# Adaptive Strategies for Mining the Positive Border of Interesting Patterns: Application to Inclusion Dependencies in Databases

Fabien De Marchi[1], Frédéric Flouvat[2], and Jean-Marc Petit[2]

[1] LIRIS, UMR CNRS 5205,
Université Lyon 1, 69622 Villeurbanne, France
[2] LIMOS, UMR CNRS 6158,
Université Clermont-Ferrand II, 63177 Aubière, France

**Abstract.** Given the theoretical framework of Mannila and Toivonen [26], we are interested in the discovery of the positive border of interesting patterns, also called the most specific interesting patterns. Many approaches have been proposed among which we quote the levelwise algorithm and the *Dualize and Advance* algorithm. In this paper, we propose an adaptive strategy – complementary to these two algorithms – based on four steps: 1) In order to initialize the discovery, eliciting some elements of the negative border, for instance using a levelwise strategy until a certain level $k$. 2) From the negative border found so far, inferring the *optimistic positive border* by dualization, i.e. the set of patterns whose all specializations are known to be not interesting patterns. 3) Estimating the distance between the positive border to be discovered and the optimistic positive border. 4) Based on these estimates, carrying out an adaptive search either bottom-up (the jump was too optimistic) or top-down (the solution should be very close).

We have instantiated this proposition to the problem of inclusion dependency (IND) discovery. IND is a generalization of the well known concept of *foreign keys* in databases and is very important in practice. We will first point out how the problem of IND discovery fits into the theoretical framework of [26]. Then, we will describe an instantiation of our adaptive strategy for IND discovery, called `Zigzag`, from which some experiments were conducted on synthetic databases. The underlying application of this work takes place in a project called *DBA Companion* devoted to the understanding of existing databases at the logical level using data mining techniques.

## 1 Introduction

Given the theoretical framework for data mining given in [26], we are interested in the discovery of the positive border of interesting patterns, also called the most specific interesting patterns. Many approaches have been proposed among which we quote the levelwise algorithm [26] and the *Dualize and Advance* algorithms [17,30]. In this paper, we propose an adaptive strategy – complementary to these two main algorithms – based on four steps:

J.-F. Boulicaut et al. (Eds.): Constraint-Based Mining, LNAI 3848, pp. 81–101, 2005.
© Springer-Verlag Berlin Heidelberg 2005

1. In order to initialize the discovery, eliciting some elements of the negative border, for instance using a levelwise strategy until a certain level $k$.
2. From the negative border found so far, inferring the *optimistic positive border* which is the set of patterns whose all specializations are known to be not interesting patterns. In the spirit of the *Dualize and Advance* algorithm, this part exploits the idea of monotone dualization, involving the generation of minimal transversals of an hypergraph.
3. Estimating the distance between the positive border to be discovered and the optimistic positive border.
4. Based on these estimates, carrying out an adaptive search either bottom-up (the jump was too optimistic) or top-down (the solution should be very close).

The basic idea of our proposition is to combine the strength of both levelwise algorithm and *Dualize and Advance* algorithm in such a way that:

- "small" maximal interesting patterns may be found efficiently as well as large ones, which is drawback of levelwise strategies.
- the number of dualization may be tuned with our adaptive strategy whereas the number of dualization performed by *Dualize and Advance* is always in the size of the positive border (tight bound).

The dualization performed in step 2 is quite similar to that proposed in the *Dualize and Advance* algorithm. Nevertheless, instead of starting from interesting patterns as *Dualize and Advance* algorithm does, we use not interesting patterns to perform the dualization. As a consequence, our proposition contributes to clarify many works dealing with related problems (e.g. maximal frequent itemsets [22,5,16,10]) since it gives an exact characterization of the optimistic positive border of interesting patterns from some subset of interesting patterns.

We have instantiated this proposition to the problem of inclusion dependency (IND) discovery. IND is a generalization of the well known concept of *foreign keys* in databases and is very important in practice. We first point out how the problem of IND discovery fits into the theoretical framework of borders of theories only if IND with repeated attributes are allowed. Then, an instantiation of our adaptive strategy for IND discovery is proposed. From our general proposition, a specific algorithm called `Zigzag` has been devised for IND discovery. Some experiments conducted on synthetic databases have been performed and results are given.

The underlying application of this work takes place in a project called *DBA Companion* devoted to the understanding of existing databases at the logical level using data mining techniques. Whereas physical database design has always received a lot of attention by the database community, one can quote that, rather surprisingly, logical database analysis has been less studied despite its importance for practical applications such as logical database tuning, semantic query optimization or simply database auditing.

From this simple remark, we have developed a project called `DBA Companion` devoted to the understanding of logical database constraints from which logical

database tuning can be achieved [25,24,11,12]. In this setting, two main data mining issues need to be addressed: the first one is the design of efficient algorithms for functional dependencies and inclusion dependencies discovery and the second one is about the interestingness of the discovered knowledge.

Clearly, the contribution made in this paper fits into this project and has been integrated in our GUI prototype available on-line [23]: its objective is to be able to connect a database in order to give some insights to DBA/analyst such as:

- the functional dependencies and inclusion dependencies satisfied in her/his database,
- small examples of her/his database, thanks to *Informative Armstrong Databases*. The same benefits when the design by example were introduced are also expected in this slightly different context (database maintenance vs database design).

*Chapter organization.* The chapter is organized as follows: Section 2 recalls the framework of borders of a theory. Section 3 introduces the principle of our approach for discovering the positive border of interesting patterns within this theoretical framework. Section 4 applies our proposition on a particular application: the discovery of inclusion dependency. Based on this proposition, the algorithm `Zigzag` and some experimental results are given. Section 5 quickly introduces related contributions and we conclude in Section 6.

## 2 Preliminaries: Borders of a Theory

We recall below some notations and basic results used among this chapter. For more details, the reader is invited to refer to [26].

Given a database $\mathbf{d}$, a finite language $\mathcal{L}$ for expressing patterns or defining subgroups of the data, and a predicate $Q$ for evaluating whether a pattern $\varphi \in \mathcal{L}$ is true or "interesting" in $\mathbf{d}$, the discovery task is to find the theory of $\mathbf{d}$ with respect to $\mathcal{L}$ and $Q$, i.e. the set $Th(\mathcal{L}, \mathbf{d}, Q) = \{\varphi \in \mathcal{L} | Q(\mathbf{d}, \varphi) \text{ is true}\}$.

A specialization/generalization relation does often exist between patterns of $\mathcal{L}$. Such a relation is a partial order $\preceq$ on the patterns of $\mathcal{L}$. We say that $\varphi$ is more general (resp. more specific) than $\theta$, if $\varphi \preceq \theta$ (resp. $\theta \preceq \varphi$).

The relation $\preceq$ is a anti-monotone relation with respect to $Q$ if the predicate $Q$ is anti-monotone wrt $\preceq$, i.e. for all $\theta, \varphi \in \mathcal{L}$ if $Q(\mathbf{d}, \theta)$ is true and $\varphi \preceq \theta$ then $Q(\mathbf{d}, \varphi)$ is true.

Given a partial order $\preceq$, the set $Th(\mathcal{L}, \mathbf{d}, Q)$ can be represented by enumerating only its maximal elements, that is the set

$$MTh(\mathcal{L}, \mathbf{d}, Q) = \{\varphi \in Th(\mathcal{L}, \mathbf{d}, Q) | \text{ for no } \theta \in Th(\mathcal{L}, \mathbf{d}, Q), \varphi \prec \theta\}$$

A set $S$ of patterns from $\mathcal{L}$ such that $S$ is closed downwards under the relation $\preceq$ can be represented by two borders: the *positive border* of $S$, denoted by $\mathcal{B}d^+(S)$, and the *negative border* of $S$, denoted by $\mathcal{B}d^-(S)$. They are defined as follows: $\mathcal{B}d^+(S) = \{\sigma \in S \mid \nexists \varphi \in S, \sigma \prec \varphi\}$ and $\mathcal{B}d^-(S) = \{\sigma \in \mathcal{L} \setminus \mathcal{S} \mid \nexists \varphi \in \mathcal{L} \setminus \mathcal{S}, \varphi \prec \sigma\}$.

Obviously, we have $\mathcal{B}d^+(Th(\mathcal{L}, \mathbf{d}, Q)) = MTh(\mathcal{L}, \mathbf{d}, Q)$.

Let us consider $\mathcal{C}$ as the set of all patterns from $\mathcal{L}$. In that case, $(\mathcal{C}, \preceq)$ is a poset and let $R$ be a set (the powerset of $R$ is denoted by $\mathcal{P}(R)$). Sometimes an isomorphism between the posets $(\mathcal{C}, \preceq)$ and $(\mathcal{P}(R), \subseteq)$ may exist. In that case, the problem $MTh(\mathcal{L}, \mathbf{d}, Q)$ is said to be *representable as sets*.

A function $f : \mathcal{C} \to \mathcal{P}(\mathcal{R})$ is said to be a *representation of* $(\mathcal{C}, \preceq)$ *as sets* if $f$ is bijective and its inverse is computable, and for all $\theta$ and $\varphi$ we have $\theta \preceq \varphi$ iff $f(\theta) \subseteq f(\varphi)$.

With this supplementary constraint, we have a relationship between the positive and negative border through the notion of minimal transversal[1] of hypergraphs.

Consider the hypergraph $\mathcal{H}(S)$ on $R$ containing as edges the sets $f(\varphi)$ for $\varphi \in \mathcal{B}d^+(S)$, i.e. $\mathcal{H}(S) = \{f(\varphi) \mid \varphi \in \mathcal{B}d^+(S)\}$ also noted as $\mathcal{H}(S) = f(\mathcal{B}d^+(S))$. Let $\mathbf{TrMin}(\mathcal{H})$ be all minimal transversals of the hypergraph $\mathcal{H}$ and $\overline{\mathcal{H}(S)} = \{R \setminus X \mid X \in \mathcal{H}(S)\}$ the complements of the edges of $\mathcal{H}(S)$ in $R$.

Now the relationship between the positive and negative border may be given:

***Theorem 1.*** *[26]*

$$f^{-1}(\mathbf{TrMin}(\overline{\mathcal{H}(S)})) = \mathcal{B}d^-(S)$$

Note that $S$ can be reduced to its positive border, i.e. we have: $f^{-1}(\mathbf{TrMin}(\overline{\mathcal{H}(\mathcal{B}d^+(S))})) = \mathcal{B}d^-(\mathcal{B}d^+(S))$.

## 3 Principle of Our Approach

The basic idea is to combine the strength of both levelwise algorithm and *Dualize and Advance* algorithm in such a way that "small" maximal interesting patterns may be found efficiently by a levelwise strategy, while "large" maximal interesting patterns may be discovered by dualization.

The proposed method consists of a pessimistic exploration of the most general patterns until a given level $k$, and then a "zigzag" between the negative border in construction and the corresponding optimistic positive border.

### 3.1 Step 1: A k-Levelwise Approach

In order to initialize the discovery, we would like to elicit the largest possible subset of the negative border. Therefore, we have chosen to apply a levelwise strategy since it may be optimal whenever large interesting patterns do not exist. We apply it *until a certain level* $k$, which may be specified by the user or dynamically defined. As an example, the following heuristic may be used : "As soon as the negative border does not change *enough* between two iterations, stop the levelwise search". This can be done efficiently without any overhead by counting at a given level $k$, the ratio of the number of interesting patterns of

[1] A minimal transversal of an hypergraph $H$ is a set of elements $X$ such that (1) $X$ has a non empty intersection with every element of $H$ and (2) $X$ is minimal w.r.t. this property.

size $k$ on the number of candidates patterns of size $k$ whose all generalizations are interesting.

More formally, this heuristic can be stated as follows:

Given a threshold $\epsilon \in [0,1]$, at any iteration of a levelwise algorithm, let $C_k$ (resp. $F_k$) be the candidate patterns (resp. interesting patterns) of size k. If $\frac{|F_k|}{|C_k|} \geq \epsilon$, then stop the levelwise search and the current level gives the value of k. The choice of $\epsilon$ should be typically close to 1.

At the end, whatever the criterion used to get the value $k$, the levelwise algorithm provides 1) the set $\mathcal{B}d_k^+(Th(\mathcal{L}, \mathbf{d}, Q))$, which can be seen as a subset of $\mathcal{B}d^+(Th(\mathcal{L}, \mathbf{d}, Q))$ (in fact, some elements of size $k$ will be removed latter) and 2) the set $\mathcal{B}d_k^-(Th(\mathcal{L}, \mathbf{d}, Q))$, which is a subset of $\mathcal{B}d^-(Th(\mathcal{L}, \mathbf{d}, Q))$.

### 3.2 Step 2: The Optimistic Positive Border

The simple remark on which this step is founded is the following: a set of not interesting patterns makes it possible to prune a certain number of candidates by anti-monotony, and thus to define an *optimistic set of interesting patterns*, as being the set of sentences whose all specializations do not verify the predicate.

**Definition 1.** *Let $C$ be the search space associated to $\mathcal{L}$ for the problem of enumerating $MTh(\mathcal{L}, \boldsymbol{d}, Q)$. Let $NI \subseteq C$ be a set such that $\forall \varphi \in NI, Q(\boldsymbol{d}, \varphi)$ is false, i.e. $\varphi$ is not interesting in $\boldsymbol{d}$.*

*The* optimistic set of interesting patterns *with respect to $NI$, denoted by $I_{opt}(NI)$, is defined by: $I_{opt}(NI) = \{\varphi \in C \mid \nexists \sigma \in NI, \sigma \preceq \varphi\}$.*

Moreover, the *optimistic positive border*, denoted by $\mathcal{B}d^+(I_{opt}(NI))$, is the set of most specific patterns in $I_{opt}(NI)$. When clear from context, we will note $\mathcal{B}d^+_{opt}(NI)$ instead of $\mathcal{B}d^+(I_{opt}(NI))$. Remark that $\mathcal{B}d^+_{opt}(NI)$ is the same if $NI$ is restricted to its most general patterns.

In the spirit of the dualization proposed in the *Dualize and Advance* algorithm [17], the next theorem states the relation between the optimistic positive border and the minimal transversals of an hypergraph.

***Theorem 2.*** *Let $NI \subseteq C$ be a set of non-interesting patterns in $\boldsymbol{d}$. The optimistic positive border w.r.t. $NI$ is such that:*

$$\mathcal{B}d^+_{opt}(NI) = f^{-1}(\overline{\mathbf{TrMin}(\mathcal{H}(NI))})$$

Proof. *Let $i \in C$ be a pattern.*
*First, we show that $i \in I_{opt}(NI) \Leftrightarrow \overline{f(i)}$ is a transversal of $\mathcal{H}(NI)$ :*
$i \in I_{opt}(NI)$
$\Leftrightarrow \forall j \in NI, j \not\preceq i$
$\Leftrightarrow \forall j \in NI, f(j) \not\subseteq \overline{f(i)}$
$\Leftrightarrow \forall j \in NI, f(j) \cap \overline{f(i)} \neq \emptyset$
$\Leftrightarrow \overline{f(i)}$ *is a transversal of* $\mathcal{H}(NI)$.

*Then we show that $i$ is maximal in $I_{opt}(NI) \Leftrightarrow \overline{f(i)}$ is a minimal transversal of $\mathcal{H}(NI)$ :*
*Let $i \in \mathcal{B}d^+_{opt}(NI)$. Since $i \in I_{opt}(NI)$, $\overline{f(i)}$ is a transversal of $\mathcal{H}(NI)$. Suppose $\overline{f(i)}$ is not minimal: $\exists X \subseteq R$, $X$ transversal of $\mathcal{H}(NI)$ and $X \subset \overline{f(i)}$, and thus $f(i) \subset \overline{X}$. Then $f^{-1}(\overline{X}) \in I_{opt}(NI)$ and $i \prec f^{-1}(\overline{X})$, which contradict the fact that $i \in \mathcal{B}d^+_{opt}(NI)$.*
*Now, let $X \in \mathbf{TrMin}(\mathcal{H}(NI))$. $X$ is a transversal, then $f^{-1}(\overline{X}) \in I_{opt}(NI)$. Suppose that $f^{-1}(\overline{X})$ is not maximal: $\exists j \in I_{opt}(NI)$ such that $f^{-1}(\overline{X}) \prec j$. Then $\overline{f(j)}$ is a transversal of $\mathcal{H}(NI)$ with $\overline{X} \subseteq f(j)$, and thus $\overline{f(j)} \subseteq X$, which contradicts the fact that $X$ is a minimal transversal.*

*Remark 1.* This result can also be proved as a simple corollary of the theorem 1 since $\mathbf{TrMin}(\mathbf{TrMin}(H)) = H$ for any hypergraph $H$ [7].

Thanks to this result, the optimistic positive border computation can exploit the numerous works and results about minimal transversals computation; recent results can be found in [14,4,9].

An optimization can also be brought to the calculation of $\mathcal{B}d^+_{opt}(NI)$. Indeed, at each iteration, this set contains at the same time the largest possible interesting patterns, but also all interesting patterns already discovered. The idea is thus to characterize only new elements of $\mathcal{B}d^+_{opt}(NI)$, ignoring those already explored. The following result just follows from the theorem 2.

**Proposition 1.** *Let $I_k$ be the set of interesting patterns of size less or equal to $k$, $NI$ a set of non-interesting patterns, and $n = |R|$.*
*We have:*
$i \in (\mathcal{B}d^+_{opt}(NI) \setminus I_k) \Longleftrightarrow \overline{f(i)} \in \mathbf{TrMin}(\mathcal{H}(NI))$ *and* $|\overline{f(i)}| \leq n - k$

In practice, this condition leads to optimize the generation of minimal transversals since candidates exceeding the size allowed can be safely removed.

### 3.3 Step 3: Getting Estimates on the Optimistic Positive Border

We try to estimate the distance between the positive border to be discovered and the optimistic positive border in order to guide the search in the next step.

For $\varphi \in \mathcal{B}d^+_{opt}(NI)$, two main cases do exist:

- either $Q(\mathbf{d}, \varphi)$ is true: the "jump" was successful.
- or $Q(\mathbf{d}, \varphi)$ is false. In that case, we propose to estimate a degree of error in order to qualify the jump.

Given a new user-defined threshold $\delta$, a database $\mathbf{d}$ and a predicate $Q$, an error measure $\psi$ defined from $\mathcal{L}$ to $\Re$, noted $\psi_{\mathbf{d},Q}(\varphi)$, we can easily devised two sub-cases when $Q(\mathbf{d}, \varphi)$ is false:

- either $\psi_{\mathbf{d},Q}(\varphi) \leq \delta$ : the "jump" was not successful but solutions should exist among the nearest generalizations of $\varphi$.
- or $\psi_{\mathbf{d},Q}(\varphi) > \delta$ : In that case, the jump was over-optimistic and probably, no solution does exist among the nearest generalizations of $\varphi$.

Moreover, error measures must be restricted to those verifying the following property:

***Property 1.*** *Let* $\varphi, \theta \in L$. $\varphi \preceq \theta \Rightarrow \psi_{\mathbf{d},Q}(\varphi) \leq \psi_{\mathbf{d},Q}(\theta)$

Clearly the definition of such error measures is quite application-dependent and should be done carefully.

Nevertheless, a generic idea to build such error measures can be stated as follows: "Computing the ratio of the size of the largest subset of the database so that the pattern becomes interesting on the size of the database". More formally, let $\varphi \in \mathcal{L}$ such that $Q(\mathbf{d}, \varphi)$ is false.

$$\psi_{\mathbf{d},Q}(\varphi) = 1 - \frac{max\{|\mathbf{d}^{'}| \mid \mathbf{d}^{'} \subseteq \mathbf{d}, Q(\mathbf{d}^{'}, \varphi) \text{ true}\}}{|\mathbf{d}|}$$

The interested reader may refer to [19] for other error measure definitions in the restricted setting of functional dependencies.

### 3.4 Step 4: Adaptive Behavior

Based on these estimates, many different strategies can be devised in order to guide the search. Basically, the traversal of the unexplored search space can be carried out either bottom-up (the jump was too optimistic) or top-down (the solution should be very close). Many other strategies could be applied to converge as soon as possible to the positive border of interesting patterns. The basic idea is to avoid the enumeration of the largest parts of the search space.

Note that many propositions have been made in the setting of maximal frequent itemsets, see for example [22,5,16,10]. The discussion done in [22] is quite relevant in our context.

Once again, this step is also application-dependent and will not be described further in this chapter. More details will be given in section 4 in the context of the discovery of INDs in databases.

## 4 Application to Inclusion Dependency Discovery

### 4.1 Preliminaries

Some concepts of the relational databases are briefly recalled (see for example [1,21] for more details).

Let $R$ be a finite set of *attributes*. For each attribute $A \in R$, the set of all its possible values is called the *domain of* $A$ and denoted by $Dom(A)$. A *tuple* over $R$ is a mapping $t : R \rightarrow \cup_{A \in R} Dom(A)$, where $t(A) \in Dom(A), \forall A \in R$. A *relation* is a finite set of tuples. The cardinality of a set $X$ is denoted by $|X|$. We say that $r$ is a relation *over* $R$ and $R$ is the *relation schema* of $r$. If $X \subseteq R$ is an attribute set[2] and $t$ is a tuple, we denote by $t[X]$ the restriction of

[2] Letters from the beginning of the alphabet introduce single attributes whereas letters from the end introduce attribute sets.

$t$ to $X$. The projection of a relation $r$ onto $X$, denoted as $\pi_X(r)$, is defined by $\pi_X(r) = \{t[X] \mid t \in r\}$.

A *database schema* $\mathbf{R}$ is a finite set of *relation schemes* $R_i$. A *relational database instance* $\mathbf{d}$ (or *database*) over $\mathbf{R}$ corresponds to a set of relations $r_i$ over each $R_i$ of $\mathbf{R}$.

An attribute sequence (e.g. $X =< A, B, C >$ or simply $ABC$) is an ordered set of attributes. When it is clear from context, we do not distinguish a sequence from its underlying set.

Two attributes $A$ and $B$ are said to be *compatible* if $Dom(A) = Dom(B)$. Two distinct attribute sequences $X$ and $Y$ are *compatible* if $|X| = |Y| = m$ and if for $j = [1, m]$, $Dom(X[j]) = Dom(Y[j])$.

An *inclusion dependency* (IND) over a database schema $\mathbf{R}$ is a statement of the form $R_i[X] \subseteq R_j[Y]$, where $R_i, R_j \in \mathbf{R}, X \subseteq R_i, Y \subseteq R_j$, $X$ and $Y$ are compatible sequences[3].

The size (or arity) of an IND $i = R[X] \subseteq R[Y]$, noted $|i|$ is such that $|i| = |X| = |Y|$. We call *unary inclusion dependency* an IND of size 1.

Let $\mathbf{d}$ be a database over a database schema $\mathbf{R}$, where $r_i, r_j \in \mathbf{d}$ are relations over $R_i, R_j \in \mathbf{R}$ respectively. An inclusion dependency $R_i[X] \subseteq R_j[Y]$ is *satisfied* in a database $\mathbf{d}$ over $\mathbf{R}$, denoted by $\mathbf{d} \models R_i[X] \subseteq R_j[Y]$, if and only if $\forall u \in r_i, \exists v \in r_j$ such that $u[X] = v[Y]$ (or equivalently $\pi_X(r_i) \subseteq \pi_Y(r_j)$).

Let $I_1$ and $I_2$ be two sets of inclusion dependencies, $I_1$ is a *cover* of $I_2$ if $I_1 \models I_2$ (this notation means that each dependency in $I_2$ holds in any database satisfying all the dependencies in $I_1$) and $I_2 \models I_1$.

A sound and complete axiomatization for INDs was given in [27]. If $I$ is a set of INDs, we have:

1. (reflexivity) $I \models R[A_1, ..., A_n] \subseteq R[A_1, ..., A_n]$
2. (projection and permutation) if $I \models R[A_1, ..., A_n] \subseteq S[B_1, ..., B_n]$ then $I \models R[A_{\sigma 1}, ..., A_{\sigma m}] \subseteq S[B_{\sigma 1}, ..., B_{\sigma m}]$ for each sequence $\sigma 1, ..., \sigma m$ of distinct integers from $\{1, ..., n\}$
3. (transitivity) if $I \models R[A_1, ..., A_n] \subseteq S[B_1, ..., B_n]$ et $I \models S[B_1, ..., B_n] \subseteq T[C_1, ..., C_n]$ then $I \models R[A_1, ..., A_n] \subseteq T[C_1, ..., C_n]$
4. (attribute equality) if $I \models R[AB] \subseteq S[CC]$, then $A$ and $B$ can be substituted to each other in all satisfied IND expressions.
5. (redundancy) if $I \models R[X] \subseteq S[Y]$, then $I \models R[XU] \subseteq S[YV]$, where $R[U] \subseteq S[V]$ can be obtained from $R[X] \subseteq S[Y]$ using second inference rule.

### 4.2 Adequacy to the Framework of Borders of Theories

The finite language $\mathcal{L}$ corresponds to the language defining INDs in a database schema, i.e. a pattern is an IND. The database $\mathbf{d}$ is the relational database on which the discovery of INDs has to be performed and, the predicate $Q(\mathbf{d}, \varphi)$ is

[3] Usually, the IND definition excludes repeated attributes in the sequences on left and right-hand sides. In this paper, we adopt a less restrictive framework in order to ensure a representation as sets for INDs (cf Section 4.2) ; the exclusion of the repeated attributes is considered after the presentation of the algorithm (cf. Section 4.5).

the satisfaction of an IND against the database, i.e. $Q(\mathbf{d}, \varphi)$ is true $\Longleftrightarrow \mathbf{d} \models \varphi$. In other words, an interesting pattern is a satisfied IND.

The problem of IND discovery can be formulated as follows:

Let $\mathbf{d}$ be a database, find a cover of all satisfied INDs in $\mathbf{d}$

The number of potentially satisfied INDs, which constitutes the basic search space, is more than factorial in the number of attributes [18].

In the sequel, we denote by $C$ the search space of INDs, made up of a set of IND expressions. The aim of this section is to reach a formal definition of $C$, in such a way that the IND discovery problem fits into the framework previously presented.

In order to structure the search space, a specialization / generalization relation between INDs is proposed in the following definition.

**Definition 2.** *Given $i = R[X] \subseteq S[Y]$ and $j = R[X'] \subseteq S[Y']$ two IND expressions, we say that $j$* generalizes *$i$ (or $i$* specializes *$j$), noted $j \preceq i$, if $X =< A_1, ..., A_n >$, $Y =< B_1, ..., B_n >$, and there exists a set of integers $k_1 < ... < k_l \in \{1, ..., n\}$ with $l \leq n$ such that $X' =< A_{k_1}, ..., A_{k_l} >$ and $Y' =< B_{k_1}, ..., B_{k_l} >$.*

For example, $R[AC] \subseteq S[DF] \preceq R[ABC] \subseteq S[DEF]$. We note $j \prec i$ for $j \preceq i$ and $j \neq i$.

Moreover, the satisfaction of an IND in a database $\mathbf{d}$ turns out to be anti-monotone with respect to the relation $\preceq$, which is a requirement to comply with the theoretical framework introduced in Section 2.

***Property 2.*** *Let $i, j \in C$ such that $j \preceq i$.*

$$\boldsymbol{d} \not\models j \Rightarrow \boldsymbol{d} \not\models i$$

Thus, any set $I$ of INDs can be represented by two borders: its most specialized elements, i.e. its positive border $\mathcal{B}d^+(I)$ and the most general elements which does not belong to $I$, i.e. its negative border $\mathcal{B}d^-(I)$.

Clearly, when $I$ is the set of the satisfied INDs in $\mathbf{d}$, $\mathcal{B}d^+(I)$ answers the IND discovery problem.

In order to apply our proposition, we need to exhibit a representation as sets of the search space $C$ of INDs, i.e. to find a subset lattice $(\mathcal{P}(R), \subseteq)$ isomorph to $(C, \preceq)$. The basic idea is to consider the powerset of unary INDs as a possible candidate in order to build a bijective function between $(\mathcal{P}(R), \subseteq)$ and $(C, \preceq)$. We shall see in the sequel that we will need to restrict somehow $(C, \preceq)$ to comply with the requirements given in Section 2.

We first define a function, called *ens*, to transform a given IND into a set of unary INDs.

**Definition 3.** *Let $I_1$ be the set of unary INDs over $\boldsymbol{R}$.*
*The function $ens : C \longrightarrow \mathcal{P}(I_1)$ is defined by:*

$$ens(i) = \{j \in I_1 \mid j \preceq i\}$$

Therefore, each IND can be associated with a set of unary INDs.

***Example 1.*** *Consider* $i = R[ABC] \subseteq S[EFG]$, $i_1 = R[A] \subseteq S[E]$, $i_2 = R[B] \subseteq S[F]$ *and* $i_3 = R[C] \subseteq S[G]$. *Then:* $ens(i) = \{i_1, i_2, i_3\}$.

Then, the following example points out that, if the domain of *ens* (i.e. $C$) is not carefully defined, the function *ens* is not injective and $ens^{-1}$ is not computable.

***Example 2.*** *Suppose* $\boldsymbol{d} = \{r_1, r_2, r_3\}$ *over the schema* $\boldsymbol{R} = \{R_1, R_2, R_3\}$, *with* $R_1 = ABC, R_2 = DEF$ *and* $R_3 = GHI$. *For sake of clearness, let us note* $i_1 = R_1[A] \subseteq R_2[D]$, $i_2 = R_1[A] \subseteq R_2[E]$ *and* $i_3 = R_1[B] \subseteq R_3[H]$. *Then:*

- $ens(R_1[AA] \subseteq R_2[DE]) = ens(R_1[AA] \subseteq R_2[ED]) = \{i_1, i_2\}$, *i.e. the function ens is not injective.*
- $ens^{-1}(\{i_1, i_3\})$ *is not computable since* $i_1$ *and* $i_3$ *are not defined over the same right-hand side schema.*

It is also worth noting that we can now justify through this example the necessity of accepting repeated attributes in IND definition, since otherwise $ens^{-1}(\{i_1, i_2\})$ could not be defined.

To cope with the first point of the example 2, the search space $C$ has to be restricted to only one permutation of each IND. Hopefully, thanks to the second inference rule for INDs (cf Section 4.1), this restriction does not imply any lost in the discovered knowledge and can be fixed easily: a total order on attributes has to be enforced on one side of IND. We choose to fix an order on the left-hand side, cf Definition 4 below.

Now, an interesting property allows us to deal with the second point of the example 2, making it possible to break up our exploration method into several independent courses.

***Property 3.*** *Let* $\boldsymbol{d}$ *be a database over a schema* $\boldsymbol{R}$ *and* $I$ *the set of satisfied INDs in* $\boldsymbol{d}$. *Let* $I_{R \to S}$ *be INDs from* $R$ *to* $S$. *Then*

$$\mathcal{B}d^+(I) = \bigcup_{(R,S) \in \boldsymbol{R}^2} \mathcal{B}d^+(I_{R \to S})$$

Thus, the IND discovery can be made through independent tasks, one for each couple of relations in the database. During one execution, only INDs defined from a given relation to another (possibly the same) relation are considered, and thus the second difficulty pointed out by the example 2 does not occur any more.

We can now restrict the search space $C$ of INDs to a couple of relations in a database schema. We suppose that a total order exists over attributes, for instance the lexicographic order can always be used up to a renaming.

**Definition 4.** *Let* $\boldsymbol{R}$ *be a database schema and* $(R, S)$ *a couple of relation schema of* $\boldsymbol{R}$. *The search space of INDs over* $(R, S)$, *denoted by* $C(R, S)$ *or just* $C$ *when* $(R, S)$ *is clear from the context, is defined by:*

$$C(R,S) = \{R[< A_1 ... A_n >] \subseteq S[< B_1 ... B_n >] \mid \forall 1 \leq i < j \leq n,$$
$$(A_i < A_j) \vee (A_i = A_j \wedge B_i < B_j)\}$$

*where* $n = min(|R|, |S|)$.

Until the end of this chapter, we will use the following notation:

- $(R, S)$ is a couple of relations of $\mathbf{R}$,
- $C$ is the search space of INDs from $R$ to $S$ and
- $I_1$ the set of unary INDs from $R$ to $S$, i.e. $I_1 \subseteq C$.

We can now give the main result of this section, that is to say that under the previous assumptions, the IND search space is *representable as sets.*

***Property 4.*** *The function* $ens : C \longrightarrow \mathcal{P}(I_1)$ *is bijective and its inverse function* $ens^{-1}$ *is computable.*

This result can be easily derived from the definition of *ens* and from the definition of the search space $C$. The following property follows from the definition of the function *ens* and the definition of the relation $\preceq$:

***Property 5.*** *Let* $i$ *and* $j$ *be two INDs expressions of* $C$*.*

$$i \preceq j \Leftrightarrow ens(i) \subseteq ens(j)$$

Thus, we have highlighted an isomorphism from $(C, \preceq)$ to $(\mathcal{P}(I_1), \subseteq)$, that is to say that the search space of INDs is representable as sets. As a consequence, each set of INDs in $C$ can be associated with an hypergraph:

**Definition 5.** *Let* $I \subseteq C$*. The hypergraph associated with* $I$*, denoted by* $\mathcal{H}(I) = \{V, E\}$*, is defined by:* $V = I_1$ *and* $E = \{ens(i) \mid i \in I\}$*.*

### 4.3 Applying Our Four Steps Approach

In this section, we customize our four steps approach to the problem of IND discovery. Some properties of INDs will be given to justify our choices.

**Step 1: A k-levelwise Approach.** Several factors justify to use a levelwise approach for INDs of "small" size. The first one is that in practice, a great proportion of unary IND candidates is not satisfied. Thus a significant part of the search space is disqualified by anti-monotony, justifying a levelwise approach for this level.

The second one is that an efficient method was proposed in [11] for unary IND discovery, based on a data reorganization. The salient feature of this approach is not to make as many database passes as candidates exist (as it is the case in general for dependency discovery [26]), but only one database pass for all the candidates (as it is for example the case for frequent itemsets).

In [26], a levelwise approach is suggested to discover $\mathcal{B}d^+(I)$. The algorithm MIND [11] is based on this idea, using an *AprioriGen* like candidate generation [2]. Its effectiveness is based on the presence at each level of many not satisfied INDs, in order to prune a great part of the remaining space. The experiments conducted in [11] show that such an approach is scalable according to the number of tuples and attributes: the greatest database had 90000 tuples and 200 attributes, the IND positive border of the database was composed of four unary IND and one IND of size 6. Nevertheless, such an approach is not adapted when large INDs have to be discovered; indeed, to discover an IND $i$ of size $n$, it is necessary to have discovered the $2^n$ INDs which generalize $i$.

As a consequence, we decided to use MIND until a given level $k$ in order to initialize the search.

**Step 2: The optimistic positive border.** From the negative border already discovered, we may apply the Theorem 2 to infer the so-called optimistic positive border of INDs.

In fact, a justification for an optimistic approach does exist for IND discovery and will be formaly stated in Proposition 3. Intuitively, it can be expressed as follows:

> if all generalizations of size $k$ of a candidate IND $i$ are satisfied, then $i$ has more chances to be satisfied when $k$ increases.

This result is justified by an inference rule[4] of Functional Dependencies (FDs) and INDs given by the following proposition.

**Proposition 2.** *Let $\{r, s\}$ be a database, $C$ the corresponding IND search space, and $I_k = \{i \in C \mid |i| = k, \{r, s\} \models i\}$, $k \geq 2$.*

*Let $i = R[X] \subseteq S[Y], i \in C, |i| = n, n > k$ such that $\forall j \in C, |j| = k, j \prec i$, we have $j \in I_k$.*

$$\text{if } \exists Y_1 \subseteq Y, |Y_1| = k-1 \text{ and } s \models Y_1 \rightarrow Y \setminus Y_1 \text{ then } \{r, s\} \models i$$

Proof. *Let $\boldsymbol{d} = \{r, s\}$ be a database and $C$ the corresponding IND search space. Let $i = R[X] \subseteq S[Y] \in C$ an IND expression of arity $n \geq 3$, and an integer $k < n$. Suppose that all INDs which generalize $i$, of size lower or equal to $k$, are satisfied. And let $Y_1 \subseteq Y$ be such that $|Y_1| = k - 1$ and $s \models Y_1 \rightarrow Y \setminus Y_1$.*
*Let us put $Y \setminus Y_1 = B_1...B_{n-k+1}$. We note $X_1$ the sub-sequence of $X$ in which the position of elements in $X$ correspond to the position of elements of $Y_1$ in $Y$, and $A_1, ..., A_{n-k+1}$ the elements of $X$ in which the position of elements in $X$ correspond to the position respectively of $B_1, ..., B_{n-k+1}$ in $Y$.*
*Let $t \in r$. We have $\boldsymbol{d} \models R[X_1A_1] \subseteq S[Y_1B_1]$, since this IND is of arity $k$. Thus $\exists u_1 \in s$ such that $u_1[Y_1B_1] = t[X_1A_1]$. In the same way, $\boldsymbol{d} \models R[X_1A_2] \subseteq S[Y_1B_2]$, then $\exists u_2 \in s$ such that $u_2[Y_1B_2] = t[X_1A_2]$. We know that $s \models Y_1 \rightarrow B_2$ and thus $u_1[B_2] = u_2[B_2]$ since $u_1[Y_1] = u_2[Y_1]$. Thus, $u_1[Y_1B_1B_2] = t[X_1A_1A_2]$. We can repeat $n - k + 1$ times the same reasoning, to show that $u_1[Y_1B_1B_2...B_{n_k+1}] = t[X_1A_1A_2...A_{n_k+1}]$, and then $u_1[Y] = t[X]$. This is true for all tuple in $r$, and we have $\boldsymbol{d} \models R[X] \subseteq S[Y]$.*

***Example 3.*** *Consider the IND $i = R[ABCDEF] \subseteq S[GHIJKL]$. Suppose that the 20 INDs of size 3 which generalize $i$ are satisfied; then if there exists two attributes of $GHIJKL$ that determine the others, for example $GH \rightarrow IJKL$, we have $\boldsymbol{d} \models i$.*

Thus, the principle justified by this rule is, starting from an explored level $k$, to build the highest IND expressions for which all sub-INDs of size $k$ are true. Notice that the larger $k$ is, the more there are chances that sets of attributes of size $k - 1$ determine the others, meeting the conditions of the Proposition 2.

[4] The inference rule stated by the Proposition 2 does not form part of the Mitchell system [27], but of course is inferred by this system which is sound and complete. The demonstration suggested here seems to be more comprehensible and shorter.

However, when a suspected large IND $i$ of size $n$ is detected as false, it is necessary to choose between two alternatives: going back to the level $k+1$ "to consolidate" basic knowledge, or maintaining an optimistic attitude by testing INDs which generalize $i$.

**Step 3: Getting estimates on the optimistic positive border.** Each time a candidate generated by an optimistic approach is detected to be false against the database, we try "to estimate" the distance between this element and the positive border of satisfied INDs. The idea is to count the number of tuples which does not satisfy the IND; we propose for that to use the error measure $g'_3$ [25] given by:

$$g'_3(R[X] \subseteq S[Y], \mathbf{d}) = 1 - \frac{max\{|\pi_X(r')| \mid r' \subseteq r, (\mathbf{d} - \{r\}) \cup \{r'\} \models R[X] \subseteq S[Y]\}}{|\pi_X(r)|}$$

Intuitively, $g'_3$ is the proportion of distinct values one has to remove from $\pi_X(r)$ to obtain a database $\mathbf{d}'$ such that $\mathbf{d}' \models R[X] \subseteq S[Y]$. Such a computation can be implemented with SQL queries on top of RDBMS. Clearly, $g'_3$ complies with the requirement given in the Property 1 (Section 3.3), i.e. $j \preceq i \Rightarrow g'_3(j) \leq g'_3(i)$.

**Step 4: Adaptive behavior.** When an IND $i$ of the optimistic positive border is false, but with a very small error, one can reasonably hope to find a satisfied IND among its nearest generalizations. Thus we consider the generalizations of $i$ from the more specific ones to the most general ones, i.e. implementing a top-down approach. Inversely when the error is large, i.e. a great number of values contradicts the IND, we start again the search in a bottom-up fashion. This step is described in much more details in Algorithm 1 (next section).

### 4.4 The Algorithm *Zigzag*

The principle of the Algorithm 1 is to mix top-down and bottom-up approaches for eliciting the positive border of satisfied INDs. As explained before, one search is performed for each couple of relation in the input database.

Initially (line 1) a purely pessimistic approach is performed from an adaptation of the levelwise algorithm $MIND$ [11], until the level $k$ fixed by the user is reached. We then know $I_k$ and $NI_k$, the set of the most specialized satisfied INDs and the set of the most general not satisfied INDs (resp.) of size smaller or equal to $k$. $I_k$ thus corresponds to an initialization of $\mathcal{B}d^+(I)$ and $NI_k$ to an initialization of $\mathcal{B}d^-(I)$, $I$ being the set of all satisfied INDs. The optimistic positive border $\mathcal{B}d^+_{opt}(NI_k)$ is then computed thanks to the Theorem 2 (line 3)[5]. The algorithm terminates when every element of $\mathcal{B}d^+_{opt}(NI_k)$ has already been

[5] The optimistic positive border generation is not detailed here. We used an adaptation of the algorithm proposed in [13].

tested as true in previous passes, i.e. $\mathcal{B}d^+_{opt}(NI_k) \setminus \mathcal{B}d^+(I)$ is empty (line 4). Otherwise, INDs of $\mathcal{B}d^+_{opt}(NI_k) \setminus \mathcal{B}d^+(I)$ are evaluated against the database: Those satisfied are added to $\mathcal{B}d^+(I)$, the others are divided into two groups according to the committed error: the "almost true" ones in the optimistic set *optDI* and the others in the pessimistic set *pessDI*. The INDs which generalize the INDs of *optDI* are traversed in a top-down fashion, from the most specific to the more general; $\mathcal{B}d^+(I)$ and $\mathcal{B}d^-(I)$ are updated accordingly (lines 14 to 21). Lastly, INDs of size $k+1$ which generalize the INDs of *pessDI* are tested, $\mathcal{B}d^+(I)$ and $\mathcal{B}d^-(I)$ are also updated (lines 23 to 26). $\mathcal{B}d^+_{opt}(NI_k)$ is then updated for the next iteration (line 28).

***Example 4.*** *Let us consider a database* $\mathbf{d} = \{r_1, r_2\}$ *over a schema* $\mathbf{R} = \{R_1, R_2\}$*, with* $R_1 = ABCDE$ *and* $R_2 = FGHIJ$*. Suppose that the set of satisfied unary INDs in* $\mathbf{d}$ *are:* $\{i_1 = A \subseteq F, i_2 = B \subseteq G, i_3 = C \subseteq H, i_4 = D \subseteq I, i_5 = E \subseteq J\}$*. The Figure 1 represents a subset of the search space of INDs over* $\mathbf{R}$*. For sake of clarity, not satisfied unary INDs are not represented since they are discarded by anti-monotony.*
*Let us illustrate Algorithm 1 with* $k = 2$ *over this toy example. After a levelwise search until level* 2*, the initialization is :*
$\mathcal{B}d^+(I) = \{AB \subseteq FG, AC \subseteq FH, AD \subseteq FI, AE \subseteq FJ, BC \subseteq GH, BD \subseteq GI, BE \subseteq GJ, CD \subseteq HI, DE \subseteq IJ\}$*;*
$\mathcal{B}d^-(I) = \{CE \subseteq HJ\}$*.*
$\mathcal{B}d^+_{opt}(I)$ *is then computed from* $\mathcal{B}d^-(I)$*, i.e.* $\mathcal{B}d^+_{opt}(I) = \{ABCD \subseteq FGHI, ABDE \subseteq FGIJ\}$ *(we omit the details).*
*These two INDs are tested over the database and let us assume that one is satisfied while the other one is not:*

- $ABCD \subseteq FGHI$ *is satisfied and added to* $\mathcal{B}d^+(I)$*, its generalizations being dropped from* $\mathcal{B}d^+(I)$*. Thus,* $\mathcal{B}d^+(I) = \{ABCD \subseteq FGHI, AE \subseteq FJ, BE \subseteq GJ, DE \subseteq IJ\}$*.*
- $ABDE \subseteq FGIJ$ *is not satisfied and added to* $\mathcal{B}d^-(I)$*:* $\mathcal{B}d^-(I) = \{CE \subseteq HJ, ABDE \subseteq FGIJ\}$*. Let us assume now that* $g'_3(ABDE \subseteq FGIJ)$ *is less than a user-supplied threshold. In that case, the generalizations of* $ABDE \subseteq FGIJ$ *of size 3 are generated and if they are not already specialized by an IND of* $\mathcal{B}d^+(I)$*, they are tested against the database. Thus,* $ABE \subseteq FGJ, ADE \subseteq FIJ$ *and* $BDE \subseteq GIJ$ *are tested, and if we assume they are satisfied,* $\mathcal{B}d^+(I)$ *is updated accordingly:*
  $\mathcal{B}d^+(I) = \{ABCD \subseteq FGHI, ABE \subseteq FGJ, ADE \subseteq FIJ, BDE \subseteq GIJ\}$*.*

### 4.5 Practical Aspects and Optimizations

*Dealing with not satisfied unary INDs.* In line 2 of algorithm 1, not satisfied unary INDs are added in the initialization of $\mathcal{B}d^-(I)$. In practice, we do not need to take them into account at one condition: they have to be removed from the set of unary INDs used during the computation of the complements of minimal transversal of the hypergraph associated with $\mathcal{B}d^-(I)$.

**Algorithm 1** Zigzag : IND cover discovery

**Require:** a database $\mathbf{d}$ over a schema $\mathbf{R}$, an integer $k$ and $R, S$ in $\mathbf{R}$
**Ensure:** $\mathcal{B}d^+(I)$ cover of the satisfied INDs from $R$ to $S$
1: Compute $I_k$ and $NI_k$ from $R$ to $S$ using a levelwise algorithm.
2: $\mathcal{B}d^+(I) = I_k$; $\mathcal{B}d^-(I) = NI_k$;
3: Compute $\mathcal{B}d^+_{opt}(I)$ from $\mathcal{B}d^-(I)$;
4: **while** $\mathcal{B}d^+_{opt}(I) \setminus \mathcal{B}d^+(I) \neq \emptyset$ **do**
5: $optDI = pessDI = \emptyset$;
6: **for all** $i \in \mathcal{B}d^+_{opt}(I) \setminus \mathcal{B}d^+(I)$ **do**
7: **if** $(g'_3(i, \mathbf{d}) = 0)$ **then** $\mathcal{B}d^+(I) = \mathcal{B}d^+(I) \cup \{i\} \setminus \{j \in \mathcal{B}d^+(I) \mid j \prec i\}$;
8: **else**
9: $\mathcal{B}d^-(I) = \mathcal{B}d^-(I) \cup i$;
10: **if** $(g'_3(i, \mathbf{d}) \leq \epsilon$ and$|i| > k + 1)$
11: **then** $optDI = optDI \cup \{i\}$;
12: **else** $pessDI = pessDI \cup \{i\}$;
13: **end for**
14: **while** $optDI \neq \emptyset$ **do**
15: $candidats = \cup_{i \in optDI}\{j \mid j \preceq i, |j| = |i| - 1$ and$|j| > k\}$;
16: **for all** $i \in candidats$ **do**
17: **if** $(\mathbf{d} \models i)$ **then** $\mathcal{B}d^+(I) = \mathcal{B}d^+(I) \cup \{i\} \setminus \{j \in \mathcal{B}d^+(I) \mid j \prec i\}$; $candidats = candidats \setminus \{i\}$;
18: **else** $\mathcal{B}d^-(I) = \mathcal{B}d^-(I) \cup \{i\} \setminus \{j \in \mathcal{B}d^-(I) \mid i \prec j\}$;
19: **end for**
20: $optDI = candidats$;
21: **end while**
22: $C_{k+1} = \cup_{i \in pessDI}\{j \mid j \prec i, |j| = k + 1\}$;
23: **for all** $i \in C_{k+1}$ **do**
24: **if** $(\mathcal{B}d^+(I) \models i$ or $\mathbf{d} \models i)$ **then**$\mathcal{B}d^+(I) = \mathcal{B}d^+(I) \cup \{i\} \setminus \{j \in \mathcal{B}d^+(I) \mid j \prec i\}$;
25: **else** $\mathcal{B}d^-(I) = \mathcal{B}d^-(I) \cup \{i\} \setminus \{j \in \mathcal{B}d^-(I) \mid i \prec j\}$;
26: **end for**
27: $k = k + 1$;
28: Compute $\mathcal{B}d^+_{opt}(I)$ from $\mathcal{B}d^-(I)$;
29: **end while**
30: Return $\mathcal{B}d^+(I)$.

*Dealing with repeated attributes in INDs.* The usual IND definition rejects repeated attributes in the left ot right-hand sides, since their pratical interest remains rather limited in databases. Nevertheless, we have pointed out that we had to have duplicated attributes in order to obtain a representation as sets for INDs.

In fact we are still able to answer the problem of IND discovery *without duplicate attributes*. For that, it is enough to add into the negative border, during its initialization (line 2 of algorithm 1) the set of INDs of size 2 with repeated attributes made up of two satisfied unary INDs.

Indeed, consider an IND $i$ having at least one repeated attribute on the left-hand side[6], i.e. $i = R[X_1AAX_2] \subseteq S[Y_1BCY_2]$. Thus there exists at least one

[6] The same justification still holds for right-hand side.

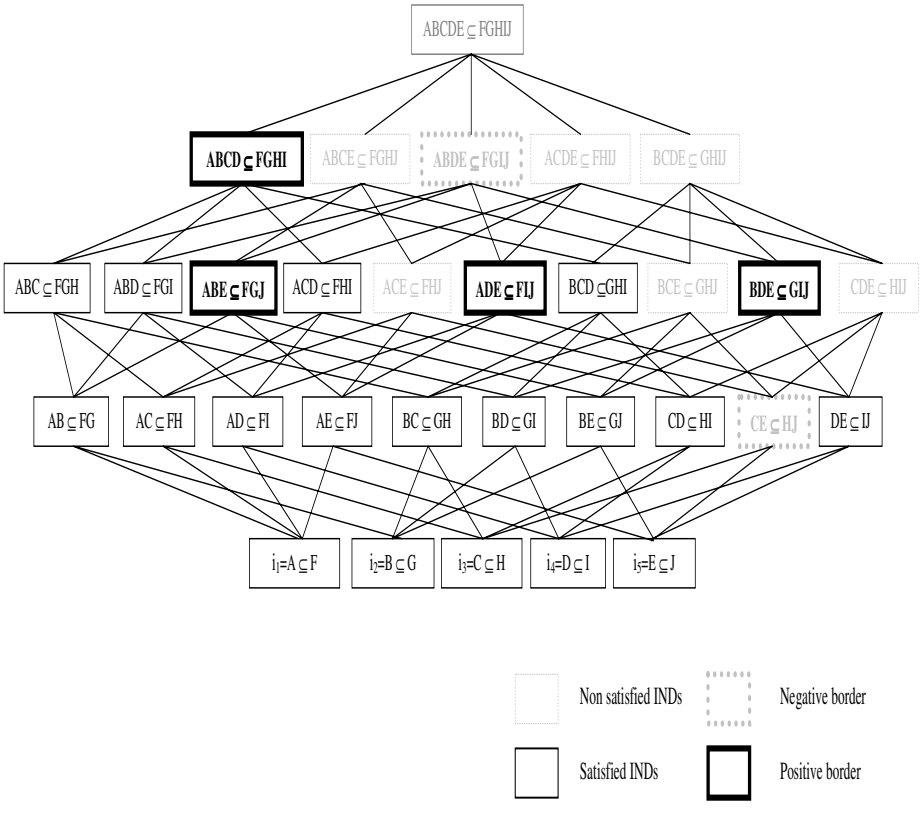


**Fig. 1.** A subset of the search space for INDs

IND of size 2, here $j = R[AA] \subseteq S[BC]$, which generalizes $i$. If $j$ belongs to the negative border, then $i$ cannot belong to the corresponding optimistic positive border, according to definition 1.

### 4.6 Experimental Results

Tests were carried out on synthetic databases in order to show the feasability of our proposition given in Section 3 on the IND discovery problem.

They were performed on an INTEL Pentium III 500 MHz, with 384 MB of main memory and running Windows 2000 Pro. The algorithms were implemented using C++/STL language. The test databases are stored under Oracle 9i, and data accesses were carried out via ODBC drivers. The tests were conducted on three databases having 2 relations, with 25 attributes and 90000 tuples in each relation. The databases differ on the constitution of the positive border of satisfied INDs to discover:

– database 1: 10 INDs, with arities of 2,5,6 and 7;
– database 2: 10 INDs, with arities of 3,5,6 and 11;
– database 3: 20 INDs, with arities of 6,8,13,17 and 18;

Table 1 gives execution times for IND discovery using algorithm Zigzag with $k = 2$. Times are compared with those given by the levelwise algorithm Mind [11]. The value for Mind on the third database is an estimate: it multiplies the number of tests to be carried out with the average cost of a test.

**Table 1.** Experimental results

| database | *Zigzag* | *Mind* |
|---|---|---|
| 1 | 1 754 s. | 2 790 s. |
| 2 | 3 500 s. | 25 626 s. |
| 3 | 7 729 s. | $\geq$ 1 year (estimate) |

First of all, these results confirm the failure of levelwise approach for large IND discovery, and thus reinforce the interest of proposing alternatives. Moreover, algorithm Zigzag makes it possible to reach INDs of size 18 in about only two hours (while Mind would have taken more than one year!), and thus shows the feasibility of the approach.

Nevertheless, we were not able to get feedbacks from our experiments on key paramaters of our propositions such as the impact of adaptive strategies (step 4). This is mainly due to the fact that "real-life" or synthetic databases are often not freely available and difficult to generate.

## 5 Related Works

*Maximal interesting pattern mining.* Several algorithms exist for discovering maximal interesting patterns; most of them were proposed in the specific case of maximal frequent itemsets mining in a transactional database. The goal is always to avoid an exponential search in the search space by characterizing as fast as possible large frequent itemsets without exploring their subsets. $MaxMiner$ [5] uses a levelwise approach to explore the candidate itemsets, using the Rymon's enumeration system [29] - in which itemsets are arranged in a non redundant tree. But when a candidate $X$ is counted over the database, the greatest candidate in the subtree of $X$ is also counted; if it is frequent, then all the subtree can be pruned by anti-monotony of the "is frequent" property. Jumps done by $MaxMiner$ depend on the ordering of items used to build the tree and are therefore quite different from jumps proposed in this paper.

The algorithms $Mafia$ [10] and $GenMax$ [16] use the same technique as $MaxMiner$ to "explore" the top of the search space. A difference lies in the fact that they reduce the number of tests by checking, for each candidate, if it is not a subset of a frequent itemsets already found. Moreover, $Mafia$ stores the

database in vertical bitmaps which appear to be extremely effective in practice. With respect to our optimistic positive border, the pruning of $GenMax$ appears to be more precise than the $MaxMiner$ pruning, thanks to a lemma which limits the size of the largest itemset to be explored. Despite of this optimization, their pruning remains always less precise than our pruning.

The $Pincer-Search$ Algorithm [22] uses a search strategy very close to ours. After a levelwise initialization, the principle is also to look at the largest not yet eliminated candidates. However, these large candidates are not characterized in a formal way.

In [17], the authors propose the *Dualize and Advance* algorithm. In their approach, the positive border in construction is always a subset of the positive border to be discovered. At each step, from some elements of the positive border already discovered, they generate the corresponding negative border. If one element of the negative border appears to be satisfied, they generate a specialization of it which belongs to the positive border and they re-iterate the process until each element of the negative border is indeed not satisfied. The same strategy is always made to explore the candidates, i.e. they cannot be guided by an estimation of the distance to the positive border and the number of dualization, i.e. minimal transversals computation, cannot be tuned.

*Adaptive data mining algorithms.* Some algorithms like $Mafia$ [10] or $DCI$ [28] can adapt themselves to mine frequent itemsets, with respect to the dataset density and some architectural characteristics (e.g. available memory). Even if these aspects improve performances, it only concerns choices for data structures; the mentioned algorithms do not really adapt their *strategy* to explore the search space.

In [8,3], the authors studied the addition of user-defined monotone constraint to facilitate exploration and reduce computation in the frequent pattern mining problem. If pushing monotone constraints can improve the pruning, it can also reduce the effectiveness of anti-monotone pruning, depending on the characteristics of the dataset. To cope with this difficulty, an adaptive algorithm was proposed based on an auto-adaptive search strategy.

*Inclusion dependency mining.* To our knowledge, only few contributions address a subset of the initial problem of IND discovery: problem declaration [18], unary IND discovery [6], or theoretical frameworks in which the problem of IND discovery could be solved [26,17]. An interesting contribution addressed the problem of large IND discovery [20]; the idea is to build an optimistic positive border starting from a set of known satisfied INDs, by introducing the concept of maximal hyperclique of a regular hypergraph, a concept very similar to the monotone dualization.

Note that an ongoing work based on results given in this chapter is currently done for maximal frequent itemsets from which an adaptive algorithm called ABS has been proposed [15].

## 6 Conclusion

From the theoretical framework of borders of theories, we have proposed a four steps approach to discover the positive border of interesting patterns. The key idea is to combine the strength of levelwise algorithms for small "maximal" interesting patterns with the strength of algorithms based on monotone dualization [17,30] for large maximal interesting patterns. We have introduced an adaptive behavior to guide the search from which "zigzaging" in the search space becomes possible.

We have applied our proposition to a data mining problem: the discovery of INDs in databases. An interesting aspect has been to point out the main steps in order to fit into the theoretical framework of borders. The principle of an optimistic attitude has been justified by a structural property of the relational model based on an interaction property between functional dependencies and inclusion dependencies. An algorithm called `Zigzag` has been devised and some experiments performed. Due to the very high cost of testing IND satisfaction against a database, `Zigzag` turns out to be more efficient in all configurations tested, even when the positive border to be discovered is not too far from the most generalized IND.

This work is integrated in a more general project devoted to DBA assistance and relational databases logical tuning, called "DBA Companion" [12].

## References

1. S. Abiteboul, R. Hull, and V. Vianu. *Fondements des bases de données*. Addison Wesley, 2000.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the* $20^{th}$ *International Conference on Very Large Databases, Santiago de Chile, Chile*, pages 487–499, 1994.
3. H. Albert-Lorincz and J.-F. Boulicaut. Mining frequent sequential patterns under regular expressions: A highly adaptive strategy for pushing contraints. In *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, CA, USA, 2003. SIAM.
4. J. Bailey, T. Manoukian, and K. Ramamohanarao. A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In *IEEE International Conference on Data Mining (ICDM'03), Floride, USA*, pages 485–488. IEEE Computer Society, November 2003.
5. R. Bayardo. Efficiently mining long patterns from databases. In L. M. Haas and A. Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 85–93. ACM Press, 1998.
6. S. Bell and P. Brockhausen. Discovery of Data Dependencies in Relational Databases. Technical report, LS-8 Report 14, University of Dortmund, 18p, April 1995.
7. C. Berge. *Graphs and Hypergraphs*. North-Holland Mathematical Library 6. American Elsevier, 2d rev. ed. edition, 1976.

8. F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Adaptive constraint pushing in frequent pattern mining. In *PKDD, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, volume 2838 of *Lecture Notes in Computer Science*, pages 47–58. Springer, 2003.
9. E. Boros, K. Elbassioni, V. Gurvich, and L. Khachiyan. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *special issue of Discrete Applied Mathematics.*
10. D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the* $17^{th}$ *IEEE International Conference on Data Engineering*, pages 443–452, Heidelberg, Germany, 2001. IEEE Computer Society.
11. F. De Marchi, S. Lopes, and J.-M. Petit. Efficient algorithms for mining inclusion dependencies. In *Proceedings of the* $7^{th}$ *International Conference on Extending Database Technology*, volume 2287 of *Lecture Notes in Computer Science*, pages 464–476, Prague, Czech Republic, 2002. Springer-Verlag.
12. F. De Marchi, S. Lopes, J.-M. Petit, and F. Toumani. Analysis of existing databases at the logical level: the dba companion project. *ACM Sigmod Record*, 32(1):47–52, 2003.
13. J. Demetrovics and V. Thi. Some remarks on generating Armstrong and inferring functional dependencies relation. *Acta Cybernetica*, 12(2):167–180, 1995.
14. T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. In *STOC 2002, Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, June 2-4, 1998, Montreal, Quebec, Canada*, pages 14 – 22. ACM Press, 2002.
15. F. Flouvat, F. D. Marchi, and J.-M. Petit. Abs: Adaptive borders search of frequent itemsets. In *FIMI'04*, 2004.
16. K. Gouda and M. J. Zaki. Efficiently mining maximal frequent itemsets. In *ICDM 2001, Proceedings IEEE International Conference on Data Mining*, pages 163–170. ACM Press, 2001.
17. D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharma. Discovering all most specific sentences. *ACM Transaction on Database Systems*, 28(2):140–174, 2003.
18. M. Kantola, H. Mannila, K.-J. Räihä, and H. Siirtola. Discovering functional and inclusion dependencies in relational databases. *International Journal of Intelligent Systems*, 7:591–607, 1992.
19. J. Kivinen and H. Mannila. Approximate inference of functional dependencies from relations. *Theoritical Computer Science*, 149(1):129–149, 1995.
20. A. Koeller and E. Rundensteiner. Discovery of high-dimentional inclusion dependencies (poster). In *International Conference on Data Engineering (ICDE'03).* IEEE Computer Society, 2003.
21. M. Levene and G. Loizou. *A Guided Tour of Relational Databases and Beyond.* Springer-Verlag, 1999.
22. D.-I. Lin and Z. M. Kedem. Pincer search: A new algorithm for discovering the maximum frequent set. In H.-J. Schek, F. Saltor, I. Ramos, and G. Alonso, editors, *Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology, Valencia, Spain, March 23-27, 1998, Proceedings*, volume 1377 of *Lecture Notes in Computer Science*, pages 105–119. Springer-Verlag, 1998.
23. S. Lopes, F. De Marchi, and J.-M. Petit. DBA companion: A tool for logical database tuning (demo). In $20^{th}$ *Proceedings of the IEEE International Conference on Data Engineering*, page 859, Boston, USA, 2004. IEEE Computer Society.

24. S. Lopes, J.-M. Petit, and L. Lakhal. Functional and approximate dependencies mining: Databases and FCA point of view. *Special issue of Journal of Experimental and Theoretical Artificial Intelligence*, 14(2/3):93–114, 2002.
25. S. Lopes, J.-M. Petit, and F. Toumani. Discovering interesting inclusion dependencies: Application to logical database tuning. *Information Systems*, 17(1):1–19, 2002.
26. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
27. J.-C. Mitchell. The implication problem for functional and inclusion dependencies. *Information and Control*, 56(3):154–173, 1983.
28. S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. Adaptive and resource-aware mining of frequent sets. In *International Conference on Data Mining (ICDM'02), Maebashi City, Japan*, pages 338–345. IEEE Computer Society, 2002.
29. R. Rymon. Search through systematic set enumeration. In B. Nebel, C. Rich, and W. R. Swartout, editors, *International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Cambridge, USA*, pages 539–550. Morgan Kaufmann, 1992.
30. T. Uno and K. Satoh. Detailed description of an algorithm for enumeration of maximal frequent sets with irredundant dualization. In B. Goethals and M. J. Zaki, editors, *FIMI '03, Frequent Itemset Mining Implementations, ICDM'03 Workshop*, volume 90 of *CEUR Workshop Proceedings*, Melbourne, Florida, USA, 2003.