On the Complexity of Inducing Categorical and Quantitative Association Rules

Fabrizio Angiulli

ICAR-CNR

c/o DEIS, Università della Calabria,

Via Pietro Bucci, 41C

87036 Rende (CS), Italy

E-mail: angiulli@icar.cnr.it

Phone: +39 0984 831738

Fax: +39 0984 839054

Giovambattista Ianni

Dip. di Matematica, Università della Calabria,

Via Pietro Bucci, 30B

87036 Rende (CS), Italy

E-mail: ianni@mat.unical.it

Phone: +39 0984 496430

Fax: +39 0984 496410

Luigi Palopoli

DIMET, Università di Reggio Calabria,

Via Graziella, Loc. Feo di Vito

89100 Reggio Calabria (RC), Italy

E-mail: palopoli@ing.unirc.it

Phone: +39 0965 875235

Fax: +39 0965 875481

October 9, 2003

Abstract

Inducing association rules is one of the central tasks in data mining applications. Quantitative association rules induced from databases describe rich and hidden relationships to be found within data that can prove useful for various application purposes (e.g., market basket analysis, customer profiling, and others). Although association rules are quite widely used in practice, a thorough analysis of the related computational complexity is missing. This paper intends to provide a contribution in this setting. To this end, we first formally define quantitative association rule mining problems, which include boolean association rules as a special case; we then analyze computational complexity of such problems. The general problem as well as some interesting special cases are considered.

1 Introduction

The enormous growth of information available in database systems has led to a significant development of techniques for knowledge discovery in databases. At the heart of the knowledge discovery process is the application of data mining algorithms that are in charge of extracting hidden relationships among pieces of information stored in a given database [11]. The most widely used data mining techniques include classification algorithms, cluster analysis and association rule induction [2]. In this paper, we focus on this last data mining technique. Informally speaking, an association rule states that, in the database at hand, a conjunction of conditions implies a consequence. For instance, the rule *hamburger, fries* $\Rightarrow$ *soft−drink* induced from a purchase database, tells that a customer purchasing a hamburger and fries also purchases a soft-drink. An association rule induced from a database is interesting if it describes a relationship that is "valid" as far as the information stored in the database is concerned. To state such a validity, *indices* are used, that is, functions with values usually in $[0, 1]$. An index tells to what extent an extracted association rule describes knowledge valid in the database at hand. For instance, a *confidence* value of 0.7, associated to the rule above, tells that 70 percent of purchases including hamburgers and fries also include a soft-drink. In the literature, several indices have been proposed (see e.g. [7], where several quality criteria are proposed). Clear enough, information patterns expressed in the form of association rules and associated indices indeed represent knowledge that might be useful in several application areas, such as, market basket analysis and fraud detection, just to mention a few. In some application areas, however, *boolean* association rules, like the one above, are not expressive enough for the purposes of the given knowledge discovery task. In order to obtain more expressive association rules, one can allow more general forms of conditions to occur therein. Given a *categorical attribute* A (an attribute having a discrete, unordered domain associated), a *numeric* attribute A' (an attribute associated with an ordered domain of numbers), a categorical domain value u, and two numeric values l' and u' ($l' \leq u'$), *quantitative association rules* [23] are such that both the premise and the consequence use conditions of the following forms: (i) $A = u$; (ii) $A \neq u$; (iii) $A' \in [l', u']$; (iv) $A' \notin [l', u']$. For instance, the quantitative rule

$$(\text{hamburger} \in [2, 4]), (\text{ice-cream-taste} = \text{chocolate}) \Rightarrow (\text{soft-drink} \in [1, 3])$$

induced from a purchase database, specifies a pattern telling that a customer purchasing from 2 to 4 hamburgers and a chocolate ice-cream also purchases from 1 to 3 soft-drinks. In either forms, inducing association rules is a quite widely used data mining technique, several systems have been developed based from them [3, 18], and several successful applications in various contexts have been described [9]. Despite the wide spread utilization of association rule induction in practical applications, a thorough analysis of the complexity of the associated computational tasks has not been yet developed. However, such analysis appears to be important since, as in other contexts, an appropriate understanding of the computational characteristics of the problem at hand makes it possible to single out tractable cases of generally untractable problems, isolate hard complexity sources and, overall, to devise more effective approaches to algorithm development.

We define a form of association rules that generalizes over the quantitative, categorical and the boolean attributes. We allow the null values (in the following indicated by ϵ) to occur in the database denoting the absence of information. When we induce association rules from databases with nulls, it is forbidden to specify conditions on null values. A

boolean association rule can be thus regarded as a special case of quantitative or categorical association rule mined on a database with nulls. Indeed, according to the definitions in [2], given a set of items I, a transaction t on I is a subset of I, a boolean database T on I is a set of transactions on I, and a boolean association rule on I is an expression of the form $X \Rightarrow Y$, where X and Y are disjoint subsets of I. We capture this formal framework by calling *boolean* a database defined on a set of attributes taking value over $\{c, \epsilon\}$, where c is an arbitrary constant. In this setting, an association rule $(B_1 = c) \wedge \ldots \wedge (B_p = c) \Rightarrow (H_1 = c) \wedge \ldots \wedge (H_q = c)$ will encode the boolean association rule $B_1, \ldots, B_p \Rightarrow H_1, \ldots, H_q$. According to the formalization introduced in the following section, this is the only kind of rule allowed on boolean databases, since conditions on ϵ values are forbidden (and a condition like $A \neq c$ is equivalent to $A = \epsilon$). We analyze the computational complexity of inducing association rules by the most frequently used rule quality indices, namely, confidence, support, θ-gain and h-laplace [7, 2]. Specifically, we shall show that, depending on the chosen index of reference, the complexity of the problem is either in P or NP-complete. When databases with nulls are considered, independently of the reference index, the rule induction task is NP-complete. However, we show that there are cases where the association rule problem is very easy to solve. To permit a better understanding of the new complexity results introduced in this work, we describe them in Section 3, after giving the preliminary definitions that will be used throughout the paper, in Section 2. The rest of this paper is organized as follows. In the next section preliminary definitions are given. In Section 3 related works and details of the new complexity results introduced in this work are described. In Section 4 general complexity results about inducing association rules are stated. Sparse databases and Fixed-schema complexity of rule induction are dealt with in Sections 5 and 6, respectively. Finally, in Section 7 further complexity results about some interesting special cases are collected.

2 Preliminaries

We begin by defining several concepts that will be used throughout the paper, including, among others, those of association rule induction problems and indices.

Definition 2.1 (Domain) A *domain* is a finite or countable set of values augmented with the special value ϵ, called *null value*. A *categorical domain* (respectively, *numerical domain*) is one whose values are unordered (respectively, totally ordered with respect to an order relation $\leq$).

Definition 2.2 (Attribute) An *attribute* A is an identifier with an associated domain $\mathbf{dom}(A)$. If $\mathbf{dom}(A)$ is a categorical (respectively, numeric) domain, then we say that A is a *categorical* (respectively, *numeric*) *attribute*. We say that A is *boolean* if $\mathbf{dom}(A) = \{\epsilon, c(A)\}$, where $c(A)$ denotes an arbitrary fixed constant associated to A.

Definition 2.3 (Tuple) Let $I = A_1, \ldots, A_m$ be a set of attributes. A *tuple* t on I is a m-ple $(v_1, \ldots, v_m)$, where $v_i \in \mathbf{dom}(A_i)$, for $i = 1, \ldots, m$. The *value of the attribute* A_i *in* t, denoted by $t[A_i]$, is v_i, for $i = 1, \ldots, m$. The *size* $|t|$ of $t \in T$ is $|\{A \in I \mid t[A] \neq \epsilon\}|$.

Definition 2.4 (Database) Let I be a set of attributes. A *database* T on I is a collection of tuples on I. We say that T is a *database without nulls* if, for each $t \in T$, $|t| = |I|$. Otherwise we say that T is a *database with nulls*.

Name	Year	Mathematics	Computer Science	Physics
John	2001	A	B	ϵ
Anastasia	2001	ϵ	ϵ	ϵ
Lawrence	2001	A	ϵ	ϵ
Gabriel	2001	B	A	A
John	2002	A	B	E
Anastasia	2002	A	C	ϵ
Lawrence	2002	A	A	A
Gabriel	2002	B	A	A

Figure 1: The example database $\mathbf{DB}_1$.

Definition 2.5 (Boolean database and sparse family of boolean databases) Let I be a set of attributes, and let T be a database on I. We say that T is a *boolean database* if every attribute $A \in I$ is boolean. Given a database T defined on a set of attributes I, by m_T we denote the tuple of T having the largest size. A family S of boolean databases is *sparse* if, for any $T \in S$, $|m_T|$ is $\mathcal{O}(\log |I|)$ where I is the set of attributes which T is defined upon. Given a family S of sparse databases, we will call *sparse database* each element $T \in S$.

Definition 2.6 (Active domain of an attribute) Let I be a set of attributes, let A be an attribute in I, and let T be a database on I. The *active domain of A in T*, denoted by $\mathbf{dom}(A, T)$, is the set $\{t[A] \mid t \in T\} - \{\epsilon\}$.

Thus, given an attribute A and a database T, with $\mathbf{dom}(A, T)$ we denote the set of the values assumed by the attribute A in the tuples of T (null value excluded), while by $\mathbf{dom}(A)$ we denote the set of all the possible values that A can assume in any database (null value included). For example, if $\mathbf{dom}(A)$ is $\mathbb{N} \cup \{\epsilon\}$, where $\mathbb{N}$ denotes the set of the integer numbers, then $\mathbf{dom}(A, T)$ is always a subset of $\mathbb{N}$ of size at most $|T|$.

Definition 2.7 (Atomic condition) Let A be an attribute. An *atomic condition* on A is:

- an expression of the form $A = u$ or $A \neq u$, where A is a categorical attribute and $u \in (\mathbf{dom}(A) - \{\epsilon\})$ is a value in the domain of A distinct from the ϵ value, or
- an expression of the form $A \in [l, u]$ or $A \notin [l, u]$, where A is a numeric attribute, $l, u \in (\mathbf{dom}(A) - \{\epsilon\})$ and $l \leq u$.

Whenever numerical attributes are involved, the notation $A = u$ (respectively $A \neq u$) can be used and is regarded as a syntactic shortcut for $A \in [u, u]$ (respectively $A \notin [u, u]$).

Definition 2.8 (Active domain of an atomic condition) Given a set of attributes I, an attribute A in I, an atomic condition C_A on A, and a database T on I, the *active domain of C_A in T*, denoted by $\mathbf{dom}(C_A, T)$, is:

Mathematics	Computer Science	Physics	Geography
Yes	Yes	ϵ	Yes
ϵ	ϵ	ϵ	Yes
Yes	ϵ	ϵ	Yes
Yes	Yes	Yes	Yes
Yes	Yes	Yes	Yes
Yes	Yes	ϵ	ϵ
Yes	Yes	Yes	ϵ
Yes	Yes	Yes	Yes

Figure 2: The example database $\mathbf{DB}_2$.

	I_1	I_2	I_3	...	I_{k-2}	I_{k-1}	I_k	I_{k+1}	...	I_{n-3}	I_{n-2}	I_{n-1}	I_n
t_1	ϵ	ϵ	ϵ	...	ϵ	ϵ	1	ϵ	...	ϵ	ϵ	ϵ	ϵ
t_2	1	ϵ	ϵ	...	ϵ	ϵ	ϵ	ϵ	...	ϵ	1	ϵ	ϵ
t_3	ϵ	1	ϵ	...	ϵ	ϵ	ϵ	ϵ	...	ϵ	1	ϵ	ϵ
							...						
t_{n-1}	ϵ	ϵ	1	...	ϵ	ϵ	ϵ	ϵ	...	ϵ	ϵ	ϵ	1
t_n	ϵ	ϵ	ϵ	...	ϵ	1	ϵ	ϵ	...	ϵ	ϵ	ϵ	1

Figure 3: An example of sparse database

- for $C_A \equiv (A = u)$, the set $\mathbf{dom}(A, T) \cap \{u\}$;
- for $C_A \equiv (A \neq u)$, the set $\mathbf{dom}(A, T) - \{u\}$;
- for $C_A \equiv (A \in [l, u])$, the set $\mathbf{dom}(A, T) \cap \{x \in \mathbf{dom}(A) \mid l \leq x \leq u\}$;
- for $C_A \equiv (A \notin [l, u])$, the set $\mathbf{dom}(A, T) - \{x \in \mathbf{dom}(A) \mid l \leq x \leq u\}$.

Definition 2.9 (Condition) A *condition* C on a set of distinct attributes $A_1, \ldots, A_n$ is an expression of the form $C = C_1 \wedge \ldots \wedge C_n$, where each C_i is an atomic condition on A_i, for each $i = 1, \ldots, n$. We denote by $\mathbf{att}(C)$ the set $A_1, \ldots, A_n$. The *size* $|C|$ of C is n.

Definition 2.10 (Satisfaction of a condition) Let I be a set of attributes, let T be a database on I, and let t be a tuple of T. Let A be an attribute in I, and let C_A be an atomic condition on A. Then, we say that t *satisfies* C_A, written $t \vdash C_A$, iff $t[A] \in \mathbf{dom}(C_A, T)$. Let $C = C_1 \wedge \ldots \wedge C_n$ be a condition on a subset of I, we say that t *satisfies* C, written $t \vdash C$,

UserID	Carrier	PlcdCalls	RcvdCalls	SpntMoney
K	Omnitel	80	40	$23.33
K	Tim	10	5	$4.30
A	Omnitel	110	81	$30.04
L	Wind	90	20	$51.51
V	Wind	95	112	$70.70
V	Omnitel	1	0	$.05
G	Wind	50	2	$25.50
G	Omnitel	5	30	$1.25

Figure 4: The example database $\mathbf{DB}_3$

iff $t \vdash C_i$, for each $i = 1, \ldots, n$. Otherwise we say that t *does not satisfy* C, written $t \nvdash C$. By T_C we denote the set of tuples $\{t \in T \mid t \vdash C\}$.

We are now able to define association rules and their semantics.

Definition 2.11 (Association rule) Let I be a set of attributes. An *association rule* on I is an expression of the form $B \Rightarrow H$, where B and H, called *body* and *head* of the rule respectively, are two conditions on the sets of attributes I_B and I_H respectively, such that $\emptyset \subset I_B, I_H \subset I$, and $I_B \cap I_H = \emptyset$. The *size* $|B \Rightarrow H|$ of the rule is $|B| + |H|$.

Definition 2.12 (Trivial condition and trivial association rule) Let I be a set of attributes, and let T be a database on I, and let C be a condition on a subset of I. We say that C is *trivial* if it contains at least one atomic condition C_A such that $T_{C_A} = T$. Let $B \Rightarrow H$ be an association rule on I. We say that $B \Rightarrow H$ is *trivial* if $B \wedge H$ is trivial.

Following are examples of rules in the database $\mathbf{DB}_3$ shown in Figure 4:

$$\mathbf{Carrier} = Omnitel \implies \mathbf{RcvdCalls} \in [50, 100], \quad (1)$$

$$\mathbf{RcvdCalls} \in [0, 50] \wedge \mathbf{PlcdCalls} \in [0, 50] \implies \mathbf{Carrier} = Wind, \quad (2)$$

$$\mathbf{PlcdCalls} \in [50, 80] \implies \mathbf{SpntMoney} \in [\$50, \$100], \quad (3)$$

$$\mathbf{SpntMoney} \in [\$0, \$100] \implies \mathbf{Carrier} = Omnitel. \quad (4)$$

Note that rule 4 is trivial. A database allowing nulls is shown in Figure 1 ($\mathbf{DB}_1$), whereas Figure 2 describes a boolean database ($\mathbf{DB}_2$). Examples of allowed rules on $\mathbf{DB}_2$ are:

$$\mathbf{Mathematics} = Yes \implies \mathbf{Physics} = Yes \wedge \mathbf{Computerscience} = Yes, \quad (5)$$

$$\mathbf{Geography} = Yes \implies \mathbf{Physics} = Yes. \quad (6)$$

When inducing association rules from databases in data mining applications, one is usually interested in obtaining rules that describe knowledge "largely" valid in the given database. This idea is captured by several notions of *indices*, which

have been defined in the literature. In the following, we shall consider the most widely used indices, whose definitions are given next.

Definition 2.13 (Indices) Let I be a set of attributes, let T be a database on I, and let $B \Rightarrow H$ be an association rule on I. Then:

1. the *support* of $B \Rightarrow H$ in T, written $sup(B \Rightarrow H, T)$, is $\frac{|T_{B \wedge H}|}{|T|}$;
2. the *confidence* of $B \Rightarrow H$ in T, written $cnf(B \Rightarrow H, T)$, is $\frac{|T_{B \wedge H}|}{|T_B|}$;
3. Let θ be a rational number, $0 < \theta \leq 1$, then the *θ-gain* of $B \Rightarrow H$ in T, written $gain_\theta(B \Rightarrow H, T)$, is $\frac{|T_{B \wedge H}| - \theta \cdot |T_B|}{|T|}$;
4. Let h be a natural number, $h \geq 2$. Then the *h-laplace* of $B \Rightarrow H$ in T, written $laplace_h(B \Rightarrow H, T)$, is $\frac{|T_{B \wedge H}|+1}{|T_B|+h}$.

Let C be a condition on I. By analogy with the above definition, we define the *support* of C in T, written $sup(C, T)$, as $\frac{|T_C|}{|T|}$.

Support and confidence are classical indices employed in the data mining field to establish rules' quality (see, e.g. [17]). Intuitively, when a rule scores a high support, an evaluation algorithm may conclude that it is worth to further consider the rule at hand, since there exist a significant fraction of the database tuples that satisfy the conjunction of the atoms in the rule. Confidence shows to what extent a given rule is true within the database at hand. The gain index [7, 13] is employed as a combined measure of support and confidence. Intuitively, it is desirable to have rules with both high confidence and support. Indeed, gain can be seen as a combined measure of rules' quality in terms of both support and confidence (note that gain can be rewritten as $gain_\theta(R, T) = sup(R, T)(cnf(R, T) - \theta)$). The Laplace index [7] is inspired from the statistical Laplace's rule, and provides a measure of the probability for a new inserted tuple to satisfy the rule at hand. Having defined association rules and associated indices (that, in different forms, measure the validity of an association rule w.r.t. a database where it has been induced from), we are able to formally define next the association rule induction problems.

Definition 2.14 (Association rule induction problem) Let I be a set of attributes, let T be a database on I, let k, $1 \leq k \leq |I|$, be a natural number, and let s, $0 < s \leq 1$, be a rational number. Furthermore, let $\rho \in \{sup, cnf, laplace_h, gain_\theta\}$. The association rule induction problem $\langle I, T, \rho, k, s \rangle$, also called *$\rho$-problem*, is as follows: *Is there a non-trivial association rule R such that $|R| \geq k$ and $\rho(R, T) \geq s$?*

In general, we shall measure the complexity of association rule induction problems for the various index forms we have defined above. As a special case, we shall also consider the complexity of the induction problems when the attribute set I is assumed not to be part of the input, in which case we will talk about *fixed schema complexity* of the association rule induction problem.

Remarks.

1. In the literature it is usually assumed that, in answering an association rule induction problem, one looks for rules that meet some criteria in terms of two or more indices [7]. Here we prefer to consider one index at a time. Indeed, this allows to identify complexity sources; moreover, complexity measures for problems involving more than one index can be obtained fairly easily from problems involving only one index.

2. Highest indices values can be easily obtained building ad hoc trivial rules. Thus, in the following, we will focus our attention on non-trivial association rules. As an example, consider rule 4 above, regarding database $\mathbf{DB}_3$, which is trivial, because of the condition $\mathbf{SpntMoney} \in [\$0, \$100]$ (the whole domain of the attribute **SpntMoney** is captured).

3. In the following, we shall study complexities of association rule induction by defining several suitable *decision* problems. It can be objected that inducing association rules is an enumeration problem rather than a decision problem. However, we observe that complexity of computational problems is studied usually by examining their decision problem versions. Indeed, computational complexity theory has focused mainly on complexity of decision problems [14, 21]. In any case, this approach allows us on one hand to state a reasonable form of lower bound over the enumeration problem and, on the other hand, to single out the source of complexity characterizing the problems at hand which is the necessary premise to devise algorithms solving the problem as efficiently as possible. In the following, in Section 3, we shall briefly comment on complexity sources of rule induction problems.

Definition 2.15 (Domain Tailoring) Let I be a set of numerical attributes, and let T be a database on I. Let A be an attribute in I, and let u be a value. Define

- $\mathbf{lub}(u, A, T) = \min\{v \in \mathbf{dom}(A, T) \mid u \leq v\}$, and
- $\mathbf{glb}(u, A, T) = \max\{v \in \mathbf{dom}(A, T) \mid v \leq u\}$.

Let $C = A \in [l, u]$ (respectively $C = A \notin [l, u]$) be a nontrivial atomic condition such that $|T_C| > 0$. Define

$$\begin{aligned} \mathbf{bot}(C, T) &= A \in [\mathbf{lub}(l, A, T), \mathbf{glb}(u, A, T)] \\ &\quad (A \notin [\mathbf{lub}(l, A, T), \mathbf{glb}(u, A, T)] \text{ respectively}). \end{aligned}$$

Let $C = C_1 \wedge \ldots \wedge C_n$ be a nontrivial condition such that $|T_C| > 0$. Define

$$\mathbf{bot}(C, T) = \mathbf{bot}(C_1, T) \wedge \ldots \wedge \mathbf{bot}(C_n, T).$$

Proposition 2.16 *Let I be a set of numerical attributes, let T be a database on I, and let C be a nontrivial condition on a subset of I such that $|T_C| > 0$. Then $T_C = T_{\mathbf{bot}(C,T)}$.*

Proof. Let $C = C_1 \wedge \ldots \wedge C_n$. Simply observe that $\mathbf{dom}(C, T) = \mathbf{dom}(\mathbf{bot}(C, T), T)$. □

Proposition 2.16 has the technically important consequence that we can restrict our attention to conditions and association rules including only values from the input database. In this paper, we will refer to conditions and association rules of this kind only. The same assumption holds for conditions defined on categorical attributes.

2.1 Complexity Classes

We assume the reader is familiar with basic concepts regarding computational complexity and, in particular, the complexity classes P (the decision problems solved by polynomial-time bounded deterministic Turing machines), NP (the decision problems solved by polynomial-time bounded non-deterministic Turing machines) and L (the decision problems solved by logspace-bounded deterministic Turing machines).

Definition 2.17 Let C be a boolean circuit. The *size* of C is the total number of gates in it. The *depth* of C is the number of gates in the longest path in C.

Definition 2.18 MAJORITY gates are unbounded fan-in logic gates (with binary input and output) that output 1 if and only if more than a half of their inputs are non-zero.

Definition 2.19 A family $\{C_i\}$ of circuits, where C_i accepts strings of size i, is uniform if there exists a Turing machine $\mathcal{T}$ which on input i produces the circuit C_i. $\{C_i\}$ is said to be *logspace uniform* if $\mathcal{T}$ carries out its work using $O(\log i)$ space.

Definition 2.20 Define AC^0 (respectively TC^0) as the class of decision problems solved by logspace uniform families of circuits of polynomial size and constant depth, with AND, OR, and NOT (respectively AND, OR, and MAJORITY) gates of unbounded fan-in [1, 6, 22].

Definition 2.21 For any $k > 0$, $\#\mathrm{AC}^0_{\mathrm{k}}$ is the class of functions $f : \{0,1\}^* \to \mathbb{N}$ computed by depth k, polynomial size logspace uniform families of circuits with $+,\times$-gates (the usual arithmetic sum and product in $\mathbb{N}$) having unbounded fan-in, where the inputs to the circuit consist of x_i and $1 - x_i$ for each input bit x_i and of the constants 0 and 1. Let $\#\mathrm{AC}^0 = \bigcup_{k>0} \#\mathrm{AC}^0_k$ [1].

Note that $\#\mathrm{AC}^0$ circuits take the values 1 and 0 as inputs, which are considered as natural numbers.

Definition 2.22 GapAC^0 is the class of all functions $f : \{0,1\}^* \to \mathbb{N}$ that can be expressed as the difference of two functions in $\#\mathrm{AC}^0$ [1, 5]. PAC^0 is the class of languages $\{A \mid \exists f \in \mathrm{GapAC}^0, x \in A \iff f(x) > 0\}$ [1].

Definition 2.23 Let $\{C_i\}$ be a uniform family of boolean circuits, and let $f(n)$ and $g(n)$ be functions from the integers to the integers. We say that the *parallel time* of $\{C_i\}$ is at most $f(n)$ if for all n the depth of C_n is at most $f(n)$. We say that the *total work* of C is at most $g(n)$ if for all $n \geq 0$ the size of C_n is at most $g(n)$.

Definition 2.24 Define $\mathbf{PT}/\mathbf{WK}(f(n), g(n))$ to be the class of all languages $L \subseteq \{0,1\}^*$ such that there is a uniform family of circuits $\{C_i\}$ deciding L with $\mathcal{O}(f(n))$ parallel time and $\mathcal{O}(g(n))$ work. NC is the class $\mathbf{PT}/\mathbf{WK}(\log^k, n^k)$ of all problems solvable in polylogarithmic parallel time with polynomial amount of total work. For any $j \geq 0$, NC_j is the class $\mathbf{PT}/\mathbf{WK}(\log^j n, n^k)$, that is, the subset of NC in which the parallel time is $\mathcal{O}(\log^j n)$; the free parameter k means that it is allowed any degree in the polynomial accounting for the total work.

The above defined classes are of practical relevance since they identify with precision many problems related to simple arithmetic calculations (e.g. $\#\mathrm{AC}^0$); furthermore, these classes enclose problems with highly parallelizable algorithmic structure. For further details, see [25].

3 Related work and contributions

As far as we know, some computational complexity results pertaining to association rules were presented in [15, 19, 20, 26, 27, 8]. We briefly survey the results presented in these works and then pinpoint relationships with this paper. In [15], is stated the NP-completeness of the problem $\langle I, T, sup, k, s\rangle$ on boolean databases, therein called $0/1$ relations. This is done through reducing the Balanced Bipartite Clique problem to it. Moreover, it is stated the #P-hardness of the problem of counting the number of association rules scoring enough support on a boolean database. In [26] the authors defined the QARMINE(D) decision problem as a sextuple $\langle I, T, L, \pi_r, s, c\rangle$, where I is a set of attributes, T is a quantitative database on I, $L \subseteq I$, π_r is a pattern over I, and s, c are two real numbers such that $0 < s \leq c \leq 1$. They defined a *pattern* over a set of attributes $A_1, \ldots, A_m$ as a condition of the form $A_1 \in [l_1, u_1] \wedge \ldots \wedge A_m \in [l_m, u_m]$ where $l_i < u_i$ $(1 \leq i \leq m)$ are two distinct real numbers. The answer to the instance $\langle I, T, L, \pi_r, s, c\rangle$ of the problem QARMINE(D) is "yes" iff there exists a pattern π_l on a subset of L, such that $sup(\pi_l \Rightarrow \pi_r, T) \geq s$ and $cnf(\pi_l \Rightarrow \pi_r, T) \geq c$. The QARMINE(D) problem has been proved to be NP-complete under the general complexity measure, while it is polynomial time solvable under the fixed schema complexity measure. In [27] a boolean database T on I is interpreted as the encoding of a bipartite graph $G = (U, V, E)$. Here U is the set of items I; there is a node n_t in V for each tuple t of T, and there is an edge (A, n_t) in E for each tuple t of T and for each attribute A of I such that $t[A] \neq \epsilon$. The authors argued that the problem of enumerating all boolean association rules with high support corresponds to the task of enumerating all the bipartite cliques (a bipartite clique is a complete bipartite subgraph) of the form $I_c \times T_c$, with $I_c \subseteq U$ and $T_c \subseteq V$, subject to the constraint that $|T_c|$ is greater than a specified threshold. Then, they recall the complexity of some decision problems for maximal bipartite cliques, and the complexity of the best algorithms for the enumeration of all the maximal bipartite cliques of a bipartite graph. In [19] and [20], an NP-hardness result is stated regarding the induction of boolean association rules (or, in general, of *conditions*) having an optimal *entropy* or *chi-square*, although entropy and chi-square are indices that we do not consider in this work. The authors of [8] dealt with the complexity of computing all the maximal frequent sets and all the minimal infrequent sets in a boolean database. Given a set of boolean attributes I, a database T on I, and a threshold t $(1 \leq t \leq |T|)$, a subset X of I is said to be *frequent*, if $|T_{\mathcal{C}(X)}| \geq t$, while is said to be *infrequent*, if $|T_{\mathcal{C}(X)}| < t$, where $\mathcal{C}(X)$ denotes the condition $\bigwedge_{Y \in X} Y = c(Y)$. Let $\mathcal{M}_t$ and $\mathcal{I}_t$ denote the family of all the maximal frequent sets and minimal infrequent sets respectively. It is proved that, if $\mathcal{I}_t \neq \emptyset$, then $|\mathcal{M}_t| \leq (|T| - t + 1)|\mathcal{I}_t|$, and, hence, that the complexity of generating $\mathcal{M}_t \cup \mathcal{I}_t$ is equivalent to that of the transversal hypergraph problem (see [10] for the definition of this problem). As the latter problem is known to be solvable in incremental quasi-polynomial time [12], then the same result holds for the joint generation of maximal frequent and minimal infrequent sets: for each $k \leq |\mathcal{M}_t \cup \mathcal{I}_t|$, k sets belonging to $\mathcal{M}_t \cup \mathcal{I}_t$ can be generated in $poly(|I|, |T|) + k^{\mathcal{O}(\log k)}$ time. We summarize next the contribution of this paper. Relationships between our contribution and the abovementioned works will be outlined.

- Consider categorical and quantitative databases without nulls; in this setting, we prove that the problem $\langle I, T, sup, k, s\rangle$ is NP-complete (Theorem 4.2). From the NP-completeness of the problem $\langle I, T, sup, k, s\rangle$ on boolean databases stated in [15] there follows the NP-completeness of the same problem on databases with nulls. We note that databases without nulls form a subset of databases with nulls, thus the result in [15] does not immediately apply.

Furthermore, in [26] is stated the NP-completeness of the problem of inducing association rules with high support (and high confidence) on quantitative databases without nulls; however, the proof is carried out under the artificial assumption that each condition within an association rule must involve an interval containing at least two distinct numbers. We also observe that in the QARMINE(D) problem the head of the rule is an input parameter.

- We show that, under the general complexity measure, the problem $\langle I, T, cnf, k, s\rangle$ is NP-complete on databases with nulls (Theorem 4.11), while it is in TC^0 (Theorem 4.7), and hence in P, when databases without nulls are considered. The analysis of the computational complexity of this problem has been, so far, missing in the literature. Furthermore, in [26] the problem of inducing quantitative association rules on databases without nulls with simultaneously greater confidence and support than two given thresholds is proven to be NP-complete. Here we prove that the problem of inducing quantitative association rules on databases without nulls with a confidence greater than a given threshold is in P. Hence, we can conclude that in the problem dealt with in [26], the additional source of complexity arising from the presence of a constraint on the confidence value is hidden by the contemporary presence of the same constraint on the support value.

- We prove that $\langle I, T, gain_\theta, k, s\rangle$ and $\langle I, T, laplace_h, k, s\rangle$ are NP-complete when both databases with nulls and databases without nulls are considered (Theorem 4.13 and Corollary 4.14).

- We single out an interesting subset of boolean databases, called *sparse*, for which the problem $\langle I, T, \rho, k, s\rangle$, $\rho \in \{sup, cnf, gain_\theta, laplace_h\}$, is solvable in logarithmic space under the general complexity measure (Theorems 5.1 and 5.2).

- We strengthen a result presented in [26], showing that the problem $\langle I, T, \rho, k, s\rangle$, $\rho \in \{sup, cnf, gain_\theta, laplace_h\}$, is solvable in logarithmic space under the fixed schema complexity measure both on databases with nulls and databases without nulls (Theorems 6.1 and 6.2).

- Finally, we prove complexity results for some interesting special cases of the general rule induction problem, namely:

  - $\langle I, T, sup, k, s\rangle$ where $s \in (0, 1)$ is a fixed constant and T is a database with nulls is NP-complete (Theorem 7.1);
  - $\langle I, T, sup, k, s\rangle$ where k is a fixed constant and T is boolean database is in TC^0 (Theorem 7.4);
  - $\langle I, T, sup, k, s\rangle$ where $\lceil s|T|\rceil$ is a fixed constant and T is boolean database is in TC^0 (Theorem 7.7);
  - $\langle I, T, sup, k, s\rangle$ where k and $\lceil s|T|\rceil$ are two fixed constants and T is a boolean database is in AC^0_2 (Theorem 7.8).

  We recall that in [27] is proved that the decision problems associated to the induction of boolean association rules $B \Rightarrow H$ such that $|B \Rightarrow H| \geq k$ (problem A) or $sup(B \Rightarrow H, T) \geq k$ (problem B) or $|B \Rightarrow H| + sup(B \Rightarrow H, T) \geq k$ (problem C), where k is a constant, are polynomial time solvable. While it is not immediately obvious

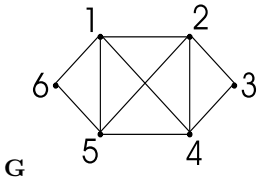
Index	Database Type	Constraint	Complexity	Reference
sup,$gain_\theta$,$laplace_h$	no nulls		NP-complete	Th. 4.2, 4.13
cnf	no nulls		TC^0	Th. 4.7
all	with nulls		NP-complete	Cor. 4.4, Th. 4.11, Cor. 4.14
all	sparse		L	Th. 5.1, 5.2
all	any	$\|I\|$ fixed	L	Th. 6.1, 6.2
sup	with nulls	s fixed	NP-complete	Th. 7.1
sup	boolean	k fixed	TC^0	Th. 7.4
sup	boolean	$s\|T\|$ fixed	TC^0	Th. 7.7
sup	boolean	$s\|T\|$ and k fixed	AC^0_2	Th. 7.8

Figure 5: Summary of complexity results for $\langle I, T, \rho, k, s\rangle$.

to compare these results with ours, we note that, in any case, generally speaking, the results presented here seem to state stronger complexity bounds.

In conclusion, we recall that problems belonging to classes as AC^0, TC^0 and L are very efficiently parallelizable (indeed $\mathrm{AC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}_1 \subseteq \mathrm{L} \subseteq \mathrm{NC}_2$), so that the algorithm design effort could be addressed accordingly. These complexity results are summarized in Table 5. Before proceeding, it is worth briefly commenting on the results presented in the table:

- Under the general complexity measure, all the problems considered are NP-complete in the presence of null values; therefore, dealing with databases where null appear makes, "per se" the task of rule induction very demanding from the computational point of view;
- Under the general complexity measure, the problem $\langle I, T, cnf, k, s\rangle$ becomes tractable when databases without nulls are considered, while the other problems remain intractable; this means that looking for rules with high-confidence is easier than generating rules scoring high values for the other indexes; the reason here (as implicitly shown in the proof of Lemma 4.5 and Theorem 4.7) is that well-suited rules can be easily enumerated using a polynomial method;
- If we impose a bound on the length of the tuples appearing in the database or a bound on the number of attributes on which a database is defined, then all the problems become highly parallelizable; intuitively speaking, this kind of result can be understood if one considers that the number of candidate rules is polynomial and different rules can be generated independently one from another, when such bounds are imposed;
- The problem $\langle I, T, sup, k, s\rangle$ on databases with nulls becomes highly parallelizable, when either k or $s|T|$ is held fixed; in this case, the same considerations drawn for the item above apply.



$\mathbf{T^{clq}}$	I_1	I_2	I_3	I_4	I_5	I_6
$t_{\{v_1,v_2\}}$	0	0	1	1	1	1
$t_{\{v_1,v_4\}}$	0	1	1	0	1	1
$t_{\{v_1,v_5\}}$	0	1	1	1	0	1
$t_{\{v_1,v_6\}}$	0	1	1	1	1	0
$t_{\{v_2,v_3\}}$	1	0	0	1	1	1
$t_{\{v_2,v_4\}}$	1	0	1	0	1	1
$t_{\{v_2,v_5\}}$	1	0	1	1	0	1
$t_{\{v_3,v_4\}}$	1	1	0	0	1	1
$t_{\{v_4,v_5\}}$	1	1	1	0	0	1
$t_{\{v_5,v_6\}}$	1	1	1	1	0	0

Figure 6: An example of the reduction used in Theorem 4.2

4 General complexity results

Here we investigate the complexity of solving $\langle I, T, \rho, k, s\rangle$ when I, T, k and s are inputs.

4.1 Support-problems

Here, we prove that, when support is used as the reference index, the association rule mining problem is NP-complete both in the presence and in the absence of nulls.

In [15] (see Theorem 4) is stated the NP-completeness of the problem $\langle I, T, sup, k, s\rangle$ when T is a boolean database, therein called *0/1-relation*. From this result immediately follows the NP-completeness of the problem $\langle I, T, sup, k, s\rangle$ on databases with nulls. The following theorem states that the problem $\langle I, T, sup, k, s\rangle$ remains intractable even if we restrict our attention to databases without nulls (we note that databases without nulls form a subset of the most general case, those of databases with nulls). In particular, the next result, extends those presented in [26], that applies only to numerical databases without nulls with conditions on intervals containing at least two distinct numbers. Furthermore, Theorem 4.2 can be quite immediately extended to prove the NP-completeness of the general case, as stated by the subsequent Corollary 4.4 which is presented below.

Proposition 4.1 *Consider the problem $\mathcal{P} = \langle I, T, sup, k, s\rangle$. If there is a rule $B \Rightarrow H$ that is a solution for $\mathcal{P}$, then for each k', $1 < k' \leq k$, there is a rule $B' \Rightarrow H'$ of size k' such that $sup(B' \Rightarrow H', T) \geq s$.*

Proof. Given a condition C and a database T, such that $sup(C, T) \geq s$, simply note that it is easy to build a condition C' such that $\mathbf{att}(C') \subseteq \mathbf{att}(C)$ and $|T_{C'}| \geq |T_C|$ holds. □

Theorem 4.2 *Given a database T without nulls, the problem $\langle I, T, sup, k, s\rangle$ is* NP*-complete.*

Proof of Theorem 4.2. (*Hardness*) The proof is by reduction of the problem CLIQUE, which is well-known to be NP-complete [14]. Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, \ldots, v_n\}$ is a set of nodes, and $E = \{e_1, \ldots, e_m\}$

is a set of edges $e_i = \{v_{p_i}, v_{q_i}\}$, $p_i, q_i \in \{1, \ldots, n\}$, for $i = 1, \ldots, m$. Let h be an integer. The CLIQUE problem is: *Does there exist in G a complete subgraph (clique) of size at least h ?* W.l.o.g. suppose the graph G is connected. We build an instance $\langle I^{clq}, T^{clq}, sup, k, s\rangle$ as follows (an example of this reduction is reported in Figure 6):

- let I^{clq} be the set consisting of the attributes $I_1, \ldots, I_n$, such that I_j represents the node v_j of G, for each $j = 1, \ldots, n$;
- let T^{clq} be the database on I^{clq} consisting of a tuple t_{e_i}, for each $i = 1, \ldots, m$, such that $t_{e_i}[I_j] = 0$ if $v_j \in e_i$, and $t_{e_i}[I_j] = 1$ otherwise (so that t_{e_i} encodes the edge e_i of G).
- let k be $n - h$;
- let s be $\frac{h(h-1)}{2m}$.

Next, we prove that G has a clique of size h iff $\langle I^{clq}, T^{clq}, sup, k, s\rangle$ is a YES instance.
The following Claim holds.

Claim 4.3 *Let $I_j \in I^{clq}$, let $C' = (I_j = 0)$, and let C'' be a nontrivial condition defined on a subset of $I^{clq} - \{I_j\}$. Then $|T^{clq}_{C' \wedge C''}| \leq n - |C' \wedge C''|$.*

Proof of Claim 4.3. We distinguish two cases:

1. C'' contains a condition $I_a = 0$ $(1 \leq a \leq n)$. Then, clearly, $|T^{clq}_{C' \wedge C''}| \leq 1$.
2. C'' contains only conditions of the form $I_a = 1$ $(1 \leq a \leq n)$. Let v_j be the node corresponding to the attribute I_j. Observe that

$$|T^{clq}_{C' \wedge C''}| = |\{v \in V : \{v_j, v\} \in E\}| - |\{v_a \in V : \{v_j, v_a\} \in E \wedge I_a \in \mathbf{att}(C'')\}| = \\ = |\{v_a \in V : \{v_j, v_a\} \in E \wedge I_a \notin \mathbf{att}(C'')\}| \leq n - |C' \wedge C''|.$$

□

We can now resume the proof of the theorem. ($\Rightarrow$) Let $C = \{v_{r_1}, \ldots, v_{r_h}\}$ be a clique of size h in G. Consider the condition

$$B \wedge H = \left(\bigwedge_{v_j \in (V - C)} (I_j = 1) \right).$$

Since G is connected, for each j, $1 \leq j \leq n$, there is at least a tuple t such that $t[I_j] = 0$, thus $B \wedge H$ is not trivial. By definition of clique, there are $\frac{h(h-1)}{2}$ edges of G connecting nodes in C. Therefore, the cardinality of the set

$$T' = \{t_{\{v_{r_x}, v_{r_y}\}} \in T^{clq} \mid 1 \leq x < y \leq h\}$$

equals $\frac{h(h-1)}{2}$. Since $T' \subseteq T^{clq}_{B \wedge H}$, then $sup(B \Rightarrow H, T^{clq}) = \frac{|T^{clq}_{B \wedge H}|}{|T^{clq}|} \geq \frac{h(h-1)}{2m}$.

($\Leftarrow$) By Proposition 4.1, if $\langle I^{clq}, T^{clq}, sup, n - h, \frac{h(h-1)}{2m}\rangle$ is a YES instance then there is a nontrivial rule $B \Rightarrow H$

of size $n-h$ such that $|T^{clq}_{B\wedge H}| \geq \frac{h(h-1)}{2}$. W.l.o.g. assume $h \geq 4$. Note that conditions of the form $I_j \in [0,1]$ are trivial. Suppose that there is a condition $I_j = 0$ occurring in $B \Rightarrow H$, then, by Claim 4.3,

$$|T^{clq}_{B\wedge H}| \leq n - |B \wedge H| = h < \frac{h(h-1)}{2}.$$

Hence only conditions of the form $I_j = 1$ can appear in $B \Rightarrow H$. Let $I^{clq} - \mathbf{att}(B \wedge H) = \{I_{r_1}, \ldots, I_{r_h}\}$. In order to be $|T^{clq}_{B\wedge H}| \geq \frac{h(h-1)}{2}$, $T^{clq}_{B\wedge H}$ contains, at least, the set

$$\{t_{\{v_{r_x}, v_{r_y}\}} \in T^{clq} \mid 1 \leq x < y \leq h\},$$

i.e. the nodes $v_{r_1}, \ldots, v_{r_h}$ form a clique of G with size h.

(*Membership*) A certificate for $\langle I, T, sup, k, s\rangle$ is given by an association rule $B \Rightarrow H$ defined on a subset of I. This can be checked in polynomial time by verifying that $B \Rightarrow H$ is not trivial, that $|B \Rightarrow H| \geq k$, and that $sup(B \Rightarrow H, T) \geq s$. □

Corollary 4.4 *Given a database T with nulls, the complexity of $\langle I, T, sup, k, s\rangle$ is* NP*-complete.*

Proof of Corollary 4.4. Hardness is proved by means of Theorem 4.2, since databases with nulls are a superset of databases without nulls. Membership in NP is straightforward. □

4.2 Confidence-problems for databases without nulls

It is generally believed that when both support and confidence are measured, the task of filtering out those rules with low confidence from a set of rules having support above a certain threshold is far easier to compute [3, 27]. We prove that the problem of finding association rules with high confidence on databases without nulls is a tractable subcase, while the same problem on databases with nulls remains computationally demanding.

Lemma 4.5 *Let I be a set of attributes, let T be a database without nulls on I, and let s, $0 < s \leq 1$, be a rational number. Then there is a nontrivial association rule $B \Rightarrow H$ on I such that $cnf(B \Rightarrow H, T) \geq s$ iff there is an attribute $J_H \in I$, a value $u_H \in \mathbf{dom}(J_H, T)$, and a tuple $t \in T$, such that the rule*

$$\left(\bigwedge_{J \in (I - \{J_H\})} (J = t[J]) \right) \Rightarrow (J_H \neq u_H)$$

is nontrivial and has a confidence greater than or equal to s.

Proof. ($\Rightarrow$) Let

$$(B \Rightarrow H) = (C_1 \wedge \ldots \wedge C_h \Rightarrow C_{h+1} \wedge \ldots \wedge C_k)$$

be a nontrivial rule such that C_i is an atomic condition, for each $i = 1, \ldots, k$, and $cnf(B \Rightarrow H, T) \geq s$. Let J_H be $\mathbf{att}(C_k)$, and let $u_H \in (\mathbf{dom}(J_H, T) - \mathbf{dom}(C_k, T))$. Since C_k is not trivial, u_H exists. Consider the rule

$$(B' \Rightarrow H') = (C_1 \wedge \ldots \wedge C_{k-1} \Rightarrow (J_H \neq u_H)).$$

A	B
1	1
1	1
2	2
2	3
3	1
3	1

Figure 7: The example database $\mathbf{DB}_4$

Then, from $|T_{B' \wedge H'}| \geq |T_{B \wedge H}|$ and $|T_{B'}| \leq |T_B|$, it follows that

$$cnf(B' \Rightarrow H', T) = \frac{|T_{B' \wedge H'}|}{|T_{B'}|} \geq \frac{|T_{B \wedge H}|}{|T_B|} = cnf(B \Rightarrow H, T).$$

Let $I - \{J_H\} = J_1, \ldots, J_{n-1}$. For each $t \in T$, we denote by $\mathcal{C}(t)$ the condition

$$(J_1 = t[J_1]) \wedge \ldots \wedge (J_{n-1} = t[J_{n-1}]).$$

Let T' be a maximal subset of $T_{B'}$ such that for no $t, t' \in T'$ it holds that

$$(t[J_1] = t'[J_1]) \wedge \ldots \wedge (t[J_{n-1}] = t'[J_{n-1}]).$$

We show that for some $t \in T'$ it holds that $\frac{|T_{\mathcal{C}(t) \wedge H'}|}{|T_{\mathcal{C}(t)}|} \geq s$. Assume by way of contradiction, for each $t \in T'$, $\frac{|T_{\mathcal{C}(t) \wedge H'}|}{|T_{\mathcal{C}(t)}|} < s$. Then

$$cnf(B' \Rightarrow H', T) = \frac{\left|\bigcup_{t' \in T'} T_{\mathcal{C}(t') \wedge H'}\right|}{\left|\bigcup_{t'' \in T'} T_{\mathcal{C}(t'')}\right|} = \frac{\sum_{t' \in T'} |T_{\mathcal{C}(t') \wedge H'}|}{\sum_{t'' \in T'} |T_{\mathcal{C}(t'')}|} < \frac{\sum_{t' \in T'} s|T_{\mathcal{C}(t')}|}{\sum_{t'' \in T'} |T_{\mathcal{C}(t'')}|} = s,$$

which contradicts the fact that $cnf(B' \Rightarrow H', T) \geq s$. Then there is some $\bar{t} \in T'$ such that $\frac{|T_{\mathcal{C}(\bar{t}) \wedge H'}|}{|T_{\mathcal{C}(\bar{t})}|} \geq s$. Hence $\mathcal{C}(\bar{t}) \Rightarrow H'$ is the required rule. Indeed, $\mathcal{C}(\bar{t}) \Rightarrow H'$ is not trivial since $B \Rightarrow H$ is not trivial: note that for each i, $1 \leq i \leq k-1$, we have that $\mathbf{dom}(J_i = \bar{t}[J_i], T) \subseteq \mathbf{dom}(C_i, T)$ and, furthermore, $\mathbf{dom}(J_H \neq u_H) \subseteq \mathbf{dom}(C_k, T)$.
($\Leftarrow$) Straightforward. □

Example 4.6 To illustrate Lemma 4.5, consider the simple example database $\mathbf{DB}_4$, in Figure 7. Consider the rule R

$$\mathbf{A} \in [1, 2] \Rightarrow \mathbf{B} \in [1, 2]$$

for which we have $cnf(R, \mathbf{DB}_4) = 0.75$. The element $u = 3$ is such that

$$u \in (\mathbf{dom}(\mathbf{B}, \mathbf{DB}_4) - \mathbf{dom}(\mathbf{B} \in [1, 2], \mathbf{DB}_4))$$

By Lemma 4.5, the set S of rules of the form

$$\mathbf{A} = x \Rightarrow \mathbf{B} \neq 3$$

for $x \in \mathbf{dom}(\mathbf{A}, \mathbf{DB}_4)$, contains at least a rule R' such that $cnf(R', \mathbf{DB}_4) \geq 0.75$. Indeed, S includes the following rules

$$\mathbf{A} = 1 \quad \Rightarrow \quad \mathbf{B} \neq 3, \tag{7}$$

$$\mathbf{A} = 2 \quad \Rightarrow \quad \mathbf{B} \neq 3, \tag{8}$$

$$\mathbf{A} = 3 \quad \Rightarrow \quad \mathbf{B} \neq 3, \tag{9}$$

among which both rule 7 and rule 9 score a confidence value equal to 1.

An immediate consequence of Lemma 4.5 is that the problem $\langle I, T, cnf, k, s\rangle$ on databases without nulls is polynomial-time decidable. Indeed, it suffices to guess all the conditions of the form illustrated above, whose number is polynomially bounded, and then check if at least one of these rules scores enough confidence on the input database. Nevertheless, we are able to provide a tighter bound on the complexity of this problem, as stated by the following theorem.

Theorem 4.7 *Given a database T without nulls, the problem $\langle I, T, cnf, k, s\rangle$ is in* TC^0.

Proof. Let $I = \{I_1, \ldots, I_m\}$ be a set of attributes, and $T = \{t_1, \ldots, t_n\}$ be a database without nulls on I. In the rest of the proof, whenever we use the subscripts i, i', j, j', j'' and h, we assume that $1 \leq i, i' \leq m$, $1 \leq j, j', j'' \leq n$, and $1 \leq h \leq \lceil \log_2 n \rceil$. Let $U(n)$ denote the set $\{0, 1, \ldots, n-1\}$ of natural numbers. Let $e_i : \mathbf{dom}(I_i, T) \mapsto U(n)$ an injective function. Given $y \in U(n)$, we denote by $b_h(y)$ the h-th bit of the binary encoding of y. W.l.o.g., we can assume that T is encoded as a $m \times n \times \lceil \log_2 n \rceil$ matrix $en(T)$ of bits, whose elements $x_{i,j,h}$ are such that $x_{i,j,h} = b_h(e_i(t_j[I_i]))$. Now, we build a logspace uniform family $\{C_{m,n}\}$[1] of circuits of polynomial size and constant depth, with AND, OR, and MAJORITY gates of unbounded fan-in. The circuit $C_{m,n}$ takes as input $en(T)$ and will output 1 iff $\langle I, T, cnf, k, s\rangle$ is a YES instance. A circuit $C_{m,n}$ is constituted of a set of TC^0 circuits $R(i, j, 1), \ldots, R(i, j, n)$. A circuit $R(i, j, j')$ takes in input $en(T)$ and outputs 1 iff the rule $\chi_{i,j,j'} \equiv (\chi_{i,j} \Rightarrow (I_i \neq t_{j'}[I_i]))$ scores enough confidence, where

$$\chi_{i,j} \equiv \left(\bigwedge_{i' \in \{1,\ldots,i-1,i+1,\ldots,m\}} (I_{i'} = t_j[I_{i'}]) \right)$$

Thus, each circuit $R(i, j, j')$ is introduced in order to guess one of the rules of the form stated by Lemma 4.5. In particular, i and j' identify the attribute I_i and the value $t_{j'}[I_i]$ appearing in head of the rule, respectively, while j identifies the values $t_j[I_{i'}]$ ($i' \neq i$) appearing in the body of the rule. The output of $C_{m,n}$ is obtained by wiring the output of the circuits $R(i, j, j')$ through an OR gate. To conclude the proof, we will have to show that the circuits $R(i, j, j')$ are in TC^0. In the following we denote by $\overline{\chi}_{i,j,j'}$ the rule $\chi_{i,j} \Rightarrow I_i = (t_{j'}[I_i])$.

Claim 4.8 *For each condition $\chi_{i,j}$ (rule $\overline{\chi}_{i,j,j'}$, respectively) there is a family $\{count(\chi_{i,j})_{m,n}\}$ ($\{count(\overline{\chi}_{i,j,j'})_{m,n}\}$, respectively) of $\#\mathrm{AC}^0$ circuits computing $|T_{\chi_{i,j}}|$ ($|T_{\overline{\chi}_{i,j,j'}}|$, respectively) over any input $en(T)$.*

[1] Note that in this theorem and in the following, by a little abuse of notation, and for simplicity, we denote a circuit family recognizing inputs in the form of a $m \times n \times \lceil \log_2 n \rceil$ boolean matrix by using the subscript m, n instead of the usual subscript C_i where i denotes the input size

Proof. Consider an atomic condition of the form $I_{i'} = t_j[I_{i'}]$ and a generic tuple $t_{j''}$ of T. We define $S(j, i', j'')$ as the $\#\mathrm{AC}^0$ circuit having $2\lceil \log_2 n \rceil$ binary inputs, $x_{i',j,1}, \ldots, x_{i',j,\lceil \log_2 n \rceil}$ (the encoding of $t_j[I_{i'}]$ in $en(T)$), and $x_{i',j'',1}, \ldots, x_{i',j'',\lceil \log_2 n \rceil}$ (the encoding of $t_{j''}[I_{i'}]$ in $en(T)$), and computing the following function:

$$\prod_{h=1}^{\lceil \log_2 n \rceil} \left(x_{i',j,h} \times x_{i',j'',h} + (1 - x_{i',j,h}) \times (1 - x_{i',j'',h})\right).$$

It is immediate to verify that the circuit $S(j, i', j'')$ outputs 1 if $t_{j''}[I_{i'}] = t_j[I_{i'}]$, i.e. if $t_{j''} \vdash (I_{i'} = t_j[I_{i'}])$, and 0 otherwise. Thus, the circuit $Q(j, i, j'')$ computing the function

$$S(j, 1, j'') \times \ldots \times S(j, i-1, j'') \times S(j, i+1, j'') \times \ldots \times S(j, m, j'')$$

outputs 1 if $t_{j''} \vdash \chi_{i,j}$, and 0 otherwise. To conclude, we can build a circuit $count(\chi_{i,j})_{m,n}$ as

$$\sum_{j''=1}^{n} Q(j, i, j'')$$

and a circuit $count(\overline{\chi}_{i,j,i'})_{m,n}$ as

$$\sum_{j''=1}^{n} \left(Q(j, i, j'') \times S(j', i, j'')\right).$$

□

Claim 4.9 *For each rule $\chi_{i,j,j'}$ there is a constant-depth polynomial size uniform family $\{R(i, j, j')_{m,n}\}$ of circuits of unbounded fan-in AND, OR and MAJORITY gates, such that $R(i, j, j')_{m,n}$ outputs 1 iff $cnf(\chi_{i,j,j'}, T) \geq s$, when the input database has size $m \times n$.*

Proof. We recall that s, $0 \leq s \leq 1$, is a rational number. So, there are two natural numbers a and b, $b \geq a$, such that $s = a/b$. W.l.o.g., we can assume that s is encoded as a pair of natural numbers of the form (a', b), with $a' = b - a$. Consider the function

$$f(\chi_{i,j,j'}, T, s) = (a'|T_{\chi_{i,j}}| + 1) - b|T_{\overline{\chi}_{i,j,j'}}|$$

taking values over integers. As $|T_{\chi_{i,j,j''}}| = |T_{\chi_{i,j}}| - |T_{\overline{\chi}_{i,j,j'}}|$, we have that $cnf(\chi_{i,j,j'}, T) \geq s$ iff $f(\chi_{i,j,j'}, T, s) > 0$. We recall the following result [5]: for each integer N there is a log-time uniform $\#\mathrm{AC}^0$ circuit, which computes N, when the binary representation of N is given in input. Call this circuit $number(N)$. Since a' and b are integers, we can build two $\#\mathrm{AC}^0$ circuits computing the functions $a'|T_{\chi_{i,j}}|$ and $b|T_{\overline{\chi}_{i,j,j'}}|$, connecting $number(a')$ to $count(\chi_{i,j})_{m,n}$ and $number(b)$ to $count(\overline{\chi}_{i,j,j'})_{m,n}$. By Definition 2.22, the function $f(\chi_{i,j,j'}, T, s)$ is in the class $Gap\mathrm{AC}^0$, and the language

$$\{\chi_{i,j,j'} \mid cnf(\chi_{i,j,j'}, T) \geq s\}$$

is in the class $P\mathrm{AC}^0$ which coincides with TC^0 under log-space uniformity [1, 5]. Thus, there is a constant-depth polynomial size uniform family $\{C(i, j, j')_{m,n}\}$ of circuits of unbounded fan-in AND, OR and MAJORITY gates, such that $R(i, j, j')_{m,n}$ outputs 1 iff $sup(\chi_{i,j,j'}, T) \geq s$, when the input database has size $m \times n$. □

The number of such circuits $R(i, j, j')$ is mn^2, which is polynomial in $m \times n$, hence $C_{m,n}$ has constant depth and polynomial size as well. We observe that $C_{m,n}$ may be easily generated using logarithmic space. Thus, this proves that the problem $\langle I, T, cnf, k, s\rangle$ on databases without nulls is in TC^0 under the general complexity measure. □

4.3 Confidence-problems for databases with nulls

Proposition 4.10 *Consider the set of problems $\mathcal{P} = \{\langle I, T, \rho, k, s\rangle\}$, where $\rho \in \{cnf, gain_\theta, laplace_h\}$. If there is an association rule $B \Rightarrow H$ that is a solution for a problem $P \in \mathcal{P}$, then the rule $B' \Rightarrow H'$ also solves P, where $B' \wedge H' = B \wedge H$ and $|H'| = 1$.*

Proof. The proof is straightforward and thus omitted. □

Theorem 4.11 *Given a database T with nulls, the complexity of $\langle I, T, cnf, k, s\rangle$ is* NP*-complete.*

Proof of Theorem 4.11. (*Hardness*) The proof, as in Theorem 4.2, is by reduction of CLIQUE. Let $G = (V, E)$ be an undirected graph, with set of nodes $V = \{v_1, \ldots, v_n\}$ and set of edges $E = \{e_1 = \{v_{p_1}, v_{q_1}\}, \ldots, e_m = \{v_{p_m}, v_{q_m}\}\}$. Let h be an integer. We build an instance $\langle I^{clq}, T^{clq}, cnf, k, s\rangle$ as follows.

- Let I^{clq} be $I' \cup \{I_{n+1}\}$, where $I' = \{I_1, \ldots, I_n\}$. I_j will represents the node v_j of G, for $j = 1, \ldots, n$, and I_{n+1} is a new attribute representing a dummy node v_{n+1};
- Let $T^{clq} = T' \cup T''$, where T' includes the tuples t_{e_i} and t'_{e_i}, where $t_{e_i}[I_j] = \epsilon$ (respectively $t'_{e_i}[I_j] = \epsilon$) if $v_j \in e_i$, and $t_{e_i}[I_j] = 1$ (respectively $t'_{e_i}[I_j] = 1$) otherwise, for $i = 1, \ldots, m$, $j = 1, \ldots, n+1$, (the tuples t_{e_i} and t'_{e_i} both denote the edge e_i of G). Furthermore, T'' includes the tuples t_{v_i}, where $t_{v_i}[I_j] = \epsilon$ if $i = j$, and $t_{v_i}[I_j] = 1$ otherwise, for $i = 1, \ldots, n+1$, $j = 1, \ldots, n+1$.
- Let $k = n - h + 1$;
- Let $s = \frac{h^2}{h^2+1}$.

See Figure 7 for an example of this reduction. We have the following Claim.

Claim 4.12 *Let C be a condition on a subset of I', then*

1. $|T'_C| \leq 2\binom{n-|C|}{2}$, *and*
2. $|T''_C| \leq n + 1 - |C|$.

Proof of Claim 4.12. Point 1 follows by considering that T' contains two tuples for each edge of G, i.e. T' contains at most $2\binom{n}{2}$ tuples. A condition containing an atomic condition on a generic attribute I_j, where $1 \leq j \leq n+1$ is not satisfied by the set of tuples $\{t_{v,v_j} | v \in V\} \cup \{t'_{v,v_j} | v \in V\}$. In general, a condition of length l on T' satisfies just a subset of T' representing a subgraph of G of size $n - l$. As for point 2, consider that a condition containing an atomic condition on a generic attribute I_j, where $1 \leq j \leq n+1$ is not satisfied by the tuple t_{v_j}. In general, a condition of length l can match only $n + 1 - l$ tuples of T''. □

Next, we prove that there is a clique of size h in G iff $\langle I^{clq}, T^{clq}, cnf, k, s\rangle$ is a YES instance. ($\Rightarrow$) Let $C = \{v_{r_1}, \ldots, v_{r_h}\}$ be a clique of size h in G. Consider the condition $B = \left(\bigwedge_{v_j \in (V-C)} (I_j = 1)\right)$ such that $|B| = n - h$. By definition of clique, there exist $\frac{h(h-1)}{2}$ edges of G connecting nodes in C. Now,

$$T'_B = \left\{ t_{\{v_{r_x}, v_{r_y}\}}, t'_{\{v_{r_x}, v_{r_y}\}} \mid 1 \leq x < y \leq h \right\}.$$

Thus, $|T'_B| = 2\binom{n-|B|}{2} = h(h-1)$, whereas $|T''_B| = n+1-|B| = h+1$. Hence, $|T^{clq}_B| = h(h-1)+(h+1) = h^2+1$, and

$$cnf(B \Rightarrow (I_{n+1}=1), T^{clq}) = \frac{|T^{clq}_{B\wedge(I_{n+1}=1)}|}{|T^{clq}_B|} = \frac{|T^{clq}_B|-1}{|T^{clq}_B|} = \frac{h^2}{h^2+1}.$$

($\Leftarrow$) By Proposition 4.10, if $\langle I^{clq}, T^{clq}, cnf, n-h+1, \frac{h^2}{h^2+1}\rangle$ is a YES instance, then there is a rule $R \equiv B \Rightarrow H$ on I^{clq}, where $|H| = 1$, such that $|B| \geq n-h$.

Note that, if R would contain an atomic condition of the form $I_j \neq 1$, $cnf(R, T^{clq})$ would be 0. Hence, only atomic conditions of the form $I_j = 1$ can appear in R.

The content of T'' implies that there is no association rule having confidence 1 on T^{clq}. Furthermore, we have that $|T^{clq}_B| \geq h^2+1$, otherwise the ratio $\frac{|T^{clq}_{B\wedge H}|}{|T^{clq}_B|}$ would not be[2] greater than or equal to $\frac{h^2}{h^2+1}$. Two cases have to be considered:

1. $I_{n+1} \notin \mathbf{att}(B)$;
2. $I_{n+1} \in \mathbf{att}(B)$.

(*Case 1*) Assume that $\mathbf{att}(B) \subseteq I'$. Then $|T^{clq}_B| \geq h^2+1$ implies that $|B| \leq n-h$, and we have already noticed that $|B| \geq n-h$. Thus $|B| = n-h$ and $|T^{clq}_B| = h^2+1$. Let $I' - \mathbf{att}(B) = \{I_{r_1}, \ldots, I_{r_h}\}$. Since $|B| = n-h$, then $|T''_B| = h+1$, whereas, in order to be $|T'_B| = h(h-1)$ it is necessary that

$$T'_B = \left\{ t_{\{v_{r_x}, v_{r_y}\}}, t'_{\{v_{r_x}, v_{r_y}\}} \mid 1 \leq x < y \leq k \right\},$$

i.e. the nodes $v_{r_1}, \ldots, v_{r_h}$ form a clique of G having size h.

(*Case 2*) Suppose that $B = B' \wedge (I_{n+1} = 1)$. Then $|T^{clq}_B| \geq h^2+1$ implies that $|B'| \leq n-h-1$, and we have already noticed that $|B| \geq n-h$, i.e. $|B'| \geq n-h-1$. Thus $|B'| = n-h-1$ and (by recalling Proposition 4.12)

$$h^2+1 \leq |T^{clq}_B| \leq 2\binom{n-|B'|}{2} + (n+1-|B'|) = h^2+2h+2.$$

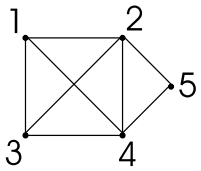
We can show that there is no tuple $t \in T'$ such that $t \not\vdash H$ and $t \vdash B$. Assume, by contradiction, that such a tuple $t \in T'$ exists. Then $|T^{clq}_{B\wedge H}| \leq |T'_B| - 2 + |T''_B| - 1$. This implies that the confidence of the association rule $B \Rightarrow H$ cannot be greater than or equal to $\frac{h^2}{h^2+1}$, since, for each h,

$$\frac{|T^{clq}_B|-3}{|T^{clq}_B|} \leq \frac{(h^2+2h+2)-3}{h^2+2h+2} < \frac{h^2}{h^2+1}$$

Indeed, simply note that, assuming $h \geq 1$ we have

$$\begin{aligned} \frac{(h^2+2h+2)-3}{h^2+2h+2} &< \frac{h^2}{h^2+1} \Rightarrow \\ \frac{h^2}{h^2+1} - \frac{(h^2+2h+2)-3}{h^2+2h+2} &> 0 \Rightarrow \\ 2h^2-2h+1 &> 0. \end{aligned}$$

[2]Consider the inequality $\frac{m}{m+1} \geq \frac{m-1}{m}$.



	I_1	I_2	I_3	I_4	I_5	I_6
$t_{\{v_1,v_2\}}$	ϵ	ϵ	1	1	1	1
$t'_{\{v_1,v_2\}}$	ϵ	ϵ	1	1	1	1
$t_{\{v_1,v_3\}}$	ϵ	1	ϵ	1	1	1
$t'_{\{v_1,v_3\}}$	ϵ	1	ϵ	1	1	1
$t_{\{v_1,v_4\}}$	ϵ	1	1	ϵ	1	1
$t'_{\{v_1,v_4\}}$	ϵ	1	1	ϵ	1	1
$t_{\{v_2,v_3\}}$	1	ϵ	ϵ	1	1	1
$t'_{\{v_2,v_3\}}$	1	ϵ	ϵ	1	1	1
$t_{\{v_2,v_4\}}$	1	ϵ	1	ϵ	1	1
$t'_{\{v_2,v_4\}}$	1	ϵ	1	ϵ	1	1
$t_{\{v_3,v_4\}}$	1	1	ϵ	ϵ	1	1
$t'_{\{v_3,v_4\}}$	1	1	ϵ	ϵ	1	1
$t_{\{v_3,v_5\}}$	1	1	ϵ	1	ϵ	1
$t'_{\{v_3,v_5\}}$	1	1	ϵ	1	ϵ	1
$t_{\{v_4,v_5\}}$	1	1	1	ϵ	ϵ	1
$t'_{\{v_4,v_5\}}$	1	1	1	ϵ	ϵ	1
t_{v_1}	ϵ	1	1	1	1	1
t_{v_2}	1	ϵ	1	1	1	1
t_{v_3}	1	1	ϵ	1	1	1
t_{v_4}	1	1	1	ϵ	1	1
t_{v_5}	1	1	1	1	ϵ	1
t_{v_6}	1	1	1	1	1	ϵ

Figure 8: An example of the reduction used in Theorem 4.11

Thus, H is such that $|T'_{B\wedge H}| = |T'_B| = |T'_{B'}| = |T'_{B'\wedge H}|$. Since $|T_B^{clq}| \geq h^2+1$ and, by Fact 4.12, we know that $|T_{B'\wedge H}^{clq}| \leq h^2+1$ (note that $|B'\wedge H| = n-h$), it follows that $|T_{B'\wedge H}^{clq}| = |T_B^{clq}| = h^2+1$. Let $I' - \mathbf{att}(B\wedge H) = \{I_{r_1},\ldots,I_{r_h}\}$. Hence

$$T'_B = \left\{ t_{\{v_{r_x},v_{r_y}\}}, t'_{\{v_{r_x},v_{r_y}\}} \mid 1 \leq x < y \leq h \right\},$$

i.e., the nodes $v_{r_1},\ldots,v_{r_h}$ form a clique of G having size h.

(*Membership*) A certificate of membership in NP is given by an association rule $B \Rightarrow H$ on I. This can be checked in polynomial time by verifying that $B \Rightarrow H$ is nontrivial, $|B\wedge H| \geq k$, and $cnf(B\Rightarrow H, T) \geq s$. □

4.4 Gain- and Laplace-problems

Despite their syntactical similarity to confidence, the laplace and the gain index "behave" more similarly to support than to confidence. For instance, consider a rule $B \Rightarrow H$ and a database T. The h-laplace index, $\frac{|T_{B\wedge H}|+1}{|T_B|+h}$, reaches its maximum

value when $|T_{B\wedge H}| = |T_B|$. Assume this is indeed the case, then, for $laplace_h(B \Rightarrow H, T) \geq s$ to hold, it must be the case that $|T_{B\wedge H}| \geq \frac{hs-1}{1-s}$. Thus, if we search for a rule scoring a high value of h-laplace, that is when s approaches 1, it is the case that $|T_{B\wedge H}|$ must approach to ∞ and consequently that $sup(B \Rightarrow H, T)$ must approach 1, or, in other words, that the rule must simultaneously score a high h-laplace value and a high support value. This argument is exploited within the proof of the following Theorem.

Theorem 4.13 *Let T be a database without nulls. Then the complexity of $\langle I, T, \rho, k, s\rangle$, with $\rho \in \{gain_\theta, laplace_h\}$, is* NP-*complete.*

Proof of Theorem 4.13. As for the hardness part of the proof, we follow the same line of reasoning as in Theorem 4.11, using a reduction of CLIQUE. Let $G = (V, E)$ be an undirected graph, consisting of a set of nodes $V = \{v_1, \ldots, v_n\}$ and of a set of edges $E = \{e_1 = \{v_{p_1}, v_{q_1}\}, \ldots, e_m = \{v_{p_m}, v_{q_m}\}\}$, and let ψ be an integer. We build an instance $\langle I^{clq}, T^{clq}, gain_\theta, k, s\rangle$ $\big(\langle I^{clq}, T^{clq}, laplace_h, k, s\rangle$ respectively$\big)$ as in Theorem 4.11, but in this case I^{clq}, T^{clq}, k and s are constructed this way:

- I^{clq} is the set of attributes $I_1, \ldots I_n, I_{n+1}$, where I_j denotes the node v_j of G $(j = 1, \ldots, n)$ and I_{n+1} is an additional attribute;
- T^{clq} includes the tuples t_{e_i}, t'_{e_i} s.t. $t_{e_i}[I_j] = t'_{e_i}[I_j] = 0$ if $v_j \in e_i$, and 1 otherwise, where t_{e_i} and t'_{e_i} both denote the edge e_i of G, for each $i = 1, \ldots, m$. Furthermore, T^{clq} contains the tuple t_0, s.t. $t_0[I_j] = 0$, for each $j = 1, \ldots, n + 1$;
- k is set to $n - \psi + 1$;
- s is set to $\frac{(1-\theta)\psi(\psi-1)}{2m+1}$ $\Big(\frac{\psi(\psi-1)+1}{\psi(\psi-1)+h}$ respectively$\Big)$.

(*Membership*) The membership of the problem in NP can be proven following a proof similar of that of Theorem 4.2. □

Corollary 4.14 *Let T be a database with nulls. Then the complexity of $\langle I, T, \rho, k, s\rangle$, where $\rho \in \{gain_\theta, laplace_h\}$, is* NP-*complete.*

Proof of Corollary 4.14. Hardness is proved by Theorem 4.13. Membership in NP is immediate. □

This closes the complexity analysis of the general association rules induction problems. In the following sections we shall analyze several interesting special cases thereof.

5 Sparse databases

There are many real world applications characterized by sparse databases. As an example, consider a database of transactions of a large store constructed for basket analysis purposes, where we have a large set of items (attributes), but a small set thereof involved in each single transaction (tuples). For databases showing this property, complexity figures are quite different from what we have seen above for the general case.

```
Input: a set of attributes I, a sparse database T, an integer k, a rational s
begin
   for i := 1 to |T| do
      if |t_i| ≥ k then
         for guess := 1 to 2^|t_i| − 1 do
            if guess has exactly k bits set to 1 then begin
               count := 0;
               for j := 1 to |T| do
                  if SATISFIES(t_j, guess, t_i) then count := count + 1;
               if count ≥ s|T| then return "yes";
            end;
   return "no";
end.

function SATISFIES(v, guess, u) : boolean;
begin
   p := 1;
   for q := 1 to |I| do
      if u[A_q] = c(A_q) then begin
         if guess[p] = 1 and v[A_q] = ε then return false;
         p := p + 1;
      end;
   return true;
end; { SATISFIES }
```

Figure 9: The algorithm of Theorem 5.1

Theorem 5.1 *Let T be a sparse database. Then the complexity of $\langle I, T, sup, k, s\rangle$ is in L.*

Proof of Theorem 5.1. We build a Turing Machine $\mathcal{T}$ employing $\mathcal{O}(\log(\max\{|I|, |T|\}))$ space, which decides $\langle I, T, sup, k, s\rangle$. Let $T = \{t_1, \ldots, t_m\}$, and let $I = \{A_1, \ldots, A_n\}$. Let $guess$ be a counter. Let $guess[p]$ denote the value of the p-th bit of $guess$, $1 \leq p \leq \lceil \log_2(guess) \rceil$. The algorithm implemented by $\mathcal{T}$ is depicted in Figure 9. $\mathcal{T}$ works as follows: each tuple t_i is considered, using the counter i, and only those conditions which can satisfied by t_i are considered. It is not necessary to represent each guessed condition explicitly; the counter $guess$ is employed instead: the p-th bit of $guess$ tells whether the p-th non null attribute value occurring in t_i belongs to the current guessed condition or not. Each guessed condition is then tested on each transaction t_j of T, using the counter j. The counter $count$ takes into account the number of tuples satisfying the current guessed condition. It is straightforward to note that the space employed corresponds to the space needed to store the variables i, j, $count$, p, q and $guess$. On the assumption that T is sparse, i, j and $count$ need $\mathcal{O}(\log |T|)$ space, whereas p, q and $guess$ need $\mathcal{O}(\log |I|)$ space. Finally, verifying whether $guess$ has at least k bits set to 1 can be easily done in logarithmic space. □

Theorem 5.2 *Let T be a sparse database. Then the complexity of $\langle I, T, \rho, k, s\rangle$, where $\rho \in \{cnf, gain_\theta, laplace_h\}$ is in L.*

Proof of Theorem 5.2. The proof follows the same line of reasoning as Theorem 5.1. In this case, two disjoint current conditions are needed (which represent the body and the head of the current association rule, respectively), and some further auxiliary logspace counters. □

6 Fixed schema complexity

In this section we improve the result reported in [26], which states the polynomial-time solvability of the association rule mining problem under the fixed schema complexity measure. For the sake of simplicity, we shall consider only the case of numerical attributes. The same results can be shown, however, using analogous proofs lines, in the other cases.

Theorem 6.1 *Let I be a set of numerical attributes. Then the fixed schema complexity of the problem $\langle I, T, sup, k, s\rangle$ is in L.*

Proof of Theorem 6.1. (*Sketch*) Let $n = |I|$, and let $m = |T|$. We can build a Turing Machine $\mathcal{T}$ employing $\mathcal{O}(\log m)$ space, which solves $\langle I, T, sup, k, s\rangle$. $\mathcal{T}$ uses $2n$ pointers p_j^l, p_j^u, to $2n$ tuples of T, of size $\mathcal{O}(\log m)$ each, and $2n$ bits o_j and i_j, for each $j = 1, \ldots, n$. An *arrangement* of $\mathcal{T}$ is a $4n$-tuple

$$(p_1^l, p_1^u, \ldots, p_n^l, p_n^u, o_1, \ldots, o_n, i_1, \ldots, i_n) \in \{1, \ldots, m\}^{2n} \times \{0, 1\}^{2n}.$$

Let t_i denote the i-th tuple of T; define $\theta(0)$ as "$\in$", $\theta(1)$ as "$\notin$", and C_j as the condition

$$I_j \ \theta(o_j) \ [t_{p_j^l}[I_j], t_{p_j^u}[I_j]]$$

for each $j = 1, \ldots, n$. An arrangement is intended in order to encode the currently guessed condition. A condition C_j will belong to the currently guessed condition if $i_j = 1$, for each $j = 1 \ldots n$. $\mathcal{T}$ works as follows: it scans, one after another, all the possible arrangements; for each candidate arrangement it checks whether it encodes a valid condition, having length at least k, and, if this succeeds, it is verified that $|T_C| \geq s|T|$, where

$$C = \bigwedge_{\substack{j=1\ldots n \\ i_j=1}} C_j.$$

We note that $\mathcal{T}$ needs an additional amount of space, to store counters and auxiliary pointers, which is logarithmic w.r.t. the input size. □

Theorem 6.2 *The fixed schema complexity of the problems $\langle I, T, \rho, k, s\rangle$, where $\rho \in \{cnf, gain_\theta, laplace_h\}$ is in L.*

Proof of Theorem 6.2. (*Sketch*) The proof uses the same line of reasoning as in Theorem 6.1. Let $n = |I|$, and let $m = |T|$. As above, we build a Turing Machine $\mathcal{T}$ employing $\mathcal{O}(\log m)$ space, which solves $\langle I, T, \rho, k, s\rangle$, where $\rho \in \{cnf, gain_\theta, laplace_h\}$. For each currently guessed rule $B \Rightarrow C$, $\mathcal{T}$ must verify that, respectively:

- $|T_{B\wedge C}| \geq s|T_B|$ if $\rho = cnf$;
- $|T_{B\wedge C}| \geq s|T| + \theta|T_B|$ if $\rho = gain_\theta$;
- $|T_{B\wedge C}| + 1 \geq s(|T_b| + h)$ if $\rho = laplace_h$.

As in Theorem 6.1, $\mathcal{T}$ employs a fixed number of log-space counters, in order to carry this out. □

7 Further complexity results

This section studies the computational complexity of several interesting special cases of mining association rules. Most of these cases assume some parameters (e.g., the lower bound on the rule length k, the index value threshold s) of the general association rule mining problem to be given constants. The relevance of the analysis we present below is two-fold. First, it contributes to understand actual complexity sources. Second, from a practical point of view, users are often interested in solving such simplified tasks, as, for instance, when one wishes to mine only rules with a support always larger than 0.75.

7.1 Support-problems with fixed threshold

As stated below, the rule mining problem remains very hard to solve even if the support thresholds is not part of the input.

Theorem 7.1 *The problem $\langle I, T, sup, k, s\rangle$ where s is a fixed constant in $(0, 1)$, and T is a database with nulls is* NP*-complete.*

Proof. (*Hardness*) The proof is by reduction of CLIQUE. Let $G = (V, E)$ be an undirected graph, consisting of a set of nodes $V = \{v_1, \ldots, v_n\}$ and set of edges $E = \{(v_{p_1}, v_{q_1}), \ldots, (v_{p_m}, v_{q_m})\}$. Let h be an integer. We build a corresponding instance $\langle I^{clq}, T^{clq}, sup, k, s\rangle$ as follows:

1. let I^{clq} be the set consisting of the attributes $I_1, \ldots I_n, I_{n+1}$, where I_j represents the node v_j of G, for $j = 1, \ldots, n$ and I_{n+1} is an additional attribute;
2. Let T^{clq} be a set built as the union of the following sets of tuples:
   - T^G, which, for each edge (v_{p_i}, v_{q_i}) of G ($i = 1, \ldots, m$) includes a tuple t_i such that $t_i[I_{p_i}] = t_i[I_{q_i}] = t_i[I_{n+1}] = \epsilon$, whereas $t_i[I_j] = 1$ otherwise ($j = 1, \ldots, n + 1$).
   - T^0, including c_0 copies of a tuple t such that $t[I_{n+1}] = 1$, and $t[I_j] = \epsilon$ otherwise ($j = 1, \ldots, n$), where c_0 is a value to be defined next;
   - T^1, consisting of c_1 copies of a tuple t such that $t[I_{n+1}] = \epsilon$, and $t[I_j] = 1$ otherwise ($j = 1, \ldots, n$), where c_1 is a value to be defined next.
3. let $k = n - h$;

As for the values c_0 and c_1 we choose two nonnegative integer values such that

$$s = \frac{\frac{h(h-1)}{2} + c_1}{m + c_0 + c_1}.$$

It can be shown that such two values exist, and are both polynomial bounded in m. Indeed, let $\alpha = h(h-1)/2$, and $s = ax/(bx)$: we have

$$\frac{ax}{bx} = \frac{\alpha + c_1}{m + c_0 + c_1}.$$

where a, b and x are positive integers and $a < b$. Thus, $c_0 = ax - \alpha$ and $c_1 = bx - m - (ax - \alpha)$. Setting x equal to, e.g., $m + \alpha$, yields the two required values. If can be shown by using the same argumentation of Theorem 4.2, that there is a clique of size h in G iff $\langle I^{clq}, T^{clq}, sup, n - h, s\rangle$ is a YES instance.
(*Membership*) Same as Theorem 4.4. □

Example 7.2 The following example shows how reduction from CLIQUE to $\langle I^{clq}, T^{clq}, sup, k, s\rangle$, used in Theorem 7.1, is applied to a given CLIQUE instance. Assume graph G of Figure 10 is given, and that we want to build a corresponding instance of $\langle I^{clq}, T^{clq}, sup, k, \frac{1}{2}\rangle$, such that G has a clique of size 3 iff $\langle I^{clq}, T^{clq}, sup, k, \frac{1}{2}\rangle$ is a YES instance. Note that G has 5 nodes and 8 edges. Thus:

1. I^{clq} is $\{I_1, \ldots, I_6\}$;
2. T^{clq} is a set composed by the union of the following sets of tuples:
   - T^G, which, for each edge (v_{p_i}, v_{q_i}) of G ($i = 1, \ldots, 8$) includes a tuple t_i such that $t_i[I_{p_i}] = t_i[I_{q_i}] = t_i[I_6] = \epsilon$, whereas $t_i[I_j] = 1$ otherwise ($j = 1, \ldots, 5$).
   - T^0, including c_0 copies of a tuple t such that $t[I_6] = 1$, and $t[I_j] = \epsilon$ otherwise ($j = 1, \ldots, 5$), where $c_0 = 3$;
   - T^1, consisting of c_1 copies of a tuple t such that $t[I_6] = \epsilon$, and $t[I_j] = 1$ otherwise ($j = 1, \ldots, 5$), where $c_1 = 5$.
3. let $k = 5 - 3 = 2$;

Note that c_0 and c_1 are such that

$$\frac{1}{2} = \frac{\frac{3(3-1)}{2} + c_1}{8 + c_1 + c_0}.$$

The resulting database T^{clq} is shown in Figure 10.

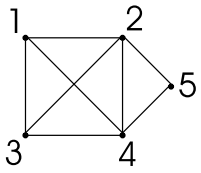
Remark. Note that the special case $\langle I, T, sup, k, 1\rangle$ can be easily shown to be in P.

7.2 Support-problems with fixed thresholds on boolean databases

Lemma 7.3 *Let C be a condition on a set of boolean attributes. Then there is a family $\{count(C)_{m,n}\}$ [3] of $\#\mathrm{AC}^0_2$ circuits computing $|T_C|$ over any input database T defined on a set of boolean attributes I such that $I \supseteq \mathbf{att}(C)$.*

Proof. Let $\mathbf{att}(C) \subseteq I = \{A_1, \ldots, A_n\}$. We define the family $\{count(C)_{m,n}\}$ of $\#\mathrm{AC}^0_2$ circuits as follows. The circuit $count(C)_{m,n}$ has $m \times n$ binary inputs $x_{i,j}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, with $m = |T|$ and $n = |I|$. The input $x_{i,j}$ is 1 if $t_i[A_j] = c(A_j)$, 0 otherwise (i.e. if $t_i[A_j] = \epsilon$). The first level of $count(C)_{m,n}$ consists of m $\times$-gates G_i, for $i = 1, \ldots, m$. Each gate G_i receives the $|C|$ inputs $\{x_{i,k} \mid A_k \in \mathbf{att}(C)\}$. Thus the output of G_i is 1 iff

[3] Note that here and elsewhere, by little abuse of notation, and for simplicity, we denote a circuit family recognizing inputs in the form of a $m \times n$ boolean matrix by using the subscript m, n instead of the usual subscript C_i where i denotes the input size



	I_1	I_2	I_3	I_4	I_5	I_6
$t_{\{v_1,v_2\}}$	ϵ	ϵ	1	1	1	ϵ
$t_{\{v_1,v_3\}}$	ϵ	1	ϵ	1	1	ϵ
$t_{\{v_1,v_4\}}$	ϵ	1	1	ϵ	1	ϵ
$t_{\{v_2,v_3\}}$	1	ϵ	ϵ	1	1	ϵ
$t_{\{v_2,v_4\}}$	1	ϵ	1	ϵ	1	ϵ
$t_{\{v_3,v_4\}}$	1	1	ϵ	ϵ	1	ϵ
$t_{\{v_3,v_5\}}$	1	1	ϵ	1	ϵ	ϵ
$t_{\{v_4,v_5\}}$	1	1	1	ϵ	ϵ	ϵ
t_1	ϵ	ϵ	ϵ	ϵ	ϵ	1
...						
t_8	ϵ	ϵ	ϵ	ϵ	ϵ	1
t_9	1	1	1	1	1	ϵ
...						
t_{20}	1	1	1	1	1	ϵ

Figure 10: An example of the reduction used in Theorem 7.1

$t_i \vdash C$. The second level of $count(C)_{m,n}$ consists of a single $+$-gate receiving in input the outputs of all the G_i gates, for $i = 1, \ldots, m$. Thus the circuit $count(C)_{m,n}$ calculates $|T_C|$ when the input has size $m \times n$. □

The following Theorems (7.4, 7.7 and 7.8) associate some task related to mining association rules to very low complexity classes such as TC^0 and AC^0. It turns out that these problems are highly parallelizable (recall that $\mathrm{AC}^0 \subset \mathrm{TC}^0 \subseteq NC^1$, [16]).

Theorem 7.4 *Let I be a set of boolean attributes, and let k be a fixed constant. Then the complexity of $\langle I, T, sup, k, s\rangle$ is in TC^0.*

Proof. Exploiting Lemma 7.3 and using the same argumentation of Claim 4.9, it can be shown that the language

$$\{B \Rightarrow H \text{ on } I \mid sup(B \Rightarrow H, T) \geq s\}$$

is in the class TC^0. Thus, there is a constant-depth polynomial size uniform family $\{C'(I_R)_{m,n}\}$ of circuits of unbounded fan-in AND, OR and MAJORITY gates, such that $C'(I_R)_{m,n}$ outputs 1 iff $sup(B \Rightarrow H, T) \geq s$, when the input database has size $m \times n$. We can build a TC^0 family of circuits solving the $\langle I, T, sup, k, s\rangle$ problem when k is fixed as follows. Consider the circuit $C(I)_{m,n}$ obtained connecting the outputs of all the circuits $C'(I_R)_{m,n}$, where $I_R \in \{S \mid S \subseteq I, |S| = k\}$, through an OR gate. Since the number of these circuits is $\binom{|I|}{k} = \mathcal{O}(|I|^k)$, hence polynomial in $|I|$, $C(I)_{m,n}$ has constant depth and polynomial size as well. The result then follows from Proposition 4.1. □

Figure 11 describes a generic circuit belonging to the above family, where $z = \binom{|I|}{k}$. Assuming $I = \{A_1, \ldots, A_n\}$ and $T = \{t_1, \ldots, t_m\}$, the generic input is represented by setting $in_{i,j}$ to 1 iff $t_i[A_j] = c(A_j)$.

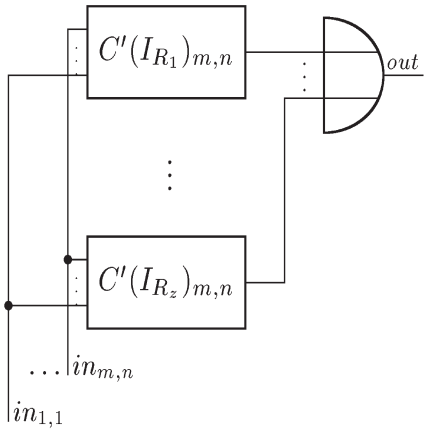


Figure 11: A generic circuit belonging to the family defined in Theorem 7.4

It is of interest to investigate the complexity of mining association rules when the value $s|T|$ is fixed. In this case $\langle I, T, sup, k, s\rangle$ corresponds to the problem of finding an association rule satisfied by almost a fixed number of transactions. Such a problem becomes of relevance when it is necessary to find a set of transactions of given size satisfying a certain property (e.g. in statistic sampling, see [24]).

Definition 7.5 Given a set of boolean attributes $I = \{A_1, \ldots, A_n\}$, and a database $T = \{t_1, \ldots, t_m\}$ defined on I, we define $\langle I, T\rangle^{-1}$ to be equal to the pair $\langle I', T'\rangle$, where $I' = \{A'_1, \ldots, A'_m\}$ is a set of boolean attributes, where each A'_j denotes the j-th tuple of T, for $j = 1, \ldots, m$, and $T' = \{t'_1, \ldots, t'_n\}$ is a database defined on I', with t'_i such that $t'_i[A'_j] = 1$ if $t_j[A_i] = c(A_i)$, and $t'_i[A'_j] = \epsilon$ otherwise (i.e. if $t_j[A_i] = \epsilon$), corresponding to the i-th attribute of I, for $i = 1, \ldots, n$, $j = 1, \ldots, m$.

Proposition 7.6 Let be I a set of boolean attributes, let T be a database on I, let k be a natural number, $1 \leq k \leq |I|$, let s, $0 \leq s \leq 1$, be a rational number, and let $\langle I', T'\rangle = \langle I, T\rangle^{-1}$. Then:

$$\langle I, T, sup, k, s\rangle \text{ is a YES instance } \iff \langle I', T', sup, \lceil s|T|\rceil, \frac{k}{|I|}\rangle \text{ is a YES instance}$$

Proof. $\langle I, T, sup, k, s\rangle$ is a YES instance iff there is an association rule $B \Rightarrow H$ on I s.t. $|B \Rightarrow H| \geq k$ and $|T_{B\wedge H}| \geq \lceil s|T|\rceil$ iff there is an association rule $B' \Rightarrow H'$ on I' s.t. $|B' \Rightarrow H'| \geq \lceil s|T|\rceil$ and $|T'_{B'\wedge H'}| \geq k$ iff $\langle I', T', sup, \lceil s|T|\rceil, \frac{k}{|I|}\rangle$ is a YES instance. □

Theorem 7.7 *Let I be a set of boolean attributes, and let $\lceil s|T|\rceil$ be a fixed constant. Then the complexity of $\langle I, T, sup, k, s\rangle$ is in* TC^0.

Proof. The result follows immediately from Theorem 7.4 and Proposition 7.6. □

Theorem 7.8 *Let I be a set of boolean attributes, and let k and $\lceil s|T|\rceil$ two fixed constants. Then the complexity of $\langle I, T, sup, k, s\rangle$ is in* AC^0_2.

Proof. Let $I = \{A_1, \ldots, A_n\}$, and let $T = \{t_1, \ldots, t_m\}$. Let $B \Rightarrow H$ be an association rule on I, and let I_R be the set $\mathbf{att}(B \wedge H)$. Define the family $\{C'(I_R)_{m,n}\}$ of AC^0_3 circuits as follows. The circuit $C'(I_R)_{m,n}$ has $n \times m$ binary inputs $x_{i,j}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, with $m = |T|$ and $n = |I|$. The input $x_{i,j}$ is 1 if $t_i[A_j] = c(A_j)$, 0 otherwise (i.e. if $t_i[A_j] = \epsilon$). The first level of $C'(I_R)_{m,n}$ consists of m AND gates G^1_i, for $i = 1, \ldots, m$. Each gate G^1_i receives the $|I_R|$ inputs $\{x_{i,k} \mid A_k \in I_R\}$. Thus the output of G^1_i is 1 iff $t_i \vdash (B \wedge H)$. The second level of $C'(I_R)_{m,n}$ consists of $\binom{m}{\lceil sm\rceil}$ AND gates G^2_j, for $j = 1, \ldots, |g|$ where

$$g = \{F \subseteq \{G^1_1, \ldots, G^1_m\} : |F| = \lceil sm\rceil\}.$$

	I_1	I_2	I_3	I_4
t_1	ϵ	1	ϵ	ϵ
t_2	ϵ	1	1	1
t_3	1	ϵ	1	1

	I'_1	I'_2	I'_3
t'_1	ϵ	ϵ	1
t'_2	1	1	ϵ
t'_3	ϵ	1	1
t'_4	ϵ	1	1

Figure 12: A database $\langle I, T\rangle$ and its transposed version $\langle I', T'\rangle^{-1}$.

The gate G_j^2 receives in input the outputs of the $\lceil sm \rceil$ gates contained within the j-th element of g. The third level consists of a single OR gate receiving in input the outputs of all the G_j^2 gates, for $j = 1, \ldots, \binom{m}{\lceil sm \rceil}$. Thus the circuit $C'(I_R)_{m,n}$ decides if $|T_{B \wedge H}| \geq \lceil sm \rceil$. The size of each circuit $C'(I_R)_{m,n}$ is polynomial, because $|g| \leq m^{\lceil sm \rceil}$, and $\lceil sm \rceil$ is fixed. We can build an AC^0 circuit solving $\langle I, T, sup, k, s \rangle$, for k and $\lceil s|T| \rceil$ fixed, as follows. Consider the circuit $C(I)_{m,n}$ obtained connecting the outputs of all the circuits $C'(I_R)_{m,n}$, with $I_R \subseteq I$ such that $|I_R| = k$ (this suffices by Proposition 4.1), through an OR gate. Since the number of these circuits is $\binom{|I|}{k} = \mathcal{O}(|I|^k)$, hence polynomial, $C_{m,n}(I)$ has constant depth and polynomial size as well. The first and second level (of AND gates), and the third and fourth level (of OR gates), can be easily each reorganized into a single level, thus giving an overall circuit family of depth 2. Hence the result follows. □

8 Conclusions

In this paper we have analyzed the computational complexity of mining association rules. We have considered the most widely accepted form of association rules that use well-known quality indices, namely, support, confidence, gain and laplace. After having formally defined association rule mining problems, we have shown that the general versions of these problems are NP-complete, except when confidence is considered over databases without nulls. Then, we have analyzed several interesting restricted cases, for most of which lower complexity bounds have been proved to hold. It is relevant to note that these cases are often related to complexity classes for which the existence of highly parallelizable algorithms has been shown. For example, for sparse databases, the complexities of the mining problems are within L. In some other cases the mining problems lie within TC^0 or within AC_2^0. The complexity analysis presented in this paper may be extended to include other forms of quality indices like, for instance, *entropy* and *improvement* [19, 18]. Moreover, other forms of association rules might be considered as, for instance, sequential patterns [4]. We leave these topics to future research.

9 Acknowledgements

We gratefully acknowledge the anonymous referees for their useful comments. We also thank John Broughton and Mirella Aquila for proofreading this manuscript.

This work was partially supported by the Italian "Ministero dell'Università e della Ricerca" within the framework of the project "D2I: Integration, Warehousing and Heterogenous sources mining".

References

[1] M. Agrawal, E. Allender, and S. Datta. On TC^0, AC^0 and arithmetic circuits. In *In Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 134–148. IEEE Computer Society Press, 1997.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28*, pages 207–216. ACM Press, 1993.

[3] R. Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[4] R. Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, Taipei, Taiwan*, pages 3–14. IEEE Computer Society, 1995.

[5] A. Ambainis, D. M Barrington, and H. LêThanh. On counting AC^0 circuits with negative constants. In *In Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 409–417. Springer, 1998.

[6] D. A. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC_1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.

[7] Roberto J. Bayardo, Jr and R. Agrawal. Mining the most interesting rules. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM*, pages 145–154, 1999.

[8] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In *STACS: Annual Symposium on Theoretical Aspects of Computer Science*, pages 134–141. Springer, 2002.

[9] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 254–260. ACM Press, 1999.

[10] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6):1278–1304, December 1995.

[11] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, 1996.

[12] M. L. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618–628, November 1996.

[13] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining using two-dimensional optimized accociation rules: Scheme, algorithms, and visualization. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 13–23. ACM Press, 1996.

[14] M. R. Garey and D. S. Johnson. *Computers and intractability; a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[15] D. Gunopulos, H. Mannila, and S. Saluja. Discovering all most specific sentences by randomized algorithms. In *ICDT '97, Proceedings of the 6th International Conference, Delphi, Greece, January 8-10*, pages 215–229. Springer, 1997.

[16] A. Hajnal, W. Maass, P. Pudlk, M. Szegedy, and G. Turan. Threshold circuits of bounded depth. *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, pages 99–110, 1987.

[17] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor, 2000.

[18] Roberto J. Bayardo Jr., R. Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 188–197. IEEE Computer Society, 1999.

[19] Shinichi Morishita. On classification and regression. *Discovery Science*, pages 40–57, 1998.

[20] Shinichi Morishita and Jun Sese. Traversing itemset lattice with statistical metric pruning. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, May 15-17, 2000, Dallas, Texas, USA*, pages 226–236. ACM, 2000.

[21] C. H. Papadimitriou. *Computatational Complexity*. Addison-Wesley, Reading, Mass., 1994.

[22] W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.

[23] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In H. V. Jagadish and I. S. Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 1–12. ACM Press, 1996.

[24] H. Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay),India*, pages 134–145. Morgan Kaufmann, 1996.

[25] J. van Leeuwen (Ed.). *Handbook of Theoretical Computer Science*, volume A. Elsevier and MIT Press, 1990.

[26] Jef Wijsen and Robert Meersman. On the complexity of mining quantitative association rules. *Data Mining and Knowledge Discovery*, 2(3):263–281, 1998.

[27] M. Zaki and M. Ogihara. Theoretical foundations of association rules. *In Proceedings of 3rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'98), Seattle, Washington, USA*, 1998.