The Data Mining and Knowledge Discovery Handbook,
O. Maimon and L. Rokach (Eds.), Springer, 2005, pp. 399-416.

Chapter 18

CONSTRAINT-BASED DATA MINING

Jean-Francois Boulicaut
INSA Lyon, LIRIS CNRS FRE 2672
69621 Villeurbanne cedex, France.
jean-francois.boulicaut@insa-lyon.fr

Baptiste Jeudy
University of Saint-Etienne, EURISE
42023 Saint-Etienne Cedex 2, France.
baptiste.jeudy@univ-st-etienne.fr

Abstract Knowledge Discovery in Databases (KDD) is a complex interactive process. The promising theoretical framework of inductive databases considers this is essentially a querying process. It is enabled by a query language which can deal either with raw data or patterns which hold in the data. Mining patterns turns to be the so-called inductive query evaluation process for which constraint-based Data Mining techniques have to be designed. An inductive query specifies declaratively the desired constraints and algorithms are used to compute the patterns satisfying the constraints in the data. We survey important results of this active research domain. This chapter emphasizes a real breakthrough for hard problems concerning local pattern mining under various constraints and it points out the current directions of research as well.

Keywords: Inductive querying, constraints, local patterns

1. Motivations

Knowledge Discovery in Databases (KDD) is a complex interactive and iterative process which involves many steps that must be done sequentially. Supporting the whole KDD process has enjoyed great popularity in recent years, with advances in both research and commercialization. We however still lack of a generally accepted underlying framework and this hinders the further development of the field. We believe that the quest for such a framework is a ma-

jor research priority and that the *inductive database* approach (IDB) (Imielinski and Mannila, 1996; De Raedt, 2003) is one of the best candidates in this direction. IDBs contain not only data, but also patterns. Patterns can be either *local patterns* (e.g., itemsets, association rules, sequences) which are of descriptive nature, or *global patterns/models* (e.g., classifiers) which are generally of predictive nature. In an IDB, ordinary queries can be used to access and manipulate data, while *inductive queries* can be used to generate (mine), manipulate, and apply patterns. KDD becomes an extended querying process where the analyst can control the whole process since he/she specifies the data and/or patterns of interests.

The IDB framework is appealing because it employs *declarative* queries instead of ad-hoc *procedural constructs*. As declarative inductive queries are often formulated using constraints, inductive querying needs for *constraint-based Data Mining* techniques and is concerned with defining the necessary constraints.

It is useful to abstract the meaning of inductive queries. A simple model has been introduced in (Mannila and Toivonen, 1997). Given a language $\mathcal{L}$ of patterns (e.g., itemsets), the *theory* of a database $\mathcal{D}$ w.r.t. $\mathcal{L}$ and a selection predicate C is the set $\mathsf{Th}(\mathcal{D},\mathcal{L},C) = \{\varphi \in \mathcal{L} \mid C(\varphi,\mathcal{D}) = \mathsf{true}\}$. The predicate selection or *constraint* C indicates whether a pattern φ is interesting or not (e.g., φ is "frequent" in $\mathcal{D}$). We say that computing $\mathsf{Th}(\mathcal{D},\mathcal{L},C)$ is the evaluation for the inductive query C defined as a boolean expression over primitive constraints. Some of them can refer to the "behavior" of a pattern in the data (e.g., its "frequency" is above a threshold). Frequency is indeed the most studied case of *evaluation function*. Some others define syntactical restrictions (e.g., the "length" of the pattern is below a threshold) and checking them does not need any access to the data. Preprocessing concerns the definition of a mining context $\mathcal{D}$, the mining phase is generally the computation of a theory while post-processing is often considered as a querying activity on a materialized theory. To support the whole KDD process, it is important to support the specification and the computation of many different but correlated theories.

According to this formalization, solving an inductive query needs for the computation of every pattern which satisfies C. We emphasized that the model is however quite general: beside the itemsets or sequences, $\mathcal{L}$ can denote, e.g., the language of partitions over a collection of objects or the language of decision trees on a collection of attributes. In these cases, classical constraints specify some function optimization. If the completeness assumption can be satisfied for most of the local pattern discovery tasks, it is generally impossible for optimization tasks like accuracy optimization during predictive model mining. In this case, heuristics or incomplete techniques are needed, which, e.g., compute sub-optimal decision trees. Very few techniques for constraint-based mining of models have been considered (see (Garofalakis and Rastogi, 2000)

for an exception) and we believe that studying constraint-based clustering or constraint-based mining of classifiers will be a major topic for research in the near future. Starting from now, we focus on local pattern mining tasks.

It is well known that a "generate and test" approach that would enumerate the patterns of $\mathcal{L}$ and then test the constraint C is generally impossible. A huge effort has been made by data mining researchers to make an active use of the primitive constraints occurring in C (*solver* design) such that useful mining query evaluation is tractable. On one hand, researchers have designed solvers for important primitive constraints. A famous example is the one of frequent itemset mining (FIM) where the data is a set of transactions, the patterns are itemsets and the primitive constraint is a minimal frequency constraint. A second major line of research has been to consider specific, say ad-hoc, techniques for conjunctions of some primitives constraints. Examples of seminal work are (Srikant *et al.*, 1997) for syntactic constraints on frequent itemsets, (Pasquier *et al.*, 1999) for frequent and closed set mining, or (Garofalakis *et al.*, 1999) for mining sequences that are both frequent and satisfy a given regular expression in a sequence database. Last but not the least, a major progress has concerned the design of generic algorithms for mining under conjunctions or arbitrary boolean combination of primitive constraints. A pioneer contribution has been (Ng *et al.*, 1998) and this kind of work consists in a classification of constraint properties and the design of solving strategies according to these properties (e.g., anti-monotonicity, monotonicity, succinctness).

Along with constraint-based Data Mining, the concept of *condensed representation* has emerged as a key concept for inductive querying. The idea is to compute $\mathcal{CR} \subset \mathsf{Th}(\mathcal{D}, \mathcal{L}, C)$ while deriving $\mathsf{Th}(\mathcal{D}, \mathcal{L}, C)$ from $\mathcal{CR}$ can be performed efficiently. In the context of huge database mining, efficiently means without any further access to $\mathcal{D}$. Starting from (Mannila and Toivonen, 1996) and its concrete application to frequency queries in (Boulicaut and Bykowski, 2000), many useful condensed representations have been designed the last 5 years. Interestingly, we can consider condensed representation mining as a constraint-based Data Mining task (Jeudy and Boulicaut, 2002). It provides not only nice examples of constraint-based mining techniques but also important cross-fertilization possibilities (combining the both concepts) for optimizing inductive queries in very hard contexts.

Section 2 provides the needed notations and concepts. It introduces the pattern domains of itemsets and sequences for which most of the constraint-based Data Mining techniques have been designed. Section 3 recalls the principal results for solving anti-monotonic constraints. Section 4 concerns the introduction of non anti-monotonic constraints and the various strategies which have been proposed. Section 5 concludes and points out the actual directions of research.

2. Background and Notations

Given a database $\mathcal{D}$, a pattern language $\mathcal{L}$ and a constraint C, let us first assume that we have to compute $\mathsf{Th}(\mathcal{D}, \mathcal{L}, C) = \{\varphi \in \mathcal{L} \mid C(\varphi, \mathcal{D}) = \mathsf{true}\}$. Our examples concern local pattern discovery tasks based on itemsets and sequences.

Itemsets have been studied a lot. Let $\mathcal{I} = \{\mathtt{A}, \mathtt{B}, ...\}$ be a set of *items*. A *transaction* is a subset of $\mathcal{I}$ and a database $\mathcal{D}$ is a multiset of transactions. An *itemset* is a set of items and a transaction t is said to *support* an itemset S if $S \subseteq t$. The *frequency* $\mathsf{freq}(S)$ of an itemset S is defined as the number of transactions that support S. $\mathcal{L}$ is the collection of all itemsets, i.e., $2^{\mathcal{I}}$. The most studied primitive constraint is the *minimum frequency constraint* $C_{\sigma\text{-}\mathsf{freq}}$ which is satisfied by itemsets having a frequency greater than the threshold σ. Many other constraints have been studied such as syntactical constraints, e.g., $\mathtt{B} \in X$ whose testing does not need any access to the data. (Ng *et al.*, 1998) is a rather systematic study of many primitive constraints on itemsets (see also Section 4). (Boulicaut, 2004) surveys some new primitive constraints based on the closure evaluation function. The closure of an itemset S in $\mathcal{D}$, $\mathbf{f}(S, \mathcal{D})$, is the maximal superset of S which has the same frequency than S in $\mathcal{D}$. Furthermore, a set S is closed in $\mathcal{D}$ if $S = \mathbf{f}(S, \mathcal{D})$ in which case we say that it satisfies C_{clos}. Freeness is one of the first proposals for constraint-based mining of closed set generators: free itemsets (Boulicaut *et al.*, 2000) (also called key patterns in (Bastide *et al.*, 2000B)) are itemsets whose frequencies are different from the frequencies of all their subsets. We say that they satisfy the C_{free} constraint. An important result is that $\{\mathbf{f}(S, \mathcal{D}) \in 2^{\mathcal{I}} \mid C_{\mathsf{free}}(S, \mathcal{D}) = \mathsf{true}\} = \{S \in 2^{\mathcal{I}} \mid C_{\mathsf{clos}}(S, \mathcal{D}) = \mathsf{true}\}$. For instance, in the toy data set of Figure 18.1, {A,C} is a free set and {A, C, D}, i.e., its closure, is a closed set.

Sequential pattern mining from sequence databases (i.e., $\mathcal{D}$ is a multiset of sequences) has been studied as well. Many different types of sequential patterns have been considered for which different *subpattern* relations can be defined. For instance, we could say that `bc` is a subpattern (substring) of `abca` but `aa` is not. In other proposals, `aa` would be considered as a subpattern of `abca`. Discussing this in details is not relevant for this chapter. The key point is that, a frequency evaluation function can be defined for sequential patterns (number of sequences in $\mathcal{D}$ for which the pattern is a subpattern). The pattern language $\mathcal{L}$ is then the infinite set of sequences which can be built on some alphabet. Many primitive constraints can be defined, e.g., minimal frequency or syntactical constraints specified by regular expressions. Interestingly, new constraints can exploit the spatial or temporal order, e.g., the `min-gap` and `max-gap` constraints (see, e.g., (Zaki, 2000) and (Pei *et al.*, 2002) for a recent survey).

Naive approaches that would compute $\mathsf{Th}(\mathcal{D}, \mathcal{L}, \mathcal{C})$ by enumerating every pattern φ of the search space $\mathcal{L}$ and test the constraint $\mathcal{C}(\varphi, \mathcal{D})$ afterwards can not work. Even though checking $\mathcal{C}(\varphi, \mathcal{D})$ can be cheap, this strategy fails because of the size of the search space. For instance, we have $2^{|\mathcal{I}|}$ itemsets and we often have to cope with hundreds or thousands of items in practical applications. Moreover, for sequential pattern mining, the search space is infinite.

For a given constraint, the search space $\mathcal{L}$ is often structured by a specialization relation which provides a lattice structure. For important constraints, the specialization relation has an anti-monotonicity property. For instance, set inclusion for itemsets or substring for strings are anti-monotonic specialization relations w.r.t. a minimal frequency constraint. Anti-monotonicity means that when a pattern does not satisfy $\mathcal{C}$ (e.g., an itemset is not frequent) then none of its specializations can satisfy $\mathcal{C}$ (e.g., none of its supersets are frequent). It becomes possible to prune huge parts of the search space which can not contain interesting patterns. This has been studied within the "learning as search" framework (Mitchell, 1980) and the generic *levelwise algorithm* from (Mannila and Toivonen, 1997) has inspired many algorithmic developments (see Section 3). In this context where we say that the constraint $\mathcal{C}$ is anti-monotonic, the most specific patterns constitute the *positive border* of the theory (denoted $\mathcal{B}d^+(\mathcal{C})$) (Mannila and Toivonen, 1997) and $\mathcal{B}d^+(\mathcal{C})$ is a condensed representation of $\mathsf{Th}(\mathcal{D}, \mathcal{L}, \mathcal{C})$. It corresponds to the S set in the terminology of versions spaces (Mitchell, 1980). For instance, the collection of the maximal frequent patterns $\mathcal{B}d^+(\mathcal{C}_{\sigma\text{-freq}})$ in $\mathcal{D}$ is generally several orders of magnitude smaller than the complete collection of the frequent patterns in $\mathcal{D}$. It is a condensed representation for $\mathsf{Th}(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\sigma\text{-freq}})$: deriving subsets (i.e., generalizations) of each maximal frequent set (i.e., each most specific pattern) enables to regenerate the whole collection of the frequent sets (i.e., the whole theory of interesting patterns w.r.t. the constraint).

In many applications, however, the user wants not only the collection of the patterns satisfying $\mathcal{C}$ but also the results of some evaluation functions for these patterns. This is quite typical for the frequent pattern discovery problem: these patterns are generally exploited in a post-processing step to derive more useful statements about the data, e.g., the popular frequent association rules which have a high enough confidence (Agrawal *et al.*, 1996). This can be done efficiently if we compute not only the collection of frequent itemsets but also their frequencies. In fact, the semantics of an inductive query is better captured by the concept of *extended theories*. An extended theory w.r.t. an evaluation function f on a domain $\mathcal{V}$ is $\mathsf{Th}_x(\mathcal{D}, \mathcal{L}, \mathcal{C}, f) = \{(\varphi, f(\varphi)) \in \mathcal{L} \otimes \mathcal{V} \mid \mathcal{C}(\varphi, \mathcal{D}) = \mathsf{true}\}$. The classical FIM problem turns to be the computation of $\mathsf{Th}_x(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\sigma\text{-freq}}, \mathsf{freq})$. Another example concerns the closure evaluation function.

For instance, $\{(\varphi, \mathbf{f}(\varphi)) \in 2^{\mathcal{I}} \otimes 2^{\mathcal{I}} \mid \mathcal{C}_{\sigma\text{-freq}}(\varphi, \mathcal{D}) = \mathsf{true}\}$ is the collection of the frequent sets and their closures, i.e., the frequent closed sets.

An alternative and useful specification for the frequent closed sets is $\{(\varphi, \mathbf{f}(\varphi)) \in 2^{\mathcal{I}} \otimes 2^{\mathcal{I}} \mid \mathcal{C}_{\sigma\text{-freq}}(\varphi, \mathcal{D}) \wedge \mathcal{C}_{\mathsf{free}}(\varphi, \mathcal{D}) = \mathsf{true}\}$.

Condensed representations can be designed for extended theories as well. Now, a condensed representation $\mathcal{CR}$ must enable to regenerate the patterns, but also the values of the evaluation function f on each pattern without any further access to the data. If the regenerated values for f are only approximated, the condensed representation is called *approximate*. Moreover, if the error on f can be bounded by ϵ, the approximate condensed representation is called an *ϵ-adequate representation* of the extended theory (Mannila and Toivonen, 1996). The idea is that we can trade off the precision on the evaluation function values with computational feasibility.

Most of condensed representations studied so far are condensed representations of the frequent itemsets. We have the maximal frequent itemsets (see, e.g., (Bayardo, 1998)), the frequent closed itemsets (see, e.g., (Pasquier *et al.*, 1999; Boulicaut and Bykowski, 2000)), the frequent free itemsets and the δ-free itemsets (Boulicaut *et al.*, 2000; Boulicaut *et al.*, 2003), the disjunction-free sets (Bykowski and Rigotti, 2003), the non-derivable itemsets (Calders and Goethals, 2002), the frequent pattern bases (Pei *et al.*, 2002), etc. Except for the maximal frequent itemsets from which it is not possible to get a useful approximation of the needed frequencies, these are condensed representations of the extended theory $\mathsf{Th}_x(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\sigma\text{-freq}}, \mathsf{freq})$ and δ-free itemsets and pattern bases are approximate representations.

Condensed representations have three main advantages. First, they contain (almost) the same information than the whole theory but are significantly smaller (generally by several orders of magnitude), which means that they are more easily stored or manipulated. Next, the computation of $\mathcal{CR}$ and the regeneration of the theory Th from $\mathcal{CR}$ is often less expensive than the direct computation of Th. One can even say that, as soon as a transactional data set is dense, mining condensed representations of the frequent itemsets is the only way to solve the FIM problem for practical applications. Last, many proposals emphasize the use of condensed representations for deriving directly useful patterns (i.e., skipping the regeneration phase). This is obvious for feature construction (see, e.g., (Kramer *et al.*, 2001)) but has been considered also for the generation of non redundant association rules (see, e.g., (Bastide *et al.*, 2000A)) or interesting classification rules (Crémilleux and Boulicaut, 2002)).

3. Solving Anti-Monotonic Constraints

In this section, we consider efficient solutions to compute (extended) theories for anti-monotonic constraints. We still focus on constraint-based mining

of itemsets when the constraint is anti-monotonic. It is however straightforwardly extended to many other pattern domains.

An *anti-monotonic* constraint on itemsets is a constraint denoted $\mathcal{C}_{\mathsf{am}}$ such that for all itemsets $S, S' \in 2^{\mathcal{I}}$: $(S' \subseteq S \wedge S$ satisfies $\mathcal{C}_{\mathsf{am}}) \Rightarrow S'$ satisfies $\mathcal{C}_{\mathsf{am}}$. $\mathcal{C}_{\sigma\text{-freq}}$, $\mathcal{C}_{\mathsf{free}}$, $\mathtt{A} \notin S$, $S \subseteq \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$ and $S \cap \{\mathtt{A}, \mathtt{B}, \mathtt{C}\} = \emptyset$ are examples of anti-monotonic constraints. Furthermore, it is clear that a disjunction or a conjunction of anti-monotonic constraints is an anti-monotonic constraint.

Let us be more precise on the useful concept of border (Mannila and Toivonen, 1997). If $\mathcal{C}_{\mathsf{am}}$ denotes an anti-monotonic constraint and the goal is to compute $T = \mathsf{Th}(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\mathsf{am}})$, then $\mathcal{B}d^{+}(\mathcal{C}_{\mathsf{am}})$ is the collection of the maximal (w.r.t. the set inclusion) itemsets of T that satisfy $\mathcal{C}_{\mathsf{am}}$ and $\mathcal{B}d^{-}(\mathcal{C}_{\mathsf{am}})$ is the collection of the minimal (w.r.t. the set inclusion) itemsets that do not satisfy $\mathcal{C}_{\mathsf{am}}$.

Some algorithms have been designed for computing directly the positive borders, i.e., looking for the complete collection of the most specific patterns. A famous one is the Max-Miner algorithm which uses a clever enumeration technique for computing depth-first the maximal frequent sets (Bayardo, 1998). Other algorithms for computing maximal frequent sets are described in (Lin and Kedem, 2002; Burdick *et al.*, 2001; Goethals and Zaki, 2003). The computation of positive borders with applications to not only itemset mining but also dependency discovery, the generic "dualize and advance" framework, is studied in (Gunopulos *et al.*, 2003).

The levelwise algorithm by Mannila and Toivonen (Mannila and Toivonen, 1997) has influenced many research in data mining. It computes $\mathsf{Th}(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\mathsf{am}})$ levelwise in the lattice ($\mathcal{L}$ associated to its specialization relation) by considering first the most general patterns (e.g., the singleton in the FIM problem). Then, it alternates candidate evaluation (e.g., frequency counting or other checks for anti-monotonic constraints) and candidate generation (e.g., building larger itemsets from discovered interesting itemsets) phases. Candidate generation can be considered as the computation of the negative border of the previously computed collection. *Candidate pruning* is a major issue and it can be performed partly during the generation phase or just after: indeed, any candidate whose one generalization does not satisfy $\mathcal{C}_{\mathsf{am}}$ can be pruned safely (e.g., any itemset whose one of its subsets is not frequent can be removed). The algorithm stops when it can not generate new candidates or, in other terms, when the most specific patterns have been found (e.g., all the maximal frequent itemsets).

The Apriori algorithm (Agrawal *et al.*, 1996) is clearly the most famous instance of this levelwise algorithm. It computes $\mathsf{Th}(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\sigma\text{-freq}}, \mathsf{freq})$ and it uses a clever candidate generation technique. A lot of work has been done for efficient implementations of Apriori-like algorithms.

Pruning based on anti-monotonic constraints has been proved efficient on hard problems, i.e., huge volume and high dimensional data sets. The many experimental results which are available nowadays prove that the minimal frequency is often an extremely selective constraint in real data sets. Interestingly, an algorithm like AcMiner (Boulicaut *et al.*, 2000; Boulicaut *et al.*, 2003) which can compute frequent closed sets (closeness is not an anti-monotonic constraint) via the frequent free sets exploits these pruning possibilities. Indeed, the conjunction of freeness and minimal frequency is an anti-monotonic constraint which enables an efficient pruning in dense and/or highly correlated data sets.

The dual property of monotonicity is interesting as well. A *monotonic* constraint on itemsets is a constraint denoted $\mathcal{C}_{\mathsf{m}}$ such that for all itemsets $S, S' \in 2^{\mathcal{I}}$: $(S \subseteq S' \wedge S \text{ satisfies } \mathcal{C}_{\mathsf{m}}) \Rightarrow S' \text{ satisfies } \mathcal{C}_{\mathsf{m}}$. A constraint is monotonic when its negation is anti-monotonic (and vice-versa). In the itemset pattern domain, the maximal frequency constraint or a syntactic constraint like $\mathtt{A} \in S$ are examples of monotonic constraints.

The concept of border can be adapted to monotonic constraints. The positive border $\mathcal{B}d^{+}(\mathcal{C}_{\mathsf{m}})$ of a monotonic constraint $\mathcal{C}_{\mathsf{m}}$ is the collection of the most general patterns that satisfy the constraint. The theory $\mathsf{Th}(\mathcal{D}, \mathcal{L}, \mathcal{C}_{\mathsf{m}})$ is then the set of patterns that are more specific than the patterns of the border $\mathcal{B}d^{+}(\mathcal{C}_{\mathsf{m}})$. For instance, we have $\mathcal{B}d^{+}(\mathtt{A} \in S) = \{\mathtt{A}\}$ and the positive border of the monotonic maximal frequency constraint is the collection of the smallest itemsets which are not frequent in the data. In other terms, a monotonic constraint defines also a border in the search space which corresponds to the G set in the version space terminology (see Figure 18.1 for an example).

The recent work has indeed exploited this duality for solving conjunctions of monotonic and anti-monotonic constraints (see Section 4.2).

4. Introducing non Anti-Monotonic Constraints

Pushing anti-monotonic constraints in the levelwise algorithm always leads to less constraint checking. Of course, anti-monotonic constraints are exploited into alternative frameworks, like depth-first algorithms.

However, this is no longer the case when pushing non anti-monotonic constraints. For instance, if an itemset does not satisfy an anti-monotonic constraint $\mathcal{C}_{\mathsf{am}}$, then its supersets can be pruned. But if this itemset does not satisfy the non anti-monotonic constraint, then its supersets are not pruned since the algorithm does not test $\mathcal{C}_{\mathsf{am}}$ on it. Pushing non anti-monotonic constraint can therefore lead to less efficient pruning (Boulicaut and Jeudy, 2000; Garofalakis *et al.*, 1999). Clearly, we have here a trade-off between anti-monotonic pruning and monotonic pruning which can be decided if the selectivity of the various constraints is known in advance, which is obviously not the case in most of

the applications. Nice contributions have considered boolean expressions over monotonic and anti-monotonic constraints. The problem is still quite open for optimization constraints.

4.1 The Seminal Work

4.1.1 MultipleJoins, Reorder and Direct. Srikant *et al.* (Srikant *et al.*, 1997) have been the first to address constraint-based mining of itemsets when the constraint $\mathcal{C}$ is not reduced to the minimum frequency constraint $\mathcal{C}_{\sigma\text{-freq}}$. They consider syntactical constraints built on two kinds of primitive constraints: $\mathcal{C}_i(S) = (i \in S)$, and $\mathcal{C}_{\neg i}(S) = (i \notin S)$ where $i \in \mathcal{I}$. They also introduce new constraints if a taxonomy on items is available. A taxonomy (also called a *is-a* relation) is an acyclic relation r on $\mathcal{I}$. For instance, if the items are products like Milk, Jackets... the relation can state that Milk *is-a* Beverages, Jackets *is-a* Outer-wear, ... The primitive constraints related to a taxonomy are: $\mathcal{C}_{\mathrm{a}(i)}(S) = (S \cap \text{ancestor}(i) \neq \emptyset)$, $\mathcal{C}_{\mathrm{d}(i)}(S) = (S \cap \text{descendant}(i) \neq \emptyset)$, and their negations. Functions ancestor and descendant are defined using the transitive closure r^* of r: we have $\text{ancestor}(i) = \{i' \in \mathcal{I} \mid r^*(i', i)\}$ and $\text{descendant}(i) = \{i' \in \mathcal{I} \mid r^*(i, i')\}$. These new constraints can be rewritten using the two primitive constraints $\mathcal{C}_i$ and $\mathcal{C}_{\neg i}$, e.g., $\mathcal{C}_{\text{desc}(i)}(S) = \bigvee_{j \in \text{descendant}(i)} \mathcal{C}_j(S)$.

It is now possible to specify syntactical constraints $\mathcal{C}_{\text{synt}}$ as a boolean combination of the primitive constraints which is written in disjunctive normal form, i.e., $\mathcal{C}_{\text{synt}} = D_1 \vee D_2 \vee \ldots \vee D_m$ where each D_k is $\mathcal{C}_{k1} \wedge \mathcal{C}_{k2} \wedge \ldots \wedge \mathcal{C}_{kn_k}$ and $\mathcal{C}_{kj}$ is either $\mathcal{C}_i$ or $\mathcal{C}_{\neg i}$ with $i \in \mathcal{I}$.

Srikant *et al.* (Srikant *et al.*, 1997) provide three algorithms to compute $\mathsf{Th}_x(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}, \mathsf{freq})$ where $\mathcal{C} = \mathcal{C}_{\sigma\text{-freq}} \wedge \mathcal{C}_{\text{synt}}$. The first two algorithms (MultipleJoins and Reorder) use a relaxation of the syntactical constraint. They show how to compute from $\mathcal{C}_{\text{synt}}$ an itemset T such that every itemset S satisfying the $\mathcal{C}_{\text{synt}}$ also satisfies the constraint $S \cap T \neq \emptyset$. This constraint is pushed in an Apriori-like levelwise algorithm to obtain MultipleJoins and Reorder (Reorder is a simplification of MultipleJoins). The third algorithm, Direct, does not use a relaxation and pushes the whole syntactical constraint at the extended cost of a more complex candidate generation phase. Experimental results confirm that the behavior of the algorithms depends clearly of the selectivity of the constraints on the considered data sets.

4.1.2 CAP. The CAP algorithm (Ng *et al.*, 1998) computes the extended theory $\mathsf{Th}_x(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}, \mathsf{freq})$ for $\mathcal{C} = \mathcal{C}_{\sigma\text{-freq}} \wedge \mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\text{succ}}$ where $\mathcal{C}_{\mathsf{am}}$ is an anti-monotonic syntactical constraint and $\mathcal{C}_{\text{succ}}$ is a *succinct* constraint. A constraint $\mathcal{C}$ is succinct (Ng *et al.*, 1998) if it is a syntactical constraint and if we have itemsets $I_1, I_2, \ldots I_k$ such that $\mathcal{C}(S) = S \subseteq I_1 \wedge S \not\subseteq I_2 \wedge \ldots \wedge S \not\subseteq I_k$. Efficient candidate generation techniques can be performed for such constraints

which can be considered as special cases of conjunctions of anti-monotonic and monotonic syntactical constraints.

In (Ng *et al.*, 1998), the syntactical constraints are conjunctions of primitive constraints which are $\mathcal{C}_i$, $\mathcal{C}_{\neg i}$ and constraints based on aggregates. They indeed assume that a value v is associated with each item i and denoted $i.v$ such that several aggregate functions can be used:

$$\text{MAX}(S) = \max\{i.v \mid i \in S\}, \qquad \text{MIN}(S) = \min\{i.v \mid i \in S\},$$
$$\text{SUM}(S) = \sum_{i \in S} i.v, \qquad \text{AVG}(S) = \frac{\text{SUM}(S)}{|S|}.$$

These aggregate functions enable to define new primitive constraints $\text{AGG}(S)\theta n$ where AGG is an aggregation function, θ is in $\{=, <, >\}$ and n is a number. In a market basket analysis application, v can be the price of each item and we can define aggregate constraints to extract, e.g., itemsets whose average price of items is above a given threshold ($\text{AVG}(S) > 10$). Among these constraints, some are anti-monotonic (e.g., $\text{SUM}(S) < 100$ if all the values are positive, $\text{MIN}(S) > 10$), some are succinct (e.g., $\text{MAX}(S) > 10$, $|S| > 3$) and others have no special properties and must be relaxed to be used in the CAP algorithm (e.g., $\text{SUM}(S) < 10$, $\text{AVG}(S) < 10$).

The candidate generation function of CAP algorithm is an improvement over Direct algorithm. However, it can not use all syntactical constraints like Direct (only conjunction of anti-monotonic and succinct constraints can be used by CAP). The CAP algorithm can also use aggregate constraints. These constraints could also be used in Direct but they would need to be rewritten in disjunctive normal form using $\mathcal{C}_i$ and $\mathcal{C}_{\neg i}$. This rewriting stage can be computationally expensive such that, in practice, we can not push aggregate constraints into Direct.

4.1.3 SPIRIT. In (Garofalakis *et al.*, 1999), the authors present several version of the SPIRIT algorithm to extract frequent sequences satisfying a regular expression (such sequences are called *valid* w.r.t. the regular expression). For instance, if the sequences consist of letters, the valid sequences with respect to the regular expression `a* (bb | cc) e` are the sequences that start with several `a` followed by either `bbe` or `cce`. In the general case, such a syntactical constraint is not anti-monotonic. The different versions of SPIRIT use more and more selective relaxations of this regular expression constraint. The first algorithm, SPIRIT(N), uses an anti-monotonic relaxation of the syntactical constraint. This constraint $\mathcal{C}_N$ is satisfied by sequences s such that all the items appearing in s also appear in the regular expression. With our running example, $\mathcal{C}_N(s)$ is true if s is built on letters `a`, `b`, `c`, and `e` only. A constraint $\mathcal{C}_L$ is used by the second algorithm, SPIRIT(L). It is satisfied by a sequence s if s is a legal sequence w.r.t. the regular expression. A sequence s is legal

if we can find a valid sequence s' such that s is a suffix of s'. For instance, cce is a legal sequence w.r.t. our running example. The SPIRIT(V) algorithm uses the constraint C_V which is satisfied by all contiguous sub-sequences of a valid sequence. Finally, the SPIRIT(R) algorithm uses the full constraint C_R which is satisfied only by valid sequences. For the three first algorithms, a final post-processing step is necessary to filter out non-valid sequences. There is a subset relationship between the theories computed by these four algorithms: $\mathsf{Th}(\mathcal{D}, \mathcal{L}, C_R \wedge C_{\sigma\text{-freq}}) \subseteq \mathsf{Th}(\mathcal{D}, \mathcal{L}, C_V \wedge C_{\sigma\text{-freq}}) \subseteq \mathsf{Th}(\mathcal{D}, \mathcal{L}, C_L \wedge C_{\sigma\text{-freq}}) \subseteq \mathsf{Th}(\mathcal{D}, \mathcal{L}, C_N \wedge C_{\sigma\text{-freq}})$. Clearly, the first two algorithms are based mostly on minimal frequency pruning while the two last ones exploit further regular expression pruning. Here again, only a prior knowledge on constraint selectivity enables to inform the choice of one of the algorithms, i.e., one of the pruning strategies.

4.2 Generic Algorithms

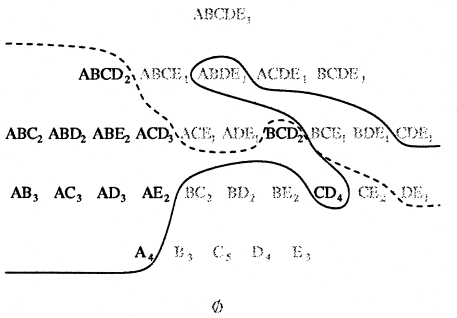
We now sketch some important results for the evaluation of quite general forms of inductive queries.

4.2.1 Conjunction of Monotonic and Anti-Monotonic Constraints.

Let us assume that we use constraints that are conjunctions of a monotonic constraint and an anti-monotonic one denoted $C_{\mathsf{am}} \wedge C_{\mathsf{m}}$. The structure of $\mathsf{Th}(\mathcal{D}, \mathcal{L}, C_{\mathsf{am}} \wedge C_{\mathsf{m}})$ is well known. Given the positive borders $\mathcal{B}d^+(C_{\mathsf{am}})$ and $\mathcal{B}d^+(C_{\mathsf{m}})$, the patterns belonging to $\mathsf{Th}(\mathcal{D}, \mathcal{L}, C_{\mathsf{am}} \wedge C_{\mathsf{m}})$ are exactly the patterns that are more specific than a pattern of $\mathcal{B}d^+(C_{\mathsf{m}})$ and more general than a pattern of $\mathcal{B}d^+(C_{\mathsf{am}})$. This kind of convex pattern collection is called a *Version Space* and is illustrated on Fig. 18.1.

Several algorithms have been developed to deal with $C_{\mathsf{am}} \wedge C_{\mathsf{m}}$. The generic algorithm presented in (Boulicaut and Jeudy, 2000) computes the extended theory for a conjunction $C_{\mathsf{am}} \wedge C_{\mathsf{m}}$. It is a levelwise algorithm, but instead of starting the exploration with the most general patterns (as it is done for anti-monotonic constraints), it starts with the minimal itemsets (most general patterns) satisfying C_{m}, i.e., the itemsets of the border $\mathcal{B}d^+(C_{\mathsf{m}})$. This is a generalization of MultipleJoins, Reorder and CAP: the constraint $T \cap S \neq \emptyset$ used in MultipleJoins and Reorder is indeed monotonic and succinct constraints used in CAP can be rewritten as the conjunction of a monotonic and an anti-monotonic constraints.

Since $\mathcal{B}d^+(C_{\mathsf{am}})$ et $\mathcal{B}d^+(C_{\mathsf{m}})$ characterize the theory of $C_{\mathsf{am}} \wedge C_{\mathsf{m}}$, these borders are a condensed representation of this theory. The Molfea algorithm



$\mathcal{D} =$

TID	Transaction
1	ABCDE
2	ABCD
3	ABE
4	ACD
5	CD
6	CE

Figure 18.1. This figure shows the itemset lattice associated to $\mathcal{D}$ (the subscript number is the frequency of each itemset in $\mathcal{D}$). The itemsets above the black line satisfy the monotonic constraint $\mathcal{C}_{\mathsf{m}}(S) = (\mathtt{B} \in S) \vee (\mathtt{CD} \subseteq S)$ and the itemsets below the dashed line satisfy the anti-monotonic constraint $\mathcal{C}_{\mathsf{am}} = \mathcal{C}_{\text{2-freq}}$. The black itemsets belong to the theory $\mathsf{Th}(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\mathsf{m}})$. They are exactly the itemsets that are subsets of an element of $\mathcal{B}d^{+}(\mathcal{C}_{\mathsf{am}}) = \{\mathtt{ABCD}, \mathtt{ABE}, \mathtt{CE}\}$ and supersets of an element of $\mathcal{B}d^{+}(\mathcal{C}_{\mathsf{m}}) = \{\mathtt{A}, \mathtt{CD}\}$.

and the Dualminer algorithms extract these two borders. They are interesting algorithms for feature extraction.

The Molfea algorithm presented in (Kramer *et al.*, 2001; De Raedt and Kramer, 2001) extract linear molecular fragments (i.e., strings) in a a partitioned database of molecules (say, active vs. inactive molecules). They consider conjunctions of a minimum frequency constraint (say in the active molecules), a maximum frequency constraint (say in the inactive ones) and syntactical constraints. The two borders are constructed in an incremental fashion, considering the constraints one after the other, using a level-wise algorithm for the frequency constraints and Mellish algorithm (Mellish, 1992) for the syntactical constraints. The Dualminer algorithm (Bucila *et al.*, 2003) uses a depth-first exploration similar to the one of Max-Miner whereas Dualminer deals with $\mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\mathsf{m}}$ instead of just $\mathcal{C}_{\mathsf{am}}$.

In (Bonchi *et al.*, 2003C), the authors consider the computation of not only borders but also the extended theory for $\mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\mathsf{m}}$. In this context, they show that the most efficient approach is not to reason on the search space only but both the search space and the transactions from the input data. They have a clever approach to data reduction based on the monotonic part. Not only it does not affect anti-monotonic pruning but also they demonstrate that the two pruning opportunities are mutually enhanced.

4.2.2 Arbitrary Expression over Monotonic and Anti Monotonic Constraints. The algorithms presented so far cannot deal with an arbitrary boolean expression consisting of monotonic and anti-monotonic constraints. These more general constraints are studied in (De Raedt *et al.*, 2002). Using

the basic properties of monotonic and anti-monotonic constraints, the authors show that such a constraint can be rewritten as $(\mathcal{C}_{\mathsf{am}_1} \wedge \mathcal{C}_{\mathsf{m}_1}) \vee (\mathcal{C}_{\mathsf{am}_2} \wedge \mathcal{C}_{\mathsf{m}_2}) \vee \ldots \vee (\mathcal{C}_{\mathsf{am}_n} \wedge \mathcal{C}_{\mathsf{m}_n})$. The theory of each conjunction $(\mathcal{C}_{\mathsf{am}_i} \wedge \mathcal{C}_{\mathsf{m}_i})$ is a version space and the theory w.r.t. the whole constraint is a union of version spaces. The theory of each conjunction can be computed using any algorithm described in the previous sections. Since there are several ways to express the constraint as a disjunction of conjunctions, it is therefore desirable to find an expression in which the number of conjunction is minimal.

4.2.3 Conjunction of Arbitrary Constraints. When constraints are neither anti-monotonic nor monotonic, finding an efficient algorithm is difficult. The common approach is to design a specific strategy to deal with a particular class of constraints. Such algorithms are presented in the next section. A promising generic approach has been however presented recently. It is the concept of *witness* presented in (Kifer *et al.*, 2003) for itemset mining. This paper does not describe a mining algorithm but rather a pruning technique for non anti-monotonic and non monotonic constraints. Considering a sub-lattice $\mathring{A}$ of $2^{\mathcal{I}}$, the problem is to decide whether this sub-lattice can be pruned. A sub-lattice is characterized by its maximal element M and its minimal element m, i.e., the sub-lattice is the collection of all itemsets S such that $m \subseteq S \subseteq M$. To prune this sub-lattice, one must prove that none of its elements can satisfy the constraint $\mathcal{C}$. To check this, the authors introduce the concept of negative witness: a negative witness for $\mathcal{C}$ in the sub-lattice $\mathring{A}$ is an itemset W such that $\neg\mathcal{C}(W) \Rightarrow \forall X \in \mathring{A},\ \neg\mathcal{C}(X)$. Therefore, if the constraint is not satisfied by the negative witness, then the whole sub-lattice can be pruned. Finding witnesses for anti-monotonic or monotonic constraints is easy : m is the witness for all anti-monotonic constraints and M for all monotonic ones. The authors then show how to compute efficiently witnesses for various tough constraints. For instance, for $\mathrm{AVG}(S) > \sigma$, a witness is the set $m \cup \{i \in M \mid i.v > \sigma\}$. The authors also gives an algorithm (linear in the size of $\mathcal{I}$) to compute a witness for the difficult constraint $(\mathrm{VAR}(S) > \sigma)$ where VAR denotes the variance.

4.3 Ad-hoc Strategies

Apart from generic algorithms, many algorithms have been designed to cope with specific classes of constraints. We select only two examples.

The FIC algorithm (Pei *et al.*, 2001) does a depth-first exploration of the itemset lattice. It is very efficient due to its clever data structure, a prefix-tree used to store the database. This algorithm can compute the extended theory for a conjunction $\mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\mathsf{m}} \wedge \mathcal{C}'$ where $\mathcal{C}'$ is *convertible anti-monotonic or monotonic*. A constraint $\mathcal{C}'$ is convertible anti-monotonic if there exists an order on the items such that, if itemsets are written using this order, every prefix of an itemset satisfying $\mathcal{C}'$ satisfies $\mathcal{C}'$. For instance, $\mathrm{AVG}(S) > \sigma$

is convertible anti-monotonic if the items i are ordered by decreasing value $i.v$. The main problem with convertible constraints is that a conjunction of convertible constraints is generally not convertible.

Another example of an ad-hoc strategy is used in the c-Spade algorithm (Zaki, 2000). This algorithm is used to extract constrained sequences where each event in the sequences is dated. One of the constraints, the $max - gap$ constraint, states that two consecutive events occurring in a pattern must not be further apart than a given maximum gap. This constraint is neither anti-monotonic nor monotonic and a specific algorithm has been designed for it.

4.4 Other Directions of Research

Among others, let us introduce here three important directions of research.

4.4.1 Adaptive Pruning Strategies. We mentioned the trade-off between anti-monotonic pruning which is known to be quite efficient and pruning based on non anti-monotonic constraints. Since the selectivity of the various constraints is generally unknown, a quite exciting challenge is to look for adaptive strategies which can decide of the pruning strategy dynamically. (Bonchi *et al.*, 2003A; Bonchi *et al.*, 2003B) propose algorithms for frequent itemsets under syntactical monotonic constraints. (Albert-Lorincz and Boulicaut, 2003) considers frequent sequence mining under regular expression constraints. These are promising approaches to widen the applicability of constraint-based mining techniques in real contexts.

4.4.2 Combining Constraints and Condensed Representations. A few papers, e.g., (Boulicaut and Jeudy, 2000; Bonchi and Lucchese, 2004), deal with the problem of extracting constrained condensed representation. In these works, the aim is to compute a condensed representation of the extended theory $\mathsf{Th}_x(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\mathsf{m}}, \mathsf{freq})$. In (Boulicaut and Jeudy, 2000), the authors use free itemsets, i.e., their algorithm computes the extended theory $\mathsf{Th}_x(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\mathsf{m}} \wedge \mathcal{C}_{\mathsf{free}}, \mathsf{freq})$. In (Bonchi and Lucchese, 2004), the authors use closed itemsets, i.e., their algorithm computes the extended theory $\mathsf{Th}_x(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\mathsf{m}} \wedge \mathcal{C}_{\mathsf{clos}}, \mathsf{freq})$. However, in these two works, the definition of free sets and closed sets have been modified to be able to regenerate the extended theory $\mathsf{Th}_x(\mathcal{D}, 2^{\mathcal{I}}, \mathcal{C}_{\mathsf{am}} \wedge \mathcal{C}_{\mathsf{m}}, \mathsf{freq})$ from the extracted theories. This kind of research combines the advantages of both condensed representations and constrained mining which result in very efficient algorithms.

4.4.3 Constraint-based Mining of more Complex Pattern Domains. Most of the recent results have concerned simple local pattern discovery tasks like the ones based on itemsets or sequences. We believe that inductive querying is much more general. Many open problems are how-

ever to be addressed. For instance, even constraint-based mining of association rules is already much harder than constraint-based mining of itemsets (Lakshmanan *et al.*, 1999; Jeudy and Boulicaut, 2002). The recent work on the MINE RULE query language (Meo *et al.*, 1998) is also typical of the difficulty to optimize constraint-based association rule mining (Meo, 2003). When considering model mining under constraints (e.g., classifier design or clustering), only very preliminary approaches are available (see, e.g., (Garofalakis and Rastogi, 2000)). We think that this will be a major issue for research in the next few years. For instance, for clustering, it seems important to go further than the classical similarity optimization constraints and enable to specify other constraints on clusters (e.g., enforcing that some objects are or are not within the same clusters).

5. Conclusion

In this chapter, we have considered constraint-based mining approaches, i.e., the core techniques for inductive querying.

This domain has been studied a lot for simple pattern domains like itemsets or sequences. Rather general forms of inductive queries on these domains (e.g., arbitrary boolean expressions over monotonic and anti-monotonic constraints) have been considered. Beside the many ad-hoc algorithms, an interesting effort has concerned generic algorithms. Many open problems are still there: how to solve tough constraints?, how to design relevant approximation or relaxation schemes? how to combine constraint-based mining with condensed representations, not only for simple pattern domains but also more complex ones?

Moreover, within the inductive database framework, the problem is to optimize sequences of queries and typically sequences of correlated inductive queries. It is crucial to consider that the optimization of a query and thus constraint-based mining must also take into account the previously solved queries. Looking for the formal properties between inductive queries, especially containment, is thus a major priority. Here again, we believe that condensed representations might play a major role.

Last but not the least, a quite challenging problem is to consider from where the constraints come. The analysts can think in terms of constraints or declarative specifications which are not supported by the available solvers: an obvious example could be unexpectedness or novelty w.r.t. some explicit background knowledge. To be able to derive appropriate inductive queries based on a limited number of primitives (and some associated solvers) from the constraints expressed by the analysts is challenging.

References

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.

H. Albert-Lorincz and J.-F. Boulicaut. Mining frequent sequential patterns under regular expressions: a highly adaptative strategy for pushing constraints. In *Proc. SIAM DM'03*, pages 316–320, 2003.

Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proc. CL 2000*, volume 1861 of *LNCS*, pages 972–986. Springer-Verlag, 2000.

Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, 2000.

R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. ACM SIGMOD'98*, pages 85–93, 1998.

F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Adaptive constraint pushing in frequent pattern mining. In *Proc. PKDD'03*, volume 2838 of *LNAI*, pages 47–58. Springer-Verlag, 2003A.

F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Examiner: Optimized level-wise frequent pattern mining with monotone constraints. In *Proc. IEEE ICDM'03*, pages 11–18, 2003B.

F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Exante: Anticipated data reduction in constrained pattern mining. In *Proc. PKDD'03*, volume 2838 of *LNAI*, pages 59–70. Springer-Verlag, 2003C.

F. Bonchi and C. Lucchese. On closed constrained frequent pattern mining. In *Proc. IEEE ICDM'04 (In Press)*, 2004.

J.-F. Boulicaut. Inductive databases and multiple uses of frequent itemsets: the cInQ approach. In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, volume 2682 of *LNCS*, pages 1–23. Springer-Verlag, 2004.

J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary Data Mining. In *Proc. PAKDD'00*, volume 1805 of *LNAI*, pages 62–73. Springer-Verlag, 2000.

J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proc. PKDD'00*, volume 1910 of *LNAI*, pages 75–85. Springer-Verlag, 2000.

J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets : a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, 7(1):5–22, 2003.

J.-F. Boulicaut and B. Jeudy. Using constraint for itemset mining: should we prune or not? In *Proc. BDA'00*, pages 221–237, 2000.

J.-F. Boulicaut and B. Jeudy. Mining free-sets under constraints. In *Proc. IEEE IDEAS'01*, pages 322–329, 2001.

C. Bucila, J. E. Gehrke, D. Kifer, and W. White. Dualminer: A dual-pruning algorithm for itemsets with constraints. *Data Mining and Knowledge Discovery*, 7(4):241–272, 2003.

D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *Proc. IEEE ICDE'01*, pages 443–452, 2001.

A. Bykowski and C. Rigotti. DBC: a condensed representation of frequent patterns for efficient mining. *Information Systems*, 28(8):949–977, 2003.

T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Proc. PKDD'02*, volume 2431 of *LNAI*, pages 74–85. Springer-Verlag, 2002.

B. Crémilleux and J.-F. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In *Proc. ES 2002*, pages 33–46. Springer-Verlag, 2002.

L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2003.

L. De Raedt, M. Jaeger, S. Lee, and H. Mannila. A theory of inductive query answering. In *Proc. IEEE ICDM'02*, pages 123–130, 2002.

L. De Raedt and S. Kramer. The levelwise version space algorithm and its application to molecular fragment finding. In *Proc. IJCAI'01*, pages 853–862, 2001.

M. M. Garofalakis and R. Rastogi. Scalable Data Mining with model constraints. *SIGKDD Explorations*, 2(2):39–48, 2000.

M. M. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: Sequential pattern mining with regular expression constraints. In *Proc. VLDB'99*, pages 223–234, 1999.

B. Goethals and M. J. Zaki, editors. *Proc. of the IEEE ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, volume 90 of *CEUR Workshop Proceedings*, 2003.

D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharm. Discovering all most specific sentences. *ACM Transactions on Database Systems*, 28(2):140–174, 2003.

T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.

B. Jeudy and J.-F. Boulicaut. Optimization of association rule mining queries. *Intelligent Data Analysis*, 6(4):341–357, 2002.

D. Kifer, J. E. Gehrke, C. Bucila, and W. White. How to quickly find a witness. In *Proc. ACM PODS'03*, pages 272–283, 2003.

S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in HIV data. In *Proc. ACM SIGKDD'01*, pages 136–143, 2001.

L. V. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. In *Proc. ACM SIGMOD'99*, pages 157–168, 1999.

D.-I. Lin and Z. M. Kedem. Pincer search: An efficient algorithm for discovering the maximum frequent sets. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):553–566, 2002.

H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. KDD'96*, pages 189–194. AAAI Press, 1996.

H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

C. Mellish. The description identification problem. *Artificial Intelligence*, 52(2):151–168, 1992.

R. Meo. Optimization of a language for Data Mining. In *Proc. ACM SAC'03 - Data Mining Track*, pages 437–444, 2003.

R. Meo, G. Psaila, and S. Ceri. An extension to SQL for mining association rules. *Data Mining and Knowledge Discovery*, 2(2):195–224, 1998.

T. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1980.

R. Ng, L. V. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. ACM SIGMOD'98*, pages 13–24, 1998.

N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.

J. Pei, G. Dong, W. Zou, and J. Han. On computing condensed frequent pattern bases. In *Proc. IEEE ICDM'02*, pages 378–385, 2002.

J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proc. IEEE ICDE'01*, pages 433–442, 2001.

R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proc. ACM SIGKDD'97*, pages 67–73, 1997.

M. J. Zaki. Sequence mining in categorical domains: incorporating constraints. In *Proc. ACM CIKM'00*, pages 422–429, 2000.