empty, then go to Step 3. Otherwise, set l to its cardinality. Let $e_{j_1}, \ldots, e_{j_l}$ be the algebraic edges of H. For $k = 1, \ldots, l$, do:

add to H^* the node k and the loop $\{k\}$, and label the loop by the label (X_{j_k}) of e_{j_k};

delete e_{j_k} from H.

If the edge set of H is empty, then Exit.

Step 3. For each node i of H, delete i from H if it is redundant. If the node set of H is empty, then Exit.

Step 4. Let $\{i_1, \ldots, i_r\}$ be the node set of H. After renaming its nodes by $l+1, \ldots, l+r$, add $(H, \boldsymbol{X})$ to $(H^*, \boldsymbol{X})$.

Example 1.1 (continued). Let us apply the reduction procedure to the map $(H, \boldsymbol{X})$ of $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_8)$.

Step 1. $H^* = \emptyset$.

Step 2. We saw that the only algebraic edges of H are e_1 and e_2. Therefore, two nodes 1 and 2 and two loops $\{1\}$ and $\{2\}$ are added to H^* with labels $\{\mathsf{A}\}$ and $\{\mathsf{B}\}$, respectively. At this point, the edges e_1 and e_2 are deleted from H. So, H has now eight nodes and seven edges (namely, $e_3, \ldots, e_9$), and the incidence matrix of H is

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{matrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \\ \mathbf{h}_4 \\ \mathbf{h}_5 \\ \mathbf{h}_6 \\ \mathbf{h}_7 \\ \mathbf{h}_8 \end{matrix}$$

Step 3. Since $\mathbf{h}_1$, $\mathbf{h}_2$ and $\mathbf{h}_3$ are all linear combinations of $\mathbf{h}_4, \ldots, \mathbf{h}_8$:

$$\mathbf{h}_1 = \mathbf{h}_6 + \mathbf{h}_7 + \mathbf{h}_8$$
$$\mathbf{h}_2 = \mathbf{0}$$
$$\mathbf{h}_3 = -\mathbf{h}_4 - \mathbf{h}_5 + \mathbf{h}_6 + \mathbf{h}_7 + \mathbf{h}_8$$

the nodes 1, 2 and 3 are redundant and are deleted. At this point, the nodes of H are 4, …, 8, and the edges of H are

$$e_3 = \{6\}$$
$$e_4 = \{7\} \qquad e_5 = \{8\} \qquad e_6 = \{4, 6\}$$
$$e_7 = \{4, 8\} \qquad e_8 = \{5, 7\} \qquad e_9 = \{5, 8\}$$

Step 4. After renaming the nodes 4, …, 8 of H as 3, …, 7, we add H with the labels of its edges to (H^*, $\boldsymbol{X}$).

The output of the reduction procedure is the map shown in Figure 1.4

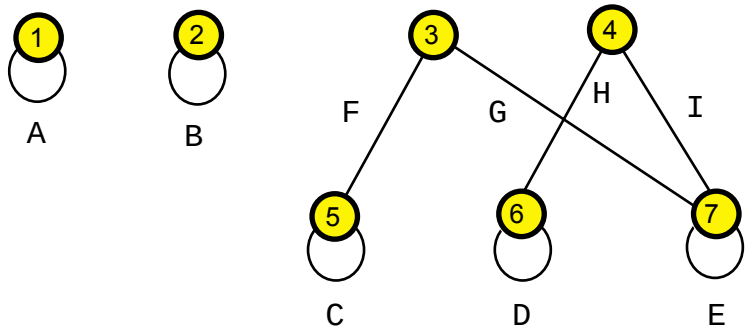


Figure 1.4

■

Chapter 2

Efficiently Answering Statistical Sum-Queries

2.1 Introduction

When a user submit a sum-query **q** with response variable SALARY, a question that naturally arises is whether the query-answering system can answer **q** from **v** without accessing the database. We saw in Chapter 1 that **q** is evaluable from **v** if the answer to **q** is uniquely determined by **v**, and is derivable from **V** if it is evaluable from any **v**.
We propose two distinct query-execution plans, based on the notions of evaluability and derivability. Given

— the current database relation I of name Rel whose scheme contains a numeric attribute σ,
— a list **V** of views over σ, which at least contains the σ-view **u**, whose target is the universal category
— the tuple **v** containing the values of views in **V** computed on I,
— the target of a sum-query **q** with response variable σ and defined over the scheme of Rel,

the query-execution plan based on derivability is as follows:

Plan D

(1) If **q** is derivable from **V**, then compute the answer to **q** from **v**.

(2) Otherwise,
find a subset Q' (or a superset) of Q such that the sum-query **q**′ with target Q' is derivable from **V**, and compute the answer to **q**′ from **v**;
evaluate the sum-query **q**″ with target $Q–Q'$ ($Q'–Q$, respectively) on I;
answer **q** by issuing the sum (difference, respectively) of the answers to **q**′ and **q**″.

The query-execution plan based on evaluability (*Plan E*) is obtained from Plan *D* by changing (the two occurrences of) "derivable from **V**" to "evaluable from **v**".

As to Plan *D* we will show that:

(*i*) the implementation of Step 1 requires $O(n^3)$ time and is independent of the domain of the response variable σ;

(*ii*) Step 2 requires polynomial time, but the problem of finding a maximal subset (a minimal superset, respectively) of Q such that sum-query with target Q' is derivable from **V** is *NP*-hard.

As to the query-execution plan *E* we will show that:

(*i*) if the domain of the response variable σ is the set of reals or the set of integers, then evaluability coincides with derivability so that the implementation of Step 1 still requires $O(n^3)$ time; if the domain of the response variable is the set of nonnegative reals, Step 1 requires solving a linear-programming problem; however, if **v** is in a suitable form ("reduced form"), then the

implementation of Step 1 still requires $O(n^3)$ time; if the domain of the response variable is the set of nonnegative integers, Step 1 requires solving a co*NP*-hard problem.

(*ii*) Step 2 requires polynomial time, but the problem of finding a maximal subset (a minimal superset, respectively) of Q such that sum-query with target Q' is evaluable from **v** is *NP*-hard.

2.1 The query-execution plan D

Let **V** be a reduced σ-view base, and let (H, $\boldsymbol{X}$, **v**) be the map of the dictionary of **V**. An edge set $\{e_j: j \in J\}$, $J \subseteq [m]$, in the map (H, $\boldsymbol{X}$, **v**)) of the dictionary of **V** is *algebraic* if the characteristic vector of J belongs to the row space of **H**. Accordingly, a single edge e_j is algebraic if the characteristic vector of the singleton $\{j\}$ belongs to the row space of **H**. For example, if H is a connected graph, then an edge is algebraic if and only if either it lies in all odd cycles or it is a bridge with a bipartite end-graph (see Chapter 4).

Henceforth, we adopt the following notation:

$\{e_j: j \in L\}$ is the set algebraic edges of H

for each $j \in L$, $i(j)$ is the node of H that the edge e_j is incident to

M is the complement of L, that is, $M = [m] - L$

$G = (N, E)$ is the subhypergraph of H induced by M; that is, $E = \{e_j: j \in M\}$

$\mathbf{G} = (g_{ij})_{i \in N, j \in M}$ is the incidence matrix of G.

It is easily seen that, for each connected component $G' = (N', E')$ of G, one has

— E' is an algebraic set, and
— no edge of G' is algebraic.

Moreover, observe that a set F of edges of H is algebraic if and only if, for each connected component $G' = (N', E')$ of G, $F \cap E'$ is an algebraic set. With the notation just introduced, the derivability criterion reads

Corollary 2.1 Let **V** be a reduced σ-view base. A σ-query **q** with support J and co-support J' is derivable from **V** if and only if

(*a*) for each connected component $G' = (N', E')$ of G, the intersection of $\{e_j: j \in J\}$ with E' is an algebraic set, and

(*b*) J' is empty.

Let $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ be a reduced σ-view base. Without loss of generality, we assume that the subhypergraph $G = (N, E)$ of H is connected. Let I be the current database relation and **v** the instance of **V** associated with I. Given a σ-query **q** with Ω-support J and Ω-cosupport J', we can decide whether **q** is or is not derivable from **V** using Corollary 2.1. If **q** is derivable from **V**, then the characteristic vector **b** of $J \cap M$ can be expressed as a linear combination of rows of the incidence matrix **G** of G, say with coefficients a_i ($i \in N$), so that the answer q to **q** can be computed from **v** as

$$q = \sum_{j \in J \cap L} a_i \, v_{i(j)} + \sum_{i \in N} a_i \, v_i$$

Consider now the case that **q** is not derivable from **V**. For the sake of simplicity, we assume that $J \cup J' \subseteq M$. With slight obvious modifications, what we will say can be easily applied to the general case $J \cup J' - M \neq \emptyset$. Let S be the set of categorical variables of **V**, S' the union of S with the set of categorical variables of **q**, and Q the extension to S' of the target of **q**. The execution plan looks for an S'-category Q' that is contained in (or containing) Q and is such that the σ-query **q**′ with target Q' is derivable from **V**. By Theorem 1.2, Q' will be the extension to S' of some S-category with Ω-support K where $K \subseteq J$ ($K \subseteq J \cup J'$, respectively). Once Q' has been determined, in order to find the answer q to **q**, the execution plan first computes the answer q' to **q**′ from **v**; next, it will evaluate on I the σ-query **q**″ with target $Q'' = Q - Q'$ ($Q' - Q$, respectively) and, finally, set $q = q' + q''$ ($q = q' - q''$, respectively) where q'' is the answer to **q**″.

Example 1.1 (continued). Consider again the σ-query **q** with Ω-support $J = \{4, 5, 9\}$ and Ω-cosupport $J' = \{1, 2, 3\}$. The S-category

$$\cup_{j \in J} X_j = X_4 \cup X_5 \cup X_9$$

contains no atoms of λ(**V**), so that the empty S-category is the only subset of $\cup_{j \in J} X_j$ that belongs to λ(**V**). The S-category

$$\cup_{j \in J \cup J'} X_j = X_1 \cup X_2 \cup X_3 \cup X_4 \cup X_5 \cup X_9$$

has two supersets that belong to λ(**V**):

$$X_1 \cup X_2 \cup X_3 \cup X_4 \cup X_5 \cup X_8 \cup X_9 \qquad\qquad dom(S) \qquad\qquad \blacksquare$$

We now show how Q' can be optimally chosen if, for each view $\mathbf{v}_i$ in **V**, we are also given the number of tuples r_i in I that fall in the category V_i. First of all, observe that, if the characteristic vector of the set K is a linear combination of rows of the incidence matrix **G** of G with coefficients a_i ($i \in N$), then the number r' of tuples in I that fall in Q' is given by

$$r' = \sum_{i \in N} a_i \, r_i \, . \qquad (2.1)$$

Let r be the (unknown) numbers of tuples in I that fall in the category Q. If Q' is required to be a subset of Q, then the number of additions required by the evaluation of **q**″ on I is given by $r - r'$. If Q' is required to be a superset of Q, then the number of additions required by the evaluation of **q**″ on I is given by $r' - r$. We now discuss the two cases separately.

Case 1: If Q' is required to be a subset of Q, then the set K is to be chosen among the subsets of J. Thus, the number of additions required by the evaluation of **q**″ on I is $r - r'$ and is minimum if and only if K maximises the sum (2.1). So, the choice of Q' is optimal if and only if the pair (**a**, **b**), where **b** stands for the characteristic vector of K, is an optimal solution of the following mixed linear-programming problem:

$$\begin{array}{lll} \text{maximise} & \sum_{i \in N} a_i \, r_i & \qquad (2.2) \\ \text{subject to} & \sum_{i \in N} a_i \, g_{ij} = b_j & \text{for } j \in M \\ & a_i \in \mathsf{R} & \text{for } i \in N \end{array}$$

$b_j \in \{0, 1\}$ for $j \in J$

$b_j = 0$ for $j \in M - J$

Case 2: If Q' is required to be a superset of Q, then the set K is to be chosen among the supersets of $J \cup J'$. Thus, the number of additions required by the evaluation of $\mathbf{q}''$ on I is $r'–r$ and is minimum if and only if K minimises the sum (2.1). So, the choice of Q' is optimal if and only if the pair (**a**, **b**), where **b** stands for the characteristic vector of K, is an optimal solution of the following mixed linear-programming problem:

$$\text{minimise} \quad \sum_{i \in N} a_i r_i \qquad (2.3)$$

$$\begin{aligned} \text{subject to} \quad & \sum_{i \in N} a_i g_{ij} = b_j && \text{for } j \in M \\ & a_i \in \mathsf{R} && \text{for } i \in N \\ & b_j = 1 && \text{for } j \in J \cup J' \\ & b_j \in \{0, 1\} && \text{for } j \in M - (J \cup J') \end{aligned}$$

We now show that problem (2.3) is reducible to problem (2.2). Since the edge set E of G is algebraic, its characteristic vector (i.e., the vector **1**) is a linear combination of rows of the incidence matrix of G, say

$$\sum_{i \in N} c_i g_{ij} = 1 \qquad (j \in M)$$

Hence, (**a**, **b**) is an optimal solution of problem (2.3) if and only if (**a**′, **b**′) where

$$\mathbf{a}' = \mathbf{c} - \mathbf{a} \qquad \mathbf{b}' = \mathbf{1} - \mathbf{b}$$

is an optimal solution of the maximisation problem

$$\text{maximise} \quad \sum_{i \in N} r_i a'_i$$

$$\begin{aligned} \text{subject to} \quad & \sum_{i \in N} a'_i g_{ij} = b'_j && \text{for } j \in M \\ & a'_i \in \mathsf{R} && \text{for } i \in N \\ & b'_j = 0 && \text{for } j \in J \cup J' \\ & b'_j \in \{0, 1\} && \text{for } j \in M - (J \cup J') \end{aligned}$$

which is the instance of the problem (2.2) where J is instantiated by $M - (J \cup J')$. Therefore, we can limit our considerations to problem (2.2). Of course, it is not less hard than the problem of finding a nonempty subset K of J such that the set $\{e_j : j \in K\}$ of edges of G is algebraic. In the next section, we will prove that the problem of finding a Nonempty Algebraic Subset of a given edge set in a connected hypergraph such as G (*NAS problem*, for short) is *NP*-complete, which implies that the problem of finding an optimal solution of problem (2.2) is *NP*-hard. To sum up, if $J \cup J' \subseteq M$ then $K := \emptyset$ is the only choice that can be made efficiently. In the general case (that is, when $J \cup J' - M \neq \emptyset$), K will be set to $J \cap L$.

2.1.1 Proving the NP-completeness of the NAS problem

In this section we prove that the NAS problem is *NP*-complete. First of all, we begin with the precise formulation the NAS problem:

Let $G = (N, E)$ be a connected hypergraph such that E is an algebraic set and no edge of G is algebraic. Given a nonempty subset F of E, find a nonempty algebraic subset of F (if any).

Our proof of its *NP*-completeness is obtained by providing a polynomial reduction of the *NP*-complete "Subset Sum" (SS) problem [25]. Before proving the statement, we first recall the SS problem and, then, sketch the proof lines.

(SS) Given a number $k \geq 2$, and an integer-valued vector $\mathbf{s} = (s_1, \ldots, s_k)$, where $s_1, \ldots, s_{k-1}$ are all positive and s_k is negative, is there a nonnull binary vector $\mathbf{a} = (a_1, \ldots, a_k)$ such that

$$\sum_{h=1,\ldots,k} a_h s_h = 0 \ ?$$

We now state a characteristic property of solutions of the SS problem which will be used to prove that the NAS problem is *NP*-complete.

Let $n \geq k$, let $F = \{i_1, \ldots, i_k\}$ be a subset of $[n]$, let $\mathbf{t} = (t_1, \ldots, t_n)$ be a vector such that $t_{i_h} = s_h$ for each $h \in [k]$, and let $\mathbf{R}$ be the $n \times (n+1)$ matrix obtained from the $n \times n$ identity matrix by adding one more column given by $\mathbf{t}$, that is,

$$\mathbf{R} = \begin{vmatrix} 1 & 0 & \ldots & 0 & t_1 \\ 0 & 1 & \ldots & 0 & t_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & t_n \end{vmatrix}$$

Note that $\mathbf{R}$ is a full row-rank matrix, that is, the rank of $\mathbf{R}$ (and, hence, the dimension of the row space of $\mathbf{R}$) is n. Then, the solutions of the SS problem can be characterised as follows.

Lemma 2.1 Let $\mathbf{a} = (a_1, \ldots, a_k)$ be a nonnull binary vector and let $\mathbf{b} = (b_1, \ldots, b_n, b_{n+1})$ be the binary vector such that $b_{i_h} = a_h$ for each $h \in [k]$, and $b_i = 0$ for each $i \in [n+1] - F$. The vector $\mathbf{a}$ is a solution of the SS problem if and only if the vector $\mathbf{b}$ belongs to the row space of $\mathbf{R}$.
Proof. Recall that $\mathbf{b}$ belongs to the row space of $\mathbf{R}$ if and only if $\mathbf{b}$ is orthogonal to the null space of $\mathbf{R}$. This space is the set of solutions $\mathbf{z} = (z_1, \ldots, z_n, z_{n+1})$ of the homogenous equation system

$$\mathbf{R}\,\mathbf{z} = \mathbf{0},$$

which consists of the n equations

$$z_i + t_i z_{n+1} = 0 \ . \qquad (i = 1, \ldots, n)$$

Therefore, the null space of $\mathbf{R}$ has dimension 1 and is spanned by the vector $(t_1, \ldots, t_n, -1)$ and $\mathbf{b}$ belongs to the row space of $\mathbf{R}$ if and only if $\mathbf{b}$ is orthogonal to $(t_1, \ldots, t_n, -1)$, that is, if and only if

$$\sum_{i=1,\ldots,n} b_i t_i - b_{n+1} = 0.$$

On the other hand, $b_i = 0$ for each $i \notin F$ so that $\mathbf{b}$ belongs to the row space of $\mathbf{R}$ if and only if

$$\sum_{h=1,\ldots,k} b_{i_h} t_{i_h} = 0.$$

Finally, since $b_{i_h} = a_h$ and $t_{i_h} = s_h$ for each $h \in [k]$, one has **b** belongs to the row space of **R** if and only if

$$\sum_{h=1,\ldots,k} a_h s_h = 0 ,$$

that is, if and only if **a** is a solution of the SS problem. □

In what follows, the matrix **R** and the vector **b** will be referred to as the matrix *associated* with **t** and the *extension* of **a**, respectively.

We will show that, given n, F, **t** and **R**, there exists an invertible linear transformation of **R** into the incidence matrix **G** of a hypergraph G with n nodes and n+1 edges. Then, by Lemma 2.1, **a** is a solution of the SS problem if and only if the extension **b** of **a** belongs to the row space of **G** or, equivalently, if and only if in G the set of edges with characteristic vector **b** is algebraic. Therefore, **a** is a solution of the SS problem if and only if in G the edge set with characteristic vector **b** is a nonempty algebraic subset of the edge set corresponding to F.

We now show how, given the parameters k and **s** of the SS problem, it is possible to construct in polynomial time (1) a vector **t**, and (2) an invertible linear transformation of the matrix **R** associated with **t** into a binary matrix **G**.

(1) The vector **t** is taken to be $\mathbf{t} = (\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_k, \boldsymbol{\tau}_{k+1})$ where the vectors $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_k, \boldsymbol{\tau}_{k+1}$ are defined as follows. We first state the procedure for $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{k-1}$, next for $\boldsymbol{\tau}_k$ and, finally, for $\boldsymbol{\tau}_{k+1}$.

— The vector $\boldsymbol{\tau}_l$ for each $l \in [k-1]$. Let $m_l = \lfloor \log_2 s_l \rfloor$, where $\lfloor x \rfloor$ denotes the highest integer that is not greater than x. If $m_l = 0$ (that is, if $s_l = 1$), then $\boldsymbol{\tau}_l$ has one component which is equal to s_l, that is, $\boldsymbol{\tau}_l = (s_l)$; otherwise, $\boldsymbol{\tau}_l$ has $3m_l + 1$ components and is defined as follows:

$$\left(s_l \quad -\lfloor s_l/2 \rfloor \quad -\lfloor s_l/2 \rfloor \quad +\lfloor s_l/2 \rfloor \quad \ldots \quad -\lfloor s_l/2^{m_l} \rfloor \quad -\lfloor s_l/2^{m_l} \rfloor \quad +\lfloor s_l/2^{m_l} \rfloor\right)$$

— The vector $\boldsymbol{\tau}_k$. Let $m_k = \lceil \log_2 |s_k| \rceil$, where $\lceil x \rceil$ denotes the least integer that is not less than x. If $m_k = 0$ (that is, if $s_k = -1$), then $\boldsymbol{\tau}_k = (s_k, 1, 1)$; otherwise (that is, if $s_k < -1$), $\boldsymbol{\tau}_k$ has $3m_k$ components and is defined as follows:

$$\left(s_k \quad +\lceil |s_k|/2 \rceil \quad +\lceil |s_k|/2 \rceil \quad -\lceil |s_k|/2 \rceil \quad \ldots \quad +\lceil |s_k|/2^{m_k} \rceil \quad +\lceil |s_k|/2^{m_k} \rceil\right)$$

— The vector $\boldsymbol{\tau}_{k+1}$. Let $|\boldsymbol{\tau}_l|$ denote the sum of components of $\boldsymbol{\tau}_l$, $k \in [l]$. Let $r = \sum_{l=1,\ldots,k} |\boldsymbol{\tau}_l|$ and let $\boldsymbol{\tau}_{k+1}$ be the $(r+1)$-dimensional vector with components

$$(-1 \quad -1 \quad \ldots -1 \quad 1)$$

So, $|\boldsymbol{\tau}_{k+1}| = 1 - r$.

Accordingly, the matrix **R** associated with **t** has

$$n = (\sum_{l=1,\ldots,k-1} 3m_l + 1) + 3m_k + r + 1$$

rows and n+1 columns.

Example 2.1. Consider the SS problem with $k = 3$ and $s_1 = s_2 = 1$ and $s_3 = -2$. Trivially, it has exactly one solution: $\mathbf{a} = (1, 1, 1)$. Then $\tau_1 = (1)$, $\tau_2 = (1)$ and $\tau_3 = (-2, 1, 1)$. Since $|\tau_1| = |\tau_2| = 1$ and $|\tau_3| = 0$, one has $\tau_4 = (-1, -1, 1)$ and the matrix $\mathbf{R}$ associated with the vector $\mathbf{t} = (\tau_1, \tau_2, \tau_3, \tau_4)$ is

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The null space of $\mathbf{R}$ is spanned by the 9-dimensional vector $\mathbf{y}^* = (1, 1, -2, 1, 1, -1, -1, 1, -1)$ so that the extension $\mathbf{b} = (1, 1, 1, 0, 0, 0, 0, 0, 0)$ of $\mathbf{a}$ is orthogonal to the null space of $\mathbf{R}$ and hence, belongs to the row space of $\mathbf{R}$. ■

Lemma 2.2 The $(n+1)$-dimensional vector $(1, 1, \ldots, 1)$ belongs to the row space of $\mathbf{R}$. Moreover, for each $j \in [n+1]$, the characteristic vector of the singleton $\{j\}$ does not belong to the row space of $\mathbf{R}$.

Proof. For each column index $j \in [n]$ of $\mathbf{R}$, one has $\sum_{i=1,\ldots,n} r_{i,j} = 1$; moreover, for $j = n+1$ one has

$$\sum_{i=1,\ldots,n} r_{i,n+1} = \sum_{l=1,\ldots,k+1} |\tau_l| = \sum_{l=1,\ldots,k} |\tau_l| + |\tau_{k+1}| = r + (1 - r) = 1,$$

which proves the first part of the statement. As to the second part, let $\mathbf{a}$ be the characteristic vector of the singleton $\{j\}$. We now prove that the equation in the unknowns $c_1, \ldots, c_n$

$$\mathbf{a} = \sum_{i=1, \ldots,n} c_i \ \mathbf{r}_i$$

has no solutions. We first consider the case $j \leq n$ and, then, the case $j = n + 1$. In the former case, the equation above reads

$$\begin{aligned} 0 &= c_1 \\ &\ldots \\ 0 &= c_{j-1} \\ 1 &= c_j \\ 0 &= c_{j+1} \\ &\ldots \\ 0 &= c_n \\ 0 &= \sum_{i=1, \ldots,n} c_i \ t_i \end{aligned}$$

Replacing the values of $c_1, \ldots, c_n$ in the last equation, one obtains

$$0 = t_j$$

which is impossible since, by construction, each t_i is nonzero. Therefore, $\mathbf{a}$ does not belong to the row space of $\mathbf{G}$. In the latter case, the equation above reads

$0 = c_1$
…
$0 = c_n$
$1 = \sum_{i=1, \ldots, n} c_i \ t_i$

Replacing the values of $c_1, \ldots, c_n$ in the last equation, we obtain

$1 = 0$

which is impossible. Therefore, **a** does not belong to the row space of **G**. □

(2) Consider the matrix **G** with rows $\mathbf{g}_1, \ldots, \mathbf{g}_n$, which results from the application of the following procedure to the matrix **R** associated with **t**. We first partition the rows of **R** into groups, each of which contains rows of **R** ending with the components of one of the vectors $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_k, \boldsymbol{\tau}_{k+1}$. Let $\{\mathbf{r}_p, \ldots, \mathbf{r}_q\}$ be the group corresponding to the vector $\boldsymbol{\tau}_l$. We first consider the case $1 \leq l \leq k-1$, then the case $l = k$ and, finally, the case $l = k+1$.

— $(1 \leq l \leq k-1)$. The vectors $\mathbf{g}_p, \ldots, \mathbf{g}_q$ are obtained as follows

$i := p$
if $q = p$, then $\mathbf{g}_i := \mathbf{r}_i$;
otherwise, while $i < q$ do

begin
$\mathbf{g}_i := \mathbf{r}_i + \mathbf{r}_{i+1} + \mathbf{r}_{i+2}$
$\mathbf{g}_{i+1} := \mathbf{r}_{i+1} + \mathbf{r}_{i+3}$
$\mathbf{g}_{i+2} := \mathbf{r}_{i+2} + \mathbf{r}_{i+3}$
$i := i + 3$
end

— $(l = k)$. The vectors $\mathbf{g}_p, \ldots, \mathbf{g}_q$ are obtained as follows

$i := p$
while $i < q$ do

begin
$\mathbf{g}_i := \mathbf{r}_i + \mathbf{r}_{i+1} + \mathbf{r}_{i+2}$
if $i + 3 \leq q$ then do

begin
$\mathbf{g}_{i+1} := \mathbf{r}_{i+1} + \mathbf{r}_{i+3}$
$\mathbf{g}_{i+2} := \mathbf{r}_{i+2} + \mathbf{r}_{i+3}$
end

$i := i + 3$
end

— $(l = k+1)$. The vectors $\mathbf{g}_p, \ldots, \mathbf{g}_q$ are obtained as follows

For $i = p$ to $q-1$, $\mathbf{g}_i := \mathbf{r}_i + \mathbf{r}_q$;
$\mathbf{g}_q := \mathbf{r}_q$.

It is easy to check that the vectors $\mathbf{g}_1, \ldots, \mathbf{g}_n$ are all binary vectors and, hence, **G** is the incidence matrix of a hypergraph.

Example 2.1 (continued). Starting from **R**, we obtain the following matrix

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

■

Lemma 2.3 The matrices **R** and **G** have the same row space.
Proof. The matrix **G** is obtained form **R** by iterating an elementary matrix operation which consist in adding a row to another row. In fact the only case in which we add two rows on another row

$$\mathbf{g}_i := \mathbf{r}_i + \mathbf{r}_{i+1} + \mathbf{r}_{i+2}$$

can be rewritten in two distinct steps as

$$\mathbf{g}_i' := \mathbf{r}_i + \mathbf{r}_{i+1}$$
$$\mathbf{g}_i := \mathbf{g}_i' + \mathbf{r}_{i+2}$$

It is very well known, from linear algebra, that iterating such elementary matrix operation on a matrix **M** preserve the row space of **M**.
In fact let **M** and **M′** be two matrix with n rows, where **M′** is obtained from **M** by adding a row $\mathbf{m}_a$ to another row $\mathbf{m}_b$. So if $\mathbf{m}_i$ are the rows of **M** then the rows of **M′** are

$$\mathbf{m}_i' = \begin{cases} \mathbf{m}_i & \text{if } i \neq b \\ \mathbf{m}_a + \mathbf{m}_b & \text{if } i = b \end{cases}$$

Suppose that **x** is a vector in the row space of **M**. Then there exist a linear combination of rows of **M** with real coefficients c_i, i=1,…,n such that

$$\sum_{i=1,\ldots,n} c_i \, \mathbf{m}_i = \mathbf{x}$$

Now let

$$\sum_{i=1,\ldots,n} d_i \, \mathbf{m}_i' = \mathbf{x}'$$

where $d_i = c_i$ if $i \neq a$ and $d_i = c_a - c_b$ if $i = a$. It is easy to check that $\mathbf{x}' = \mathbf{x}$. So **x** is also in the row space of **M′**. On the other hand let $\mathbf{x}'$ be a vector in the row space of **M′**. Then there exist a linear combination of rows of **M′** with real coefficients d_i, i=1,…,n such that

$$\sum_{i=1,\ldots,n} d_i \, \mathbf{m}_i' = \mathbf{x}'$$

now let

$$\sum_{i=1,\ldots,n} c_i \mathbf{m}_i = \mathbf{x}$$

where $c_i = d_i$ if $i \neq a$ and $c_i = d_b + d_a$ if $i = a$. Again it is easy to check that $\mathbf{x}' = \mathbf{x}$. So $\mathbf{x}'$ is also in the row space of **M**. This proves that **G** and **R** have the same row space. □

Corollary 2.2 The matrix **G** is the incidence matrix of a hypergraph which contains no algebraic edges and whose edge set is algebraic.
Proof. Let *G* be the hypergraph with incidence matrix **G**. By Lemma 2.3, the row spaces of **R** and **G** are the same. So, by Lemma 2.2, the edge set of *G* is algebraic and no edge of *G* is algebraic, which proves the statement. □

Theorem 2.1 The NAS problem is *NP*-complete.
Proof. First of all, given an edge set *F* of a hypergraph *G*, the problem of checking that another edge set is a nonempty algebraic subset of *F* is polynomial since it requires (1) testing non emptiness and inclusion in *F*, and (2) checking the consistency of an equation system such as equation system (1.3). Therefore, the problem is in *NP*. Moreover, by Lemmas 2.1, 2.2 and 2.3 the SS problem can be reduced to the NAS problem. Finally, our reduction is polynomial since the sizes of **t** and **G** are polynomial in the size of the SS problem. □

2.2 The query-execution plan E

Let $\mathbf{V}$ be a σ-view base and **v** an instance of $\mathbf{V}$ such that the map $(H, \boldsymbol{X}, \mathbf{v})$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$ is reduced with respect to evaluability, and let $\mathbf{q}$ be a σ-query. Recall that if V is a reduced σ-view base then the set of isolated loop of *H* correspond to the *d*-invariant edges of *H*. In particular the set of loop incident to zero-weighted nodes of *H* are also called the set of *null edges*. The set of variables of system $\Sigma(\boldsymbol{X}, \mathbf{v})$ associated to the set of null edges is called the set of *null variables*. We now provide a merely algebraic evaluability test, which reduces to the derivability test if *d* is R or Z. Henceforth, we adopt the following notation:

$\{e_j : j \in L\}$ is the set of isolated loops of *H*

for each $j \in L$, $i(j)$ is the node of *H* that the loop e_j is incident to

$L_o = \{j \in L : s_{i(j)} = 0\}$ is the set of isolated loops incident to zero-weighted nodes of *H*

M is the complement of *L*

$G = (N, M)$ is the subhypergraph of *H* induced by *M*

$\mathbf{G} = (\mathbf{g}_i)_{i \in N}$ is the incidence matrix of *G*.

By re-phrasing Theorem 1.1, a σ-query $\mathbf{q}$ with Ω-support *J* and Ω-cosupport *J′* is evaluable from **v** if and only if

(*a*) the sum-expression $\sum_{j \in J} x_j$ is a *d*-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$, and

(*b*) if *d* is R or Z, then $J' = \emptyset$; if $d = \mathsf{R}^+$, then *J′* is a (possibly empty) subset of L_o.

We now state an algebraic method for testing condition (*a*) for $d = \mathsf{R}^+$.

Affine Form of Farkas's Lemma (e.g., see [55], page 93). Let $\mathbf{H}\ \mathbf{x} \leq \mathbf{b}$ be a system of n linear inequalities with m unknowns such that the set $\mathbf{X}$ of its solutions is not empty. Suppose that the linear inequality $\sum_{j=1,\ldots,m} u_j\, x_j \leq d$ holds for each $\mathbf{x}$ in $\mathbf{X}$. If c is the maximum value of the linear function $\sum_{j=1,\ldots,m} u_j\, x_j$ over $\mathbf{X}$, then there exist n real nonnegative numbers $y_1, \ldots, y_n$, such that

(*i*) $u_j = \sum_{i=1,\ldots,n} y_i\, h_{ij}$ $\qquad (j = 1, \ldots, m)$

(*ii*) $c = \sum_{i=1,\ldots,n} y_i\, b_i$.

The following is a direct consequence of Farkas's Lemma.

Lemma 2.4 [39] Let $\mathbf{A}\ \mathbf{x} = \mathbf{q}$ be a system of n linear equations with m unknowns, such that the set $\mathbf{X}$ of its nonnegative solutions is not empty. If c is the maximum value of the linear function $\sum_{j=1,\ldots,m} u_j\, x_j$ over $\mathbf{X}$, then there exist n real numbers $\lambda_1, \ldots, \lambda_n$, and m nonnegative real numbers $v_1, \ldots, v_m$ such that

(*i*) $u_j = \sum_{i=1,\ldots,n} \lambda_i\, a_{ij} - v_j$ $\qquad (j = 1, \ldots, m)$

(*ii*) $c = \sum_{i=1,\ldots,n} \lambda_i\, q_i$.

Proof. First of all, we re-write the constrained equation system $\mathbf{A}\ \mathbf{x} = \mathbf{q}$, $\mathbf{x} \geq \mathbf{0}$ as a system of linear inequalities: $\mathbf{A}\ \mathbf{x} \leq \mathbf{q}$, $-\mathbf{A}\ \mathbf{x} \leq -\mathbf{q}$, $-\mathbf{x} \leq \mathbf{0}$, whose coefficient matrix $\mathbf{H}$ and constant vector $\mathbf{b}$ are

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} \\ -\mathbf{A} \\ -\mathbf{1} \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \mathbf{q} \\ -\mathbf{q} \\ -\mathbf{0} \end{bmatrix}.$$

By the Affine Form of Farkas's Lemma, there exist $2n+m$ real nonnegative numbers $y_1, \ldots, y_{2n+m}$ such that

(*i*) $u_j = \sum_{i=1,\ldots,2n+m} y_i\, h_{ij} = \sum_{i=1,\ldots,n} y_i\, a_{ij} - \sum_{i=1,\ldots,n} y_{n+i}\, a_{ij} - y_{2n+j}$

$= \sum_{i=1,\ldots,n} (y_i - y_{n+i})\, a_{ij} - y_{2n+j}$ $\qquad (j = 1, \ldots, m)$

(*ii*) $c = \sum_{i=1,\ldots,2n+m} y_i\, b_i = \sum_{i=1,\ldots,n} y_i\, q_i - \sum_{i=1,\ldots,n} y_{n+i}\, q_i = \sum_{i=1,\ldots,n} (y_i - y_{n+i})\, q_i$

The statement follows from setting $\lambda_i = y_i - y_{n+i}$, $1 \leq i \leq n$, and $v_j = y_{2n+j}$, $1 \leq j \leq m$. □

Using Lemma 2.4, we can prove the following.

Lemma 2.5 Let $\mathbf{A}\ \mathbf{x} = \mathbf{q}$ be a system of n linear equations with m unknowns, such that the set $\mathbf{X}$ of its nonnegative solutions is not empty. If a linear function $\sum_{j=1,\ldots,m} u_j\, x_j$ is constant over $\mathbf{X}$ with value c, then there exist n real numbers $\lambda_1, \ldots, \lambda_n$, and m nonnegative real numbers $v_1, \ldots, v_m$ such that

(*i*) $u_j = \sum_{i=1,\ldots,n} \lambda_i\, a_{ij} - v_j$ $\qquad (j = 1, \ldots, m)$

(*ii*) $c = \sum_{i=1,\ldots,n} \lambda_i\, q_i$

(*iii*) for each j, if $v_j \neq 0$ then x_j is a null variable.

Proof. (*If*) By Lemma 2.4, for every **x** in **X**, one has that

$$\sum_{j=1,\ldots,m} u_j x_j = \sum_{j=1,\ldots,m} (\sum_{j=1,\ldots,m} \lambda_i a_{ij} - v_j) x_j =$$

$$= \sum_{j=1,\ldots,m} \lambda_i (\sum_{j=1,\ldots,m} a_{ij} x_j) - \sum_{j=1,\ldots,m} v_j x_j =$$

$$= \sum_{i=1,\ldots,n} \lambda_i q_i$$

(*Only if*) Let c be the constant value of the function $\sum_{j=1,\ldots,m} u_j x_j$. By Lemma 2.4, one has

$$u_j = \sum_{i=1,\ldots,n} \lambda_i a_{ij} - v_j \qquad (j = 1, \ldots, m)$$

$$c = \sum_{i=1,\ldots,n} \lambda_i q_i.$$

Then, for every **x** in **X**, one has that

$$c = \sum_{j=1,\ldots,m} u_j x_j = \sum_{i=1,\ldots,n} \lambda_i q_i - \sum_{j=1,\ldots,m} v_j x_j = c - \sum_{j=1,\ldots,m} v_j x_j$$

and, hence,

$$\sum_{i=1,\ldots,m} v_j x_j = 0 .$$

By the nonnegativity constraints, each term $v_j x_j = 0$ for all j, which implies that it is not the case that for some j there exists **x** in **X** for which $v_j \neq 0$ and $x_j \neq 0$. □

Lemma 2.6 Let $\mathbf{A}\,\mathbf{x} = \mathbf{q}$ be a system of n linear equations with m unknowns, such that the set **X** of its nonnegative solutions is not empty. Let $\{x_j: j \in L_o\}$ be the set of null variables. Let $\mathbf{A}'\,\mathbf{x}' = \mathbf{q}$ be the equation system obtained from $\mathbf{A}\,\mathbf{x} = \mathbf{q}$ by deleting each occurrence of x_j, for all $j \in L_o$. A linear function $\sum_{j=1,\ldots,m} u_j x_j$ is constant over **X** if and only if the vector $\mathbf{u} = (u_j)_{j \in Z}$ is a linear combination of rows of $\mathbf{A}'$. Furthermore, if $\mathbf{u} = \sum_{i=1,\ldots,n} \lambda_i \mathbf{a}'_i$, then the value of the function is given by $\sum_{i=1,\ldots,n} \lambda_i q_i$.

Proof. Let $\mathbf{X}'$ be the set of nonnegative solutions $\mathbf{A}'\,\mathbf{x}' = \mathbf{q}$. Of course, the function $\sum_{j=1,\ldots,m} u_j x_j$ is constant over **X** if and only if the function $\sum_{j \in Z} u_j x'_j$ is constant over $\mathbf{X}'$. By parts (*i*) and (*iii*) of Lemma 2.5, the function $\sum_{j \in Z} u_j x'_j$ is constant over $\mathbf{X}'$ if and only if there exist n real numbers λ_1, …, λ_n, and, for each $j \in L_o$, there exists a nonnegative real number v_j such that

$$u_j = \sum_{i=1,\ldots,n} \lambda_i a_{ij} - v_j \qquad (j \in L_o)$$

for each $j \in L_o$, if $v_j \neq 0$ then x'_j is a null variable.

Since the equation system $\mathbf{A}'\,\mathbf{x}' = \mathbf{q}$ contains no null variables, one has that each v_j is zero so that the function $\sum_{j \in L_o} u_j x'_j$ is constant over $\mathbf{X}'$ if and only if there exist n real numbers λ_1, …, λ_n such that

$$u_j = \sum_{i=1,\ldots,n} \lambda_i a_{ij} \qquad (j \in L_o)$$

Finally, for every **x** in **X**, one has that

$$\sum_{j=1,\ldots,m} u_j x_j = \sum_{j \in L_0} u_j x'_j = \sum_{j \in L_0} (\sum_{i=1,\ldots,n} \lambda_i a_{ij}) x'_j = \sum_{i=1,\ldots,n} \lambda_i (\sum_{j \in L_0} a_{ij} x'_j)$$

$$= \sum_{i=1,\ldots,n} \lambda_i q_i \; . \qquad \square$$

The following are two straightforward consequences of Lemma 2.6.

Corollary 2.3 Let **A x** = **q** be a system of n linear equations with m unknowns, such that the set $\mathbf{X}$ of its nonnegative solutions is not empty. Let $\{x_j: j \in L_0\}$ be the set of null variables. Let $\mathbf{A'x'} = \mathbf{q}$ be the equation system obtained from **A x** = **q** by deleting each occurrence of x_j, for all $j \in L_0$. A variable x_{j*} is determined over $\mathbf{X}$ if and only if either $j^* \in L_0$ or the characteristic vector of the singleton $\{j^*\}$, that is, the $(m-|L_0|)$-dimensional binary vector **u** with

$$u_j = \begin{cases} 1 & \text{if } j = j^* \\ 0 & \text{else} \end{cases}$$

is a linear combination of rows of $\mathbf{A'}$. In the latter case, if $\mathbf{u} = \sum_{i=1,\ldots,n} \lambda_i \mathbf{a'}_i$ then the value of x_{j*} is $\sum_{i=1,\ldots,n} \lambda_i q_i$.

Corollary 2.4 Let **A x** = **q** be a system of n linear equations with m unknowns, such that the set $\mathbf{X}$ of its nonnegative solutions is not empty. Let $\mathbf{A'x'} = \mathbf{q}$ be the equation system obtained from **A x** = **q** by deleting all null variables. An equation in **A x** = **q** is redundant if and only if its corresponding equation in $\mathbf{A'x'} = \mathbf{q}$ is a linear combination of remaining equations.

We now turn to the problem of testing condition (*c*) of Theorem 1.1 Since the equation system **G** $\mathbf{x}_M$ = **w** has no null variables, by Lemma 2.6 the function $\sum_{j \in J \cap M} x_j$ is constant if and only if the characteristic vector **u** of $J \cap M$, that is, the vector **u** with

$$u_j = \begin{cases} 1 & \text{if } j \in J \cap H \\ 0 & \text{else} \end{cases}$$

is a linear combination of rows of **G**. If r is the number of rows of the matrix **G**, then we have to check the consistency of an equation system with r unknowns, say $\lambda_1, \ldots, \lambda_r$:

$$\mathbf{u} = \sum_{i=1,\ldots,r} \lambda_i \mathbf{g}_i$$

If a solution exists, then the function $\sum_{j \in J \cap M} x_j$ is constant with value

$$\sum_{i=1,\ldots,r} \lambda_i w_i$$

and the total for K amounts to $\sum_{j \in J \cap L} c_j + \sum_{i=1,\ldots,r} \lambda_i w_i$. Finally, since a solution (if any) of an equation system of size s can be found in $O(s^3)$ time [13], the following holds.

Lemma 2.7 Let $\mathbf{V}$ be a σ-view base and **v** an instance of $\mathbf{V}$ such that the map $(H, \boldsymbol{X}, \mathbf{v})$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$ is reduced. A sum-expression $\sum_{j \in J} x_j$ is an R^+-invariant of $\Sigma(\boldsymbol{X}, \mathbf{v})$ if and only if the set $\{e_j: j \in J \cap M\}$ of edges of G is algebraic. If this is the case and the characteristic vector of $J \cap M$ can be written as

$$\sum_{i \in N} a_i \mathbf{g}_i$$

then the value of the sum-expression is given by

$$\sum_{j \in J \cap L} v_{i(j)} + \sum_{i \in N} a_i v_i \,. \qquad (2.4)$$

Proof. The statement follows from Corollary 2.4.

By Lemma 2.7, the R^+-invariance of the sum-expression $\sum_{j \in J} x_j$ can be decided simply by checking the consistency of the following equation system in the unknowns a_i:

$$\sum_{i \in N} a_i \mathbf{g}_i = \mathbf{b} \qquad (2.5)$$

where **b** is the characteristic vector of the set $\{e_j : j \in J \cap M\}$ of edges of G. Combining Theorem 1.1 and Lemma 2.7, one has

Theorem 2.1 Let **V** be a σ-view base and **v** an instance of **V** such that the map $(H, \boldsymbol{X}, \mathbf{v})$ of $\Sigma(\boldsymbol{X}, \mathbf{v})$ is reduced. A σ-query **q** with Ω-support J and Ω-cosupport J' is evaluable from (**V**, **v**) if and only if

(*a*) if G is not empty, then the set $\{e_j : j \in J \cap M\}$ of edges of G is algebraic, and

(*b*) either $J' = \emptyset$ or $d = \mathsf{R}^+$ and $J' \subseteq L_o$.

Moreover, if **q** is evaluable from **v** and the characteristic vector of the edge set $\{e_j : j \in J - L\}$ can be expressed by (2.4), then the answer to **q** is given by (2.5).

By Theorem 2.1 the query-execution plan can easily decide whether **q** is or is not evaluable from **v** and, if this is the case, compute the answer to **q** from **v**. If **q** is not evaluable from **v**, then as in Section 2.1 the query-execution plan looks for an S'-category Q' such that the σ-query **q**′ with target Q' is evaluable from **v** and Q' is contained in (or contains) the extension Q to S' of the target of **q**.

Example 1 (continued). Consider again the σ-query **q** with support $J = \{4, 5, 9\}$ and co-support $J' = \{1, 2, 3\}$, and the map for **v*** (see Figure 5) corresponding to the case that the domain of SALARY is R^+. Thus, one has $L_o = \{1, 3, 5, 9\}$ and $M = \emptyset$. We saw that **q** is not evaluable from **v***. An S'-category Q' that is (maximally) contained in the target of **q** and is the target of a σ-query **q**′ evaluable from **v*** is given by the following relation over {GENDER, DEPT}

GENDER	DEPT
male	A
male	C
male	D
female	D
male	E
female	E
male	I
female	I

and the answer to **q**′ is 6. ■

If d is R or Z, then plan E is like plan D and we proved that finding an S'-category that is maximally contained in Q is an *NP*-hard problem. Of course, the case $d = \mathsf{R}^+$ is at least hard as the case $d = \mathsf{R}$, so that

$$Q' := (\cup_{j \in J' \cap L_o} Y_j) \cup (\cup_{j \in J \cap L} Y_j)$$

is the only choice for the target of $\mathbf{q}'$ that can be made efficiently; then, the answer to $\mathbf{q}'$ can be computed as

$$\sum_{j \in J \cap L} v_{i(j)}.$$

2.2.1 Proving coNp-completeness of computing a Z^+–invariant sets.

As stated at the end of Section 1.4.1, the problem of deciding if there exists some variable of the equation system (1.1) that is a Z^+-invariant is co*NP*-complete and we now give its proof. Note that an analogous result holds for {0, 1}-invariants and the result is known as the co*NP*-completeness of the Boolean Auditing Problem [37] (see Remark below).

Let $\mathbf{A}$ be an $r \times n$ dimensional binary matrix, where each row contains at most three 1's. It is well-known that deciding if the set

$$\boldsymbol{Z} = \{\mathbf{z} \in \{0, 1\}^n : \mathbf{A}\,\mathbf{z} = \mathbf{1}\}$$

is not empty, is an *NP*-complete problem [25]. We now provide a polynomial reduction of this problem to an instance of the problem of deciding if no variable of $\mathbf{H}\,\mathbf{x} = \mathbf{v}$ is a Z^+-invariant.

For each $i \in [n]$, let us introduce fifteen Z^+-valued variables $x_{i,1}, \ldots, x_{i,15}$ and the nine equations

$$x_{i,1} + x_{i,2} + x_{i,3} + x_{i,7} + x_{i,8} + x_{i,15} = 3 \qquad 1(i)$$

$$x_{i,4} + x_{i,5} + x_{i,6} + x_{i,9} + x_{i,10} + x_{i,15} = 3 \qquad 2(i)$$

$$x_{i,1} + x_{i,4} = 1 \qquad x_{i,2} + x_{i,5} = 1 \qquad x_{i,3} + x_{i,6} = 1 \qquad 3(i)$$

$$x_{i,7} + x_{i,11} = 1 \qquad x_{i,8} + x_{i,12} = 1 \qquad 4(i)$$

$$x_{i,9} + x_{i,13} = 1 \qquad x_{i,10} + x_{i,14} = 1 \qquad 5(i)$$

First of all, note that, by equations 3(i)-5(i) in every Z^+-valued solution $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,15})$ of this equation system, $x_{i,1}, \ldots, x_{i,14}$ can only assume the values 0 and 1 and the sum-expression

$$x_{i,1} + x_{i,2} + x_{i,3}$$

can only assume the values 1 and 2 for, otherwise, if $x_{i,1} + x_{i,2} + x_{i,3} = 0$, then one would have $x_{i,4} + x_{i,5} + x_{i,6} = 3$ by 3(i), $x_{i,15} = 0$ by 2(i) and equations 1(i) would be inconsistent, and if $x_{i,1} + x_{i,2} + x_{i,3} = 3$, then one would have $x_{i,15} = 0$ by 1(i), $x_{i,4} + x_{i,5} + x_{i,6} = 0$ by 3(i) and equations 2(i) would be inconsistent. Moreover, since the sum-expression $x_{i,1} + x_{i,2} + x_{i,3}$ can only assume the value 1 and 2, one has that also $x_{i,15}$ can only assume the values 0 and 1, so that every Z^+-valued solution $\mathbf{x}_i$ is binary. Explicitly, there are twelve Z^+-valued solutions with if $x_{i,1} + x_{i,2} + x_{i,3} = 1$ and twelve Z^+-valued solutions with if $x_{i,1} + x_{i,2} + x_{i,3} = 2$. Explicitly, one has that

— if $x_{i,1} + x_{i,2} + x_{i,3} = 1$, then the Z^+-valued solution set is the Cartesian product of

$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	$x_{i,4}$	$x_{i,5}$	$x_{i,6}$
0	0	1	1	1	0
0	1	0	1	0	1

1	0	0	0	1	1

with

$x_{i,7}$	$x_{i,8}$	$x_{i,9}$	$x_{i,10}$	$x_{i,11}$	$x_{i,12}$	$x_{i,13}$	$x_{i,14}$	$x_{i,15}$
1	1	0	1	0	0	1	0	0
1	1	1	0	0	0	0	1	0
0	1	0	0	1	0	1	1	1
1	0	0	0	0	1	1	1	1

— if $x_{i,1} + x_{i,2} + x_{i,3} = 2$, then the Z^+-valued solution set is the Cartesian product of

$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	$x_{i,4}$	$x_{i,5}$	$x_{i,6}$
0	1	1	1	0	0
1	0	1	0	1	0
1	1	0	0	0	1

with

$x_{i,7}$	$x_{i,8}$	$x_{i,9}$	$x_{i,10}$	$x_{i,11}$	$x_{i,12}$	$x_{i,13}$	$x_{i,14}$	$x_{i,15}$
0	1	1	1	1	0	0	0	0
1	0	1	1	0	1	0	0	0
0	0	0	1	1	1	1	0	1
0	0	1	0	1	1	0	1	1

Note that none of the fifteen variables is a Z^+-invariant of the equation system 1(i)-5(i) even if the value of $x_{i,1} + x_{i,2} + x_{i,3}$ is fixed.

At this point, for each $i \in [n]$, replace the binary variable z_i in the equation system $\mathbf{A}\,\mathbf{z} = \mathbf{1}$ by the sum-expression $x_{i,1} + x_{i,2} + x_{i,3} - 1$ and add the nine equations 1(i)-5(i). Let

$$\mathbf{B}\,\mathbf{x}' = \mathbf{b}$$

be the resulting equation system, where

$$\mathbf{x}' = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$$

$$\mathbf{b} = (b_1, \ldots, b_m) \qquad (m = r + 9n).$$

Note that each b_h is not greater than 4. The set of Z^+-valued solutions of $\mathbf{B}\,\mathbf{x}' = \mathbf{b}$ is the set of vectors $\mathbf{x}'$ such that

(a) each $\mathbf{x}_i$ is a Z^+-valued solution of the equation system 1(i)-5(i), and

(b) the vector $\mathbf{z} = (z_1, \ldots, z_n)$, where $z_i = x_{i,1} + x_{i,2} + x_{i,3} - 1$, is a binary solution of

$\mathbf{A}\,\mathbf{z} = \mathbf{1}$.

Therefore, the set of Z^+-valued solutions of $\mathbf{B}\ \mathbf{x}' = \mathbf{b}$ is empty if and only if $\boldsymbol{Z} = \emptyset$ and, if $\boldsymbol{Z} \neq \emptyset$, then no variable of $\mathbf{B}\ \mathbf{x}' = \mathbf{b}$ is a Z^+-invariant.

Finally, let us introduce seven more Z^+-valued variables: $x_{o,1}, \ldots, x_{o,7}$ and the six equations

$$x_{o,1} + x_{o,5} = 1 \quad x_{o,2} + x_{o,5} = 1 \quad x_{o,2} + x_{o,6} = 1$$
$$x_{o,3} + x_{o,6} = 1 \quad x_{o,3} + x_{o,7} = 1 \quad x_{o,4} + x_{o,7} = 1$$

Note that system of these six equations has exactly two Z^+-valued (and binary) solutions:

$$x_{o,1} = x_{o,2} = x_{o,3} = x_{o,4} = 0 \qquad x_{o,5} = x_{o,6} = x_{o,7} = 1$$
$$x_{o,1} = x_{o,2} = x_{o,3} = x_{o,4} = 1 \qquad x_{o,5} = x_{o,6} = x_{o,7} = 0.$$

Finally, for each $h \in [m]$, add to the left-hand side of the h-th equation in $\mathbf{B}\ \mathbf{x}' = \mathbf{b}$ the sum-expression

$$\sum_{k=1,\ldots,b_h} x_{o,k}$$

and add the six equations above with unknowns $x_{o,1}, \ldots, x_{o,7}$. Let $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ be the resulting equation system where $\mathbf{x} = (\mathbf{x}_o, \mathbf{x}')$. A trivial Z^+-valued solution of $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ can be obtained by taking

$$\mathbf{x}_o = (1\ 1\ 1\ 1\ 0\ 0\ 0) \qquad \mathbf{x}' = \mathbf{0}$$

Let $\boldsymbol{\mathcal{X}}$ be the set of variables of $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ that are Z^+-invariants. We now prove that $\boldsymbol{\mathcal{X}} = \emptyset$ if and only if $\boldsymbol{Z} \neq \emptyset$, which implies that deciding if there exists some variable of $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ that is a Z^+-invariant is a co*NP*-complete problem.

(*if*) If $\boldsymbol{Z} \neq \emptyset$ and $\mathbf{z} \in \boldsymbol{Z}$, then we can obtain at least 12^n nontrivial Z^+-valued solutions $\mathbf{x} = (\mathbf{x}_o, \mathbf{x}')$ of $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ by taking $\mathbf{x}_o = (0\ 0\ 0\ 0\ 1\ 1\ 1)$ and setting $\mathbf{x}_i$, $1 \leq i \leq n$, to each of the twelve Z^+-valued solutions of the equation system 1(i)-5(i) having $x_{i,1} + x_{i,2} + x_{i,3}$ equal to $z_i + 1$. It follows that no variable of $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ is a Z^+-invariant, that is, $\boldsymbol{\mathcal{X}} = \emptyset$.

(*only if*) If $\boldsymbol{Z} = \emptyset$, then the trivial solution of $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ is the only Z^+-valued solution so that each variable of $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ is a Z^+-invariant, that is, $\boldsymbol{\mathcal{X}} \neq \emptyset$ ($|\boldsymbol{\mathcal{X}}| = 7 + 15n$).

Remark. We saw that the Z^+-valued solutions of the equation system $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ are all binary. It follows that the set of variables of $\mathbf{H}\ \mathbf{x} = \mathbf{v}$ that are $\{0, 1\}$-invariants is empty if and only if $\boldsymbol{Z} \neq \emptyset$, which implies the co*NP*-completeness of the Boolean Auditing Problem [37]. (The proof of the co*NP*-completeness of the Boolean Auditing Problem in [37] contains a flaw and in a private communication the authors of [37] agreed with us on the above proof of their statement.)

Chapter 3

Auditing Sum-Queries to Make a Statistical Database Secure

3.1 Introduction

Consider a statistical database SDB containing a relation name `Personnel` with scheme {`NAME`, `SSN`, `GENDER`, `AGE`, `SALARY`}. The users of the SDB can ask for summary statistics on the attribute `SALARY` for groups of employees which at the conceptual level are specified by predicates involving the attributes `GENDER`, `AGE` and `DEPARTMENT` but not `NAME` and `SSN` which are private attributes.

Let D be an instance of the SDB containing the relation R of name `Personnel`. If `SALARY` is a confidential attribute, then answering **q** (and, more in general, answering a statistical query whose response variable is a confidential attribute) raises concerns on the compromise of individual privacy since releasing the value of **q** could lead to the ("exact" or "approximate") disclosure of the value of `SALARY` for some element of the query-set of **q** [17, 18, 61, 62]. Such a sum-query, which risks the confidentiality of the response variable and, hence, the security of the SDB, is called *intrusive* and the category S specified by its "where"-clause is said to be *sensitive* in D. Typically, for a fixed positive integer k, S is sensitive if the number of tuples in R that fall in S are less than k (exact disclosure) or there are k or fewer tuples in R that give a dominant contribution to the value of **q** (approximate disclosure) [17, 18, 61, 62].

The confidentiality of a response variable σ can be attacked either (in a directed way) by an intrusive sum-query or (in an indirect way) by a non-intrusive sum-query whose value on D, combined with the responses to previously-answered sum-queries on D, leads to an accurate estimate of the total of σ for some category that is sensitive in D. In the latter case, we call the sum-query *tricky*.

In order to make an SDB secure, when a new instance D is created, for each confidential attribute σ the sensitive categories in D are identified and each of them is assigned a fixed nonnegative number, called its *protection level* [61, 62]. Such a category S will be considered *protected* at a certain time if its protection level comes out to be less than the width of the interval, called the *feasibility range* [61, 62], of the feasible values for the total of σ for S that are permitted by the responses to previously answered sum-queries. In our proposal, if the current sum-query **q** is recognized to be intrusive or tricky, then the query system of the SDB will give a non-informative response to **q** by issuing the feasibility range for **q** determined by the responses to all previously answered sum-queries with response variable σ. Now, it is easy to decide whether **q** is or is not intrusive since it is sufficient to check the presence of the category specified by **q** in the list of the categories that are sensitive in D for σ; but, deciding whether **q** is or is not tricky requires 'auditing' [11, 12, 40, 42, 46] the previously answered sum-queries on D with response variable σ and, for each sensitive category S, comparing the protection level assigned to S with the width of the feasibility range for the total of σ for S determined by the value of **q** and by the responses to the previously answered sum-queries on σ. If each sensitive category is protected, then we say that **q** can be *safely* answered and the value of **q** will be issued. A special case occurs when **q** is evaluable from previously answered sum-queries, that is, when the value of **q** is uniquely determined by them; then, **q** is neither intrusive nor tricky and it can be safely answered.

The scenario we are considering can be depicted as a competitive game played by the query system, which has as its opponent a hypothetical statistical user, henceforth referred to as the (*data*)

snooper, who (with no prior information) attempts to pry an accurate estimate of the total of a response variable σ for some sensitive category out of the responses to all answered sum-queries on *D* with response variable σ. This assumption is not unrealistic if the query system of an SDB (also in order to lighten its workload) allows every user to look into the archive of answered sum-queries on *D*.

In most previous work [1, 11, 12, 40, 42, 46], the technique of auditing was applied under the assumption that the snooper also knows the query-set of each answered sum-query. Thus, for each answered sum-query the snooper can write down an equation, whose unknowns represent the unknown values of the response variable for the tuples in its query-set. As a consequence, the size of the snooper's model comes out to be proportional to the size of the instance *D* of the SDB, which may contain a very large number of tuples [1]. On the other hand, the hypothesis that the snooper knows the query-sets of answered sum-queries is not realistic. In order to make the snooper's model independent of the size of the instance of the SDB, some authors [9, 10, 11, 48] have suggested working with categories instead of query-sets. Accordingly, in order to model the information conveyed by a set of answered sum-queries, we shall introduce an equation system whose unknowns correspond to the classes of a suitable partition of the union of the categories specified by the answered sum-queries. Thus, typically (i.e., with a very large database) the size of our equation system comes out to be far less than the size of the equation system based on query-sets.

To repel the attacks of the snooper, the query system will make use of its own information model, which essentially is the same as the snooper's model and will be constructed incrementally as the value of a new-sum query is issued. Such an information model was described in Chapter 1. Finally, using a the information model, we shall address the question whether or not a new sum-query can be safely answered.

Answering this question raises some computational problems (recognizing evaluable sum-queries, updating the information model, computing a feasibility range), whose solutions depend on the data type of the response variable as seen in Section 1.4. If it is of real type, then standard algebraic methods can be used to solve all of them efficiently (see Section 1.4.1); moreover, if sum-queries contain all the "group-by" clause, that is, if they are table queries (or "cuboids" [58, 59]), then there also exist cardinality-based conditions that are sufficient for them to be inference free [58, 59, 63]. If it is of nonnegative type, then we can resort to linear-programming or integer linear-programming methods depending on the specific data type. In general, the case that the response variable is of nonnegative-integer type is extraordinarily difficult from a computational viewpoint as we seen in Section 2.2.1 where finding a in general computing a $\mathbb{Z}^+$-invariant set is an *coNP*-Complete problem and, a general theory has yet to be developed [22].

In this Chapter, we only consider the case that the response variable is of nonnegative real type. Then, a natural approach consists in resorting to standard linear-programming algorithms (e.g., the simplex method [13, 55]). Unfortunately, none of them is polynomial even if they are polynomial on the average and have good performances in practice [13, 55]; on the other hand, existing polynomial linear-programming algorithms (e.g., the ellipsoid method [13, 55]) have bad performances in practice. Therefore, in order to solve the computational problems raised by the security of the SDB, it is convenient to make a parsimonious use of standard linear-programming algorithms and "there is considerable interest in finding alternative techniques" [22]. Accordingly, using the techniques presented in Chapter 1 and Chapter 2, we will present a cubic evaluability test based on standard algebraic methods, and we shall show in Chapter 5 that, in the case that the current information model is "graphical", then also the problem of finding a feasibility range can be solved efficiently without resorting to linear programming at all.

3.2 Basic Definitions

Suppose we are given an instance *D* of an SDB which contains a relation *R*. Let *a* be an attribute in the schema of *R* that is used by such a sum-query. Typically, if the domain of *a* is large, then the "where"-clause of the sum-query will contain re-coded values in such a way that the size of the re-

coded domain of a is made small [61, 62]. For example, the re-coding of the attribute AGE may consist of year classes instead of numbers of years, or the re-coding of a geographic attribute with a hierarchical structure (e.g., country, state, county) may consist in 'chopping off' some digits of its values of precision [58, 62].
By V_σ we denote the set of all categorical variables corresponding to the attributes in the schema of R that can occur in the "where"-clauses of sum-queries with response variable σ.
In order to speed up the evaluation of sum-queries with response variable σ, the query system will make use of a materialized aggregate view on σ, which will be referred to as the *summary table* on σ; it is created initially and reports the total of σ for each V_σ-cell c. As we said, a user can ask for the total of a statistic of σ by submitting a sum-query **q**, where the selection criterion is specified be an arbitrary (consistent) condition P on a (possibly empty) subset S of the set V_σ of categorical variables built up with logical connectives. Owing to the assumptions of mutual exclusiveness and global exhaustiveness of cells, P uniquely determines a category Q, we call the *target* of **q**, such that P is logically equivalent [44] to the formula (see also Section 1.2)

$$\vee_{c \in Q} \ (\wedge_{a \in S} \ a = c(a)).$$

The value q of **q** can then be computed by the query system from the summary table on σ (without accessing the database relation R) simply as the total of σ for Q.

Example 3.1. Consider an instance of an SDB containing a relation of name Personnel with schema {NAME, GENDER, AGE, SALARY}. Here, SALARY is assumed to be an attribute of nonnegative real type. A sum-query with response variable SALARY will take its categorical variables from the set V_{SALARY} = {GENDER, AGE}. We assume that the domains of GENDER and AGE are {M, F} and {young, middle, old}, respectively, and that the summary table on SALARY contains the following data.

GENDER	AGE	SALARY
M	young	15.0
M	middle	9.0
M	old	7.5
F	young	6.5
F	middle	1.5
F	old	0.0

Consider the following four sum-queries with response variable SALARY:

q1: select **sum**(SALARY)
from Personnel
where GENDER = M and AGE ≠ old

q2: select **sum**(SALARY)
from Personnel
where (GENDER = M and AGE ≠ young) or (GENDER = F and AGE = middle)

q3: select **sum**(SALARY)
from Personnel
where (GENDER = M and AGE ≠ middle) or (GENDER = F and AGE = young)

q4: select **sum**(SALARY)
from Personnel

where GENDER = F and AGE ≠ middle.

Their targets are the following relations with schema {GENDER, AGE}:

V_1	V_2	V_3	V_4
(M, young)	(M, middle)	(M, young)	(F, young)
(M, middle)	(M, old)	(M, old)	(F, old)
	(F, middle)	(F, young)	

Using the summary table on SALARY, the query system is able to compute the values

$v_1 = 24$ $v_2 = 18$ $v_3 = 29$ $v_4 = 6.5$

$\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$ and $\mathbf{q}_4$ without accessing the relation of name Personnel. ■

In the next section, we shall see how the snooper can use the information released by the query system in response to sum-queries to get a more-or-less detailed information on the total of σ for a (possibly sensitive) category of interest.

3.3 The snooper's model

Let $\mathbf{V}=(\mathbf{q}_1, \ldots, \mathbf{q}_n)$ be answered sum-queries whose response variable σ is of nonnegative real type. Recalling the terminology used in Chapter 1, $\mathbf{V}$ can be seen as a set of σ-views. The amount of information conveyed by the answers to $\mathbf{V}$ will be defined in such a way to capture all the snooper's knowledge about them (that is, their targets and their values). The snooper's information model consists of a set of variables, each of which is interpreted as the total of σ for a certain category, and of a system of linear constraints (1.1).

From now on, for simplicity, we assume that V_i is the extension to V_σ of the target of $\mathbf{q}_i$. Also let v_i, $i = 1, \ldots, n$, be the value of $\mathbf{q}_i$. By $\Omega \subseteq dom(V_\sigma)$ we denote the set of cells contained in the categories V_i, that is, $\Omega = \cup_{i=1,\ldots,n} V_i$. Let $\boldsymbol{X} = \{X_1, \ldots, X_m\}$ be the classification system of Ω. The snooper's information model is described by the system of n linear equations (1.1) where, each variable x_j stands for a feasible value of the total of σ for X_j and, hence, is subject to the nonnegativity constraint: $x_j \geq 0$.

To sum up, the information model has a semantic component, given by the classification system $\boldsymbol{X}$, and an analytical component, given by system (1.1).

Example 3.1 (continued). Consider again the four sum-queries $\mathbf{q}_1, \ldots, \mathbf{q}_4$. Then $\Omega = dom(\{$AGE, GENDER$\})$ and the classification system $\boldsymbol{X}$ of $\mathbf{V}$ is formed by the following six elementary categories (i.e., singletons):

$X_1 = \{$(M, young)$\}$ $X_2 = \{$(M, middle)$\}$ $X_3 = \{$(M, old)$\}$
$X_4 = \{$(F, young)$\}$ $X_5 = \{$(F, middle)$\}$ $X_6 = \{$(F, old)$\}$.

The targets of $\mathbf{q}_1, \ldots, \mathbf{q}_4$ can be written as

$V_1 = X_1 \cup X_2$
$V_2 = X_2 \cup X_3 \cup X_5$
$V_3 = X_1 \cup X_3 \cup X_4$
$V_4 = X_4 \cup X_6$

and system (1.1) reads

$$\begin{cases} x_1 + x_2 = 24 \\ x_2 + x_3 + x_5 = 18 \\ x_1 + x_3 + x_4 = 29 \\ x_4 + x_6 = 6.5 \end{cases} \qquad (3.1) \qquad \blacksquare$$

Of course, system (3.1) has at least one nonnegative solution. Suppose now that the snooper is interested in the total of σ for a (possibly sensitive) category K of interest. He can obtain the tightest lower bound and the tightest upper bound on the total σ for K as follows. Let (see Figure 3.1)

$J = \{j \in [m]: X_j \subseteq K\}$,
$J' = \{j \in [m]: X_j \cap K \neq \emptyset \text{ and } X_j - K \neq \emptyset\}$.

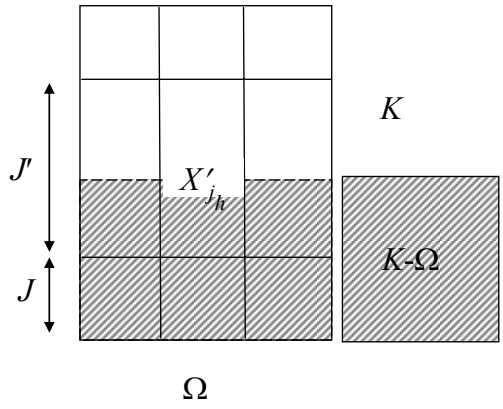


Figure 3.1. Overlap of category K with basis $\boldsymbol{X}$

As stated in Section 1.4 the tightest lower bound, denoted by *lower*(K), is set to zero if $J = \emptyset$; otherwise, it is taken to be

$$lower(K) = \mathbf{min}\ \{\textstyle\sum_{j \in J} x_j: \mathbf{H}\,\mathbf{x} = \mathbf{v}, \mathbf{x} \geq \mathbf{0}\}\ .$$

The tightest upper bound, denoted by *upper*(K), is set to $+\infty$ if K is not contained in Ω; otherwise, it is taken to be

$$upper(K) = \mathbf{max}\ \{\textstyle\sum_{j \in J \cup J'} x_j: \mathbf{H}\,\mathbf{x} = \mathbf{q}, \mathbf{x} \geq \mathbf{0}\}\ .$$

The interval [*lower*(K), *upper*(K)] is called the *feasibility range* of the total for K. Recall that if *lower*(K) = *upper*(K), then the total for K is evaluable in the information model.

Example 3.1 (continued). The general solution of system (3.1) is

$$(x_1 = 24 - x_2, x_2, x_3, x_4 = 5 + x_2 - x_3, x_5 = 18 - x_2 - x_3, x_6 = 1.5 - x_2 + x_3).$$

By the nonnegativity constraints, the couple (x_2, x_3) is any point of the region shown in Figure 3.2.

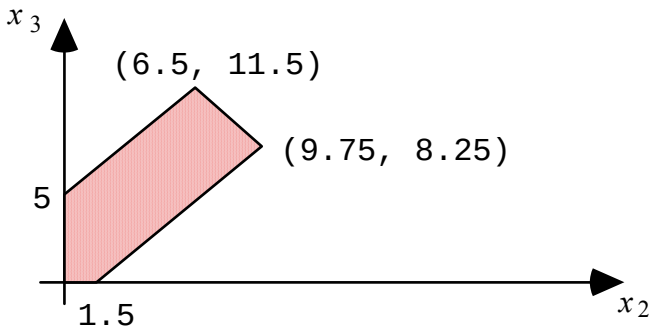


Figure 3.2. The set of nonnegative solutions of system (3.1)

Thus, if the snooper is interested in the total of SALARY for the category

K
(F, middle)
(F, old)

then, after computing the minimum and the maximum of the function $x_5 + x_6$ using a standard linear-programming method, he can find that the feasibility range for the total of SALARY for K is [0,19.5]. ■

3.4 How to beat the snooper

To repel the attacks of the snooper to the confidentiality of the response variable, the query system will make use of its own information model, which will be constructed incrementally as the value of a new-sum query is issued. Suppose that, after a certain number of answered sum-queries, every sensitive category is protected in the query system's model and a new sum-query **q** is submitted. We call the query system's model the *prior model*. As said in Section 3.1, if **q** is intrusive or tricky, then **q** should be answered by issuing the feasibility range of its value given by the prior model; otherwise, the value of **q** should be issued. Since recognizing intrusive sum-queries is a matter of routine, the most demanding task that the query system is called to carry out is how to decide whether or not **q** is tricky. Of course, if **q** is evaluable in the prior model, then **q** is not tricky. Let us assume that **q** is not evaluable. Then, the query system should add to the prior model the piece of information conveyed by the value of **q**, and in the "augmented" information model, we call the *posterior model*, it should check that each sensitive category is still protected. If this is the case, then (and only then) **q** is not tricky. The posterior model is constructed from the prior model given the target K and the value v of **q** as described in Section 1.3 of Chapter 1.

After constructing the posterior model, each sensitive category V contained in Ω' is examined by computing the feasibility range of the total for V in the posterior model which is, then, compared with the protection level assigned to V. If all the sensitive categories contained in Ω' are protected in the posterior model, then (and only then) **q** is not tricky.

*Example 3.*2. Consider again the database instance of Example 3.1. Suppose there are only two sensitive categories for SALARY:

V_1	V_2
(M, young)	(M, young)
	(F, old)

both of which have been assigned the protection level `3.0`. Assume that the four sum-queries $\mathbf{q}_1$, …, $\mathbf{q}_4$ of Example 3.1 are submitted in this order. We now show that each of them can be safely answered. The following are the four information models resulting from answering the sum-queries $\mathbf{q}_1$, …, $\mathbf{q}_4$ respectively, and the feasibility ranges for the sensitive categories. Note that answering $\mathbf{q}_1$, $\mathbf{q}_2$ and $\mathbf{q}_3$ risks V_1 only, but answering $\mathbf{q}_4$ risks V_2 too.

$$\{x_1 = 24$$

X_1= {(M, young), (M, middle)}

feasibility range for V_1: [0,24]

$$\begin{cases} x_1 + x_2 = 24 \\ x_2 + x_3 = 18 \end{cases}$$

X_1 = {(M, young)} X_2 = {(M, middle)}
X_3 = {(M, old), (F, middle)}

feasibility range for V_1: [6,24]

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

X_1 = {(M, young)} X_2 = {(M, middle)} X_3 = {(M, old)}
X_4 = { (F, young)} X_5 = {(F, middle)}

feasibility range for V_1: [6,24]

$$\begin{cases} x_1 + x_2 = 24 \\ x_2 + x_3 + x_5 = 18 \\ x_1 + x_3 + x_4 = 29 \\ x_4 + x_6 = 6.5 \end{cases}$$

X_1 = {(M, young)} X_2 = {(M, middle)} X_3 = {(M, old)}
X_4 = { (F, young)} X_5 = {(F, middle)} X_6 = {(F, old)}

feasibility range for V_1: [14.25,24]
feasibility range for V_2: [14.25,30.5]

Suppose now that the following sum-query is submitted after $\mathbf{q}_4$:

$\mathbf{q}_5$: select **sum**(SALARY)
from Personnel
where GENDER = F and AGE ≠ young.

The target of $\mathbf{q}_5$ is

K_5
(F, middle)
(F, old)

and the value of $\mathbf{q}_5$ is v_5 = 1.5 (obtained from the summary table). Neither the category K_5 is sensitive nor the total for K_5 is evaluable in the prior model. The posterior model consists of the following equation system

$$\begin{cases} x_1 + x_2 = 24 \\ x_2 + x_3 + x_5 = 18 \\ x_1 + x_3 + x_4 = 29 \\ x_4 + x_6 = 6.5 \\ x_5 + x_6 = 1.5 \end{cases} \qquad (3.2)$$

with the same meaning of the variables as in the prior model. The general solution of system (3.2) is

$$(x_1 = 15, x_2 = 9, x_3, x_4 = 14 - x_3, x_5 = 9 - x_3, x_6 = -7.5 + x_3)$$

By the nonnegativity constraints, x_3 is any number in the interval [7.5, 9]. Note that the sensitive category V_1 is exactly the category associated with x_1 and, since the value of x_1 is uniquely determined, V_1 would not be protected if the value of V_5 were released. Therefore, V_5 is tricky and, in response to V_5, the query system will issue the feasibility range [0, 19.5] for K_5 given by the prior model (see Example 3.1). ■

To sum up, a procedure for answering a sum-query **q** with target K and value v should work as follows:

(*i*) If K is sensitive, then the answer to **q** will be the feasibility range of the total for K in the prior model.

(*ii*) If the total for K is evaluable in the prior model, then the answer to **q** will be v.

(*iii*) If neither K is sensitive nor the total for K is evaluable in the prior model, then the posterior model will be constructed, and for each sensitive category V contained in $\Omega' = \Omega \cup K$, the feasibility range of the total for V in the posterior model will be computed and compared with the protection level associated with V so that, if V is not protected, then the answer to **q** will be the feasibility range of the total for K in the prior model.

(*iv*) If each sensitive category V contained in Ω' is protected in the posterior model, then the answer to **q** will be v, and the posterior model will serve as the prior model for processing the next sum-query.

From a computational point of view, tasks (*i*), (*ii*) and (*iii*) require computing the feasibility range of the total for K and for each sensitive category contained in Ω'. Accordingly, we propose the following answering procedure.

Answering Procedure

Step 1. If K is sensitive, then

compute *lower*(K) and *upper*(K) in the prior model,

answer **q** by issuing the feasibility range for the total for K, and Exit.

Step 2. If the total for K is evaluable in the prior model, then answer **q** by releasing its value, and Exit.

Step 3. Construct the posterior model and find a normal form.

Step 4. For each sensitive category V contained in $K \cup \Omega$,

if the total for V is evaluable in the posterior model, then

compute *lower*(K) and *upper*(K) in the prior model,

answer **q** by issuing the feasibility range of the total for K, and Exit;

otherwise

compute the feasibility range [*lower*(V), *upper*(V)] in the posterior model,

compare its width with the protection level associated with V,

if V is not protected, then

compute *lower*(K) and *upper*(K) in the prior model,

answer **q** by issuing the feasibility range of the total for K, and Exit.

Step 5. Answer **q** by issuing its value.

3.5 A polynomial test for evaluability

The cost of the storage representation of an information model such as system (1.1) depends on the *size* of its equation system, by which we mean the sum of the numbers of occurrence of its variables. A storage representation of the information model proposed in Chapter 1 is the hypergraph of which the coefficient matrix **H** of system (1.1) is the (node-hyperedge) incidence matrix, where the i-th node is "weighted" by v_i and the j-th hyperedge is "labelled" by X_j. Such a storage representation is called the map of **V**. In Section 1.5 we saw a method to reduce the dimension of the map of **V**. Here on the assumption that $d=\mathbb{R}^+$ we see how further reduce the size of the system (1.1) and, hence, the storage cost of the information model.

Let **V** be a σ-view base and **v** an instance of **V** such that the map (H, $\boldsymbol{X}$, **v**) of Σ($\boldsymbol{X}$, **v**) is reduced with respect to evaluability. Henceforth, we adopt the following notation:

L is the set of isolated loops incident to nonzero-weighted nodes of H

L_o is the set of isolated loops incident to zero-weighted nodes of H

M is the complement of $L \cup L_o$

$G = (N, M)$ is the subhypergraph of H induced by M

$\mathbf{G} = (\mathbf{g}_i)_{i \in N}$ is the incidence matrix of G.

Let us consider its determined variables of system (1.1) whose values are zero. They will be referred to as null variables. If $L_o \neq \emptyset$, then system (1.1) contains the set of equations

$$x_j = 0 \qquad (j \in L_o)$$

which, by the nonnegativity constraints, is equivalent to the single equation

$$\sum_{j \in L_o} x_j = 0.$$

Let us introduce a new variable x_0 which stands for the sum $\sum_{j \in L_o} x_j$ and is to be interpreted as the total for the category $X_0 = \cup_{j \in L_o} X_j$. Then, we can replace the equations $x_j = 0$, $j \in L_o$, in system (1.1) by the single equation $x_0 = 0$ so that the resulting equation system has $|L_o|-1$ variables less and $|L_o|-1$ equations less. If $L_o \neq \emptyset$ and $L \neq \emptyset$, then, the resulting equation system is like

$$\begin{cases} x_0 = 0 \\ \mathbf{x}_L = \mathbf{c} \\ \mathbf{G}\,\mathbf{x}_M = \mathbf{w} \end{cases} \qquad (3.3)$$

System (3.3) with the interpretation of its variables given by the following partition of Ω

$$\{X_0\} \cup \{X_j : j \in L \cup M\}$$

defines what we call a *normal form* of the information model. Given system (3.3) and a category K, the feasibility range of the total for K can be obtained as follows (see Figure 3.3).

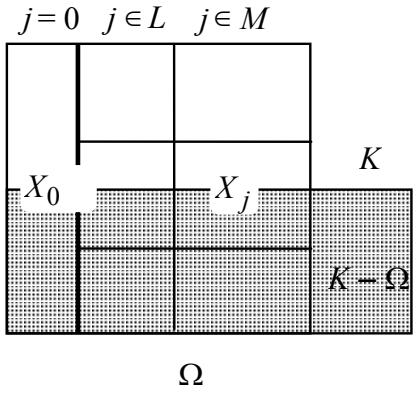


Figure 3.3. Overlap of the target with the category basis of a normal model

Let

$$J = \{j \in L \cup M : X_j \subseteq K\} \qquad J' = \{j \in L \cup M : X_j \cap K \neq \emptyset \text{ and } X_j - K \neq \emptyset\}.$$

Since the total of the response variable for the category $X_0 \cap K$ is definitely zero, the tightest lower bound for K is equal to

$$lower(K) = \sum_{j \in J \cap L} c_j + \mathbf{min}\ \{\sum_{j \in J \cap M} x_j : \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\} \qquad (3.4)$$

and, if K is contained in Ω, then the tightest upper bound for K is equal to

$$upper(K) = \sum_{j\in(J\cup J')\cap L} c_j + \mathbf{max}\ \{\sum_{j\in(J\cup J')\cap M} x_j\colon \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\}\ . \quad (3.5)$$

Remark 3.1 Since $c_j > 0$ for $j \in L$, then one has that

— $\sum_{j\in J\cap L} c_j \geq 0$ where the equality holds if and only if $J\cap L = \emptyset$, and

— $\sum_{j\in J\cap L} c_j \leq \sum_{j\in(J\cup J')\cap L} c_j$ where the equality holds if and only if $J'\cap L = \emptyset$.

The following is a characterization of evaluability.

Theorem 3.1 Let system (3.3) be the equation system of an information model in reduced form. The total for a category K is evaluable if and only if

(a) K is contained in Ω,

(b) $J' = \emptyset$, and

(c) the function $\sum_{j\in J\cap M} x_j$ is constant over the set of nonnegative solutions of the equation system $\mathbf{G}\,\mathbf{x}_M = \mathbf{w}$.

Proof. The "if"-part trivially follows from equations (3.4) and (3.5). On the other hand, if the total for K is evaluable, then by Remark 1.3 condition (a) must hold and from Remark 3.1 it follows that $J'\cap L$ must be empty. Suppose by contradiction that condition (b) does not hold. Then, $J'\cap M \neq \emptyset$ and

$$\mathbf{max}\ \{\sum_{j\in J\cap M} x_j\colon \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\} < \mathbf{max}\ \{\sum_{j\in(J\cup J')\cap M} x_j\colon \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\}$$

for, otherwise,

$$\mathbf{max}\ \{\sum_{j\in J'\cap M} x_j\colon \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\} = 0$$

and each $x_j, j \in J'\cap M$ would be a null variable (contradiction). So, condition (b) must hold. As to condition (c), suppose by contradiction that

$$\mathbf{min}\ \{\sum_{j\in J\cap M} x_j\colon \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\} < \mathbf{max}\ \{\sum_{j\in J\cap M} x_j\colon \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\}.$$

Then, the following contradiction would arise

$$lower(K) = \sum_{j\in J\cap L} c_j + \mathbf{min}\ \{\sum_{j\in J\cap M} x_j\colon \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\} <$$

$$< \sum_{j\in J\cap L} c_j + \mathbf{max}\ \{\sum_{j\in J\cap M} x_j\colon \mathbf{G}\,\mathbf{x}_M = \mathbf{w},\ \mathbf{x}_M \geq \mathbf{0}\} = upper(K). \quad \square$$

From a computational point of view, conditions (a) and (b) are easy to test and require $O(|\Omega|)$ and $O(1)$ time, respectively. As to condition (c), it can be tested by comparing

$$\mathbf{min}\ \{\sum_{j\in J\cap H} x_j\colon \mathbf{G}\,\mathbf{x}_H = \mathbf{w},\ \mathbf{x}_H \geq \mathbf{0}\}$$

and

$$\mathbf{max}\ \{\sum_{j\in J\cap H} x_j\colon \mathbf{G}\,\mathbf{x}_H = \mathbf{w},\ \mathbf{x}_H \geq \mathbf{0}\},$$

whose computation requires solving two linear-programming problems. However, as seen in Chapter 2 we can do better since we don't need to resort to linear-programming methods but can use the result of Theorem 2.1.

Example 3.3. Consider an instance of an SDB containing a relation of name `Personnel` with schema {NAME, GENDER, AGE, DEPT, SALARY}, where SALARY is assumed to be of nonnegative real type. Let {A, B, C, D, ...} be the domain of DEPT.
Consider fourteen sum-queries $\mathbf{q}_1, \ldots, \mathbf{q}_{14}$ with response variable SALARY whose targets (V_i) are as follows:

V_1	V_2	V_3	V_4
(M, young, D)	(M, middle, B)	(M, middle, D)	(F, young, A)

V_5	V_6	V_7	V_8
(F, young, D)	(F, middle, A)	(F, middle, B)	(F, middle, C)

V_9	V_{10}	V_{11}	V_{12}
(M, young, A)	(M, middle, A)	(F, young, A)	(M, young, A)
(M, young, B)	(M, middle, B)	(F, young, B)	(M, middle, A)
(M, young, C)	(M, middle, C)	(F, young, C)	(F, young, A)
(M, young, D)	(M, middle, D)	(F, young, D)	(F, middle, A)

V_{13}	V_{14}
(M, young, B)	(M, young, C)
(M, middle, B)	(M, middle, C)
(F, young, B)	(F, young, C)
(F, middle, B)	(F, middle, C)

Let us assume that the values of $\mathbf{q}_1, \ldots, \mathbf{q}_{14}$ are as follows:

$q_1 = 0$	$q_2 = 5$	$q_3 = 10$	$q_4 = 10$
$q_5 = 10$	$q_6 = 15$	$q_7 = 20$	$q_8 = 10$
$q_9 = 30$	$q_{10} = 25$	$q_{11} = 25$	$q_{12} = 30$
$q_{13} = 60$	$q_{14} = 15$		

Note that $q_1, \ldots, q_{14}$ can be viewed as being the entries in an incomplete two-dimensional table (see Figure 3.4), where $q_1, \ldots, q_8$ are the internal entries, $q_9, \ldots, q_{11}$ are the row totals and $q_{12}, \ldots, q_{14}$ are the column totals.

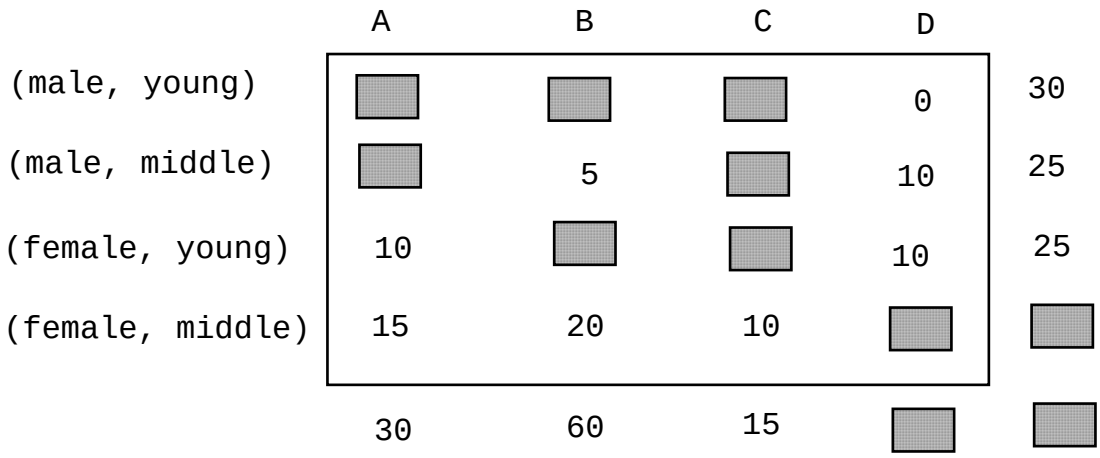


Figure 3.4. An incomplete two-dimensional table

The basis of $\{V_1, \ldots, V_{14}\}$ is formed by the following fifteen elementary categories:

$X_1 = \{(\mathtt{M}, \mathtt{young}, \mathtt{A})\}$ $X_2 = \{(\mathtt{M}, \mathtt{young}, \mathtt{B})\}$
$X_3 = \{(\mathtt{M}, \mathtt{young}, \mathtt{C})\}$ $X_4 = \{(\mathtt{M}, \mathtt{young}, \mathtt{D})\}$
$X_5 = \{(\mathtt{M}, \mathtt{middle}, \mathtt{A})\}$ $X_6 = \{(\mathtt{M}, \mathtt{middle}, \mathtt{B})\}$
$X_7 = \{(\mathtt{M}, \mathtt{middle}, \mathtt{C})\}$ $X_8 = \{(\mathtt{M}, \mathtt{middle}, \mathtt{D})\}$
$X_9 = \{(\mathtt{F}, \mathtt{young}, \mathtt{A})\}$ $X_{10} = \{(\mathtt{F}, \mathtt{young}, \mathtt{B})\}$
$X_{11} = \{(\mathtt{F}, \mathtt{young}, \mathtt{C})\}$ $X_{12} = \{(\mathtt{F}, \mathtt{young}, \mathtt{D})\}$
$X_{13} = \{(\mathtt{F}, \mathtt{middle}, \mathtt{A})\}$ $X_{14} = \{(\mathtt{F}, \mathtt{middle}, \mathtt{B})\}$
$X_{15} = \{(\mathtt{F}, \mathtt{middle}, \mathtt{C})\}$

and system (1.1) reads

$$\begin{cases} x_4 = 0, x_6 = 5, x_8 = 10, x_9 = 10, x_{12} = 10, x_{13} = 15, x_{14} = 20, x_{15} = 10 \\ x_1 + x_2 + x_3 + x_4 = 30 \\ x_5 + x_6 + x_7 + x_8 = 25 \\ x_9 + x_{10} + x_{11} + x_{12} = 25 \\ x_1 + x_5 + x_9 + x_{13} = 30 \\ x_2 + x_6 + x_{10} + x_{14} = 60 \\ x_3 + x_7 + x_{11} + x_{15} = 15 \end{cases} \qquad (3.6)$$

Using standard linear-programming methods, we find that each variable x_j is determined:

$x_1 = 0$ $x_2 = 30$ $x_3 = 0$ $x_4 = 0$
$x_5 = 5$ $x_6 = 5$ $x_7 = 5$ $x_8 = 10$
$x_9 = 10$ $x_{10} = 5$ $x_{11} = 0$ $x_{12} = 10$
$x_{13} = 15$ $x_{14} = 20$ $x_{15} = 10$

The set of these fifteen equalities is the reduced form of system (1.1), and its normal form is

$x_0 = 0$ $x_2 = 30$ $x_5 = 5$ $x_6 = 5$ $x_7 = 5$ $x_8 = 10$
$x_9 = 10$ $x_{10} = 5$ $x_{12} = 10$ $x_{13} = 15$ $x_{14} = 20$ $x_{15} = 10$

where x_0 stands for the total of SALARY for the category

X_0
(M, young, A)
(M, young, C)
(M, young, D)
(F, young, C)

It follows that the total for a category K is evaluable if and only if $K - X_0$ is the union of zero or more categories from $\{X_2, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{12}, X_{13}, X_{14}, X_{15}\}$. ■

3.6 Management of the normal form

In our Answering Procedure of Section 3.4, at Step 3, when the posterior model is constructed, it should be reduced in normal form because it becomes the next prior model if Step 5 is executed. This also allows to test the evaluability of each sensitive category (Step 4) in cubic time. In this section, we present a procedure for getting the posterior model in reduced form (Step 3). We start with the prior model, which is assumed to be in reduced form and to consist of system (3.3) with the interpretation of its variables given by the partition $\mathbf{X} = \{X_0, X_1, \ldots, X_m\}$ of Ω where the categories $X_1, \ldots, X_m$ are partitioned into the two groups $\{X_j: j \in L\}$ and $\{X_j: j \in M\}$ as shown in Figure 3.4. Let

$$J = \{j \in L \cup M: X_j \subseteq K\} \qquad J' = \{j \in L \cup M: X_j \cap K \neq \emptyset \text{ and } X_j - K \neq \emptyset\}$$

Let $p = |J'|$ and, if $p > 0$, let $J' = \{j_1, \ldots, j_p\}$. The procedure first constructs the semantic part of the posterior model using the partition $\boldsymbol{X}' = \{X'_0, X'_1, \ldots, X'_m, \ldots, X'_{m+p+1}\}$ of $\Omega' = \Omega \cup K$ defined as follows (see Figure 3.6):

If $p > 0$, then set $X'_{m+h} := X_{j_h} - K$ and $X'_{j_h} := K \cap X_{j_h}$ for $h = 1, \ldots, p$.
If $\Omega' \neq \Omega$, then set $C_{m+p+1} := K - \Omega$

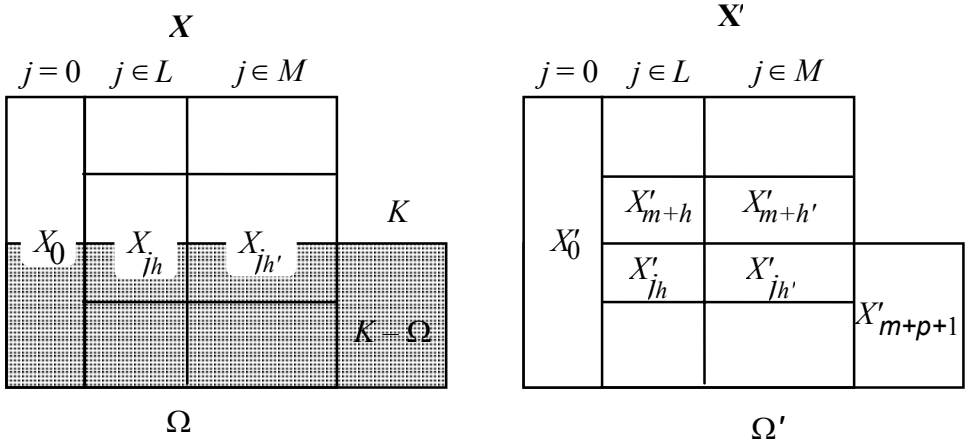


Figure 3.5. Updating the basis of a normal model

Without loss of generality, we assume that $J' \neq \emptyset$ and $\Omega' \neq \Omega$. For $h = 1, \ldots, p$, associate with the variable x_{j_h} the category X_{j_h} from $\boldsymbol{X}'$, and introduce the variable x_{m+h} and associate with x_{m+h} the category X_{m+h} from $\boldsymbol{X}'$. Next, introduce the variable x_{m+p+1} and associate with x_{m+p+1} the category

X_{m+p+1} from $\boldsymbol{X}'$. The equation system of the posterior model is then obtained from system (3.3) as follows. Re-write system (3.3) as

$$\begin{cases} x_0 = 0 \\ x_j = c_j & (j \in L) \\ \sum_{j \in M} g_{ij} x_j = w_i & (i = 1, \ldots, n) \end{cases}$$

For $h = 1, \ldots, p$, replace each occurrence of the variable x_{j_h} by the expression $x_{j_h} + x_{m+h}$. Finally, since the total for the category $X_0 \cap K$ is always zero, add the equation

$$\sum_{j \in J \cap L} c_j + \sum_{j \in J \cap M} x_j + \sum_{h=1,\ldots,p} x_{j_h} + x_{m+p+1} = q$$

which we re-write as

$$\sum_{j \in J \cap M} x_j + \sum_{h=1,\ldots,p} x_{j_h} + x_{m+p+1} = q'$$

where $q' = q - \sum_{j \in J \cap L} c_j$.

Thus, the equation system of the posterior model reads

$$\begin{cases} x_0 = 0 \\ x_j = c_j & (j \in L - J') \\ x_{j_h} + x_{m+h} = c_{j_h} & (j_h \in J' \cap L) \\ \sum_{j \in M - J'} g_{ij} x_j + \sum_{j_h \in J' \cap M} g_{ij_h} (x_{j_h} + x_{m+h}) = w_i & (i = 1, \ldots, n) \\ \sum_{j \in J \cap M} x_j + \sum_{h=1,\ldots,p} x_{j_h} + x_{m+p+1} = q' \end{cases} \qquad (3.7)$$

At this point, in order to get a normal form of the posterior model, we need a reduced form of system (3.7), that is, to find its determined variables and its redundant equations. Preliminarily, note that x_0 is a null variable of system (3.7) and that each $x_j, j \in L - J'$, is a nonnull determined variable of system (3.7). Moreover, for each $j_h \in J' \cap L$, the variables x_{j_h} and x_{m+h} appear only in the equation $x_{j_h} + x_{m+h} = c_{j_h}$ and are undetermined since the feasibility range for both is $[0, c_{j_h}]$. Let us consider the remaining variables and their equation system

$$\begin{cases} \sum_{j \in M - J'} g_{ij} x_j + \sum_{j_h \in J' \cap M} g_{ij_h} (x_{j_h} + x_{m+h}) = w_i & (i = 1, \ldots, n) \\ \sum_{j \in J \cap M} x_j + \sum_{h=1,\ldots,p} x_{j_h} + x_{m+p+1} = q' \end{cases} \qquad (3.8)$$

Finding a determined variable or a redundant equation in system (3.8) can be done using linear-programming methods. We can do better by exploiting Corollaries 2.3 and 2.4. Thus, if we first find the set of null variables of system (3.8) using a standard linear-programming algorithm, then we can decide in cubic time if a nonnull variable in equation system (3.8) is determined or if an equation in equation system (3.8) is redundant. Indeed, the search for null variables can be restricted to the variables corresponding to categories for which the summary table gives a zero total; analogously, the search for nonnull determined variables can be restricted to the variables corresponding to categories for which the summary table gives a nonzero total. Suppose that we have already found a reduced form of system (3.8), say

$$\begin{cases} x_j = 0 & (j \in Z') \\ x_j = c_j & (j \in L') \\ \mathbf{G}'\,\mathbf{x}' = \mathbf{w}' \end{cases} \qquad (3.9)$$

Finally, we are in a position to get a normal form of the posterior model. Its equation system is obtained by combining system (3.7) and system (3.9):

$$\begin{cases} x_0 = 0 \\ x_j = c_j & (j \in (L - J') \cup L') \\ x_{j_h} + x_{m+h} = c_{j_h} & (j_h \in J' \cap L) \\ \mathbf{G}'\ \ \mathbf{x}' = \mathbf{w}' \end{cases}$$

where the variable x_0 has now associated the category

$$(\cup_{j \in Z'} X_j) \cup X_0 \ .$$

Example 3.2 (continued). Consider again the posterior model constructed when the sum-query Q_5 is processed.

$$\begin{cases} x_1 + x_2 = 24 \\ x_2 + x_3 + x_5 = 18 \\ x_1 + x_3 + x_4 = 29 \\ x_4 + x_6 = 6.5 \\ x_5 + x_6 = 1.5 \end{cases}$$

X_1 = {(M, young)} X_2 = {(M, middle)} X_3 = {(M, old)}
X_4 = { (F, young)} X_5 = {(F, middle)} X_6 = {(F, old)}

Since X_6 is the only category for which the summary table on SALARY gives a zero total, the only candidate for a null variable is the variable x_6. Using the simplex method, we find that the maximum value of x_6 is 1.5 so that x_6 is not a null variable. At this point, the nonnull determined variables are found by exploiting Corollary 2.3. The variables that are candidates for nonnull determined variables are x_1, …, x_5, and one finds that the only determined variables are x_1 and x_2 with values 15 and 9, respectively. For example, the characteristic vector

[1 0 0 0 0 0]

of the singleton {1} can be written as a linear combination of the rows of the coefficient matrix of the equation system above

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

with coefficients equal to $\frac{1}{2}$ for the odd rows and $-\frac{1}{2}$ for the even rows. So, x_1 is determined with value $\frac{1}{2}(24 - 18 + 29 - 6.5 + 1.5) = 15$.

After deleting the two determined variables and adding the two equations $x_1 = 15$ and $x_2 = 9$, we obtain the following equation system

$$\begin{cases} x_1 = 15, x_2 = 9 \\ x_3 + x_5 = 9 \\ x_3 + x_4 = 14 \\ x_4 + x_6 = 6.5 \\ x_5 + x_6 = 1.5 \end{cases}$$

At this point, the redundant equations are found by exploiting Corollary 2.4. If the equation $x_3 + x_5 = 9$ is first examined, then it comes out to be redundant and none of the remaining equations is redundant. Explicitly, for the left-hand side and the right-hand side of the equation $x_3 + x_5 = 9$ one has

$$\begin{aligned} x_3 + x_5 &= (x_3 + x_4) - (x_4 + x_6) + (x_5 + x_6) \\ 9 &= (14) - (6.5) + (1.5) \end{aligned}$$

Thus, we have obtained the following reduced form of the equation system of the posterior model:

$$\begin{cases} x_1 = 15, x_2 = 9 \\ x_3 + x_4 = 14 \\ x_4 + x_6 = 6.5 \\ x_5 + x_6 = 1.5 \end{cases}$$

Since there are no null variables, this is also the equation system of the normal form of the posterior model. ■

Chapter 4

Finding The Invariant Edges In An Edge Weighted Graph

4.1. Introduction

We saw in Section 2.2 and in the Answering Procedure given in Section 3.5, that testing the evaluability of a query needs to resort to the linear-programming methods only to solve the following two problems:

(P1) given the equation system of an arbitrary information model, find the set of its null variables;

(P2) given the equation system of an information model in normal form, find the feasibility range for a given sum of variables.

However, some special instances of those problems can be very efficiently solved. For example, if the information model represents a "decomposable" set of marginals of an unknown multidimensional table, then the feasibility range for a single variable (corresponding to a cell entry) can be computed using closed formulas [20]. Also we see that the problem

(P3) given the dictionary matrix H and a nonempty subset A of column of H find a nonempty algebraic subset of A

is an *NP*-Complete problem. We will see in the next three Chapters that the three problems can be efficiently solved for a *graphical* information model, that is, for an information model where the dictionary matrix **H** can be viewed as the incidence matrix of a graph G. Note that this is the case whenever the information model represents the contents of internal and marginal cells of a (possibly incomplete) two-dimensional table (see Section 4.6 and Example 4.3).

Also it is possible to chose a set $\mathbf{V}=\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n\}$ of materialised views such that the dictionary matrix of $\mathbf{V}$ is the incidence matrix of a graph G.
The choice of such $\mathbf{V}$ can be possible by limiting the amount of overlapping between the targets of the views in $\mathbf{V}$. Let S be the set of categorical variables of $\mathbf{V}$. Let V_j be the extension of the target of the query $\mathbf{q}_j$ to S. Let $\boldsymbol{X}=\{X_1, \ldots, X_m\}$ be the classification system of $\mathbf{V}$. If P is a subset of $\{1, \ldots, n\}$ then we say that the set $\{V_1, \ldots, V_n\}$ has overlapping of *order r* if for any P such that $|P|>r$ we have $\cap_{i\in P} V_i=\varnothing$. Suppose the set $\{V_1, \ldots, V_{n-1}\}$ has an overlapping of order r and suppose that we want to add query (or view) $\mathbf{q}_n$ to $\mathbf{V}$. Then to enforce that $\{V_1, \ldots, V_n\}$ has overlapping of order r, simply check that for all subsets P of $\{1, \ldots, n\}$ with $|P|>r$, we have $\cap_{i\in P} V_i=\varnothing$. If this not the case we discard $\mathbf{q}_n$. Now if we take $r=2$, then the dictionary matrix **H** of $\mathbf{V}$ is the incidence matrix of a graph where loops are allowed.
Finally suppose the set $\{V_1,\ldots,V_{n-1}\}$ has an overlapping of order two. Let **H** be the dictionary matrix of $\mathbf{V}$. Then **H** is the node-edge incidence matrix of a graph G where loops are allowed. Given a new query $\mathbf{q}_n$ to ensure that $\{V_1,\ldots,V_n\}$ has an overlapping of order two simply check that V_n has a non empty intersection only with element X_j where j is a loop of G. This can be done by checking that $V_n\cap X_j$ is empty for all j where j is not a loop of G.

Example. A given database consists in a relation R that reports the sales of products and country. The schema of R in this example is { `sales`, `product`, `country` } where *sales* is the response variable of nonnegative real type. Suppose we have the following materialised views

```
q1 :  select sum (sales)
      from R
      where product in ("wine" ,"beer")

q2 :  select sum (sales)
      from R
      where product in ("wine" ,"liquor")

q3 :  select sum (sales)
      from R
      where product ="wine" and country ="Italy"

q4 :  select sum (sales)
      from R
      where product in ("beer" ,"liquor")
```

and let $\{V_1, V_2, V_3, V_4\}$ be the target of each query. The set $\{V_1, V_2, V_3\}$ has an overlapping of order three since $V_1 \cap V_2 \cap V_3 \neq \varnothing$. On the other hand $\{V_1, V_2, V_4\}$ has an overlapping of order two and its dictionary matrix is the incidence matrix of a graph. ■

As of problem (P1) if G is bipartite, then Gusfield [27] proved that, given a nonnegative solution of system (1.1), the set of null variables can be found in strongly linear time [27]. In this Chapter we will prove that the same holds for an arbitrary graph. We also see that the problem of finding all the determined variables of system (1.1) can be solved in time linear in the size of system (1.1). Finally in section 4.7, we also solve problem P1 in the case that the variables of system (1,1) take their values from a commutative (abelian) group.
As for the problem P2, we will prove in the Chapter 5 that can be efficiently solved if the information model is graphical. Finally problem P3 is solved when **H** is the incidence matrix of a graph in Chapter 6.

Let $G = (V, E)$ be a graph without isolated vertices (where self-loops and parallel edges may exist), and let $\boldsymbol{w} = (w(e))_{e \in E}$ be a vector of nonnegative reals. The pair $\Gamma = (G, \boldsymbol{w})$ is referred to as an *edge-weighted graph* (an EWG, for short). Let **A** be the incidence matrix of G and let $\boldsymbol{b} = (b(v))_{v \in V}$ be the vector of nonnegative reals such that, for each vertex v of G, $b(v)$ equals the sum of the weights of the edges incident to v. Consider the following system of linear equations

$$\mathbf{A}\,\boldsymbol{x} = \boldsymbol{b} \qquad (4.1)$$

For every edge e of G, let $L[x(e)]$ and $U[x(e)]$ denote the minimum and the maximum of the variable $x(e)$ over the nonnegative solutions of equation system (4.1), respectively. An edge e of G is an *invariant edge* of Γ if $L[x(e)] = U[x(e)]$. Thus, an edge e of G is an invariant edge of Γ if and only if $x(e) = w(e)$ for every nonnegative solution of equation system (4.1). The following two examples show two EWGs which have all and no invariant edges, respectively.

Example 4.1. Consider the EWG $\Gamma = (G, \boldsymbol{w})$ shown in Figure 4.1 where α, β and γ are any positive reals. By making use of standard algebraic methods, one finds there is no nonnegative solution of equation system (4.1) other than $\boldsymbol{w}$. Therefore, each edge of G is an invariant edge of Γ.

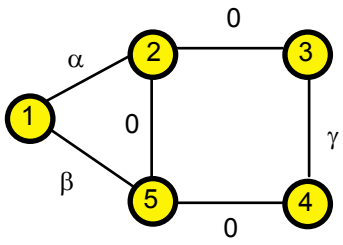


Figure 4.1

■

Example 4.2. Consider the EWG $\Gamma = (G, \boldsymbol{w})$ shown in Figure 4.2.

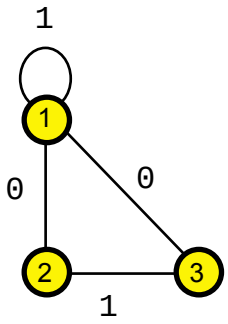


Figure 4.2

The general expression of a nonnegative solution of equation system (4.1) is

$$x(1, 1) = 1 - 2\lambda \qquad x(1, 2) = x(1, 3) = \lambda \qquad x(2, 3) = 1 - \lambda$$

where the parameter λ ranges from 0 to 1/2. Therefore, one has

$$L[x(1, 1)] = 0 \qquad U[x(1, 1)] = 1$$
$$L[x(1, 2)] = L[x(1, 3)] = 0 \qquad U[x(1, 2)] = U[x(1, 3)] = 1/2$$
$$L[x(2, 3)] = 1/2 \qquad U[x(2, 3)] = 1$$

and, hence, no edge of G is an invariant edge of Γ. ■

The problem addressed in this Chapter consists in finding the set of invariant edges of an arbitrary EWG. The following obvious fact allows us to limit our considerations to EWGs with underlying simple graphs (i.e., graphs without parallel edges).

Fact 4.1. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG where $G = (V, E)$ is a nonsimple graph. Let S be a set of two or more parallel edges and let e_0 be an arbitrarily chosen element of S. Let $G' = (V, E')$ be the graph with edge set $E' = (E - S) \cup \{e_0\}$. Consider the EWG $\Gamma' = (G', \boldsymbol{w}')$ where $\boldsymbol{w}'$ is defined as follows

$$w'(e) = \begin{cases} w(e) & e \notin S \\ \sum_{e' \in S} w(e') & e = e_0 \end{cases}$$

Then, an edge not in S is an invariant edge of Γ if and only if it is an invariant edge of Γ', and an edge in S is an invariant edge of Γ if and only if $w'(e_0) = 0$ and e_0 is an invariant edge of Γ'.

Here we present a linear-time algorithm which finds the set of invariant edges of an arbitrary EWG.

4.2. Background

Let $G = (V, E)$ be a simple graph with vertex-edge incidence matrix $\mathbf{A}$. For any vector $\boldsymbol{x} = (x(e))_{e\in E}$, the *support* of $\boldsymbol{x}$ is the set $S = \{e\in E: x(e) \neq 0\}$, and the *signed support* of $\boldsymbol{x}$ is the ordered set pair (S^+, S^-), where $S^+ = \{e\in E: x(e) > 0\}$ and $S^- = \{e\in E: x(e) < 0\}$; moreover, the set $E–S$ is called the *zero set* of $\boldsymbol{x}$. The non-zero solutions of the homogeneous equation system $\mathbf{A}\,\boldsymbol{y} = \mathbf{0}$ are referred to as *circulations* in G and the linear space of the solutions of the homogeneous equation system $\mathbf{A}\,\boldsymbol{y} = \mathbf{0}$ is referred to as the *circulation space*. Thus, a nonempty subset S of E corresponds to a set of columns of $\mathbf{A}$ that are linearly dependent (over the field of reals) if and only if S contains the support of a circulation in G. A *minimal circulation* in G is a circulation in G with inclusion-minimal support. The following is a well-known result of linear algebra.

Proposition 4.1 (e.g., see page 107 in [6]). Let S and (S^+, S^-) be the support and the signed support of a circulation in G, respectively. For each edge e in S, there is a minimal circulation in G with support C and signed support (C^+, C^-) such that e is in C, $C^+ \subseteq S^+$ and $C^- \subseteq S^-$.

The set of supports of minimal circulations in G can be viewed as the family of *circuits* of a matroid [60], we denote by $\mathcal{M}(G)$, whose rank (i.e., the rank of $\mathbf{A}$) is given by $|V|–q$ where q is the number of connected components of G that are bipartite (in that they contain no odd cycles) (see Theorem 1, page 421 in [19], or [57]). Explicitly, a subset of E is a circuit of $\mathcal{M}(G)$ if and only if it is the edge set of either an even simple cycle or a pair of two edge-disjoint odd simple cycles that either have exactly one vertex in common or are vertex-disjoint and are connected by a simple path (see Figure 4.3) [15].

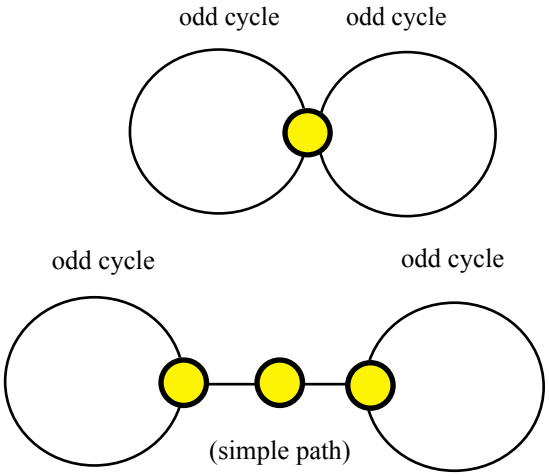


Figure 4.3

Let Z be a (proper or improper) subset of E. We say that a circuit of $\mathcal{M}(G)$ is *Z-traversable* if it is the support of a (minimal) circulation whose signed support (C^+, C^-) is such that $Z\cap C^- = \emptyset$.

Consider now the vectors that are linear combinations of rows of $\mathbf{A}$. The inclusion-minimal supports of these vectors are the *co-circuits* of $\mathcal{M}(G)$; that is, they are minimal edge sets whose removal decreases the rank of $\mathcal{M}(G)$ [60]. Moreover, an edge e of G is a *co-loop* of $\mathcal{M}(G)$ if the singleton $\{e\}$ is a co-circuit of $\mathcal{M}(G)$. In other words, an edge e of G is a co-loop of $\mathcal{M}(G)$ if and

only if the incidence vector of $\{e\}$ is a linear combination of rows of **A** or, equivalently, if and only if e is not in any circuit of $\mathcal{M}(G)$ [60].

4.3. Invariant Edges

In this section, we state a few characteristic properties of invariant edges of an arbitrary EWG which will be used later on. We need some preliminary definitions and results.

Let $\Gamma = (G, \boldsymbol{w})$ be an EWG with $G = (V, E)$ and let Z be the zero set of $\boldsymbol{w}$. A circulation in G with signed support (S^+, S^-) is said to be *legal* in Γ if $Z \cap S^- = \emptyset$. Accordingly, a circuit of $\mathcal{M}(G)$ is Z-traversable if and only if it is the support of a (minimal) circulation in G which is legal in Γ. It should be noted that, if the weights of the edges of G are all positive, then Z is empty so that each circulation in G is legal in Γ.

Theorem 4.1. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG. An edge of G is not an invariant edge of Γ if and only if it belongs to the support of a circulation in G which is legal in Γ.

Proof. ("only if") Let e be an edge of G that is not an invariant edge of Γ. Then, there exists a nonnegative solution $\boldsymbol{x}$ of eq. (1) with $x(e) \neq w(e)$. The vector $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{w}$ is then a circulation in G. Let S and (S^+, S^-) be the support and the signed support of $\boldsymbol{y}$, respectively, and let Z be the zero set of $\boldsymbol{w}$. Then e is in S and, since $y(e') = x(e') \geq 0$ for each e' in Z, one has $Z \cap S^- = \emptyset$; that is, e belongs to the support of a circulation which is legal in Γ.

("if") Let $\boldsymbol{y}$ be a legal circulation with support S and signed support (S^+, S^-), and let e be in S. Consider the solution $\boldsymbol{x} = \boldsymbol{w} + \boldsymbol{y}$ of equation system (1). If $\boldsymbol{x}$ is nonnegative everywhere, then the statement follows from the fact that e in S which implies $x(e) \neq w(e)$. Otherwise, let

$$S' = \{e' : x(e') < 0\} \qquad \text{and} \qquad \lambda = \min \{-w(e')/y(e') : e' \in S'\}.$$

Since S' is a subset of S^- and $\boldsymbol{y}$ is a legal circulation, λ is positive. Then the vector $\boldsymbol{y}' = \lambda\, \boldsymbol{y}$ is a circulation in G having the same support and the same signed support as $\boldsymbol{y}$. Consider the solution $\boldsymbol{x}' = \boldsymbol{w} + \boldsymbol{y}'$ of equation system (1). It is easily seen that $\boldsymbol{x}'$ is nonnegative everywhere since, for each e' not in S', then one has trivially $x'(e') \geq 0$ and, for each e' in S', one has

$$x'(e') = w(e') + \lambda\, y(e') = -y(e')\,(-w(e')/y(e') - \lambda) \geq 0.$$

Finally, since e is in the support S of y, one has

$$x'(e) = w(e) + \lambda\, y(e) \neq w(e),$$

which proves the statement. □

Theorem 4.2. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG and let Z be the zero set of $\boldsymbol{w}$. An edge of G is not an invariant edge of Γ if and only if it belongs to some Z-traversable circuit of $\mathcal{M}(G)$.

Proof. ("if") It follows from the "if" part of Theorem 4.1.

("only if") If e is not an invariant edge of Γ, then by the "only-if" part of Theorem 4.1 there is a circulation in G with support S and signed support (S^+, S^-) such that e is in S and $Z \cap S^- = \emptyset$. But, by Proposition 4.1, there is a minimal circulation in G such that its support contains e and its signed support (C^+, C^-) is such that $C^- \subseteq S^-$. Therefore, one has $Z \cap C^- \subseteq Z \cap S^- = \emptyset$ which proves the statement. □

Example 4.1 (continued). The zero set of $\boldsymbol{w}$ is $Z = \{(2, 3), (2, 5), (4, 5)\}$. The minimal circulations in G are summarized in Figure 4.4 by taking λ to be any nonzero real. So, $\mathcal{M}(G)$ contains one circuit which is not Z-traversable. By Theorem 4.2, each edge of G is an invariant edge of Γ.

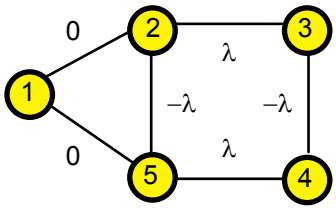


Figure 4.4

■

Example 4.2 (continued). The zero set of $\boldsymbol{w}$ is $Z = \{(1, 2), (1, 3)\}$. The minimal circulations in G are summarized in Figure 4.5 by taking λ to be any nonzero real. So, $\mathcal{M}(G)$ contains one circuit which is Z-traversable. By Theorem 4.2, no edge of G is an invariant edge of Γ.

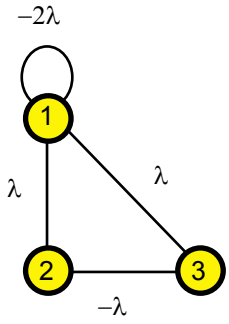


Figure 4.5

■

Note that, if the zero set Z of $\boldsymbol{w}$ is empty then, by Theorem 4.2, an edge of G is an invariant edge of Γ if and only if it is not in any circuit of $\mathcal{M}(G)$, that is, if and only if it is a co-loop of $\mathcal{M}(G)$. We now prove that the same holds in a more general case. In what follows, by the *kernel* of Γ [45] we mean the intersection of Z with the set of invariant edges of Γ.

Lemma 4.1. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG whose kernel is empty. An edge of G is an invariant edge of Γ if and only if it is a co-loop of $\mathcal{M}(G)$.

Proof. ("if") If an edge of G is a co-loop of $\mathcal{M}(G)$, then it is an invariant edge of Γ by the "only-if" part of Theorem 4.2.

("only if") Let e be an invariant edge of Γ. Suppose by contradiction that e is in some circuit of $\mathcal{M}(G)$. Then, as is shown below, e should belong to the support of a legal circulation in Γ which contradicts Theorem 4.1. To show that, suppose that e is in the circuit C_0 of $\mathcal{M}(G)$. By the "only-if" part of Theorem 4.2, C_0 cannot be Z-traversable, where Z is the zero set of $\boldsymbol{w}$. Thus, if $\boldsymbol{c}_0$ is any minimal circulation in G with support C_0 and signed support (C_0^+, C_0^-), one has $Z \cap C_0^- \neq \emptyset$. Let $Z \cap C_0^- = \{e_1, \ldots, e_p\}$. Since the kernel of Γ is empty, no edge in Z is an invariant edge of Γ. Then, by the "only-if" part of Theorem 4.1, for each e_i, $1 \leq i \leq p$, there is a legal circulation $\boldsymbol{y}_i$ of Γ such that, if S_i is the support of $\boldsymbol{y}_i$, e_i is in S_i; moreover, if (S_i^+, S_i^-) is the signed support of $\boldsymbol{y}_i$, then $y_i(e_i) > 0$ since $Z \cap S_i^- = \emptyset$. Consider now the circulation

$$\boldsymbol{c}_i = [-c_0(e_i)/y_i(e_i)]\, \boldsymbol{y}_i$$

Since $c_0(e_i) < 0$ (recall that $e_i \in C_0^-$) and $y_i(e_i) > 0$, $\boldsymbol{c}_i$ has the same support and the signed support as $\boldsymbol{y}_i$, and hence is legal in Γ. Let

$$\boldsymbol{y} = \boldsymbol{c}_0 + \sum_{i=1,\ldots,p} \boldsymbol{c}_i \ .$$

Since the circulation space of $\mathbf{A}$ is a linear space, $\boldsymbol{y}$ is still a circulation in G. Let S and (S^+, S^-) be the support and signed support of $\boldsymbol{y}$, respectively. Finally, we now prove that (*i*) e is in S, and (*ii*) the circulation $\boldsymbol{y}$ is legal in Γ.

Proof of (*i*). Since e is an invariant edge of Γ, by the "if" part of Theorem 4.2, e is in the support of none of the legal circulations $\boldsymbol{c}_i$ so that $c_i(e) = 0$ for each i, $1 \le i \le p$. Therefore, $y(e) = c_0(e)$ and, since e is in C_0, one has that e is also in S.

Proof of (*ii*). In order to prove that the intersection of Z with S^- is empty, we separately examine the edges $e_1, \ldots, e_p$ in $Z \cap C_0^-$ and the edges in $Z - C_0^-$. For each i and j, $1 \le i, j \le p$, $c_j(e_i) \ge 0$ since $Z \cap S_j^- = \emptyset$. Moreover, for each i, $1 \le i \le p$, one has

$$c_i(e_i) = -c_0(e_i)$$

and hence

$$y(e_i) = c_0(e_i) + c_i(e_i) + \sum_{j \ne i} c_j(e_i) = \sum_{j \ne i} c_j(e_i) \ge 0.$$

Therefore, each e_i is not in S^-.

We now consider the edges in $Z - C_0^-$. If e' is such an edge, then $c_0(e') \ge 0$. Moreover, since $Z \cap S_i^- = \emptyset$, $1 \le i \le p$, one has $c_i(e') \ge 0$. Therefore

$$y(e') = c_0(e') + \sum_{i=1,\ldots,p} c_i(e') \ge 0,$$

and, hence, e' is not in S^-.

After proving (*i*) and (*ii*), by the "if" part of Theorem 4.1 one has that e is not an invariant edge of Γ (contradiction). □

As a consequence of Lemma 4.1, we obtain the following characterization of invariant edges of an EWG.

Theorem 4.3. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG with kernel K. The set of invariant edges of Γ is the union of K with the set of co-loops of $\boldsymbol{M}(G-K)$.

Proof. Let $\Gamma' = (G', \boldsymbol{w}')$ where $G' = G-K$, and let $\boldsymbol{w}'$ be the restriction of $\boldsymbol{w}$ to the edge set of G'. It is clear that an edge of G is an invariant edge of Γ if and only if it either is in K or it is an invariant edge of Γ'. On the other hand, the kernel of Γ' is empty so that, by Lemma 4.1, the invariant edges of Γ' are exactly the co-loops of $\boldsymbol{M}(G')$. □

By Theorem 4.3, the set of invariant edges of $\Gamma = (G, \boldsymbol{w})$ can be found by first determining the kernel K of Γ and, then, the set of co-loops of $\boldsymbol{M}(G-K)$. We shall solve the problem of the kernel of an EWG in Section 4.5 and, in the next section, we shall give a linear algorithm for finding the set of co-loops of the matroid on a graph.

4.4. Finding The Co-Loop Set

Let $G = (V, E)$ be a simple graph. Bearing in mind that a subset of E is a co-circuit of $\mathcal{M}(G)$ if and only if it is a minimal edge set whose removal decreases the rank of $\mathcal{M}(G)$, one easily obtains the following

Proposition 4.2 [15]. An edge of G is a co-loop of $\mathcal{M}(G)$ if and only if its removal creates one more bipartite connected component.

Let e be a co-loop of $\mathcal{M}(G)$. The graph $G–e$ has or has not one more connected component than G. By Proposition 4.2, in the former case e must be a bridge, we call an *algebraic bridge* of G, and in the latter case, as is shown below, e is an *odd edge*, by which we mean that e is common to all odd cycles of G.

Lemma 4.2. An edge of a simple graph G is a co-loop of $\mathcal{M}(G)$ if and only if it is either an algebraic bridge or an odd edge.

Proof. The statement is trivial if the graph is bipartite since, by Proposition 4.2, each co-loop of $\mathcal{M}(G)$ is a bridge and *vice versa*. Consider now a graph G which is not bipartite. Without loss of generality, we assume G is connected. It is sufficient to prove that a co-loop e of $\mathcal{M}(G)$ is not a bridge if and only if it is an odd edge. If e is not a bridge, then, by Proposition 4.2, $G–e$ is bipartite and connected and, hence, every odd cycle of G must contain e; that is, e is an odd edge of G. On the other hand, if e is an odd edge of G, then $G–e$ is connected and contains no odd cycles so that, by Proposition 4.2, e is a co-loop of $\mathcal{M}(G)$. □

Example 4.1 (continued). The co-loops of $\mathcal{M}(G)$ are the two edges missing from the even simple cycle supporting the minimal circulations shown in Figure 4.4. Both of them are odd edges. ■

Example 4.2 (continued). $\mathcal{M}(G)$ has no co-loops (see Figure 4.5). ■

Let $G = (V, E)$ be a simple graph which without loss of generality we assume to be connected. We first show that the problem of finding the set of co-loops of $\mathcal{M}(G)$ is polynomial; next, we shall give a linear algorithm based on Lemma 4.2.

In [39, 46, 49] an $O(|E|)$ algorithm is given to decide whether the incidence vector of a given subset of E is orthogonal to the space of circulations in G. By applying that algorithm to each singleton, one can determine the set of co-loops of $\mathcal{M}(G)$ in $O(|E|^2)$ time. In the next two subsections, we give two linear algorithms for finding the algebraic bridges and the odd edges of G; so, by Lemma 4.2, determining the whole set of co-loops of $\mathcal{M}(G)$ requires $O(|E|)$ time.

4.4.1 Algebraic bridges

Let $G = (V, E)$ be a connected simple graph, and let B be the set of bridges of G. Consider the tree $T = (N, A)$ whose nodes represent the connected components of $G–B$ and whose arcs represent the bridges of G. A node n of T is marked if the corresponding connected component of $G–B$ is not bipartite. If no node of T is marked, then G is bipartite and the bridges of G are all and the only algebraic bridges. Otherwise, there is at least one marked node of T; then, arbitrarily choose a marked node r of T and let T_r be the directed tree obtained by rooting T at r. For each node n of T_r, $n \neq r$, let $par(n)$ be the parent of n in T_r. Of course, a bridge of G is algebraic if and only if the (directed) arc $<par(n), n>$ of T_r is such that the subtree of T_r rooted at n contains no marked nodes. Thus, in order to get the algebraic bridges of G, it is sufficient to perform a postorder traversal of T_r[1]: when node n is examined, $n \neq r$, if n is marked then the edge of G corresponding to the arc $<par(n), n>$ is removed from B and the vertex $par(n)$ is marked if it was unmarked. So, the ultimate

value of B is exactly the set of algebraic edges of G. Now, since the construction of T and B and the postorder traversal of T_r require $O(|E|)$ time, we have

Theorem 4.4. The set of algebraic bridges of a connected simple graph can be found in time linear in the number of its edges.

4.4.2 Odd edges

Let $G = (V, E)$ be a connected simple graph. Trivially, if G is bipartite, then G contains no odd edges. In the case that G is not bipartite, we shall show that the set of odd edges of G can be found in $O(|E|)$. To achieve this, we need the following technical lemmas, the first two of which refer to general properties of the symmetric difference ($\oplus$) of cycles.

Lemma 4.3 (see, e.g., [2]). The symmetric difference of two distinct nondisjoint cycles is a set of edge-disjoint cycles.

Lemma 4.4. If the symmetric difference of two or more cycles contains an odd number of edges, then the number of such cycles having odd lengths is odd.

Proof. It easily follows from the fact that, for every two sets C and C', $|C \oplus C'|$ is odd if and only if $|C|$ and $|C'|$ have different parities. □

Let T be the edge set of a spanning tree of G. For each back-edge e (i.e., e not in T), the set $T \cup \{e\}$ contains exactly one simple cycle; such simple cycles, one for each back-edge, are called the *fundamental cycles* of G with respect to T [54].

Lemma 4.5 (see, e.g., page 251 in [2]). Let T be the edge set of a spanning tree of a simple graph G. Every cycle of G can be expressed as symmetric difference of fundamental cycles of G with respect to T.

Lemma 4.6. Let G be a nonbipartite connected simple graph, and T a spanning tree of G. An edge of G is an odd edge of G if and only if it is in all odd fundamental cycles with respect to T and in no even fundamental cycle with respect to T.

Proof. ("if") Let e be an edge of G that is in all odd fundamental cycles with respect to T and in no even fundamental cycle with respect to T. Let C be any odd cycle. By Lemma 4.5, C can be expressed as symmetric difference of fundamental cycles with respect to T and, by Lemma 4.4, the number of odd fundamental cycles in its expression is odd so that, since e is in all of them and in no even fundamental cycle with respect to T, e belongs to C.

("only if") Let e be an odd edge of G. Of course e is in all odd fundamental cycles with respect to T. Suppose by contradiction there is an even fundamental cycle C' with respect to T that contains e. Let C be an odd cycle containing e. By Lemma 4.3, $C \oplus C'$ contains an odd cycle, say C'', because the lengths of C and C' have different parities. So, since e is in both C and C', e is not in C'' which contradicts the hypothesis that e is in all odd cycles of G. □

From a computational point of view, the fundamental cycles of G with respect to a given spanning tree can be constructed using an $O(|V|^3)$ algorithm (see, e.g., Algorithm 8.10 in [54]). So, by Lemma 4.6 one can resort to that algorithm to find the set of odd edges of G in $O(|V|^3)$ time. However, we shall use Lemma 4.6 to work out an algorithm which runs in $O(|E|)$ time. It consists of two phases.

PHASE I. Arbitrarily choose a vertex r of G and perform a traversal of G with the depth-first search (DFS) technique to produce

— the edge set T of a directed spanning tree of G,

— the set B of back-edges that create odd fundamental cycles of G with respect to T,

— a vertex table which, for each vertex v, reports the following information items:

- the DFS number of v, denoted by $n(v)$;
- a label, denoted by $col(v)$, which is set to 'white' or 'black' depending on whether the length of the path from r to v in the spanning tree is even or odd;
- if $v \neq r$, the parent of v, denoted by $par(v)$;
- if $v \neq r$, the tree-edge $<par(v), v>$, denoted by $arc(v)$.

PHASE II. First of all, join a back-edge to *Odd* if it is the unique element of B. Next, in order to decide if a tree-edge e can be joined to *Odd*, compute

— the number of the even fundamental cycles that contain e, denoted by NEC[e], and
— the number of the odd fundamental cycles that contain e, denoted by NOC[e]

as follows. For each vertex u, let $N(u)$ be the set of neighbours of u in G and let $C(u)$ be the set of children of u in T. Then, set (see Figure 4.6)

$$\text{NEC}[arc(u)] := |P_{even}(u)| + \sum_{v \in C(u)} \text{NEC}[arc(v)] - |S_{even}(u)|. \qquad (4.2)$$

where

$$P_{even}(u) = \{v \in N(u): par(u) \neq v \text{ and } col(v) \neq col(u) \text{ and } n(v) < n(u)\}$$

$$S_{even}(u) = \{v \in N(u): par(v) \neq u \text{ and } col(v) \neq col(u) \text{ and } n(v) > n(u)\},$$

and

$$\text{NOC}[arc(u)] := |P_{odd}(u)| + \sum_{v \in C(u)} \text{NOC}[arc(v)] - |S_{odd}(u)|. \qquad (4.3)$$

where

$$P_{odd}(u) = \{v \in N(u): par(u) \neq v \text{ and } col(v) = col(u) \text{ and } n(v) < n(u)\}$$

$$S_{odd}(u) = \{v \in N(u): par(v) \neq u \text{ and } col(v) = col(u) \text{ and } n(v) > n(u)\}.$$

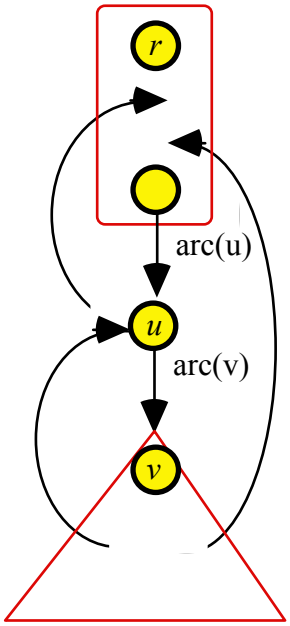


Figure 4.6

After calculating the quantities NEC[*e*] and NOC[*e*] for each edge *e* in *T*, determine the set of odd edges, denoted by *Odd*, as follows (see Lemma 4.6): for each edge *e* in *T*, join *e* to *Odd* if NEC[*e*] = 0 and NOC[*e*] = |*B*|.

The following algorithm details the steps of Phase II.

ALGORITHM 4.1

Input: A nonbipartite, connected simple graph $G = (V, E)$, a vertex r of G, T, B and the vertex table of G.

Output: The set *Odd* of odd edges of G.

(1) Set $Odd := \emptyset$. Set $k := |B|$. If $k = 1$, then $Odd := Odd \cup B$.
For each edge e in T, set NEC[e] := NOC[e] := 0.

(2) For each child u of r in T, TRAVERSE (G, u).
(3) For each edge e in T, if NEC[e] = 0 and NOC[e] = k then add e to *Odd*.

Procedure TRAVERSE (G, u)
For each neighbour v of u, do:

begin
if v is a child of u then do:

begin
TRAVERSE (G, v);
NEC[*arc*(u)] := NEC[*arc*(u)] + NEC[*arc*(v)];
NOC[*arc*(u)] := NOC[*arc*(u)] + NOC[*arc*(v)]
end;

otherwise, if v is not the parent of u then do:

Case 1. if $n(v) > n(u)$ and $col(v) \neq col(u)$ then set NEC[$arc(u)$] := NEC[$arc(u)$] – 1;

Case 2. if $n(v) > n(u)$ and $col(v) = col(u)$ then set NOC[$arc(u)$] := NOC[$arc(u)$] – 1;

Case 3. if $n(v) < n(u)$ and $col(v) \neq col(u)$ then set NEC[$arc(u)$] := NEC[$arc(u)$] + 1;

Case 4. if $n(v) < n(u)$ and $col(v) = col(u)$ then set NOC[$arc(u)$] := NOC[$arc(u)$] + 1

end

Theorem 4.5. Let G be a nonbipartite, connected simple graph. The value of *Odd* computed by Algorithm 4.1 with input G and vertex r is exactly the set of odd edges of G.

Proof. It is sufficient to prove that the quantities NEC[e] and NOC[e], for each tree-edge e, equal the number of even fundamental cycles containing e and the number of odd fundamental cycles containing e, respectively. The statement is proven by structural induction.

BASIS. Assume that u is a leaf of T. Then, u has no children so that, if v is a neighbour of u then v must be an ancestor of u. If $v = u$ then the self-loop (u, u) contributes to neither NEC[$arc(u)$] nor NOC[$arc(u)$]. If v is a proper ancestor of the parent of u, then $n(v) < n(u)$ and the back-edge (u, v) correctly adds 1 to either NEC[$arc(u)$] or NOC[$arc(u)$].

INDUCTIVE STEP. Let u be not a leaf of T and assume that statement holds for each one of the children of u. Thus, if v is a child of u, then values of both NEC[$arc(v)$] and NOC[$arc(v)$] are right. It is then easily seen that, by formulae (2) and (3), the statement also holds for u. □

From the complexity-theoretic point of view, it is easily seen that the time of Algorithm 4.1 is dominated by the time required by the DFS traversal and, hence, is $O(|E|)$. So, by Theorem 4.5 one has

Corollary 4.1. Let $G = (V, E)$ be a nonbipartite, connected simple graph. The set of odd edges of G can be found in $O(|E|)$ time.

4.5. Finding The Kernel

Let $\Gamma = (G, \boldsymbol{w})$ be an EWG with $G = (V, E)$ and kernel K. If the zero set Z of $\boldsymbol{w}$ is empty, then K is empty too and we are done. Assume that Z is not empty. If G is bipartite, Gusfield [27] proved that K equals the set of directed edges joining strongly connected components of the *mixed graph* $G(Z)$ obtained from G by directing all the edges in Z from one side of the bipartition to the other one so that it can be computed in time linear in the size of G. In this section we show that, even in the case that G is not bipartite, the kernel of Γ can be computed in time linear in the size of G.

With Γ we associate a bipartite EWG $\Gamma' = (G', \boldsymbol{w}')$ we call a *bipartite EWG associated* with Γ. The graph $G' = (V', E')$ is constructed as follows. Let B be a maximal bipartite partial graph of G and let $\{V_1, V_2\}$ be a bipartition of V such that each edge of B has one end in V_1 and the other end in V_2.

Let $\overline{V}$ be a "copy" of V, that is, $\overline{V} \cap V = \emptyset$ and $|\overline{V}| = |V|$. If v is a vertex of G, then by $\overline{v}$ we denote the copy of v. The vertex set of G' is taken to be $V' = V \cup \overline{V}$, and the edge set of G' is taken to be

$$E' = \cup_{e \in E} f(e)$$

where f is function defined on E as follows:

— if e is a self-loop, say (v, v), then $f(e) = \{(v, \overline{v})\}$;

— if $e = (u, v)$ is an edge of B then $f(e) = \{(u, v), (\overline{u}, \overline{v})\}$;

— if $e = (u, v)$ is neither a self-loop nor an edge of B then $f(e) = \{(u, \overline{v}), (\overline{u}, v)\}$.

The set $f(e)$ will be referred to as the *image* of e in G'. Let $V'_1 = V_1 \cup \overline{V}_2$ and $V'_2 = \overline{V}_1 \cup V_2$, where $\overline{V}_i = \{\overline{v} : v \in V_i\}$, $i = 1, 2$. The graph G' is bipartite and the partition $\{V'_1, V'_2\}$ of V' is such that each edge of G' has one end in V'_1 and the other end in V'_2. Furthermore, G' is connected if and only if G is not bipartite. Finally, to each edge e' of G' we assign the weight $w'(e') = w(e)$ where e is the edge of G for which $e' \in f(e)$. Let $\mathbf{A}'$ be the incidence matrix of G' and let $\boldsymbol{b}' = (b'(v'))_{v' \in V'}$ where $b'(v')$ equals the sum of the weights $w'(e')$ of the edges of G' incident to v'. Consider the equation system

$$\mathbf{A}' \boldsymbol{x}' = \boldsymbol{b}' \qquad (4.4)$$

For every edge e' of G', let $L[x'(e')]$ and $U[x'(e')]$ denote the minimum and the maximum of the variable $x'(e')$ over the nonnegative solutions of equation system (4.4), respectively. Moreover, for every edge e of G, let $L[f(e)]$ and $U[f(e)]$ denote the minimum and the maximum of the expression $\sum_{e' \in f(e)} x'(e')$ over the nonnegative solutions of equation system (4.4), respectively. First of all, observe that, if $\boldsymbol{x}'$ is a (nonnegative) solution of equation system (4.4), then a (nonnegative) solution $\boldsymbol{x}''$ of equation system (4.4) can be obtained by setting for each edge e' of G'

$$x''(e') = x'(e') \quad \text{if } \{e'\} \text{ is the image of a self-loop of } G$$

and

$$x''(e') = x'(e'') \text{ if } \{e', e''\} \text{ is the image of an edge of } G \text{ that is not a self-loop.}$$

It follows that, if $\{e', e''\}$ is the image of an edge of G that is a not a self-loop, then

$$L[x'(e')] = L[x'(e'')] \quad \text{and} \quad U[x'(e')] = U[x'(e'')] \,. \qquad (4.5)$$

Second, if $\boldsymbol{x}$ is a (nonnegative) solution of equation system (4.1), then a (nonnegative) solution $\boldsymbol{x}'$ of equation system (4.4) can be obtained by setting for each edge e' of G'

$$x'(e') = x(e) \text{ where } e \text{ is the edge of } G \text{ whose image } f(e) \text{ contains } e'.$$

On the other hand, if $\boldsymbol{x}'$ is a (nonnegative) solution of equation system (4.4), then a (nonnegative) solution $\boldsymbol{x}$ of equation system (4.1) can be obtained by setting for each edge e of G

$$x(e) = [\sum_{e' \in f(e)} x'(e')]/|f(e)|.$$

Therefore, one has

$$L[x(e)] = (1/|f(e)|)\, L[f(e)] \quad \text{and} \quad U[x(e)] = (1/|f(e)|)\, U[f(e)] \,. \qquad (4.6)$$

Example 4.2 (continued). By choosing as maximal bipartite partial graph of G the graph shown in Figure 4.7

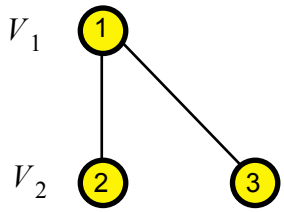


Figure 4.7

we associate with Γ the bipartite EWG $\Gamma' = (G', \boldsymbol{w}')$ shown in Figure 4.8.

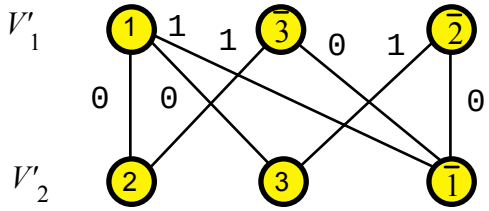


Figure 4.8

The general expression of a nonnegative solution of equation system (4.4) is

$$x'(1, 2) = x'(\bar{1}, \bar{3}) = \mu$$
$$x'(1, 3) = x'(\bar{1}, \bar{2}) = \nu$$
$$x'(1, \bar{1}) = 1 - \mu - \nu$$
$$x'(2, \bar{3}) = 1 - \mu$$
$$x'(\bar{2}, 3) = 1 - \nu$$

where μ and ν are bounded as shown in Figure 4.9:

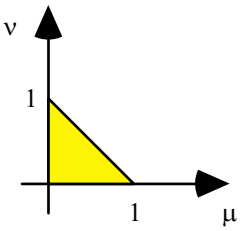


Figure 4.9

At this point, it is easy to check formulae (4.5) and (4.6). ■

We now state some technical results to relate the kernels of Γ and Γ'.

Lemma 4.7. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG and $\Gamma' = (G', \boldsymbol{w}')$ a bipartite EWG associated with Γ. An edge e of G is an invariant edge of Γ if and only if $L[f(e)] = U[f(e)]$, where $f(e)$ is the image of e in G'.

Proof. By formula (4.6). □

Lemma 4.8. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG and $\Gamma' = (G', \boldsymbol{w}')$ a bipartite EWG associated with Γ. An edge e of G belongs to the kernel of Γ if and only if its image in G' is contained in the kernel of Γ'.

Proof. If e is a self-loop of G then the statement immediately follows from formula (4.6) and Lemma 4.7. We now prove the statement in the case that e is not a self-loop and $f(e) = \{e', e''\}$.

(*if*) If both e' and e'' belongs to the kernel of Γ' then $x'(e') = x'(e'') = 0$ for every solution $\boldsymbol{x}'$ of equation system (4.4). Therefore, $L[x'(e')+x'(e'')] = U[x'(e')+x'(e'')] = 0$ and the statement follows from formula (4.6).

(*only if*) If e belongs to the kernel of Γ then, by formula (4.6), one has

$$x'(e')+x'(e'') = 0$$

for every nonnegative solution $\boldsymbol{x}'$ of equation system (4.4). By the nonnegativity of $\boldsymbol{x}'$, $x'(e') = x'(e'') = 0$, which proves that both e' and e'' belong to the kernel of Γ'. □

Corollary 4.2. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG and $\Gamma' = (G', \boldsymbol{w}')$ a bipartite EWG associated with Γ'. An edge of G belongs to the kernel of Γ if and only if an element of its image in G' belongs to the kernel of Γ'.

Proof. By Lemma 4.8 and formula (4.5). □

Example 4.2 (continued). The zero set of $\boldsymbol{w}'$ is $Z' = \{(1, 2), (1, 3), (\bar{1}, \bar{2}), (\bar{1}, \bar{3})\}$. The mixed graph $G'(Z')$ is strongly connected (see Figure 4.10).

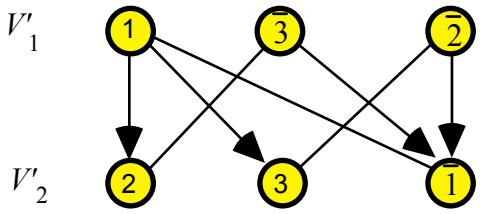


Figure 4.10

So, the kernel of Γ' is empty. By Corollary 4.2, the kernel of Γ is empty. ■

Theorem 4.6. The kernel of an EWG can be found in time linear in the size of G.

Proof. Let $\Gamma = (G, \boldsymbol{w})$ be an EWG and $\Gamma' = (G', \boldsymbol{w}')$ a bipartite EWG associated with Γ. If G is bipartite, then the statement was proven by Gusfield [27]. Otherwise, since G' is bipartite, the kernel K' of Γ' can be found in time linear in the size of G' and, hence, of G. So, it is sufficient to prove that both constructing G' and determining K from K' take a linear time. In order to construct G', we perform a depth-first-search traversal of G, which allows us to find both a maximal bipartite partial graph B of G and the nontree edges that create odd cycles when added to B. When an edge e' of G' is created, we get e' to point to the edge e of G for which $e' \in f(e)$. Finally, by Corollary 4.2, the set K can be obtained as follows. Initially, each edge e of G is unmarked. For each element e' of K', if the edge e of G that e' points to is unmarked, then e is marked and added to K. □

4.6. Security Of Statistical Data

In the security analysis of statistical data [8, 27, 29, 30, 33, 34, 35, 36, 45, 46, 49], EWGs and, more in general, weighted hypergraphs can be used to control the amount of information that is implicitly released when statistical data are made public, in order to avoid disclosure of confidential data. We now illustrate this application by discussing a typical case. Suppose that we are given a data set $\{b_i: i \in I\}$, where b_i is the value of a confidential variable b of nonnegative real type (e.g., salary) for individual i in the population I. The sum of values of b over a subset I' of I is called *sensitive* if I' contains exactly one individual. Now, given a statistical summary $\sigma = \{b(v): v \in V\}$, where $b(v)$ is the sum of b over a subset $I(v)$ of I containing at least two people, for each v in V, the problem that naturally arises consists in checking that no sensitive sum is implicitly released. We now present a graph-theoretic approach to this problem. Let $I_\sigma = \cup_{v \in V} I(v)$, we call the set of individuals *covered* by σ. The *basic partition* of I_σ is the coarsest partition of I_σ such that each $I(v)$ can be obtained as the union of one or more classes of the partition. A class J of the basic partition of I_σ will be indexed by the set $e = \{v \in V: J \subseteq I(v)\}$. If E is the index set of the classes of the basic partition of I_σ, the pair $G = (V, E)$ defines a hypergraph where hyperedge e is incident to vertex v if and only if v belongs to e. Consider the weighted hypergraph $\Gamma = (G, \boldsymbol{w})$ where, for each hyperedge e of G, $w(e)$ is given by the sum of the values of the variable b over the class $J(e)$ of the basic partition of I_σ indexed by e; that is,

$$w(e) = \sum_{i \in J(e)} b_i \ .$$

Finally, a hyperedge e of G is *marked* if $|J(e)| = 1$. Then, no sensitive sum is implicitly released given σ if and only if no invariant hyperedge of Γ is marked. If this is the case, the statistical summary σ is said to be *safe*. Since the invariant edges of an EWG can be found in linear time, one has that, if G is a graph, then one can decide whether σ is or is not safe in linear time too.

Example 4.3. Consider five individuals with salaries $b_1 = 2.0$, $b_2 = 2.5$, $b_3 = 3.8$, $b_4 = 3.7$ and $b_5 = 3.0$. Suppose that the two sums $b_1 + b_2 + b_3$ and $b_3 + b_4 + b_5$ are made public. Let $\sigma_1 = \{b_1 + b_2 + b_3, b_3 + b_4 + b_5\}$. The set of individuals covered by σ_1 is $\{1, \ldots, 5\}$ and its basic partition consists of the three classes $\{1, 2\}$, $\{3\}$ and $\{4, 5\}$. Thus, the weighted hypergraph Γ_1 associated with σ_1 is the EWG shown in Figure 4.11, where the edge (1, 2) is marked.

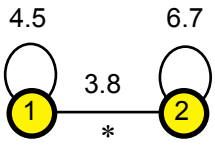


Figure 4.11

Since the set of invariant edges of Γ_1 turns out to be empty, σ_1 is safe. Next, suppose that the sum $b_1 + b_2 + b_4 + b_5$ is also made public. Let $\sigma_2 = \{b_1 + b_2 + b_3, b_3 + b_4 + b_5, b_1 + b_2 + b_4 + b_5\}$. Again, the set of individuals covered by σ_2 is $\{1, \ldots, 5\}$ and its basic partition consists of the three classes $\{1, 2\}$, $\{3\}$ and $\{4, 5\}$. The weighted hypergraph Γ_2 associated with σ_2 is the EWG shown in Figure 12, where the edge (1, 2) is marked.

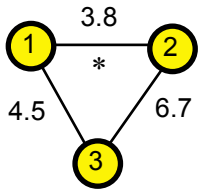


Figure 4.12

Since each edge is an invariant edge of Γ_2, σ_2 is not safe (and the salary b_3 is unprotected).

■

Let $\sigma = \{b(v): v \in V\}$ be a statistical summary of the data set $\{b_i: i \in I\}$ and let $\Gamma = (G, \boldsymbol{w})$ be the associated weighted hypergraph. It is worth noting that, if v is a "leaf" of G, that is, if v belongs to exactly one hyperedge e of G, then the class of the basic partition of I_σ indexed by E coincides with $I(v)$ so that $w(e) = b(v)$; furthermore, the hyperedge e is definitely an invariant hyperedge of Γ and, since $|I(v)| > 1$, it is not marked. As we are interested in checking the existence of marked invariant hyperedges of Γ (if any), we can reduce Γ by deleting all leaves of G and their incident hyperedges. Let $\Gamma' = (G', \boldsymbol{w}')$ be the resulting weighted hypergraph. Of course, σ is safe if and only if no invariant hyperedge of Γ' is sensitive. We now show that, if σ is a two-dimensional table with suppressions, then Γ' is always a graph so that one can decide whether σ is or is not safe in linear time. Let σ be obtained from a complete two-dimensional table T by suppressing all sensitive cells ("primary suppressions") as well as additional (internal or marginal) cells to exclude the possibility of arriving at the contents of sensitive cells by indirect methods ("complementary suppressions"). Denote by

$T(r, c)$ the value of internal cell (r, c), $1 \le r \le m$ and $1 \le c \le n$,

$T(r, +)$ the r.th row total, $1 \le r \le m$, and

$T(+, c)$ the c.th column total, $1 \le c \le n$.

Assume that each $T(r, c)$ is the sum of the values of a confidential variable of nonnegative real type over the set $I(r, c)$ of individuals. So, a cell (r, c) of T is sensitive if $|I(r, c)| = 1$. We first detail the structure of the weighted hypergraph $\Gamma = (G, \boldsymbol{w})$ associated with σ and, then, show that the reduction of Γ results in an EWG. Let U, R and C be the set of unsuppressed internal cells, the set of marginal cells corresponding to unsuppressed row totals and the set of marginal cells corresponding to unsuppressed column totals, respectively. Then the vertex set of G is

$$V = U \cup R \cup C.$$

Let $S = \{(r, c) \in U: r \notin R \text{ and } c \notin C\}$. Moreover, for each $r \in R$, let $C_r = \{c \notin C: (r, c) \notin U\}$; analogously, for each $c \in C$, let $R_c = \{r \notin R: (r, c) \notin U\}$. Then, the set of individuals covered by σ is $I_\sigma = \cup_{(r,c)\notin S} I(r, c)$ and the basic partition of I_σ contains:

one class $I(r, c)$ for each (r, c) in U and for each (r, c) not in U with $r \in R$ and $c \in C$,
one class $\cup_{c \in C_r} I(r, c)$ for each $r \in R$ with $C_r \neq \emptyset$, and
one class $\cup_{r \in R_c} I(r, c)$ for each $c \in C$ with $R_c \neq \emptyset$.

Recall that the hyperedges of G are the indices of these classes. The hyperedge e indexing a class such as $I(r, c)$ is

$e = \{(r, c), (r, +), (+, c)\}$ if $(r, c) \in U, r \in R, c \in C$

$e = \{(r, c), (r, +)\}$ if $(r, c) \in U, r \in R, c \notin C$

$e = \{(r, c), (+, c)\}$ if $(r, c) \in U, r \notin R, c \in C$

$e = \{(r, c)\}$ if $(r, c) \in U, r \notin R, c \notin C$

$e = \{(r, +), (+, c)\}$ if $(r, c) \notin U, r \in R, c \in C$

and $w(e)$ is always set to $T(r, c)$. For the hyperedge e indexing a class such as $\cup_{c \in C_r} I(r, c)$ one has $e = \{(r, +)\}$ and

$$w(e) = \sum_{c \in C_r} T(r, c),$$

and for the hyperedge e indexing a class such as $\cup_{r \in R_c} I(r, c)$ one has $e = \{(+, c)\}$ and

$$w(e) = \sum_{r \in R_c} T(r, c).$$

At this point, it should be clear that the leaves of G are all and the only vertices of the type (r, c), of the type $(r, +)$ with $r \in R$ and $C_r = \{1, \ldots, n\}$ and of the type $(+, c)$ with $c \in C$ and $R_c = \{1, \ldots, m\}$. Let L be the set of leaves of G, and let $R' = R–L$ and $C' = C–L$. After deleting all the leaves of the hypergraph G, we remain with the hypergraph graph $G' = (V', E')$ whose hyperedges are incident to at most two vertices. More precisely, one has that $V' = R' \cup C'$ and E' consists of the edges

$\{(r, +), (+, c)\}$ if $(r, c) \notin U, r \in R, c \in C$

$\{(r, +)\}$ with $r \in R'$

$\{(+, c)\}$ with $c \in C'$

To sum up, the reduction of the weighted hypergraph associated with σ is an EWG and, therefore, the safety of σ can be tested in linear time.

Example 4.4. Consider Table 4.1 whose entries are assumed to be nonnegative reals.

	j=1	j=2	j=3	j=4	
i=1	0	10	0	20	T(1,+) = 30
i=2	2	3	0	20	T(2,+) = 25
i=3	17	13	5	0	T(3,+) = 35
i=4	16	15	14	5	T(4,+) = 50
	T(+,1) = 35	T(+,2) = 41	T(+,3) = 19	T(+,4) = 45	

Table 4.1

Suppose that the following cells of Table 4.1

(1, 1), (1, 2), (2, 1), (2, 2), (2, 3), (4, 4)

are all sensitive. Table 4.2 is obtained from Table 4.1 by suppressing all the six sensitive cells and the additional cells (3, 3), (3, +), (4, +) and (+, 4).

	j=1	j=2	j=3	j=4	
i=1			0	20	T(1,+) = 30
i=2				20	T(2,+) = 25
i=3	17	13		0	
i=4	16	15	14		
	T(+,1) = 35	T(+,2) = 41	T(+,3) = 19		

Table 4.2

The reduced weighted hypergraph $\Gamma' = (G', \boldsymbol{w}')$ associated with Table 4.2 is the EWG shown in Figure 4.13. The invariant edges of Γ' are the edge joining the vertices (2, +) and (+, 3), and the self-loop at vertex (+, 3). One of these two edges is marked and, therefore, Table 4.2 is not safe.

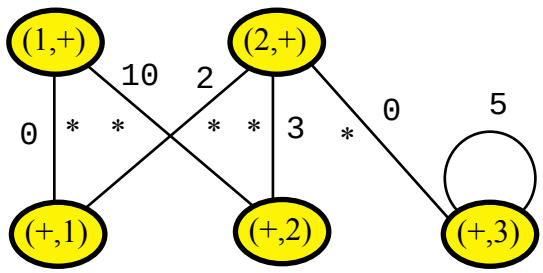


Figure 4.13

■

4.7 Protecting privacy for data of additive type

In this section we will solve the problem of finding the invariant edges of a graph when the values of the weights of the edges are taken from a commutative group.

Let A be an (additive) commutative group with zero 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 $y(e) = 0$ 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 $(G, \boldsymbol{q})$.

4.7.1 Find a compatible edge labeling

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∈A: 2a = b}. Accordingly, *half*(0) = {a∈A: 2a = 0} = {a∈A: a = –a}. For example, if A is the set {0, 1, …, 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 [21] where only loopless graphs are considered.

Proposition 4.3 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 4.2

(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 $E(T) = Ø$ (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.

Example. In fig 4.14 the algorithm 4.2 is applied to the map of step 1.

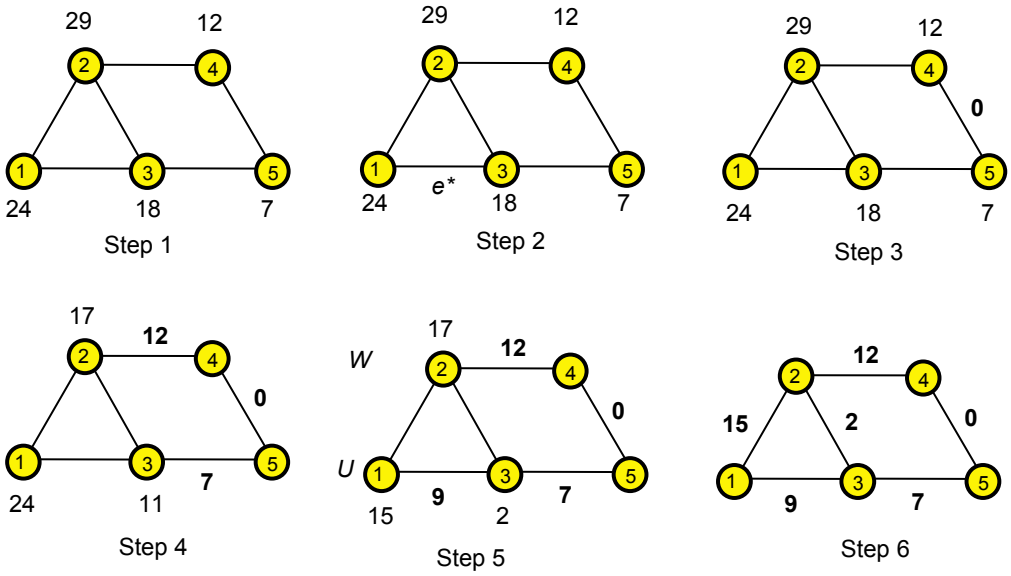


Figure 4.14

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 4.3) obtained from Algorithm 4.2 by replacing 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 $x(e^*) := q(v^*)$.

Example. In fig 4.15 the algorithm 4.3 is applied to the map of step 1.

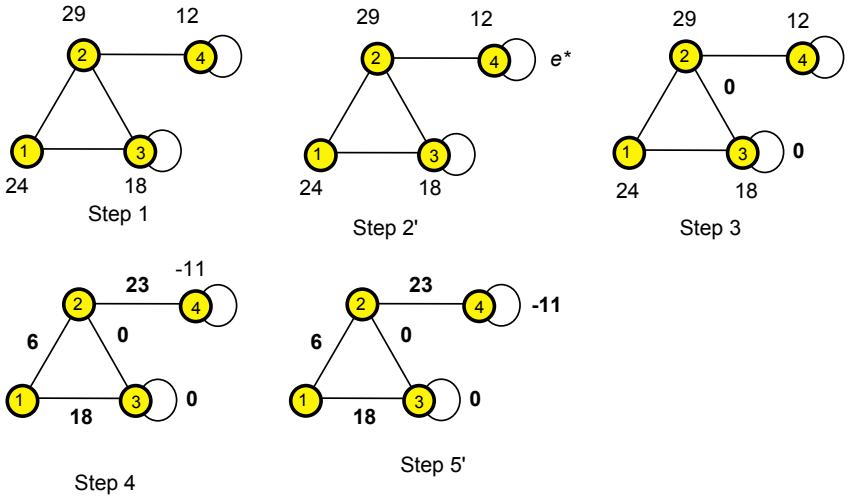


Figure 4.15

Proposition 4.4 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.

4.7.2 Characterization of invariant edges

Let G be a connected graph and let $\boldsymbol{Y} = \boldsymbol{X}(\mathbf{0})$. An edge labeling 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 $\{e \in E(G): y(e) \neq 0\}$ 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 4.2) 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 4.16) whose support is C.

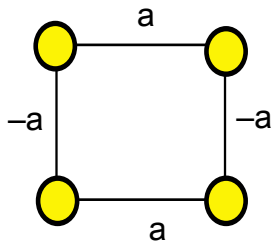


Figure 4.16 An A-circulation associated with an even cycle

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

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

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 4.2) creating an odd cycle, say C. Given an arbitrary element a of *half*(0), with C

we can associate an A-circulation (see Figure 4.17), whose support is empty or C depending on whether $\mathsf{a} = 0$ or $\mathsf{a} \neq 0$, respectively.

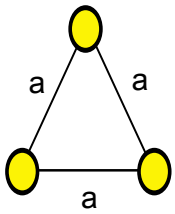


Figure 4.17 An A-circulation associated with an odd cycle

Let $\boldsymbol{c}_{\mathsf{a}}$ be such an A-circulation associated with C. If $G = T$, then $\boldsymbol{Y} = \{\boldsymbol{c}_{\mathsf{a}}: \mathsf{a} \in \mathit{half}(0)\}$ (see Algorithm 4.2) so that, if $\mathit{half}(0) = \{0\}$ then each edge of G is A-invariant; otherwise (that is, if $\mathit{half}(0) \neq \{0\}$), 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* [15], 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. 4.16; if C_i is an L-odd set, then the A-circulation is of the form shown in Fig. 4.18.

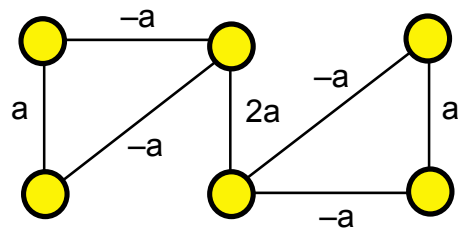


Figure 4.18 An A-circulation associated with an L-odd set

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

Case 2(*i*): $\mathit{half}(0) = \{0\}$. 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.

Case 2(*ii*): $\mathit{half}(0) \neq \{0\}$. Then, the A-invariant edges are all bridges since they belong to no cycles. Furthermore, if $\mathit{half}(0) \neq \mathsf{A}$ 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 $\mathit{half}(0) = \mathsf{A}$) then $2\mathsf{a} = 0$ 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 4.3). If $G = T$ then $\boldsymbol{Y} = \{\boldsymbol{0}\}$ 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. 4.16; if C_i is an L-odd set, then the A-circulation is of either form shown in Fig. 4.19.

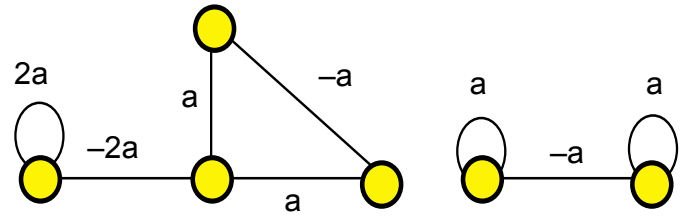


Figure 4.19 An A-circulation associated with an L-odd set containing a loop

Let $\boldsymbol{c}_{i,\mathsf{a}_i}$ 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.

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 4.5 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.

A consequence of Proposition 4.5 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 and the set of bridges whose removal creates one more loopless component can be found in linear time as shown in section 4.4;

— 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 as shown in section 4.4;

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 4.2 or Algorithm 4.3, depending on whether G is or is not loopless.

Chapter 5

Computing Feasible Ranges

5.1 Introduction

Recall the two problems at the beginning of Chapter 4 were:

(P1) given the equation system of an arbitrary information model (e.g., the posterior model), find the set of its null variables;

(P2) given the equation system of an information model in normal form (e.g., the prior model), find the feasibility range for a given sum of variables.

We saw in the previous Chapter that there exist a linear time algorithm to solve problem (P1) when the coefficient matrix **H** of system (1.1) is the incidence matrix of a graph. Problem (P2) has been solved for a single variable by Gusfield [27] if G is bipartite, using a maximum-flow algorithm. We shall show that, more in general, in a graphical information model the problem of finding the feasibility range for an arbitrary sum of variables can be solved using a strongly polynomial algorithm.

Using the same terminology of Chapter 4 we deal with a graph $G = (V, E)$ without isolated vertices (where self-loops may exist). Let $\boldsymbol{s} = (s(e))_{e \in E}$ be a vector of nonnegative reals. The pair $(G, \boldsymbol{s})$ is referred to as an edge-weighted graph (an EWG, for short). Let **A** be the incidence matrix of G and let $\mathbf{b} = (b(v))_{v \in V}$ be the vector of nonnegative reals such that, for each vertex v of G, $b(v)$ equals the sum of the weights of the edges incident to v.

In this Chapter we address the problem of computing the tightest bounds on the sum of weights of an arbitrary set of edges under the assumption that the edge weights are nonnegative real numbers. In other word we want to solve the following two linear programming problems **max** $\sum_{e \in J} x(e)$ and **min** $\sum_{e \in J} x(e)$ subject on the constraint

$$\begin{aligned} &\mathbf{A}\,\mathbf{x} = \mathbf{b} \\ &\mathbf{x} \geq \mathbf{0} \\ &\mathbf{x} \in \mathbb{R}^{|E|} \end{aligned} \qquad (5.1)$$

where J is an arbitrary non empty subset of edges of E. Here we shall show that for a nonbipartite graph the tightest bounds on the weight of an edge, that is in the case that $|J|=1$, can be found with two or four maximum-flow computations depending on whether the edge is or is not a loop.

Finally we consider the problem of finding the feasible tightest bounds on the sum of the weights of a set of edges, that is when $|J|>1$, effectively solving (P2) in the general case.

5.2 The bipartite case

In this section, we review the maximum-flow technique proposed by Gusfield [27] to compute the tightest bounds on the weight of a given edge of a EWG $(G, \mathbf{s})$ where $G = (V, E)$ is a bipartite, connected graph. Let $\{V_1, V_2\}$ be the vertex bipartition of G. First, a flow network [2] is built up as follows. Let $b(v) = \sum_{a \in E(v)} s(a)$ for each v in V, where $E(v)$ is the set of edges incident to v. Let M be a finite number larger than **max** $\{b(v): v \in V_1\}$. First, each edge (u, v) of G is directed both from V_1 to V_2 and from V_2 to V_1. Then, the capacity of each edge $u \rightarrow v$ is set to M if $u \in V_1$, and to $s(u, v)$ otherwise. We denote the resulting flow network by $N(G, \mathbf{s}; V_1, M)$.

Example 5.1. Consider the EGW $(G, \mathbf{s})$ shown in Figure 5.1.

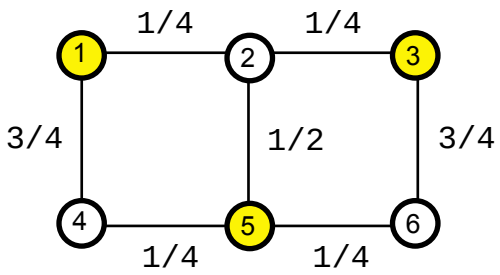


Fig. 5.1. A bipartite graph

Let $V_1 = \{2, 4, 6\}$ and $V_2 = \{1, 3, 5\}$, and let $M = 2$. Figure 5.2 shows the network $(G, s; V_1, M)$.

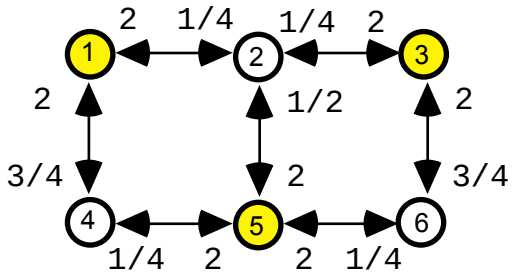


Fig. 5.2. The network associated with the EWG of Figure 5.1 ■

Let $f_{u,v}$ be a maximum flow in $N(G, \mathbf{s}; V_1, M)$ from vertex u to vertex v, and let $F_{u,v}$ be the value of $f_{u,v}$, that is,

$$F_{u,v} = \sum_w f_{u,v}(u \to w) - \sum_w f_{u,v}(w \to u) = \sum_w f_{u,v}(w \to v) - \sum_w f_{u,v}(v \to w)$$

Note that if $u \in V_1$ then

$$F_{u,v} = M + \sum_{w \in V_2 - \{v\}} f_{u,v}(u \to w) - \sum_{w \in V_2 - \{v\}} f_{u,v}(w \to u) .$$

The following three propositions were proven by Gusfield [27]. Recall that $\mathbf{X}$ is the set of solutions of the system of constraints (5.1) where $\mathbb{R}^+$ is the set of nonnegative reals.

Proposition 5.1. Given a EWG (G, s) where G is a bipartite connected graph with bipartition $\{V_1, V_2\}$, let a be an edge of G with end-points $u \in V_1$ and $v \in V_2$. Then

$$\min_X x(a) = \max \{0, s(a) + M - F_{u,v}\} \text{ and } \quad \max_X x(a) = F_{v,u}$$

where $F_{u,v}$ and $F_{v,u}$ are the values of maximum flows in $N(G, \mathbf{s}; V_1, M)$ from u to v and from v to u, respectively.

If $\mathbf{x} \in \mathbf{X}$ has $x(a) = \min_X x(a)$ ($x(a) = \max_X x(a)$, respectively), we call the map $(G, \mathbf{x})$ an *a-minimal* (*a-maximal*, respectively) *variant* of the EGW $(G, \mathbf{s})$.

Proposition 5.2. Given a EGW $(G, \mathbf{s})$ where G is a bipartite connected graph with bipartition $\{V_1, V_2\}$, let a be an edge of G with end-points $u \in V_1$ and $v \in V_2$.

(*i*) Given a maximum flow $f_{u,v}$ in $N(G, \mathbf{s}; V_1, M)$ from u to v, an a-minimal variant $(G, \mathbf{x})$ of $(G, \mathbf{s})$ can be constructed as follows. For each edge a' of G with end-points $u' \in V_1$ and $v' \in V_2$ take

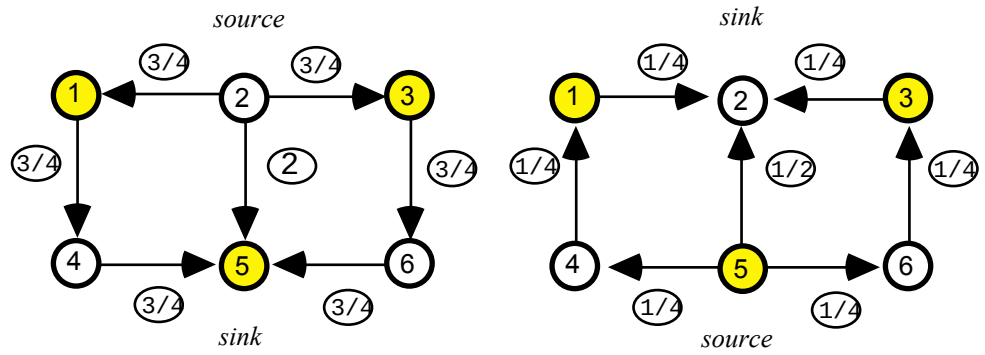
$$x(a') = \begin{cases} \max\{0, s(a) + M - F_{u,v}\} & \text{if } a' = a \\ s(a') + \tau[f_{u,v}(u' \to v') - f_{u,v}(v' \to u')] & \text{else} \end{cases}$$

where $\tau = \min\left\{1, \dfrac{s(a)}{F_{u,v} - M}\right\}$.

(*ii*) Given a maximum flow $f_{v,u}$ in $N(G, V_1, V_2; \mathbf{s}, M)$ from v to u, an a-maximal variant $(G, \mathbf{x})$ of $(G, \mathbf{s})$ can be constructed as follows. For each edge a' of G with end-points $u' \in V_1$ and $v' \in V_2$ take

$$x(a') = \begin{cases} F_{v,u} & \text{if } a' = a \\ s(a') + f_{v,u}(u' \to v') - f_{v,u}(v' \to u') & \text{else} \end{cases}$$

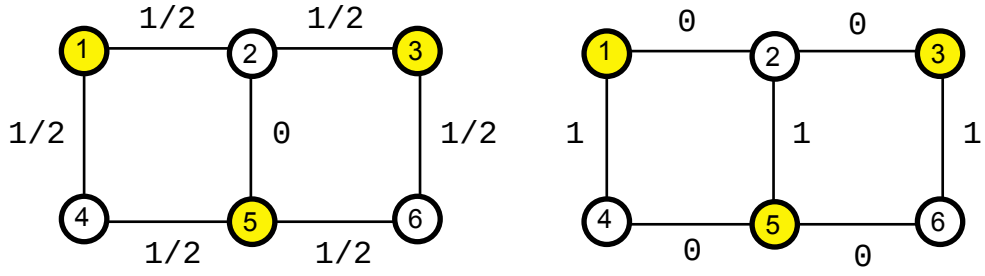
Example 5.1 (continued). Figure 5.3(*a*) shows a maximum flow $f_{2,5}$ from the vertex 2 to the vertex 5 and Figure 5.3(*b*) shows a maximum flow $f_{5,2}$ from the vertex 5 to the vertex 2. So, $F_{2,5} = 7/2$ and $F_{5,2} = 1$.



(*a*) (*b*)

Fig. 5.3.

By Proposition 5.1, the tightest lower and upper bounds on the weight of the edge (2, 5) are 0 and 1, respectively. Using Proposition 5.2, we obtain a (2, 5)-minimal variant and a (2, 5)-maximal variant of the EWG of Figure 5.1, which are shown in Figures 5.4(*a*) and 5.4(*b*) respectively.



(*a*) (*b*)

Fig. 5.4. Minimal and maximal variants ■

Proposition 5.3. Given a EGW $(G, \mathbf{s})$ where G is a complete bipartite graph with bipartition $\{V_1, V_2\}$, let $b(v) = \sum_{a \in E(v)} s(a)$, $v \in V$, and let $N = \sum_{v \in V_1} b(v)$. For each edge a of G with end-points u and v, one has

$$\min_X x(a) = \max\{0, b(u) + b(v) - N\} \qquad \max_X x(a) = \min\{b(u), b(v)\}.$$

5.3 Bounds on the weight of an edge

In this section we show that the tightest bounds on the weight of an edge of a nonbipartite EWG can be computed with two or four maximum-flow computations depending whether the edge is a loop or a link. In Section 4.5 we have seen that given an EWG $(G, \mathbf{s})$ we can derive an EWG $(G', \mathbf{s}')$ called the bipartite EWG associated to $(G, \mathbf{s})$ such that $G'=(V', E')$ is bipartite. Let $\boldsymbol{A}'$ be the node edge incidence matrix of G'. Consider the following system of linear constraint

$$\begin{aligned} &\boldsymbol{A}'\,\mathbf{y} = \boldsymbol{b}' \\ &\mathbf{y} \geq \mathbf{0} \\ &\mathbf{y} \in \mathbb{R}^{|E'|} \end{aligned} \qquad (5.2)$$

where $b'(v)=\sum_{a\in E(v)} s'(a)$ for each v in V' and let $\mathbf{Y}$ be the set of solution of (5.2). By formula (4.6), the tightest bounds on the weight of a loop of $(G, \mathbf{s})$ coincide with the tightest bounds on the weight of the corresponding edge of $(G', \mathbf{s}')$ and, hence, by Proposition 5.1 they can be determined with two maximum-flow computations.

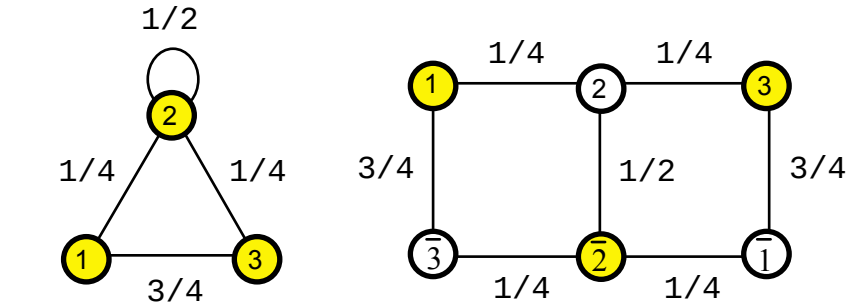


Fig. 5.6. (a) A nonbipartite EWG $(G, \mathbf{s})$; (b) an bipartite transform of $(G, \mathbf{s})$

Example 5.3. Consider the nonbipartite EWG $(G, \mathbf{s})$ of Figure 5.6(a). We want to compute the tightest bounds on the weight of the loop (2, 2). The bipartite transform of G of Figure 5.6(b) looks like the graph of Figure 5.1. Therefore, the tightest bounds on the weight of the loop (2, 2) of the map of Figure 5.6(a) coincide with the tightest bounds on the weight of the arc (2, 5) of the map of Figure 5.1, which were `0` and `1`. ■

Consider now the case of a link a of $(G, \mathbf{s})$ where $\{e', e''\}$ is its image in G'. By formula (4.6), we have to compute

$$\min_Y [y(e') + y(e'')] \quad \text{and} \quad \max_Y [y(e') + y(e'')].$$

subject to system (5.2). We shall show that they can be obtained as follows: The minimum (maximum, respectively) of the function $y(e') + y(e'')$ over $\mathbf{Y}$ equals the tightest lower (upper, respectively) bound on the weight of the arc e' of G' plus the tightest lower (upper, respectively) bound on the weight of the arc e'' of the EWG that is obtained from a e'-minimal (e'-maximal, respectively) variant of $(G', \mathbf{s}')$ by deleting e'.

We now prove the correctness for $\max_Y [y(e') + y(e'')]$. The proof of the correctness for $\min_Y [y(e') + y(e'')]$ is similar. In the next Lemma we use the terminology of Section 4.2.

Lemma 5.1. Let $(G', \mathbf{s}')$ be a bipartite EWG and let e' and e'' be two edges of G'. There exists a nonnegative real-valued solution $\mathbf{y}$ of equation system (5.2) that maximises both $y(e')$ and $y(e') + y(e'')$.

Proof. Let $(G', \mathbf{y}_o)$ be a e'-maximal variant of $(G', \mathbf{s}')$. Moreover, let Y_1 be the set of nonnegative real-valued solutions of equation system (5.2) that maximise $y(e') + y(e'')$, and let $\mathbf{y}_1$ be in Y_1 and such that $y_1(e') \geq y(e')$ for every $\mathbf{y} \in Y_1$. Of course, one has $y_1(e') \leq y_o(e')$. We shall prove that $y_1(e') = y_o(e')$. Consider the circulation (see Section 4.2), $\mathbf{z} = \mathbf{y}_o - \mathbf{y}_1$, and let (S^+, S^-) be the signed support of $\mathbf{z}$. Suppose, by contradiction, that $y_1(e') < y_o(e')$. Then e' belongs to S^+. By Proposition 4.1, e' lies in a cycle C which is the support of a minimal circulation having signed support (C^+, C^-), where $C^+ = C \cap S^+$ and $C^- = C \cap S^-$. Such a minimal circulation can be explicitly constructed as follows. Let

$$\varepsilon = \min \{|c(e)|: e \in C^-\}$$

and let ζ be the circulation with

$$\zeta(e) = \begin{cases} +\varepsilon & e \in C^+ \\ -\varepsilon & e \in C^- \\ 0 & \text{else} \end{cases}$$

Let $\mathbf{y}_2 = \mathbf{y}_1 + \boldsymbol{\zeta}$. Of course, $\mathbf{y}_2$ is a solution of equation system (5.2). Indeed, $\mathbf{y}_2$ is nonnegative everywhere because, for each edge e of G', if e is not in C^-, then

$$y_2(e) = y_1(e) + \zeta(e) \geq y_1(e) \geq 0;$$

otherwise,

$$y_2(e) = y_1(e) - \varepsilon \geq y_1(e) + z(e) = y_o(e) \geq 0 \ .$$

We now show that from the foregoing a contradiction always follows. Consider the following three cases that can occur for e'':

Case 1. e'' is in C^+. Then $y_2(e'') = y_1(e'') + \varepsilon$ which leads to the following

$$y_2(e') + y_2(e'') = y_1(e') + y_1(e'') + 2\varepsilon > y_1(e') + y_1(e'')$$

which contradicts the membership of $\mathbf{y}_1$ in Y_1.

Case 2. e'' is in C^-. Then $y_2(e'') = y_1(e'') - \varepsilon$ and, hence, one has

$$y_2(e') + y_2(e'') = y_1(e') + \varepsilon + y_1(e'') - \varepsilon = y_1(e') + y_1(e'')$$

so that $\mathbf{y}_2$ belongs to Y_1. But, since e' is in C^+, one has

$$y_2(e') = y_1(e') + \zeta(e') = y_1(e') + \varepsilon > y_1(b')$$

which contradicts the choice of $\mathbf{y}_1$.

Case 3. $e'' \notin C^+ \cup C^-$. Then $y_2(e'') = y_1(e'')$ which leads to the following

$$y_2(e') + y_2(e'') = y_1(e') + \varepsilon + y_1(e'') > y_1(e') + y_1(e'')$$

which contradicts the membership of $\mathbf{y}_1$ in Y_1. □

The following is an immediate consequence of Lemma 5.1.

Lemma 5.2. The maximum of the function $y(e') + y(e'')$ over the set $\mathbf{Y}$ of nonnegative real-valued solutions of equation system (5.2) equals the tightest upper bound on the weight of the edge e' of $(G', \mathbf{s}')$ plus the tightest upper bound on the weight of the edge e'' of the EWG obtained from a e'-maximal variant of $(G', \mathbf{s}')$ by deleting e'.

Combining Lemma 5.2 with Proposition 5.1, we obtain the following algorithm which, given a bipartite transform $(G', \mathbf{s}')$ of $(G, \mathbf{s})$ and a link a of G where $\{e', e''\}$ is the image of a in G', computes the tightest upper bound on the weight of a link a of a nonbipartite EWG $(G, \mathbf{s})$. The input data are:

$(G', \mathbf{s}')$	a bipartite transform associated with $(G, \mathbf{s})$
$\{W_1, W_2\}$	the bipartition of G'
$N(G', s'; W_1, M)$	a network associated with $(G', \mathbf{s}')$

$e' = (w_1', w_2')$ with $w_1' \in W_1$ and $w_2' \in W_2$, $e'' = (w_1'', w_2'')$ with $w_1'' \in W_1$ and $w_2'' \in W_2$.

Algorithm MAX

(1) Find a maximum flow f in $N(G', \mathbf{s}'; W_1, M)$ from w_2' to w_1', and let F be the value of f.

(2) Given f and using Proposition 5.2(ii), construct a e'-maximal variant $(H, \mathbf{y})$ of $(G', \mathbf{s}')$.

(3) Let $(H, \mathbf{t})$ be the EWG obtained from $(G', \mathbf{s}')$ by deleting e'. Find a maximum flow f' in $N(H, \mathbf{t}; W_1, M)$ from w_2'' to w_1'', and let F' be the value of f'.

(4) Set the tightest upper bound on the weight of a to $\dfrac{F + F'}{2}$.

Analogously, the following algorithm correctly computes the tightest lower bound on the weight of a.

Algorithm MIN

(1) Find a maximum flow f in $N(G', \mathbf{s}'; W_1, M)$ from w_1' to w_2' .

(2) Given f and using Proposition 2(i), construct a e'-minimal variant $(H, \mathbf{y})$ of $(G', \mathbf{s}')$.

(3) Let $(H, \mathbf{s})$ be the EWG obtained from $(G', \mathbf{s}')$ by deleting e'. Find a maximum flow f' in $N(H, \mathbf{t}; W_1, M)$ from w_1'' to w_2'', and let F' be the value of f'.

(4) Set the tightest lower bound on the weight of a to

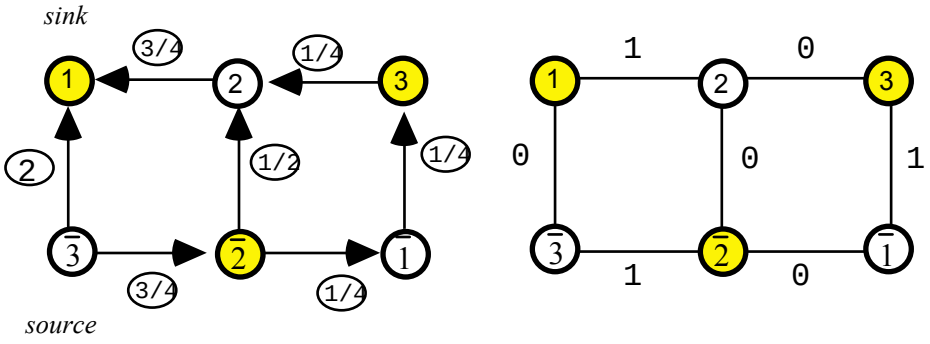
$$\frac{\max\{0, s'(e') + M - F\} + \max\{0, t(e'') + M - F'\}}{2}.$$

To sum up we have the following

Theorem 5.1. The tightest bounds of an edge of a nonbipartite EWG can be found with two or four maximum-flow computations depending on whether the edge is a loop or a link.

Example 5.3 (continued). We now apply the procedure above to compute the tightest lower and upper bounds on the weight of the link (1, $\bar{3}$) of the nonbipartite EWG of Figure 5.6(*a*). Recall that this edge corresponds to the two edges ($\bar{3}$, 1) and ($\bar{1}$, 3) of the associated bipartite transform shown in Figure 5.6(*b*), and that the vertices $\bar{1}$ and $\bar{3}$ are on the side W_1 and the vertices 1 and 3 are on the side W_2.

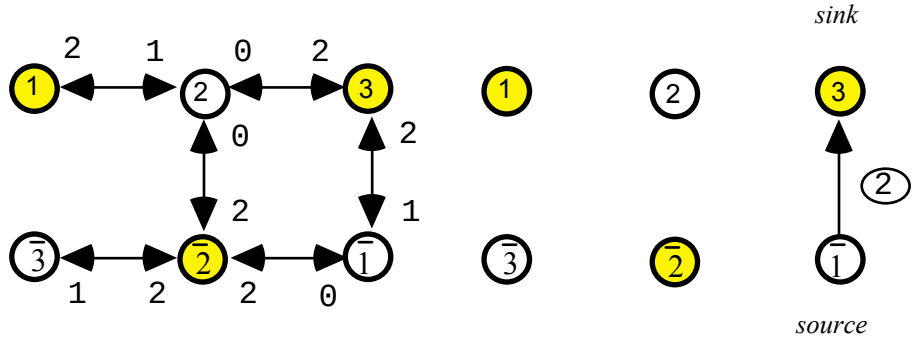
We first apply Algorithm MIN. Figure 5.7(*a*) shows a maximum flow in the network associated with the bipartite EWG shown in Figure 5.6(*b*) from the vertex $\bar{3}$ to the vertex 1, and Figure 5.7(*b*) shows the corresponding ($\bar{3}$, 1)-minimal variant of the EWG of Fig. 5.6(*b*).



(*a*) (*b*)

Fig. 5.7

So, the tightest lower bound on the weight of the edge ($\bar{3}$, 1) is max $\{0, \frac{3}{4}+2-\frac{11}{4}\} = 0$. Figure 5.8(*a*) shows the network associated with the bipartite EWG of Figure 5.7(*b*) with the edge ($\bar{3}$, $\bar{1}$) deleted, and Figure 5.8(*b*) shows a maximum flow in this network from the vertex 3 to the vertex $\bar{1}$.



(*a*) (*b*)

Fig. 5.8

So, the tightest lower bound on the weight of the edge ($\bar{1}$, 3) is max $\{0, 1+2-2\} = 1$, and the tightest lower bound on the weight of its corresponding edge (1, 3) is $(0+1)/2 = \frac{1}{2}$.

We now apply Algorithm MAX. Figure 5.9(*a*) shows a maximum flow in the network associated with the bipartite EWG shown in Figure 5.6(*b*) from the vertex 1 to the vertex $\bar{3}$, and Figure 5.9(*b*) shows the corresponding ($\bar{3}$, 1)-maximal variant of the map of Fig. 5.6(*b*).

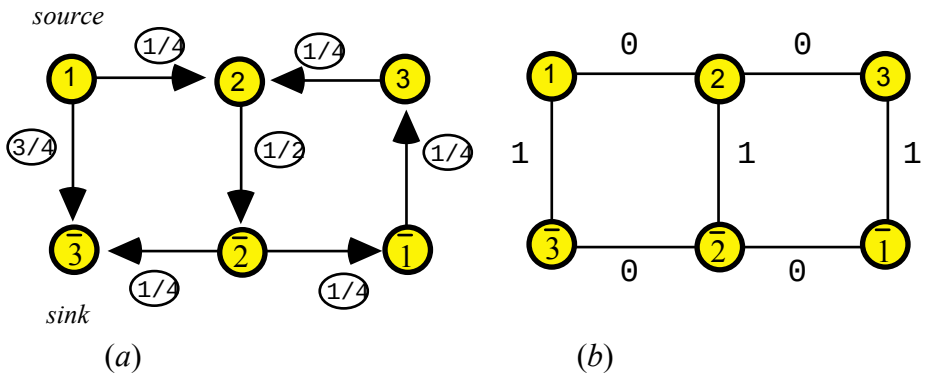


Fig. 5.9

So, the tightest upper bound on the weight of the edge $(\bar{3}, 1)$ is equal to 1. Figure 5.10(*a*) shows the network associated with the bipartite EWG of Figure 5.9(*b*) with the edge $(\bar{3}, 1)$ deleted, and Figure 5.10(*b*) shows a maximum flow in this network from the vertex 3 to the vertex $\bar{1}$.

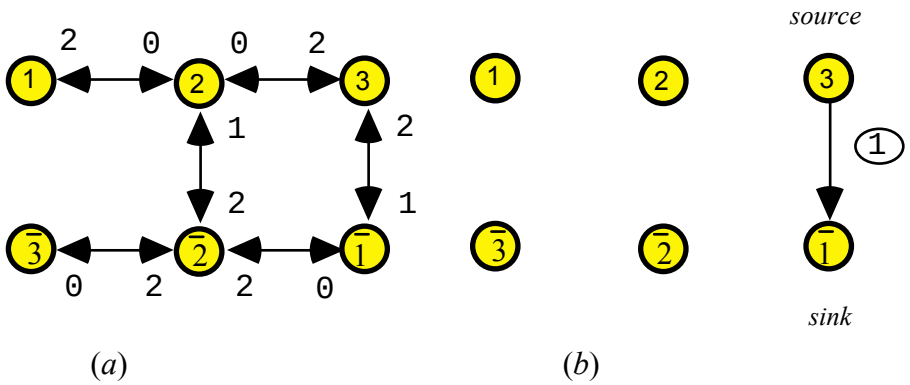


Fig. 5.10

So, the tightest upper bound on the weight of the edge $(\bar{1}, 3)$ is 1, and the tightest upper bound on the weight of the edge (1, 3) is equal to $(1+1)/2 = 1$. ■

5.4 A special Case

In this section, we consider the special case of a complete graph with the addition of one loop for each vertex. We now prove that the tightest bounds on the weight of an edge can be computed with a number of arithmetic operations and comparisons proportional to the number of vertices.

Let $(G, \mathbf{s})$ be a EWG where $G = (V, E)$ is a complete graph with the addition of one loop for each vertex. Let $b(v) = \sum_{a \in E(v)} s(a)$ for each v in V. The constraint system (5.1) always admits the solution $\mathbf{x}$ (see Figure 5.11) with

$$x(u, v) = \begin{cases} b(v) & \text{if } u = v \\ 0 & \text{else} \end{cases}$$

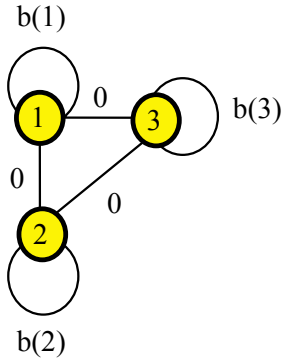


Fig. 5.11

Moreover, given a link (u^*, v^*) of G with $b(u^*) \leq b(v^*)$, then the constraint system (5.1) always admits the solution **x** (see Figure 5.14) with

$$x(u, v) = \begin{cases} b(u^*) & \text{if } (u, v) = (u^*, v^*) \\ 0 & \text{else} \end{cases}$$

for each link (u, v), and

$$x(u, u) = \begin{cases} 0 & \text{if } u = u^* \\ b(v^*) - b(u^*) & \text{if } u = v^* \\ b(u) & \text{else} \end{cases}$$

for each loop (u, u).

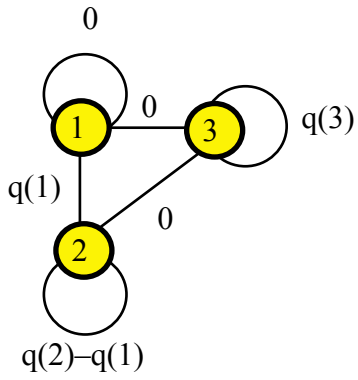


Fig. 5.12

It follows that the tightest upper bound on the weight of loop (u, u) is equal to $b(u)$, and the tightest lower and upper bounds on the weight of link (u, v) are 0 and **min** $\{b(u), b(v)\}$, respectively. What remains is a formula for the tightest lower bound on the weight of loop (u, u). We now prove that it is given by **max** $\{0, 2b(u) - N\}$, where $N = \sum_{v \in V} b(v)$. Let $(H, \mathbf{t})$ be a bipartite EWG associated with (G, s) so that the loop (u, u) corresponds to the edge $(u, \overline{u})$ of H. For each vertex w of H, let $r(w) = q(v)$ if v is the vertex of G corresponding to w. By Proposition 5.3, the tightest lower bound on the weight of $(u, \overline{u})$ is

$$\max \{0, r(u) + r(\overline{u}) - \sum_{w \in W_1} r(w)\} .$$

But $r(u) = r(\overline{u}) = b(u)$ and $\sum_{w \in W_1} r(w) = \sum_{v \in V} b(v) = N$ so that the statement follows from part (*i*) of Lemma 5.1. To sum up, we have the following result.

Theorem 5.2. Let (G, **s**) be an EWG where G is a complete graph with the addition of one loop for each vertex. Let $b(v) = \sum_{a \in E(v)} s(a)$ for each v in V, and let $N = \sum_{v \in V} b(v)$. Then,

(*i*) the tightest lower and upper bounds on the weight of loop (u, u) are respectively max {0, $2b(u) - N$} and $b(u)$;
(*ii*) the tightest lower and upper bounds on the weight of link (u, v) are respectively 0 and min {$b(u)$, $b(v)$}.

5.5 Computing feasible range of a sum for an arbitrary set of variables

Suppose that, given a subset J of edges of G, that is a subset of {1,…,m} one wants to compute the tightest lower bound or the tightest upper bound on the sum of variables $\sum_{j \in J} x_j$ over the set of solutions of system (5.1). They can be obtained by solving the linear-programming problem

$$\text{minimize} \quad \sum_{j=1,\ldots,m} c_j x_j \qquad (5.3)$$

subject to system (5.1)

where each c_j is set to the j-th component of the characteristic vector of J (for the tightest lower bound) or to its opposite (for the tightest upper bound). Now, if the graph G is bipartite, then problem (5.3) can be naturally viewed as a *bipartite transportation problem* [3] and can be efficiently solved using the *network simplex method*, which is a strongly polynomial algorithm [3]. If G is not bipartite, consider a bipartite transform G' of G. Let **y** be a solution of system (5.2). Since for any edge j of G we have that (see Section 4.5)

$$x_j = [\sum_{j' \in f(j)} y_{j'}] \,/\, |f(j)|.$$

where $f(j)$ is the image of j in G' (see Section 4.5) then we can translate problem (5.3) into the following bipartite transportation problem. Then we have

$$\sum_{j=1,\ldots,m} c_j x_j = \sum_{j=1,\ldots,m} c_j [\sum_{j' \in f(j)} y_{j'}] / |f(j)|$$

Therefore we can rewrite problem (5.3) as

$$\text{minimize} \sum_{j=1,\ldots,m} c_j [\sum_{j' \in f(j)} y_{j'}] / |f(j)| \qquad (5.4)$$

subject to system (5.2)

It is easily seen that, if **x** is a nonnegative solutions of problem (5.3) and **y** is the nonnegative solution of problem (5.4) associated with **x**, then

$$\sum_{j=1,\ldots,m} c_j x_j = \sum_{j=1,\ldots,m} c_j [\sum_{j' \in f(j)} y_{j'}] \,/\, |f(j)|$$

and *vice versa*. So, every optimal solution of problem (5.3) corresponds to an optimal solution of problem (5.4) and *vice versa*, and the minima of problems (5.3) and (5.4) do coincide.

Before closing this section, it is worth considering the special case that the components of the vector **b** in system (5.1) are all nonnegative integers. If G is bipartite then, by the integrality theorem and by the total unimodularity [2, 13, 55] of the incidence matrix of G, problem (5.3) has an integral optimal solution. If G is not bipartite then, since problem (5.4) has an integral optimal

solution, problem (5.3) has an optimal solution whose components are either integers or half-integers.

Chapter 6

A linear time algorithm to solve the NAS problem in a graph

6.1. Introduction

Let H be a hyper-graph with edge set $E(H)$ and node set $V(H)$. Let **M** be the node-edge incidence matrix of H. Let A be a subset of $E(H)$ and let **a** be the characteristic vector of A , that is

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

Recall that a subset A of edges of $E(H)$ is said to be algebraic if its characteristic vector can be expressed as a linear combination of rows of **M**, that is if there exist real coefficients $(c_v)_{v\in V(H)}$ such that

$$\mathbf{a} = \sum_{v\in V(H)} c_v \, \mathbf{m}_v \qquad (6.1)$$

where $\mathbf{m}_v$ is the row of **M** corresponding to node v. In this chapter we address the problem to decide if a set of edges F of H contains a proper non-empty algebraic subset. This problem was proved to be NP-*Complete* in Chapter 2.

We will show that the NAS problem can be solved in linear time when H is a graph. This result allow us to find a maximal algebraic set contained in F in at most quadratic time in the size of the graph.

6.2. Algebraic set

Let $G=(V(G), E(G))$ be a graph without parallel edges where loops may exist. An edge (u,v) is a *link* if it is not a loop. If U and W are two non empty subsets of $V(G)$ with $[U, W]$ we denote all the edges of $E(G)$ with one end point in U and the other in W.

A graph G is said to be *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 $[U, U]=[V, V]=\varnothing$. *W*e 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 class of algebraic sets (see also [33, 34, 35, 39]). An *open-flower set* is an open starset or the proper difference of two open starsets $S(W_1)$ and $S(W_2)$ where $S(W_2)\subseteq S(W_1)$ (see Fig 6.1). An open-flower set is algebraic. In fact its characteristic vector can be written as

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

If G is loopless, a *closed-flower* set is the proper difference of two starsets $S(W_1)$ and $S(W_2)$ where

$S(W_1) \cup S(W_2) = E(G)$ and

$S(W_2)$ is a open starset or is empty

Note that if both $S(W_1)$ and $S(W_2)$ are open starsets, $S(W_2) \neq \varnothing$ and $S(W_1) \cup S(W_2) = E(G)$ then $[W_1, W_1] = [W_2, W_2] = \varnothing$, G is bipartite and the proper difference of $S(W_1)$ and $S(W_2)$ is empty. Therefore we consider closed-flower set only if G is loopless and not bipartite. Note that if A is a closed-flower set then $G–A$ is a bipartite graph.

Also note that a closed-flower set is algebraic since its characteristic vector can be written as $1/2(\sum_{v \in W_1} \mathbf{m}_v - \sum_{v \in W_2} \mathbf{m}_v)$ (see Fig. 6.1).

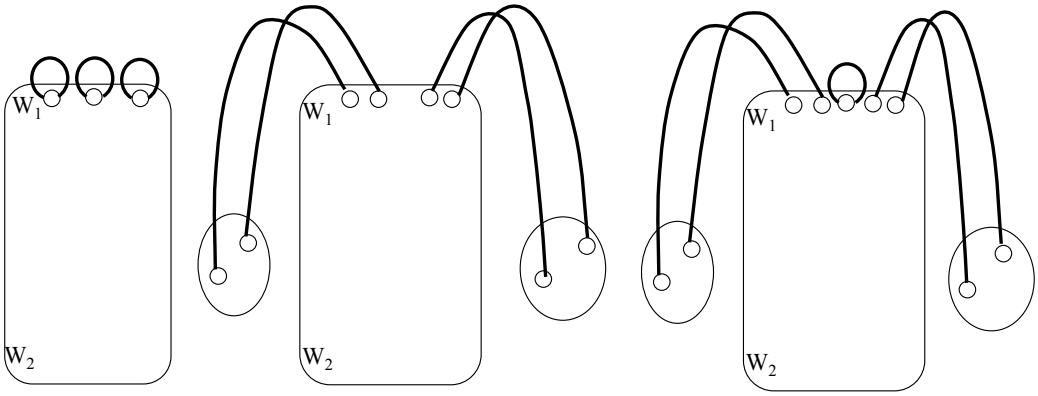


Figure 6.1. Open-flower sets.

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

Given a real valued vector $\mathbf{c}=(c_v)_{v \in V(G)}$, the *signed support* [6] 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 $P \cup N$.

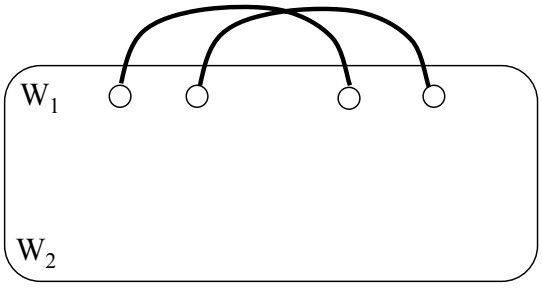


Figure 2. Closed-flower set.

Let A be an algebraic set and let $\mathbf{c}$ be a solution of (6.1). Let (P, N) be the signed support of $\mathbf{c}$. Since each equation of (6.1) is in the form

$$a((u, v)) = c_v + c_u$$
$$a((u, u)) = c_u$$

then its easily seen that

$$[N, V(G)–P] = \varnothing$$

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

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

so that

$$c_{v_k}=(-1)^{k-1}c_{v_1}+ \pi \qquad (6.2)$$

where π is an integer ([44]). 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

Lemma 6.1 Let G be a connected graph and A be a nonempty algebraic set. Let (P, N) be the signed support of a solution of (6.1). 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.
Proof Let **c** be a solution of (6.1). By formula (6.2) is easy to see that if $P\cup N\neq V(G)$ or G contains loops then every component of **c** is integer and if $[P, P]$ contains a link (u, v) then no component of **c** is an integer since $c_u+c_v=1$ and $c_u, c_v >0$. It follows that if $P\cup N\neq V(G)$ and $[P, P]$ is not empty it must contains only loops. Moreover if $[P, P]$ is not empty then either G is loopless or $[P, P]$ contains all loops of G. ♦

Theorem 6.1 Every nonempty algebraic set of a connected graph G, 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 bipartite then it is known ([34, 39]) that a nonempty edge subset of $E(G)$ is algebraic 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 (6.1). 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 is bipartite. By Lemma 6.1, $[P, P]$ 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 6.1, if $[P, P]$ is not empty then is a set of loops. Therefore the subset $[P, V(G)-N]$ of A is an open-flower set. ♦

Example 6.1 Referring to Fig 6.3: (a) the bold edges form an open-flower set and (b) a closed-flower set. For both (a) and (b) the signed support of the solution of (1) are shown. A solution of (6.1) is: for (a) $c_v = +1$ if $v\in P$, $c_v =-1$ if $v\in N$. For (b) , $c_v =+1/2$ if $v\in P$, $c_v =-1/2$ if $v\in N$. The set Z is given by $V(G)-P\cup N$.

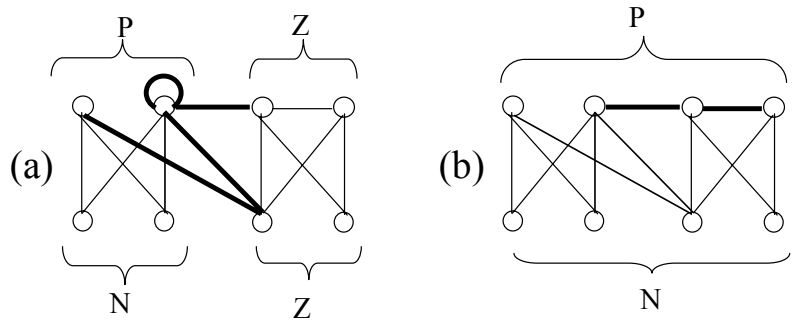


Figure 6.3. Example of (a) open-flower set and (b) closed flower set.

6.3. The kernel of an edge set

In this section we introduce a particular subset of a given edge set F we call the *kernel* of F, which has the property of containing all the algebraic subsets of F. Consider the following linear system

$$\mathbf{M}\,\mathbf{x} = \mathbf{b} \qquad (6.3)$$

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

$$b_v = |star(v) - F| \qquad v\in V(G)$$

A solution of system (6.3) 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 (6.3) is given by

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

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

$$\mathbf{M}\,\mathbf{y} = \mathbf{0} \qquad (6.4)$$

The set of solutions $\mathbf{y}$ of (6.4) is called the *null space* of $\mathbf{M}$. According to the terminology introduced in Chapter 4 [51] if $\mathbf{X}$ is the (non-empty) set of nonnegative solutions of (6.3), 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.

Theorem 6.2 Let F a nonempty edge set of a graph. An algebraic set A is a subset of F if and only if it is a subset of K.

Proof (*if*) Trivially if A is a subset of K then A is a subset of F since $K\subseteq F$.

(*only if*) First we show that if A is an algebraic subset of F then $\sum_{e\in A} x(e)$ takes on the same value for every nonnegative solution $\mathbf{x}$ of (6.3). 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 nonnegative solution of (3), then $\mathbf{x}_1-\mathbf{x}_2$ is a solution of (6.4). 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 (6.3) 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 (6.3). It follows that $A\subseteq K$. ♦

Consider the set of edges $F–K$. By the kernel definition, for every edge e of $F–K$, there exists a nonnegative solution $\mathbf{x}^{\#}$ of (6.3) such that $x^{\#}(e)>0$. It follows that there exists a solution $\mathbf{y}=\mathbf{x}^{\#}-\mathbf{x}^*$ of system (6.4) such that $y(e)>0$. More generally we have the following

Lemma 6.2 If K is the kernel of F then there exists a nonnegative solution $\mathbf{x}'$ of (6.3) such that $x'(e)=0$ if and only if e is in K.
Proof 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 (6.4) such that $y_i(e_i)>0$ and $\mathbf{x}^* + \mathbf{y}_i \geq \mathbf{0}$, $i=1,\ldots,p$. 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

$$x'(e)>0 \qquad \text{if and only if } e\notin K$$

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

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

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 all the edges of $E(G) –F$. 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 $x^*(e) + \varphi\, y(e)>0$. ♦

Now we state a useful property of the edges of the kernel of F that can be obtained from Theorem 4.1 (see also [27, 41]).

Lemma 6.3 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 6.2, there exist a nonnegative solution $\mathbf{x}'$ of (6.3) such that $x'(e)=0$ if and only if $e \in K$. Now let $0 < \varepsilon < \mathbf{min}\{ x'(e) : x'(e)>0\}$ and let $\mathbf{y}=(y(e))_{e \in E(G)}$ be defined as follows

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

Clearly $\mathbf{y}$ is a solution of system (6.4). But then we have that $\mathbf{x}'+\mathbf{y}$ is a nonnegative solution of (6.3) and $x'(e)+y(e)=+\varepsilon>0$ 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 2.5.

Lemma 6.4 Let K be the kernel of F. Then there always exist a real valued solution $\mathbf{c}$ to the following system of linear constraints

$$\sum_{v \in V(G)} c_u \, m_v(e) \; = \begin{cases} >0 \text{ if } e \in K \\ \\ =0 \text{ if } e \notin K \end{cases} \qquad (6.5)$$

The proof of this Lemma will be given in the Section 6.5, where we will give an algorithm that always compute a solution of (6.5).

Let (P, N) be the signed support of a solution of (6.5). Since each equations of (6.5) is in the form $c_u+c_v \geq 0$ or $c_v \geq 0$, we have that $[N, V(G)-P]=\varnothing$ and that $[P, V(G)-N] \subseteq K \subseteq [P, V(G)]$. Also if $Z=V(G)-(P \cup N)$ then $K \cap [Z, Z]= \varnothing$

Example 6.2 In the graph of Fig.6.2(a) the edges of a subset F of $E(G)$ are shown in bold. Fig.6.2(b) shows the edges of the kernel K along with a solution of (6.5).

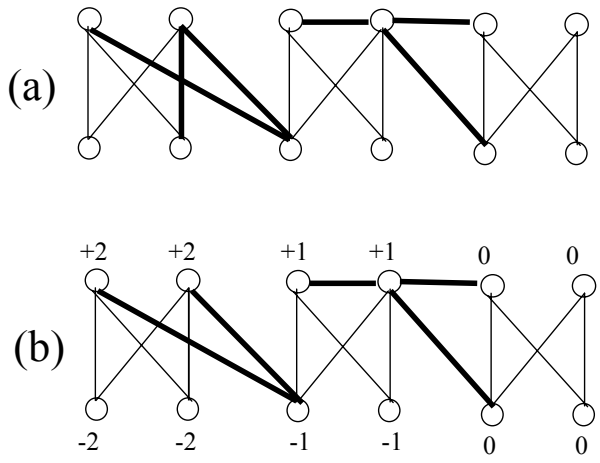


Figure 6.4

6.4. Finding a nonempty algebraic subset of the kernel

By Theorem 6.2 all the algebraic subset of a set F are contained in the kernel of F. Here we will show how to find a closed-flower or a open-flower set contained in K.

Lemma 6.5 If G is a non-bipartite and loopless graph then F contains a closed-flower set A if and only if $G–K$ is bipartite.

Proof (*if)* Suppose A is a closed-flower subset of F. By Theorem 6.2, A is a subset of K. By definition of closed-flower set, $G–A$ is bipartite and, then, $G–K$ must be bipartite too.

(*only if)* Let $G–K$ be non-bipartite and loopless. Let (P, N) be the signed support of a solution of system (6.5). 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, then the subgraph induced by Z is bipartite too because $K\cap[Z, Z]=\varnothing$. Now if P' and N' are two sides of the subgraph induced by Z, then let $A=[P\cup P', P\cup P']$. Since $[N, V(G)–P]=\varnothing$, 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 6.3 (cont.) Fig. 6.5(a) highlights the signed support of a solution of (6.5). Fig. 6.5(b) $[P\cup P', P\cup P']$ is the set of bold edges. Note that $G–K$ is a bipartite graph.

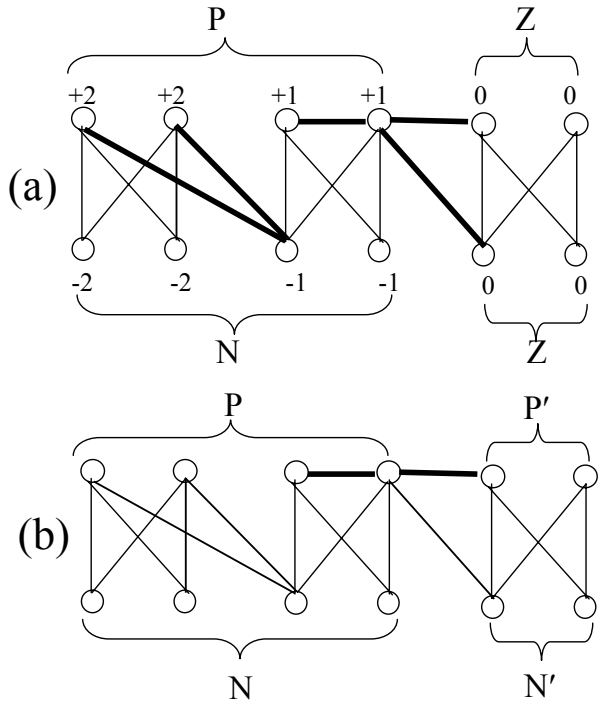


Figure 6.5. (a) a solution of (6.5). (b) **a** closed-flower set contained in K

Lemma 6.6 Let G a connected graph and F a subset of $E(G)$. If K is the kernel of F, then K 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 opposites side 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 (6.1). Let B the subgraph induced by $[P, N]$. If B contains no edges of K then the statement is trivially true since we can take $B^*=B$. Then let $\{B_1,\ldots,B_h\}$ 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 $N_i \subseteq N$, $i=1,\ldots,h$.
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 $B_1=B_{i_1}$. Since $[N, V(G)-P]=\emptyset$ every edge of K attached to N_1 has the other end point attached to some other component B' of $\{B_1,\ldots,B_h\}$. If, for contradiction, an edge of K has both endpoints in the same component B_i clearly it close an even cycle C such that $|C\cap K|=1$, a contradiction of Lemma 6.3.
Let e_{i_1} be one of such edge attached to B_{i_1} and also attached to $B'=B_{i_2}\neq B_{i_1}$. 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 component of $B-K$ and edges e_{i_j} of K (see Fig 6.6(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. 6.6(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 trough B_{i_j} from $u_{i_{j-1}}$ to v_{i_j} and let p_{i_h} be a simple path trough B_{i_h} from $u_{i_{k-1}}$ to v_{i_h}. Clearly, we have obtained an even cycle $C = p_{i_h}, e_{i_h}, p_{i_{h+1}}, e_{i_{h+1}}, \ldots, e_{i_{k-1}}$. 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 6.3, all the edge $e_{i_h}, e_{i_{h+1}},\ldots, e_{i_{k-1}}$ are not in the kernel, a contradiction. ♦

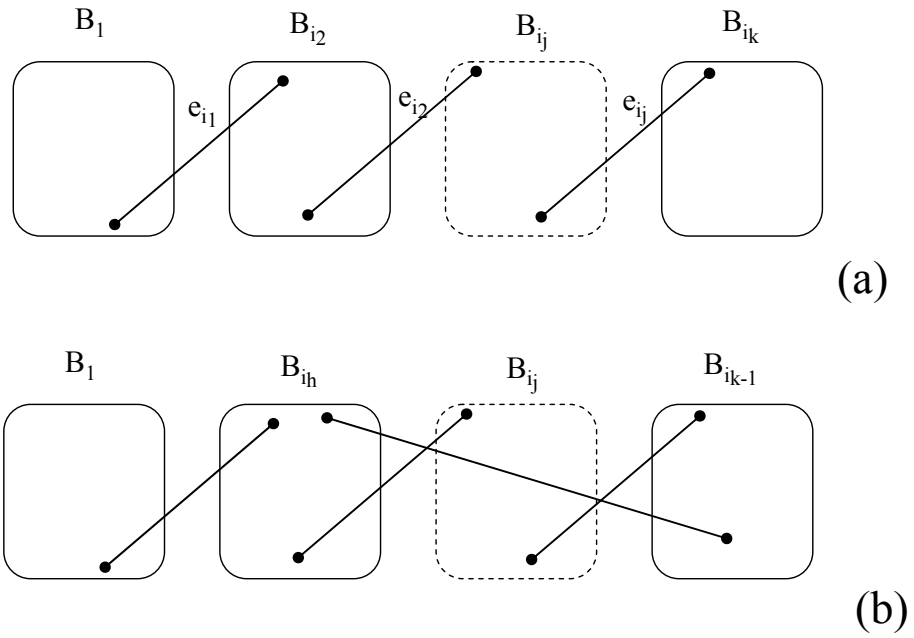


Figure 6.6. The connected bipartite components of B-K and some edges of K connecting those bipartite components.

Example 6.4 (cont.) Clearly the subgraph of $G-K$ induced by $\{u, w, x, z\}$ satisfy the conditions of Lemma 6.6 (see Fig. 6.7).

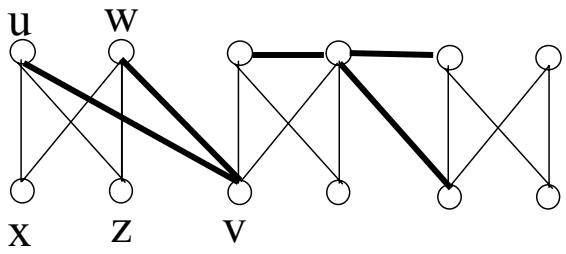


Figure 6.7. $\{(u, v), (w, v)\}$ is an example of a open-flower set

Tu sum up we have the following algorithm to find a nonempty algebraic subset of F.

FIND_ALGEBRAIC_SUBSET
input : Graph G and a subset F of edges of $E(G)$
output : An 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 (6.5);
      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
      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 6.6 are satisfied for A^* **then output** A
    end
  end
end

Theorem 6.3 Algorithm FIND_ALGEBRAIC_SUBSET correctly finds a nonempty algebraic subset of a given edge set.
Proof The correctness follows from Lemma 6.5, Lemma 6.6. ♦

6.5. Computational aspects

Gusfield [27] gave an algorithm to find the kernel in the case of bipartite graph. 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 6.1[27]. All the edges not in any strongly connected component of G' are in the kernel of F.

So the algorithm to find the kernel of F if G is bipartite is:

FIND_KERNEL
input : 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

To demonstrate Lemma 6.4 we give an algorithm that always finds a solution of (6.5). First we see how to find a solution of (6.5), when the graph is bipartite, next we extend the algorithm to non-bipartite graphs. We will use the concept of the *abstract* [23] of a directed graph. If G is a directed graph its abstract H is the directed graph where the node set is the set of strongly connected components of G and the edge set $E(H)=\{(u, v) \mid$ there exist in G at least one directed edge from component u to component $v\}$.

COMPUTE_SUPPORT
input : bipartite connected graph G and a subset F of $E(G)$
output: a solution of (6.5) 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 abstract 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 $N_i \subseteq N$, $i=1,\ldots,h$. Then let

$$c_v = \begin{cases} h-i & \text{if } v \in P_i \\ -h+i & \text{if } v \in N_i \end{cases} \qquad i=1,\ldots,h$$

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

Lemma 6.7 The algorithm COMPUTE_SUPPORT correctly finds a solution of (6.5).
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 6.1, 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 $k(e)=c_u+c_v=h-i-h+j=j-i>0$ as supposed to be. ♦

Algorithms FIND_KERNEL and COMPUTE_SUPPORT apply to bipartite graphs. In the case of non bipartite graphs we can use what is called in Chapter 4 the bipartite transform of G which is a bipartite graph. Then we can apply the above two algorithms to the bipartite transform to find both the kernel and a solution of (6.5). Here the details.

Let $H = (V(H), E(H))$ be the bipartite transform of G and let D be the image of F in H, that is $D=\{ f(e) : e \in F \}$. By Lemma 4.8, if K' is the kernel of D then the kernel K of F is the set $K=\{ e :$

$f(e) \subseteq K'\}$. The following Lemma gives a method to find a solution of (6.5) when G is a non bipartite graph. Since $V(H) = V' \cup V''$ where V' and V'' are copies of $V(G)$ (see Section 4.5) we denote with $v' \in V'$ and $v'' \in V''$ the copies of v in H. With this notation if $e=(u, v) \in E(G)$ is not a loop then the image of e in H is $f(e)=\{ (u'', v'), (u', v'')\}$ and if $e=(u, u)$ is a loop then $f(e)=\{ (u', u'') \}$.

Lemma 8 Let $(g_v)_{v \in V(H)}$ be the coefficients of a solution of (6.5) for the kernel of D. Then

$$c_v = g_{v'} + g_{v''} \qquad\qquad v \in V(G)$$

are the coefficients of a solution of (6.5) for the kernel of F.
Proof. Let K' the kernel of D. If $(u', v'') \in K'$ then, by formula 4.5, $(u'', v') \in K'$, $g_{u'} + g_{v''} > 0$ and $g_{v'} + g_{u''} > 0$. Therefore $(u, v) \in K$ and

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

Also if $(u', v'') \notin K'$ then, by formula 4.5, $(u'', v') \notin K'$, $g_{u'} + g_{v''} = 0$ and $g_{v'} + g_{u''} = 0$. Therefore $(u, v) \notin K$ and

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

To sum up $(c_v)_{v \in V(G)}$ are the coefficients of a solution of (6.5) for the kernel of F. ♦

The strongly connected components and a topological sort of a graph can be found using standard graph algorithms [16] and all takes time linear in the size of the graph. Therefore we have

Remark 6.1 The time complexity of COMPUTE_SUPPORT and FIND_KERNEL is linear in the size of the graph G.

Example 6.5 Consider the graph G of Fig. 6.8. Let F be the set f bold edges. We have that $K=F$. Fig. 6.9 shows the bipartite transform H of G where the edges weighting zero 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 (6.5) are $c_1=+5$, $c_2=-5$, $c_3=+3$, $c_4=-3$, $c_5=+1$ and $c_6=-1$ as is easily checked using algorithm COMPUTE_SUPPORT and Lemma 6.8.
.

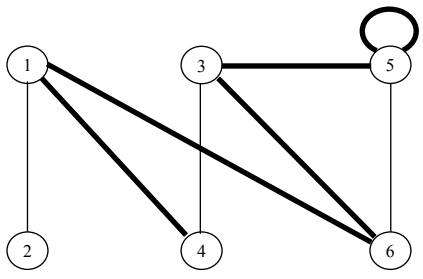


Figure 6.8. A graph and, in bold, a subset F of edges

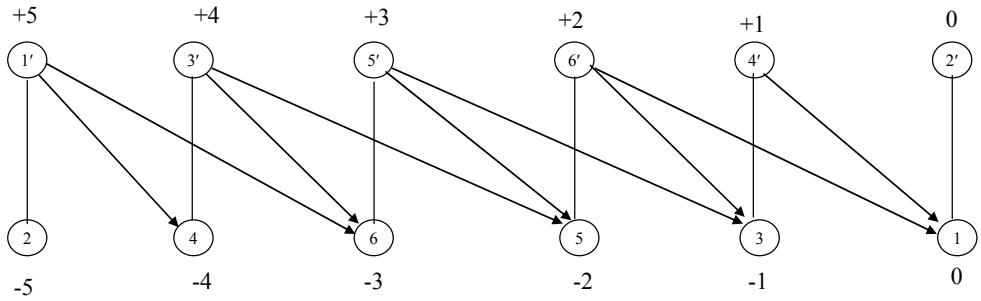


Figure 6.9. The bipartite transform H of the graph of Fig. 6.8. The edges of H in correspondence of F are directed from the upper part to the lower part.

To find if there exist a bipartite component of $G–K$ that satisfy the condition (i) and (ii) of Lemma 6.6, we can proceed as follows.
Let $(B_1,\ldots B_h)$ the bipartite connected components of $G–K$. Suppose we have for each node v of B_i, $i=1,\ldots,h$ the following

$cn(v)$	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 let

$AT_1(B_i)\in\{0,1,2\}$	set to 1 if at least one edge of K is attached to side 1 of B_i set to 2 if an edge of K has both end point attached to side 1 of B_i set to 0 otherwise
$AT_2(B_i)\in\{0,1,2\}$	set to 1 if at least one edge of K is attached to side 2 of B_i set to 2 if an edge of K has both end point attached to side 2 of B_i set to 0 otherwise

Then 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 in 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 6.6 are satisfied. Since all those tasks takes time linear to the size of the graph and for Remark 6.1, we have the following

Theorem 6.4 Algorithm FIND_ALGEBRAIC_SUBSET has time complexity linear in the size of the graph.

If H is a graph then we can use the algorithm FIND_ALGEBRAIC_SUBSET to find a maximally contained algebraic subset of F as in the following algorithm :

MAXIMALLY_CONTAINED
input :a graph G and a subset F of edges
output :a maximally contained algebraic subset of F

begin
$M:=\varnothing$;
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_CONTAINED is at most $|F|$ times the time complexity of FIND_ALGEBRAIC_SUBSET. Therefore MAXIMALLY_CONTAINED takes at most quadratic time in the size of **H**.

Chapter 7

Minimal invariant sets in a vertex-weighted graph

7.1. Introduction

Let G be a graph with vertex set $V(G)$ and an edge set $E(G)$, which contains no isolated vertices and no parallel edges but loops are allowed. A (*vertex*) *weighting* of G is a $|V(G)|$-dimensional vector of real numbers. A weighting $\boldsymbol{a}$ is an *admissible* R-*weighting* (or an *admissible* R_+-*weighting*) if there exists at least one real-valued (a nonnegative real-valued, respectively) solution of the following system of $|V(G)|$ linear equations

$$\boldsymbol{G}\,\boldsymbol{x} = \boldsymbol{a} \qquad (7.1)$$

where $\boldsymbol{G}$ is the (vertex-edge) incidence matrix of G. If $\boldsymbol{a}$ is an admissible R-weighting (or an admissible R_+-weighting), then real-valued (nonnegative real-valued, respectively) solutions of system (7.1) are called (*edge*) R-*labellings* (R_+-*labellings*, respectively) *constrained* by $\boldsymbol{a}$. In the theory of magic graphs [21, 32], a weighting $\boldsymbol{a}$ is called an "indexing vector" or a "vertex labelling", and a labelling constrained by $\boldsymbol{a}$ is also called a labelling "induced" by or "compatible" with $\boldsymbol{a}$. It is well-known [21, 32] that an R-weighting $\boldsymbol{a}$ is always admissible unless G is a bipartite graph and $\sum_{v\in U} a_v \neq \sum_{v\in W} a_v$ where (U, W) is a bipartition of G, and the same holds if $\boldsymbol{a}$ is an R_+-weighting of G [27, 41].

Given an admissible R-weighting (or an admissible R_+-weighting) $\boldsymbol{a}$ of G, a subset S of $E(G)$ is an R-*invariant set* (an R_+-*invariant set*, respectively) if either $S = \emptyset$ or $\sum_{e\in S} l(e) = \sum_{e\in S} l'(e)$ for every two R-labellings (R_+-labellings, respectively) $\boldsymbol{l}$ and $\boldsymbol{l'}$ of G constrained by $\boldsymbol{a}$; if this is the case, the *value* of S is taken to be 0 if $S = \emptyset$, and to be the sum $\sum_{e\in S} l(e)$ where $\boldsymbol{l}$ is any R-labelling (any R_+-labelling, respectively) constrained by $\boldsymbol{a}$, otherwise. If the singleton $\{e\}$ is an R-invariant set (or an R_+-invariant set), we call e an R-*invariant edge* (an R_+-*invariant edge*, respectively). A nonempty R-invariant set (or R_+-invariant set) is *minimal* if none of its nonempty proper subsets is an R-invariant set (an R_+-invariant set, respectively). It is easy to see that the family of R-invariant sets as well as the family of R_+-invariant sets are closed under disjoint union and proper difference. So, by the closure under proper difference and disjoint union, a nonempty edge set is an invariant set if and only if it is the disjoint union of one or more minimal invariant sets.

In this Chapter we address the problem of finding a characterisation of minimal R-invariant sets (or R_+-invariant sets) for a given admissible R-weighting (or R_+-weighting) of a graph. The interest in (minimal) invariant sets was first motivated by the security issues connected with the publication of statistical data [18, 27, 33, 34, 35, 36, 40, 42, 43, 45, 46, 49, 50]. Recently, invariant sets have found applications in sum-query processing [51].

At the present, the only known result about minimal algebraic sets is a graphical characterisation of minimal invariant sets in a bipartite weighted graph [36, 39]. In this paper we first give a polynomial test for recognising invariant sets and, next, state a graphical characterisation of minimal invariant sets in a nonbipartite weighted graph.

7.2. Definitions

In this section, we recall some more-or-less standard definitions on graphs, which will be used in the sequel. A graph G is defined by a nonempty finite set, denoted by $V(G)$ and called the set of its *vertices*, and by a (possibly empty) set, denoted by $E(G)$ and called the set of its *edges*, where each edge is an unordered couple of vertices which are called its *endpoints*. A *trivial* graph is a graph G with $E(G) = \emptyset$. A *subgraph* of graph G is a graph H with $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. An edge in $E(G)\backslash E(H)$ having at least one endpoint in $V(H)$ is said to be *attached* to H, and the vertices of H that are endpoints of edges attached to H are called the *attachments* of H. The subgraph of G *induced* by a nonempty subset U of $V(G)$, denoted by $G(U)$, is the graph with vertex set U whose edges are the edges of G whose endpoints belong to U.

An edge with identical endpoints is called a *loop*, and an edge with distinct endpoints a *link*. The *star* of a vertex v, denoted by $star(v)$, is the set of edges incident with v. An *independent set* is a nonempty set vertices whose stars are pairwise disjoint. A *starset* is the (disjoint) union of the stars of vertices from an independent set; if U is an independet set, then by $starset(U)$ we denote the starset over U, that is, $starset(U) = \cup_{v \in U} star(v)$. A *complete graph* is a loopless graph with an edge for each pair of distinct vertices.

A *walk* is a sequence $(v_0, e_1, v_1, \ldots, e_k, v_k)$ whose terms are alternately vertices and edges, such that if $k > 0$ then, for $1 \leq i \leq k$, the endpoints of e_i are v_{i-1} and v_i; the vertices v_0 and v_k are its *start* and *end* vertices, respectively, and k is its *length*. A *cycle* is a walk whose start and end vertices are the same. A walk is a *path* if no vertex appears on it more than once. A cycle is a *circuit* if no vertex, other than the start-end vertex, appears more than once. Accordingly, a loop defines a circuit of length 1. A graph is *connected* if any two vertices are joined by a path. A *component* of a graph is a maximal connected induced subgraph.

A graph G is *bipartite* if G contains no odd cycles, that is, if and only if either G is a trivial graph or there is a bipartition (U, W) of $V(G)$ such that each edge of G has one endpoint in U and the other endpoint in W. The components of a graph that are bipartite will be referred to as its *bipartite components*. A nontrivial bipartite graph with bipartition (U, W) is a *complete bipartite graph* if there is an edge for each pair of vertices u and v with $u \in U$ and $v \in W$.

For subsets U and W of $V(G)$, by $[U, W]$ we denote the (possibly empty) set of edges of G with one endpoint in U and the other endpoint in W. According to notation used in [15], if U is a nonempty subset of $V(G)$ and W is the complement of U, then $D(U)$ is used for $[U, W]$ and $B(U)$ is used for $E(G)\backslash D(U)$. Accordingly, if G is a nontrivial bipartite graph with bipartition (U, W), then $E(G) = D(U) = D(W)$ and, if G is an arbitrary graph, then $E(G) = B(V(G))$. A nonempty edge set S is a *cut set* (an "edge cut" in [7]) if $S = D(U)$ for some nonempty proper subset of $V(G)$. A *bond* [7] is a minimal cut set. An edge e of G is a *bridge* if the singleton $\{e\}$ is a bond of G.

A nonempty edge set S is a *bip set* if $S = B(U)$ for some nonempty proper subset of $V(G)$. Note that, since $E\backslash B(U) = D(U)$, the graph $G–S$ is bipartite. If e is an edge of a nonbipartite and connected graph G such that the singleton $\{e\}$ is a bip set of G, then e is a *handle* on $G–e$ [32]; in other words, an edge of a nonbipartite and connected graph is a handle if and only if $G–e$ is bipartite. Note that there is at most one handle that is a loop, and there may be one or more handles that are links but, if this is the case, then G is loopless.

Let U be a nonempty proper subset of $V(G)$ such that $D(U) \neq \emptyset$. The union of the cut set $D(U)$ with a bip set of $G(U)$ is called a *cut-bip set*.

We now introduce special versions of cut sets, bip sets and cut-bip sets.

We say that a cut set $D(U)$ is *simple* if $G(U)$ is bipartite and either $G(U)$ is a trivial graph or there is a bipartition (P, N) of $G(U)$ such that the attachments of $G(U)$ are all in P (see Figure 7.1); if this is the case, we call $G(U)$ and the components of the subgraph of G induced by $V(G)\backslash U$ the *root* and the *branches* of the simple cut set, respectively.

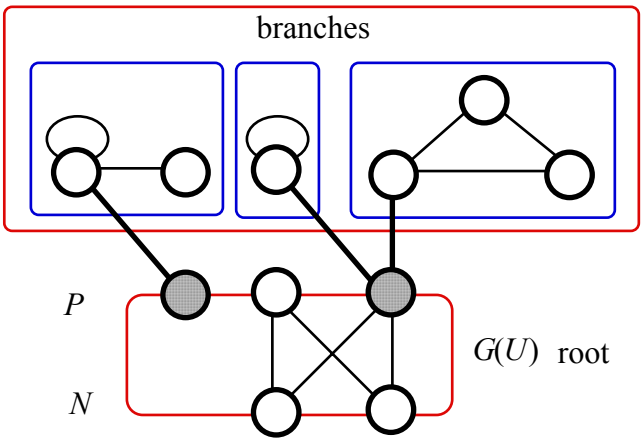


Figure 7.1. A simple cut set of a nonbipartite graph.

Remark 7.1 If the root of a simple cut set is a trivial graph, then the simple cut set is a starset given by the disjoint union of the stars of vertices of its root; otherwise (see Figure 7.1), the simple cut set is the proper difference of two starsets, that is, $starset(P) \setminus starset(N)$.

We say that a bip set $B(U)$ of G is *simple* if either $G–B(U)$ is a trivial graph or there is a bipartition (P, N) of $G–B(U)$ such that the attachments of $G–B(U)$ are all in P, that is, $B(U) = [P, P]$. A simple bip set $B(U)$ is a *simple loop bip set* if $B(U)$ is a set of loops (see Figure 7.2).

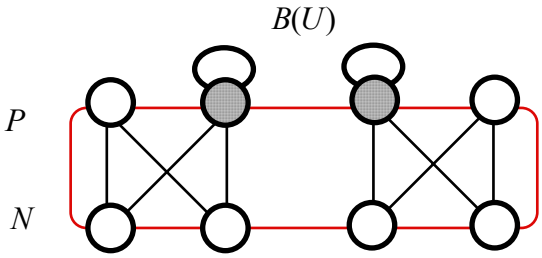


Figure 7.2. A simple loop bip set.

Remark 7.2 Let $B(U)$ be a simple loop bip set. If $G–B(U)$ is a trivial graph, then $E(G) = B(U)$ and the simple bip set is a starset given by the union of the stars of vertices of G; otherwise (see Figure 2), the simple loop bip set is the proper difference of two starsets, that is, $starset(P) \setminus starset(N)$.

A simple bip set $B(U)$ is a *simple link bip set* if $B(U)$ is a set of links (see Figure 7.3).

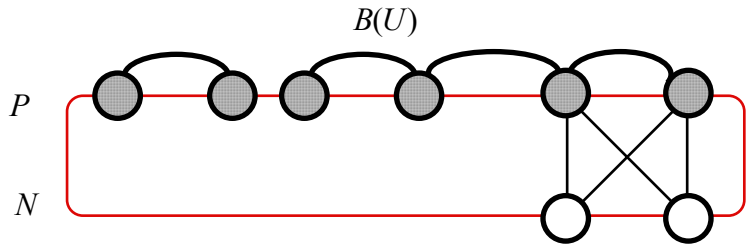


Figure 7.3. A simple link bip set of a nonbipartite graph.

Let U be a nonempty proper subset of $V(G)$ such that $D(U) \neq \emptyset$, and let B be a bipset of $G(U)$. We say that the cut-bip set $D(U) \cup B$ of G is *simple* if either $G(U)–B$ is a trivial graph or there is a bipartition (P, N) of $G(U)–B$ such that the attachments of $G(U)–B$ are all in P. A simple cut-bip set is a *simple cut-loop set* if B is a set of loops (see Figure 7.4); if this is the case, we call the subgraph $G(U)–B$ and the components of the subgraph of G induced by $V(G)\backslash U$ the *root* and the *branches* of the simple cut-loop set, respectively.

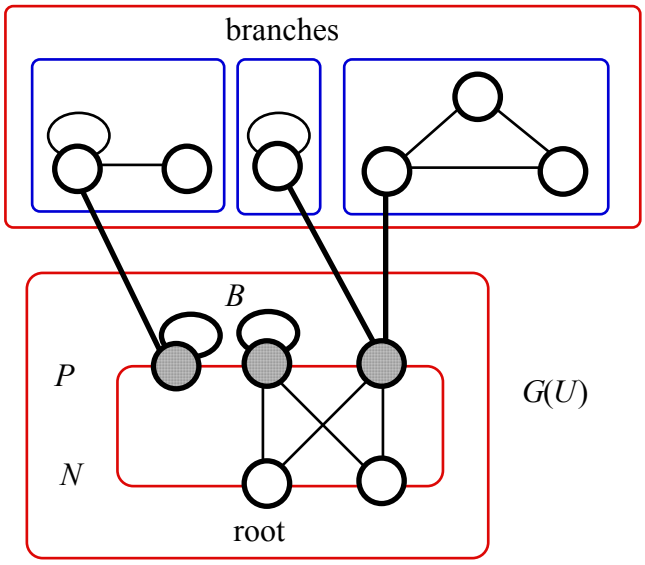


Figure 7.4. A simple cut-loop set.

Remark 7.3 If the root of a simple cut-loop set is a trivial graph, then the simple cut-loop set is a starset given by the disjoint union of the stars of vertices of its root; otherwise (see Figure 7.4), the simple cut-loop set is the proper difference of two starsets, that is, *starset*(P) \ *starset*(N).

7.3. Invariance tests

Let S be a subset of $E(G)$. The binary vector $\boldsymbol{s} = [s(e)]_{e \in E(G)}$ with

$$s(e) = \begin{cases} 1 & \text{if } e \in S \\ 0 & \text{else} \end{cases}$$

will be referred to as the *characteristic vector* of S in G. The following two results characterise R-invariant sets and R_+-invariant sets in terms of their characteristic vectors.

Proposition 7.1 [12] Let $\boldsymbol{a}$ be an admissible R-weighting of a graph G. A subset S of $E(G)$ is an R-invariant set for $\boldsymbol{a}$ if and only if the characteristic vector of S is a linear combination of rows of the incidence matrix of G.

Proposition 7.2 [46] Let $\boldsymbol{a}$ be an admissible R_+-weighting of a graph G, and let Z be the set of the R_+-invariant edges with value zero. A subset S of $E(G)$ is an R_+-invariant set if and only if the characteristic vector of $S\backslash Z$ in $G–Z$ is a linear combination of rows of the incidence matrix of $G–Z$.

According to the terminology introduced in [39], a subset of $E(G)$ whose characteristic vector belongs to the *row space* of the incidence matrix of G (i.e., the linear space spanned by the rows of the incidence matrix of G) is called an *algebraic* (*edge*) *set* of G. Thus, Proposition 7.1 states that an edge set S is an R-invariant set if and only if S is an algebraic set of G, and Proposition 7.2 states that an edge set S is an R_+-invariant set if and only if $S\backslash Z$ is an algebraic set of $G–Z$.

Propositions 7.1 and 7.2 lead to efficient tests for recognising invariant sets and computing their values. We first consider R-invariant sets and, then, R_+-invariant sets.

(R-invariant sets) Let S be a subset of $E(G)$. By the very definition, one has that S is an algebraic set of G if and only if the following equation system

$$\begin{cases} c_u + c_v = s(u,v) & \text{for each link } (u,v) \\ c_w = s(w,w) & \text{for each loop } (w,w) \end{cases} \tag{7.2}$$

has a real-valued solution $\boldsymbol{c} = [c_v]_{v \in V(G)}$. The following result states that one can recognise an algebraic set in linear time.

Proposition 7.3 [46] A solution (if any) of system (7.2) can be found in time linear in the size of G.

By Propositions 7.1 and 7.3, one can decide whether or not a subset S of $E(G)$ is an R-invariant set in time linear in the size of G. Moreover, if S is an R-invariant set then, given a solution $\boldsymbol{c}$ of system (7.2), using basic linear algebra one can compute the value of S as

$$\sum_{v \in V(G)} c_v \, a_v$$

and, hence, in time linear in the size of $\boldsymbol{a}$.

(R_+-invariant sets) The set Z of R_+-invariant edges with value zero can be found in time linear in the size of G [41]. So, by Propositions 7.2 and 7.3, the problem of deciding whether or not a subset S of $E(G)$ is an R_+-invariant set has the same complexity as the problem of finding an R_+-labelling of G constrained by $\boldsymbol{a}$. If G is bipartite, then Gusfield [27] proved that R_+-labellings of G constrained by $\boldsymbol{a}$ correspond to maximum flows of a network which can be obtained from G and $\boldsymbol{a}$ in linear time, so that finding an R_+-labelling of G constrained by $\boldsymbol{a}$ requires cubic time [3]. If G is not bipartite, two of the authors [41] proved that R_+-labellings of G constrained by $\boldsymbol{a}$ correspond to R_+-labellings of the so-called "bipartite transform" of G, which is a bipartite graph that can be obtained from G in linear time, so that finding an R_+-labelling of G constrained by $\boldsymbol{a}$ still requires cubic time. To sum up, using Propositions 7.2 and 7.3, one can decide whether or not a subset S of $E(G)$ is an R_+-invariant set in cubic time. Moreover, if S is an R_+-invariant set then, given a solution $\boldsymbol{c}$ of system (7.2) for the set $S\backslash Z$ of edges of the graph $G–Z$, one can compute the value of S as

$$\sum_{v \in V(G–Z)} c_v \, a_v$$

and, hence, in time linear in the size of $\boldsymbol{a}$.

7.4. Algebraic sets

By Propositions 7.1 and 7.2, a characterisation of invariant sets passes through a characterisation of algebraic sets. Like the family of invariant sets of a weighted graph, also the family of algebraic sets of a graph is closed under disjoint union and proper difference [9, 10, 52]. Trivial examples of an algebraic set of a graph G are the empty subset of $E(G)$ and the star of each vertex of G. By the closure under disjoint union and proper difference, one has that every starset and the proper difference of two starsets are algebraic sets.

Lemma 7.1 Simple cut sets, simple loop bip sets, simple link bip sets and simple cut-loop sets are all algebraic sets.
Proof. By Remarks 7.1, 7.2 and 7.3, simple cut sets, simple loop bip sets and simple cut-loop sets are all algebraic sets. What remains to prove is that every simple link bip set is an algebraic set. Let S be a simple link bip set of a graph G. Of course G is loopless. If $S = E(G)$, then the characteristic vector $\boldsymbol{s}$ of S in G can be written as

$$\frac{1}{2} \sum_{v \in V(G)} \boldsymbol{g}_v \ ;$$

otherwise, $\boldsymbol{s}$ can be written as (see Figure 7.2)

$$\frac{1}{2} \sum_{v \in P} \boldsymbol{g}_v - \frac{1}{2} \sum_{v \in N} \boldsymbol{g}_v \ .$$

So, in both cases, S is an algebraic set of G. □

Remark 7.4 If G is a loopless graph, then the family of algebraic sets of G is also closed under complementation since $E(G)$ is an algebraic set. On the other hand, if G is a connected graph with loops, then it is easy to see that $E(G)$ is an algebraic set if and only if the set of loops of G is a simple (loop) bip set of G, so that the family of algebraic sets of G is closed under complementation if and only if the set of loops of G is a simple (loop) bip set of G.

We now state some general properties of an algebraic set of a graph G. The first property we now state follows from the following two facts involving the incidence matrix $\boldsymbol{G}$ of G.

Fact 7.1 [15] The rank of $\boldsymbol{G}$ is $|V(G)|–p$, where p is the number of bipartite components of G.

Let $\boldsymbol{r} = [r(e)]_{e \in E(G)}$ be a vector of the row space of $\boldsymbol{G}$. The *support* of $\boldsymbol{r}$ is the edge set $||\boldsymbol{r}|| = \{e \in E(G): r(e) \neq 0\}$, and $\boldsymbol{R}$ is a *minimal vector* of the row space of $\boldsymbol{G}$ is if $\boldsymbol{r} \neq \boldsymbol{0}$ and the support of no nonzero vector of the row space of $\boldsymbol{G}$ is a proper subset of $||\boldsymbol{r}||$. By Fact 1 one has

Fact 7.2 [15] The supports of minimal vectors of the row space of $\boldsymbol{G}$ are exactly the minimal edge sets whose removal from G creates one more bipartite component.

Theorem 7.1 The removal of a nonempty algebraic set creates at least one more bipartite component.
Proof. Let S be a nonempty algebraic set, and $\boldsymbol{s}$ its characteristic vector. Since $\boldsymbol{s}$ is a vector of the row space of $\boldsymbol{G}$, S contains the support of a minimal vector of the row space of $\boldsymbol{G}$. Then, the statement follows from Fact 7.2. □

Let S be a subset of $E(G)$. Of course, S is an algebraic set of G if and only if, for each component G' of G, the intersection of S with $E(G')$ is algebraic too. Therefore, without loss of generality, henceforth we assume that G is connected. Then, the rank of $\boldsymbol{G}$ is equal to either $|V(G)|-1$ or $|V(G)|$ depending on whether or not G is bipartite. It follows that, if the constant term of system (7.2) is the characteristic vector of an algebraic set of G, then the equation system has either ∞^1 solutions or exactly one solution depending on whether or not G is bipartite. Let S be an algebraic set of G with characteristic vector $\boldsymbol{s}$, and let $\boldsymbol{c}$ be a solution of system (7.2). Let (P, N) be the *signed support* of $\boldsymbol{c}$; that is,

$$P = \{v \in V(G): c_v > 0\} \qquad N = \{v \in V: c_v < 0\}.$$

Since $c_u + c_v \in \{0, 1\}$ for each link (u, v), and $c_w \in \{0, 1\}$ for each loop (w, w), it is easily seen that

Fact 7.3 There is no edge (u, v) of G with $u \in N$ and $v \in V(G)\backslash P$.

Fact 7.4 Each edge (u, v) of G with $u \in P$ and $v \in V(G)\backslash N$ belongs to S.

Moreover, for every two vertices u and v of G, if $(v_1, \ldots, v_k)$ is a path with start-vertex u and end-vertex v then, since $\boldsymbol{c}$ is a solution of system (2), one has

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

so that, since $\boldsymbol{s}$ is a binary vector, one has

$$c_v = \alpha + (-1)^{k-1}\, c_u \qquad (7.3)$$

where α is an integer. Therefore, the following holds (see also lemma 6.1).

Fact 7.5 For every two vertices u and v of G, either both c_u and c_v are integers or neither c_u nor c_v is an integer.

Lemma 7.2 Let S be an algebraic set of a connected graph G with characteristic vector $\boldsymbol{s}$. If G is bipartite, then there exists an integral solution of system (7.2). If G is not bipartite, then the solution of system (7.2) is integral whenever G contains a loop; but, if G is not bipartite and loopless, then the solution of system (7.2) is either integral or half-integral.
Proof. Let us distinguish two cases depending on whether or not G is bipartite.

Case 1: G is bipartite. Then, one component, say c_u, of a solution $\boldsymbol{c}$ of system (7.2) can be chosen arbitrarily, so that c_u can be taken to be 0 and, then, by (7.3) all the remaining components of $\boldsymbol{c}$ will be integers.

Case 2: G is not bipartite. Then, there is exactly one solution of system (7.2), say $\boldsymbol{c}$. Now, if G contains a loop, say (w, w), then c_w is equal to either 1 or 0 depending on whether the loop is or is not in S, so that by Fact 7.5 all the remaining components of $\boldsymbol{c}$ will be integers. If G is loopless and v is the start-end vertex of an odd circuit, then by (7.3) one has $c_v = \frac{\alpha}{2}$ so that c_v is either an integer or a half-integer (that is, a fraction having an odd integer as a numerator and 2 as a denominator); in the former case, by Fact 7.5 each component of $\boldsymbol{c}$ is an integer, and in the latter case each component of $\boldsymbol{c}$ is a half-integer. □

Lemma 7.3 Let S be an algebraic set of a connected graph G with characteristic vector $\boldsymbol{s}$. Let $\boldsymbol{c}$ be a solution of system (2), and (P, N) the signed support of $\boldsymbol{c}$. If $[P, P] \neq \emptyset$, then either

(*i*) $[P, P]$ is a set of loops, or

(*ii*) $[P, P]$ is a set of links, G is loopless and $V(G) = P \cup N$.

Proof see Lemma 6.1. □

With the notation above, consider the set S' of edges with one endpoint in P and the orther endpoint in $V(G)\backslash N$, that is, $S' = [P, V(G)\backslash N]$. By Fact 7.4, S' is a subset of S. Moreover, if G is bipartite, then S' may be empty; but, if G is not bipartite, then S' cannot be empty for, otherwise, $E(G) = [P, N]$ and, hence, G would be bipartite.

Lemma 7.4 Let G be a nonbipartite and connected graph, and S a nonempty algebraic set. Let $S' = [P, V(G)\backslash N]$ and $H = G(P \cup N) - [P, P]$. Then

(*i*) if $V(G) = P \cup N$, then S' is a simple loop bip set or a simple link bip set of G;

(*ii*) if $V(G) \neq P \cup N$, then S' is either a simple cut set or a simple cut-loop set (both with root H) depending on whether or not $[P, P]$ is an empty set, and

(*iii*) S' is a nonempty algebraic subset of S.

Proof By Fact 7.3, $[N, N] = \emptyset$ so that H is bipartite with bipartition (P, N). Again, by Fact 7.3, $[N, Z] = \emptyset$ so that the attachments of H are all in P. Then, statements (*i*) and (*ii*) follow from parts (*i*) and (*ii*) of Lemma 7.3. Finally, as noted above, S' is a nonempty subset of S so that statement (*iii*) follows from statements (*i*) and (*ii*) and from Lemma 7.1. □

7.5. Minimal algebraic sets

A nonempty algebraic set of a graph G is *minimal* if none of its nonempty proper subsets is an algebraic set of G, and an edge e is *algebraic* if $\{e\}$ is an algebraic set. Note that, by the closure under disjoint union and proper difference, one has then that every nonempty algebraic set of a graph is a disjoint union of minimal algebraic sets. From Propositions 7.1 and 7.2 it follows that

Corollary 7.1 Let $\boldsymbol{a}$ be an admissible R-weighting of a graph G. A nonempty subset S of $E(G)$ is a minimal R-invariant set if and only if S is a minimal algebraic set of G.

Corollary 7.2 Let $\boldsymbol{a}$ be an admissible R-weighting of a graph G. A nonempty subset S of $E(G)$ is a minimal R_+-invariant set if and only if either $S = \{e\}$ for some edge e in Z or $S\backslash Z$ is a minimal algebraic set of $G–Z$.

By Corollaries 7.1 and 7.2, a characterisation of minimal invariant sets passes through a characterisation of minimal algebraic sets. The following is a graphical characterisation of minimal algebraic sets of a bipartite graph.

Theorem 7.2 [36, 40] The minimal algebraic sets of a bipartite graph are exactly its simple bonds. The minimal algebraic sets of a complete bipartite graph are exactly the stars of its vertices.

Corollary 7.3 The family of algebraic sets of a bipartite graph is the smallest family containing stars, and is closed under disjoint union and proper difference. The family of algebraic sets of a complete bipartite graph is formed by the empty set and by starsets.

Proof. Let G be a bipartite graph. Since a simple bond can be obtained as a proper difference of two distinct starsets (see Remark 7.1) and every nonempty algebraic set is the disjoint union of minimal algebraic sets, the first statementent follows from Theorem 7.2. If G is a complete bipartite graph then, by Theorem 7.2, the disjoint union of minimal algebraic sets is always a starset, and the proper difference of two starsets is either the empty set or a starset. □

By Theorem 7.2, one also has that an edge of a bipartite graph is algebraic if and only if it is a bridge. At the present, the only known result about minimal algebraic sets of a nonbipartite and connected graph is the following characterisation of algebraic edges.

Theorem 7.3 [41] An edge of a nonbipartite and connected graph is algebraic if and only if it is a handle or a simple bridge.

We shall state a characterisation (see Theorem 7.5 below) of minimal algebraic sets of a nonbipartite and connected graph, which subsumes Theorem 7.3.

Theorem 7.4 Let G be a nonbipartite and connected graph, and S a minimal algebraic set of G with characteristic vector $\boldsymbol{s}$. Let (P, N) be the signed support of the solution of system (7.2), and $H = G(P \cup N) - [P, P]$. Then, one has that

(a) if $V(G) = P \cup N$, then S is a simple loop bip set or a simple link bip set of G and, for each component B of H, there is an edge in S having both endpoints in B;

(b) if $V(G) \neq P \cup N$, then S is either a simple cut set or a simple cut-loop set (both with root H) depending on whether or not $[P, P]$ is an empty set, and

($b1$) H is connected,

($b2$) for each bipartite branch C of S, the set of edges attached to C is not a simple cut set, and

($b3$) if S is a simple cut set, then there is at least one nonbipartite branch of S.

Proof. First of all, observe that, by part (iii) of Lemma 7.4 and by the minimality of S, one has that $S = [P, V(G) \backslash N]$.

(a) By part (i) of Lemma 7.4, if $V(G) = P \cup N$, then S is either a simple loop bip set or a simple link bip set of G. If S is a simple loop bip set, then $H = G - S$ and, since G is connected, H is connected too so that it is trivially true that there is an edge in S having both endpoints in each component of H. Consider now the case that S is a simple link bip set. Suppose, by contradiction, that there is a component B of H such that no edge in S has both endpoints in B (see Figure 7.5).

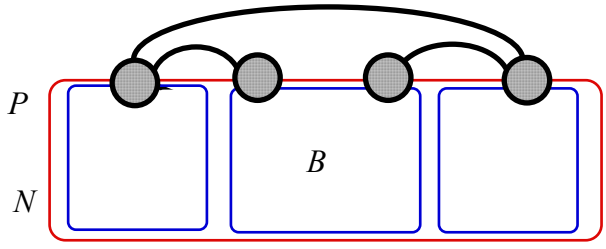


Figure 7.5

Since G is not bipartite and, by Lemma 7.3, is loopless, the set $S' = D[V(B)]$ is a nonempty proper subset of S. Moreover, S' is a simple cut set and, by Lemma 7.1, S' is an algebraic set, which contradicts the minimality of S.

(b) By part (ii) of Lemma 7.4, if $V(G) \neq P \cup N$, then S is either a simple cut set or a simple cut-loop set of G (both with root H) depending on whether or not $[P, P] = \emptyset$.

($b1$) Suppose, by contradiction, that H is not connected and let B be a component of H (see Figure 7.6).

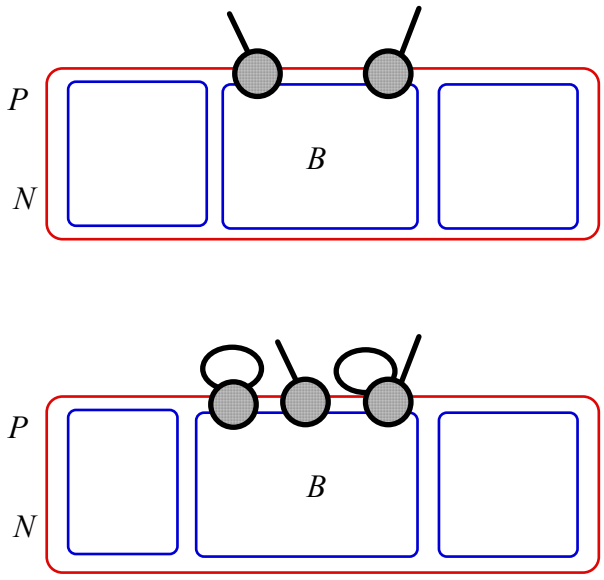


Figure 7.6

Then, the set $S' = D[V(B)]$ is a nonempty proper subset of S and is either a simple cut set or a simple cut-loop set. By Lemma 7.1, S' is an algebraic set, which contradicts the minimality of S.

($b2$) Suppose, by contradiction, that there is a bipartite branch C of S such that the set S' of edges attached to C is a simple cut set (see Figure 7.7).

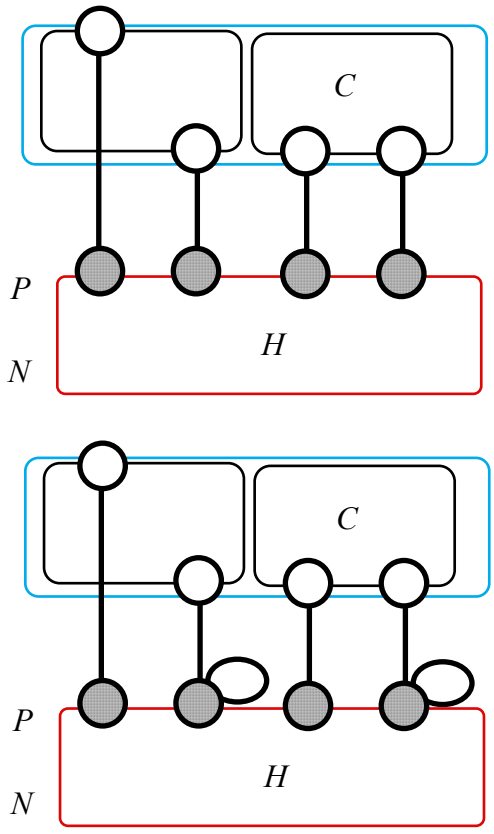


Figure 7.7

By Lemma 7.1, S' is an algebraic set. Since G is not bipartite, either C is the unique branch of S and S is a cut-loop set, or C is not the unique branch of S; in both cases, S' is a nonempty proper subset of S, which contradicts the minimality of S.

($b3$) Suppose, by contradiction, that S is a simple cut set and that the subgraph K of G induced by $V(G) \setminus (P \cup N)$ is bipartite with bipartition (U, W). By ($b2$), the set of edges attached to no component of K is a simple cut set (see Figure 7.8).

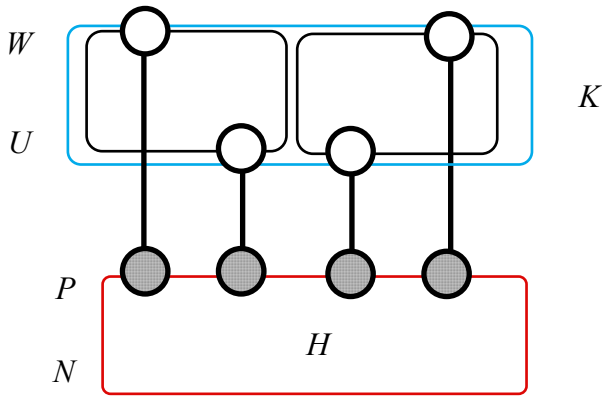


Figure 7.8

So, neither $[P, U]$ nor $[P, W]$ is an empty set. Then, the set $S' = [P, U]$ is a nonempty proper subset of S and is a simple link bip set of G; so, by Lemma 7.1, S' is an algebraic set, which contradicts the minimality of S. □

We shall show that the converse of Theorem 7.4 also holds. We begin by proving the statement for a simple loop bip set or a simple link bip set.

Lemma 7.5 Let G be a nonbipartite and connected graph, and let S be either a simple loop bip set or a simple link bip set. Then, S is a minimal algebraic set of G if, for each component B of $G–S$, there is an edge in S having both endpoints in B.
Proof. By Lemma 7.1, S is an algebraic set. Let us distinguish two cases depending on whether S is a simple loop bip set or a simple link bip set. In the former case, S is the loop set of G so that, if S' is a nonempty proper subset of S, then $G–S'$ is connected and nonbipartite and, by Theorem 7.1, S' is not an algebraic set. Let S be a simple link bip set and let S' be a minimal algebraic set contained in S. By Theorem 7.4, S' may be either a simple link bip set or a simple cut set. We first prove that (*i*) S' cannot be a simple cut set and, next, (*ii*) $S' = S$.

(*i*). Suppose by contradiction that S' is a simple cut set with root H'. Then, H' is an induced subgraph of G and, by part (b1) of Theorem 7.4, H' is connected. Since S' is a subset of S, there exists a component B of $G–S$ that is a subgraph of H'. By hypothesis, there is an edge e in S whose endpoints are both in B, which excludes that B is a one-point graph. Since H' is an induced subgraph of G, e is an edge of H' so that H' is not bipartite, which contradicts the fact that H' is the root of a simple cut set.

(*ii*). Let S' be a simple link bip set. Since $G–S'$ is bipartite, each edge in S having both endpoints in the same component of $G–S$ must belong to S' (for, otherwise, $G–S'$ would not be bipartite). We now prove that also each edge in S whose endpoints are in distinct components of $G–S$ must belong to S', which completes the proof that $S' = S$. Suppose, by contradiction, that there is an edge (u, v) in $S\backslash S'$ such that u and v are in two distinct components of $G–S$, say B_1 and B_2, respectively. By hypothesis, there is an edge (u_1, v_1) in S having both endpoints in B_1 and there is an edge (u_2, v_2) in S having both endpoints in B_2 (see Figure 7.9).

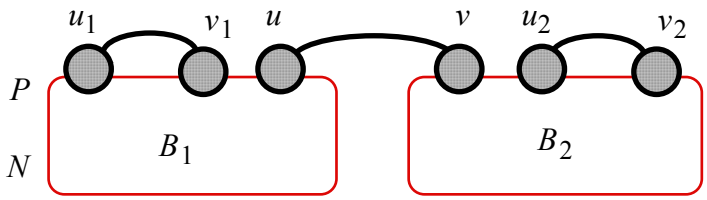


Figure 7.9

Since (u, v) does not belongs to S' and since S' is a subset of S, there is a component B' of $G–S'$ containing both B_1 and B_2. Since (u_1, v_1) and (u_2, v_2) are both in S' (see above), the vertices u_1 and u_2 are attachments of B' which are joined by an odd path in B' (see Figure 7.10),

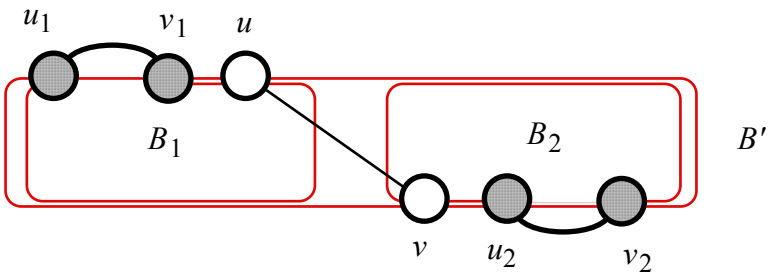


Figure 7.10

which contradicts the fact that S' is a simple (link) bip set of G. □

The next lemma states a sufficient condition for a simple cut set or a simple cut-loop set to be a minimal algebraic set.

Lemma 7.6 Let G be a nonbipartite and connected graph, and let S be either a simple cut set or a simple cut-loop set. Then, S is a minimal algebraic set of G if

(*i*) the root of S is connected, and

(*ii*) for each bipartite branch C of S, the set of edges attached to C is not a simple cut set, and

(*iii*) if S is a simple cut set, then there is at least one nonbipartite branch of S.

Proof. By Lemma 7.1, S is an algebraic set. Let S' be a minimal algebraic set that is contained in S. We shall show that $S' = S$. Let us distinguish two cases depending on whether S is a simple cut set or a simple cut-loop set.

Case 1: Let S be a simple cut set with root H. By Theorem 7.4, S' may be either a simple link bip set or a simple cut set. But, S' cannot be a simple link bip set because, by (*iii*), $G–S$ is not bipartite and, since S' is a subset of S, $G–S'$ is not bipartite. Therefore, S' is a simple cut set. Let H' be the root of S'. In order to prove that $S' = S$, it is sufficient to show that $H' = H$. First of all, note that H' is a connected by part (b1) of Theorem 7.4 and H is connected by (*i*). Since S' is a subset of S, H' contains at least one component of $G–S$. Since H' is bipartite, H' can contain only bipartite components of $G–S$. Suppose, by contradiction, that H' contains a bipartite branch C of S. Since the set of edges attached to C is not a simple cut set (see Figure 7.11), at least one edge attached to C is not in S'.

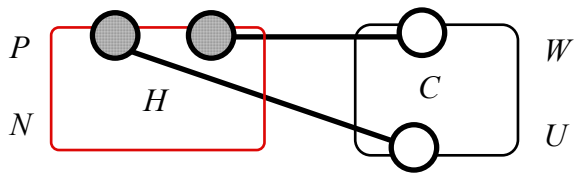


Figure 7.11

But, then, H' should also contain H and, since the root of a simple cut set is an induced subgraph, H' should also contain all the edges attached to C (see Figure 7.12), which would make H' a nonbipartite graph. Therefore, $H' = H$ and, hence, $S' = S$.

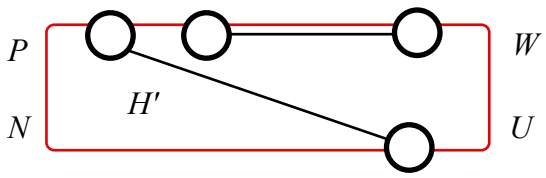


Figure 7.12

Case 2: Let S be a simple cut-loop set with root H. By Theorem 7.4, S' may be either a simple (loop or link) bip set, or a simple cut set, or a simple cut-loop set. First, S' cannot be a simple loop bip set for, otherwise, S' would be the set of loops of G but, even if each branch of S were bipartite, $G–S'$ would not be bipartite by (*ii*). Second, since G contains loops, S' cannot be a simple link bip set by part (*ii*) of Lemma 7.3. Third, S' cannot be a simple cut set because, using the same argument as in Case 1, the root of S' would contain H but, then, would also contain the loops attached to H and, hence, would not be bipartite. Therefore, S' must be a simple cut-loop set. Let H' be the root of S'. Using the same argument as in Case 1, one can prove that H' contains no branches of S so that H is the unique component of $G–S$ contained in H'. Moreover, since H' is bipartite, each loop in S must be in S' for, otherwise, H' would not be bipartite. Therefore, $S' = S$. □

Finally, we are in a position to state the following characterization of minimal algebraic sets of a nonbipartite graph.

Theorem 7.5 Let G be a nonbipartite and connected graph. A nonempty edge set S is a minimal algebraic set of G if and only if either

(*a*) S is a simple loop bip set or a simple link bip set such that, for each connected component B of $G–S$, there is an edge in S having both endpoints in B; or

(*b*) S is a simple cut set or a simple cut-loop set such that

(*b*1) the root of S is connected,

(*b*2) for each bipartite branch C of S, the set of edges attached to C is not a simple cut set, and

(*b*3) if S is a simple cut set, then there is at least one nonbipartite branch of S.

Proof. (*only if*) By Theorem 7.4. (*if*) By Lemmas 7.5 and 7.6. □

Corollary 7.4 The family of algebraic sets of a nonbipartite and loopless, connected graph is the smallest family containing stars and the whole edge set, and is closed under disjoint union and proper difference. The family of algebraic sets of a graph with loops is the smallest family containing stars, and is closed under disjoint union and proper difference.

Proof. If G is a loopless graph then, by Remark 7.4, $E(G)$ is an algebraic set. Moreover, every simple cut set can be obtained as a proper difference of two starsets, and every simple link bip set that is properly contained in $E(G)$ can be obtained as $E(G)$ minus a starset. If G a graph with loops, then every minimal algebraic set is either a simple cut set, or a simple loop bip set or a simple cut-loop set and, hence, is either a starset or the proper difference of two starsets. □

As a consequence of the first part of Corollary 7.4, one has

Corollary 7.5 The family of algebraic sets of a complete graph G contains exactly $2\,(|V(G)| + 1)$ sets; namely, $\emptyset$, $E(G)$ and, for each vertex v of G, the two sets $star(v)$ and $E(G)\backslash star(v)$.

Conclusions

The Performance Problem
Existing approaches to answering statistical queries using aggregate views adopt as rewriting language the very query language. We have presented an analytic approach which proves to be more powerful. Two query-execution plans have been given for sum-queries: one, *plan D*, is independent of the domain of the response variable; the other, *plan E*, is sensitive to the domain of the response variable. We have shown that plan *D* always succeeds in recognising answerable sum-queries in polynomial time, and that plan *E* succeeds in doing so only if the domain of the response variable is the set of reals, or the set integers or the set nonnegative reals (the case of the set nonnegative integers requires solving a co*NP*-hard problem). Moreover, both in plan *D* and in plan *E* the problem of finding a maximally contained sum-query proves to be *NP*-hard, so that more-or-less trivial solutions should be adopted when a sum-query turns out to be unanswerable.
Future research at least includes the following two directions:

- special classes of (maps of) view bases for which the two above-mentioned intractable problems above can be solved in polynomial time

- the extension of the analytic approach to semi-additive statistical queries such as `max` and `min` [9, 10, 37].

The Security Problem
In order to protect the confidentiality of individual data, the query system of a statistical database should be sure that no confidential piece of information runs the risk of being disclosed in an exact or approximate way from responses to sum-queries. To achieve this, the query system should audit sum-queries and issue non-informative answers to sum-queries that directly or indirectly would lead to the disclosure of confidential data. We proposed an answering procedure, which makes a parsimonious use of standard linear-programming methods.

Possible directions of future research are:

- auditing sum-queries with a response variable that is of a general additive type (e.g., an Abelian group [9, 10, 43]), or of a specific type (e.g., a nonnegative integer type or a binary type [37]);

- auditing max- or min-queries, by relaxing some restrictive assumptions such as the individual values of the response variable are all distinct [11] and there is a single tuple falling in every sensitive category [11, 37].

Finally, note that auditing count-queries requires solving the same integer linear-programming problems as auditing sum-queries with a response variable of a nonnegative integer type.

Problems solved when the information model is graphical
We solved the problem of finding the set of invariant edges of an EWG under the assumption that edge weights are nonnegative reals. It is an open problem the case that edge weights are nonnegative integers. However, if the underlying graph of the EWG is bipartite, then Gusfield's algorithm still holds owing to the total unimodularity of the incidence matrix.
A natural generalization of the problem dealt with in Chapter 4 is the search of invariant edges of an edge-weighted hypergraph. It should be noted that *mutatis mutandis* Theorem 4.3 (see Section 4.3) applies to edge-weighted hypergraphs too. So, in order to find the invariant edges of an edge-

weighted hypergraph (G, $\boldsymbol{w}$), we have to devise a procedure for computing its kernel, say K, and the co-loops of the matroid $\boldsymbol{M}(G–K)$. It should be clear that, in order to find to co-loops of $\boldsymbol{M}(G–K)$, we need a formula for the rank of the incidence matrix of G. At the present, such a formula is known only for special classes of hypergraphs, e.g., for the class of connected uniform hypergraphs [5].
Also we solved the problem of finding feasible ranges of the sums of the weights for an arbitrary set of edges on an edge weighted graph.

We solved the NAS problem in linear time and also solved the problem of finding a maximal algebraic set in quadratic time. However we left open the question of whether the problem of finding a maximum algebraic set of a given edge set can be solved in polynomial time or not.
Finally, we stated the graphical characterisation of minimal algebraic sets. This characterisation suggests viewing a minimal algebraic set S as being a "chemical bounding" among the components of $G–S$ that is "easy to loose". Future research is required to find an analogous characterisation for the case that weights are nonnegative integers, or to consider invariant sets of a weighted hypergraph.

References

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

2 Aho, A.V., Hopcroft J.E. and J.F. Ullman: Data Structures and Algorithms, *Addison-Wesley Pub. Co., Reading* 1987.

3 Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. *Prentice Hall, Englewood Cliffs*, 1993.

4 Billingsley, P.: Probability and measure. *Wiley*, 1995.

5 Björner A. and J. Karlander: The mod p rank of incidence matrices for connected uniform hypergraphs, *European J. Combinatorics*, 14 (1993), 151-155.

6 Bjorner A., M. Las Vergnas, B. Sturmfels, N. White, and G. Ziegler: Oriented Matroids, *Cambridge University Press, Cambridge, MA*,1993.

7 Bondy, J.A., Murty, U.S.R Graph theory with applications. *Nort Holland, New York, 1976*.

8 Brankovic L., P. Horak and M. Miller, An optimization problem in statistical databases, *SIAM J. Discrete Mathematics*, 13 (2000), 346-353.

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

10 Chang Chen, M., McNamee, L.: A model of summary data and its applications in statistical databases, *Proc. IV Int. Working Conf. on "Statistical & Scientific Database Management", (M. Rafanelli, J.C. Klensin and P. Svensson, eds.), Lecture Notes in Computer Sciences* 339, 354-372 (*Springer-Verlag*, 1989).

11 Chin, F.: Security problems on inference control for SUM, MAX, and MIN queries. *J. ACM* 33 (1986) 451-464.

12 Chin, F.Y., Ozsoyoglu, G.: Auditing and inference control in statistical databases. *IEEE Trans. on Software Engineering* 8 (1982) 574-582.

13 Chvátal, V.: Linear Programming. *Freeman, New York*, 1983.

14 Cohen, S., Nutt, W., Serebrenik, A.: Rewriting aggregate queries using views, *Proc. XVIII ACM Symp. on "Principles of Database Systems"*, 155-166, 1999.

15 Conforti M. and Rao M.R., Some new matroids on graphs: Cut sets and the max cut problem, *Mathematics of Operations Research*, 12 (1987), 193-204.

16 Cormen T. H., Leiserson C. E., Rivest R. L., Introduction to Algorithms, *McGraw-Hill*, 1990.

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

18 Cox, L.H., Zayatz, L.V.: An agenda for research on statistical disclosure limitation. *J. Official Statistics* 11 (1995) 205-220.

19 Dantzig G.B., Linear Programming and Extensions, *Princeton University Press, Princeton*, 1963.

20 Dobra, A., Fienberg, S.E.: Bounds for cell entries in contingency tables given the marginal totals and decomposable graphs, *Proceedings of the National Academy of Sciences of the United States of America* 97 (2000) 11885-11892.

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

22 Duncan, G.T., Fienberg, S.E., Krishnan, R., Padman, R., Roehrig, S.F.: Disclosure limitation methods and information loss for tabular data, in Confidentiality, Disclosure and Data Access *(Doyle, P., Lane, J., Theeuwes, J., Zayatz, L., eds.), Elsevier* (2001), 135-166.

23 Even S., Graph Algorithms, *Computer Science Press*, 1979.

24 Faloutsos, C., Jagadish, H.V., Sidiropoulos, N.D.: Recovering information from summary data, *Proc. of the XXIII Int. Conf. on "Very Large Data Bases*" 1997, 36-45.

25 Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. *W. H. Freeman, San Francisco*, 1979.

26 Grumbach, S., Tininini, L.: On the content of materialized aggregate views. *J. of Computer and System Sciences 66 (2003), 133-168 (A preliminary version appeared in Proc. XIX ACM Symp. on "Principles of Database Systems"*, 2000.)

27 Gusfield, D., A Graph Theoretic Approach to Statistical Data Security, *SIAM J. on Computing.* 17(3): 552-571 (1988)

28 Halevy, A.Y.; Answering queries using views: a survey. *The VLDB Journal* 10 (2001), 270-294.

29 Horak P., Brankovic L. and M. Miller, A combinatorial problem in database security, *Discrete Applied Mathematics*, 91 (1999), 119-126.

30 Hsu T.-S., and Kao M. Y., Security problems for statistical databases with general cell suppression, in *Proc. of the IX Int. Conf. on "Scientific & Statistical Database Management" (IEEE Comp. Society, Los Alamitos, CA), Olympia*, 1997, 155-164.

31 Hurtado, C.A., Mendelzon, A.O.: Reasoning about summarizability in heterogeneous multidimensional schemas, *Proc. Int. conf. on "Database Theory" 2001, (Van de Bussche, J., Vianu, V., eds.) LNCS* 1973, 375-389, 2001.

32 Jeurissen, R.H.: Magic graphs, a characterization. *Europ. J. Combinatorics* (1988) **9**, 363-368.

33 Kao, M. Y. and D. Gusfield, Efficient detection and protection of information in cross-tabulated tables I: Linear invariant test, *SIAM J. Discrete Mathematics*, 6 (1993), 460-476.

34 Kao, M. Y., Data security equals graph connectivity, *SIAM J. Discrete Mathematics*, 9 (1996), 87-100.

35 Kao M. Y., Efficient detection and protection of information in cross-tabulated tables II: minimal linear invariants, *J. Combinatorial Optimization*, 1 (1997), 187-202.

36 Kao M. Y., Total protection of analytic-invariant information in cross-tabulated tables, SIAM J. Computing, 26 (1997), 231-242.

37 Kleinberg, J.M., Papadimitriou, C.H., Raghavan, P.: Auditing Boolean attributes. *J. of Computer and System Sciences* 66 (2003), 244-253 (*A preliminary version appeared in Proc. XIX ACM Symp. on "Principles of Database Systems*", 2000)

38 Maier, D.: The Theory of Relational Databases. *Computer Science Press, Rockville*, 1983.

39 Malvestuto, F.M., A universal-scheme approach to statistical databases containing homogeneous summary tables, *ACM Transactions on Database Systems*, 18 (1993), 678-708.

40 Malvestuto, F.M., Mezzini, M.: On the hardness of protecting sensitive information in a statistical database. *Proc. World Multiconference on "Systemics, Cybernetics and Informatics", vol. XIV* (2001) 504-509.

41 Malvestuto, F.M., M.Mezzini, A Linear time algorithm for finding the invariant edges of edge-weighted graph, *SIAM J. on Computing* 31(5): 1438-1455 (2002)

42 Malvestuto, F.M., Mezzini, M.: Auditing sum-queries. *Proc. International Conference on "Database Theory"* (2003) 504-509, *Lecture Notes in Computer Sciences*.

43 Malvestuto, F.M., Mezzini, M.: Privacy preserving and data mining in an on-line statistical database of additive type. *Proc. International Conference on "Privacy in Statistical Databases" (2004)*, Barcelona.

44 Malvestuto, F.M., Moscarini, M.: Query evaluability in statistical databases. *IEEE Transactions on Knowledge and Data Engineering* 2 (1990) 425-430.

45 Malvestuto, F.M., M. Moscarini, Suppressing marginal totals from a two-dimensional table to protect sensitive information, *Statistics and Computing*, 7 (1997), 101-114.

46 Malvestuto, F.M., M. Moscarini, An Audit Expert for large statistical databases, *in Proc. of the First Conf. on "Statistical Data Protection", Lisbon, 1998 (edited by EUROSTAT*, 1999), 29-43.

47 Malvestuto, F.M., Moscarini, M.: Computational issues connected with the protection of sensitive statistics by auditing sum-queries, *Proc. X Int. Conf. on "Scientific & Statistical Database Management", (M. Rafanelli, M. Jarke, eds.), IEEE Computer Science* (1998), 134-144.

48 Malvestuto, F.M., Moscarini, M.: Privacy in multidimensional databases, in Multidimensional Databases *(Rafanelli, M., editor), Idea Group Pub. (2003), Hershey, USA*, 310-360.

49 Malvestuto, F.M., M. Moscarini and M. Rafanelli, Suppressing marginal cells to protect sensitive information in a two-dimensional statistical table, in *Proc. of the X ACM Symp. on "Principles of Database Systems", Denver*, 1991, 252-258.

50 Malvestuto F. M., Mezzini M., M.Moscarini, Auditing Sum-Queries To Make A Statistical Database Secure*, accepted for publication to transaction of ACM on Information and System Security* (2004)

51 Malvestuto, F.M., Mezzini M., M.Moscarini, Answering Statistical Sum-Queries Using Materialised Sum-View: An Analytic Approach, *Submitted to Transaction of ACM on Database Systems* (2004)

52 Malvestuto, F.M., Zuffada, C.: The classification problem with semantically heterogenous data, *Proc. IV Int. Working Conf. on "Statistical & Scientific Database Management", (M. Rafanelli, J.C. Klensin and P. Svensson, eds.), Lecture Notes in Computer Sciences* 339, 157- 176 (Springer-Verlag, 1989).

53 Ng, W.K., Ravishankar, C.V.: Information synthesis in statistical databases, *Proc. IV Int. Conf. on "Informaton & Knowledge Management*", 355-361, 1995.

54 Reingold E.M., J. Nievergelt and N. Deo, Combinatorial Algorithms: Theory and Practice, *Prentice-Hall, Englewood Cliffs*, 1977.

55 Schrijver, A.: Theory of Linear and Integer Programming. *Wiley, New York*, 1986.

56 Srivastava, D., Dar, S., Jagadish, H.V., and Levy, A.: Anwering queries with aggregation using views, *Proc. XXII Conf. on "Very Large Data Bases*", 318-329, 1996.

57 Van Nuffelen C., On the incidence matrix of a graph, *IEEE Trans. Circuits and Systems*, 23 (1976), 572.

58 Wang, L., Wijekera, D., Jajodia, S.: Cardinality-based inference control in datacubes. *J. of Computer Security* 12 (2004) 655-692.

59 Wang, L., Wijekera, D., Jajodia, S.: Cardinality-based inference control in sum-only data cubes. Proc. European Symposium on "Computer Security" (ESORICS 2002). Lecture Notes in Computer Science, Vol. 2502. Springer-Verlag, New York (2002), 55-71.

60 Welsh, D.J.A. Matroids: Fundamental Concepts, *Chapter 9 of Handbook of Combinatorics (R.L. Graham, M. Groötschel and L. Lovász, eds.) vol. 1, North-Holland, Amsterdam*, 1995.

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

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

63 Zhang, N., Zhao, W., Chen, J.: Cardinality-based inference control in OLAP systems: an information theoretic approach. *Proc. ACM Int. Workshop on "Data Warehousing and OLAP" (DOLAP 2004*), 59-64.